\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Filename: Abbreviature.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.WebPages;

using KPIWeb;

namespace KPIWeb

{

public class Abbreviature

{

public static string deleteSpaces(string input)

{

string tmpStr = input;

tmpStr = tmpStr.Replace(" ", "");

return tmpStr;

}

public static string[] splitString(string input)

{

string tmpStr = input;

string[] abbArray = tmpStr.Split('/', '^', '+', '-', '(', ')', '\*', ' ');

return abbArray;

}

public static List<CalculatedParametrs> GetCalculatedList(string input)

{

KPIWebDataContext kpiweb = new KPIWebDataContext();

List<CalculatedParametrs> tmpList = new List<CalculatedParametrs>();

string tmpStr;

tmpStr = input;

deleteSpaces(tmpStr);

tmpStr = tmpStr.Replace("\r", "");

tmpStr = tmpStr.Replace("\n", "");

string[] abbArray = splitString(tmpStr);

foreach (string str in abbArray)

{

if ((str != null) && (str != " ") && (!str.IsEmpty()))

{

if (!str.IsFloat())

{

CalculatedParametrs result = (from a in kpiweb.CalculatedParametrs

where a.AbbreviationEN == str

select a).FirstOrDefault();

if (result != null)

{

tmpList.Add(result);

}

}

}

}

// LogHandler.LogWriter.WriteLog(LogCategory.ERROR, errorList.ToString());

return tmpList;

}

public static List<BasicParametersTable> GetBasicList(string input)

{

KPIWebDataContext kpiweb = new KPIWebDataContext();

List<BasicParametersTable> tmpList = new List<BasicParametersTable>();

string tmpStr;

tmpStr = input;

deleteSpaces(tmpStr);

tmpStr = tmpStr.Replace("\r", "");

tmpStr = tmpStr.Replace("\n", "");

string[] abbArray = splitString(tmpStr);

foreach (string str in abbArray)

{

if ((str != null) && (str != " ") && (!str.IsEmpty()))

{

if (!str.IsFloat())

{

BasicParametersTable result = (from a in kpiweb.BasicParametersTable

where a.AbbreviationEN == str

select a).FirstOrDefault();

if (result != null)

{

tmpList.Add(result);

}

}

}

}

// LogHandler.LogWriter.WriteLog(LogCategory.ERROR, errorList.ToString());

return tmpList;

}

public static string ReturnCalcFormula(string abb)

{

// string abbTmp = abb;

//string[] tmpArr = abbTmp.Split('#');

KPIWebDataContext KPIWebDataContext = new KPIWebDataContext();

string Formula = (from a in KPIWebDataContext.CalculatedParametrs

where a.Active == true

&& a.AbbreviationEN == abb

select a.Formula).FirstOrDefault();

if (Formula == null) // так быть не должно)

{

Formula = "0";

}

// Formula = "" + Formula + "";

return "(" + Formula + ")"; ;

}

public static string CalculatedAbbToFormula(string IndicatorFormulaString)

{

// DateTime time1 = DateTime.Now; //Точка начала отсчета времени

string BigFormulaString = IndicatorFormulaString;

List<CalculatedParametrs> CalculatedList = GetCalculatedList(IndicatorFormulaString);

foreach (CalculatedParametrs Calculated in CalculatedList)

{

int idx = BigFormulaString.IndexOf(Calculated.AbbreviationEN);

if (idx < 0)

{

//FORMULA\_ERROR

}

else

{

BigFormulaString = BigFormulaString.Remove(idx, Calculated.AbbreviationEN.Length)

.Insert(idx, ReturnCalcFormula(Calculated.AbbreviationEN));

}

}

/\* DateTime time2 = DateTime.Now; //Точка окончания отсчета времени

long elapsedTicks = time2.Ticks - time1.Ticks; // подсчитываем число тактов, один такт соответствует 100 наносекундам

double tmpp = elapsedTicks \* 1E-7; // делим на 10^7 для отображения времени в секундах

int j =0;\*/

return BigFormulaString;

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Filename: About.aspx.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using System;

using System.Configuration;

using System.Collections.Generic;

using System.Data;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

namespace KPIWeb

{

public partial class About : Page

{

protected void Page\_Load(object sender, EventArgs e)

{

if (!IsPostBack)

{

KPIWebDataContext kPiDataContext = new KPIWebDataContext();

List<ManualTable> man = (from a in kPiDataContext.ManualTable where a.Active == true select a).ToList();

LinksLable.Text = "";

foreach(ManualTable link in man)

{

LinksLable.Text += "<a href=\"" + ConfigurationManager.AppSettings.Get("SiteName") + "/manuals/" + link.ManualLink + "\" target=\"\_blank\" >" + link.ManualName + "</a> </br>";

}

}

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Filename: About.aspx.designer.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//------------------------------------------------------------------------------

// <автоматически создаваемое>

// Этот код создан программой.

//

// Изменения в этом файле могут привести к неправильной работе и будут потеряны в случае

// повторной генерации кода.

// </автоматически создаваемое>

//------------------------------------------------------------------------------

namespace KPIWeb {

public partial class About {

/// <summary>

/// LinksLable элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Label LinksLable;

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Filename: Action.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using System;

using System.Collections.Generic;

using System.Configuration;

using System.Linq;

using System.Net;

using System.Net.Mail;

using System.Text;

using System.Text.RegularExpressions;

using System.Web;

namespace KPIWeb

{

public static class Action

{

public static int Encode(string code) // расшифровка кода специальности

{

string pattern = @"\b(\.+\d+\.)";

Regex regex = new Regex(pattern);

Match match = regex.Match(code);

char[] charsToTrim = { '.', ' ', '\'' };

string result = match.Groups[1].Value.Trim(charsToTrim);

switch (result)

{

case "03": return 1; // бакалавр

case "04": return 3; // магистр

case "05": return 2; // специалист

case "06": return 4; // аспирант

case "08": return 4; // аспирант

}

return 5;

}

public static string EncodeToStr(string code) // расшифровка кода специальности

{

string pattern = @"\b(\.+\d+\.)";

Regex regex = new Regex(pattern);

Match match = regex.Match(code);

char[] charsToTrim = { '.', ' ', '\'' };

string result = match.Groups[1].Value.Trim(charsToTrim);

switch (result)

{

case "03":

return "бакалавриат";

case "04":

return "магистратура";

case "05":

return "специалитет";

case "06":

return "аспирантура";

case "08":

return "аспирантура";

}

return "не опр.";

}

private static int SendMail(string smtpServer, int port, string from, string password,

string mailto, string caption, string message, string attachFile)

{

try

{

MailMessage mail = new MailMessage();

mail.From = new MailAddress(from, "ИАС. КФУ-Программа развития");

mail.To.Add(new MailAddress(mailto));

mail.Subject = caption;

mail.Body = message;

if (!string.IsNullOrEmpty(attachFile))

mail.Attachments.Add(new Attachment(attachFile));

SmtpClient client = new SmtpClient();

client.Host = smtpServer;

client.Port = port;

client.EnableSsl = true;

client.Credentials = new NetworkCredential(from.Split('@')[0], password);

client.DeliveryMethod = SmtpDeliveryMethod.Network;

client.Send(mail);

mail.Dispose();

}

catch (Exception e)

{

return 0;

}

return 1;

}

public static int MassMailing(string emailto, string caption, string message, string attachFile)

{

int errors=0;

KPIWebDataContext kpiWeb = new KPIWebDataContext();

var emails = (from a in kpiWeb.EmailSendTable select a).ToList();

StringBuilder messageBuilder = new StringBuilder();

messageBuilder.Append(Environment.NewLine);

messageBuilder.Append(Environment.NewLine);

messageBuilder.Append(message);

messageBuilder.Append(Environment.NewLine);

messageBuilder.Append(Environment.NewLine);

messageBuilder.Append("По вопросам заполнения форм отчетности обращайтесь по телефону: +7-978-823-14-32");

messageBuilder.Append(Environment.NewLine);

messageBuilder.Append("Техническая поддержка: +7-978-117-53-98");

messageBuilder.Append(Environment.NewLine);

messageBuilder.Append("E-mail адрес: otdel-avtomatizatsii-kfu@yandex.ru");

messageBuilder.Append(Environment.NewLine);

messageBuilder.Append("С уважением администрация \"ИАС.КФУ-Программа развития\"");

foreach (var ems in emails)

{

if (SendMail(ems.SMTPName, 587, ems.Email, ems.Password, emailto, caption, messageBuilder.ToString(), attachFile) == 1)

{

ems.SendOk++;

LogHandler.LogWriter.WriteLog(LogCategory.INFO, "0MS0: From mail " + ems.Email + " success send an email to " + emailto + " with caption:\" " + caption + " \"");

break;

}

ems.SendError++;

errors++;

LogHandler.LogWriter.WriteLog(LogCategory.ERROR, "0MS1: From mail " + ems.Email + " failed send an email to " + emailto + " with caption:\" " + caption + " \"");

}

kpiWeb.SubmitChanges();

return errors;

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Filename: CalculateAbb.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using System;

using System.Collections.Generic;

using System.Dynamic;

using System.Linq;

using System.Text;

using System.Web;

using System.Web.WebPages;

using System.Text.RegularExpressions;

namespace KPIWeb

{

public class CalculateAbb

{

public static List<string> errorList = new List<string>();

public static string replaseAbbWithValueForLevel(int type, string input, int reportId, int specID, int Lv0,

int Lv1, int Lv2, int Lv3,

int Lv4, int Lv5)

{

string abbTmp = input;

KPIWebDataContext kpiWebDataContext = new KPIWebDataContext();

double? a = 0;

if (specID == 0)

{

a = (from collect in kpiWebDataContext.CollectedBasicParametersTable

join basic in kpiWebDataContext.BasicParametersTable

on collect.FK\_BasicParametersTable equals basic.BasicParametersTableID

where

(collect.FK\_ZeroLevelSubdivisionTable == Lv0 || Lv0 == 0)

&& (collect.FK\_FirstLevelSubdivisionTable == Lv1 || Lv1 == 0)

&& (collect.FK\_SecondLevelSubdivisionTable == Lv2 || Lv2 == 0)

&& (collect.FK\_ThirdLevelSubdivisionTable == Lv3 || Lv3 == 0)

&& (collect.FK\_FourthLevelSubdivisionTable == Lv4 || Lv4 == 0)

&& (collect.FK\_FifthLevelSubdivisionTable == Lv5 || Lv5 == 0)

&& basic.AbbreviationEN == abbTmp

&& collect.FK\_ReportArchiveTable == reportId

select collect.CollectedValue).Sum();

}

else

{

a = (from collect in kpiWebDataContext.CollectedBasicParametersTable

join fourth in kpiWebDataContext.FourthLevelSubdivisionTable

on collect.FK\_FourthLevelSubdivisionTable equals fourth.FourthLevelSubdivisionTableID

join basic in kpiWebDataContext.BasicParametersTable

on collect.FK\_BasicParametersTable equals basic.BasicParametersTableID

where

fourth.FK\_Specialization == specID

&& (collect.FK\_ZeroLevelSubdivisionTable == Lv0 || Lv0 == 0)

&& (collect.FK\_FirstLevelSubdivisionTable == Lv1 || Lv1 == 0)

&& (collect.FK\_SecondLevelSubdivisionTable == Lv2 || Lv2 == 0)

&& (collect.FK\_ThirdLevelSubdivisionTable == Lv3 || Lv3 == 0)

&& (collect.FK\_FourthLevelSubdivisionTable == Lv4 || Lv4 == 0)

&& (collect.FK\_FifthLevelSubdivisionTable == Lv5 || Lv5 == 0)

&& basic.AbbreviationEN == abbTmp

select collect.CollectedValue).Sum();

}

if (a == null) // так быть не должно)

{

a = 0;

// LogHandler.LogWriter.WriteLog(LogCategory.ERROR, "Замена аббревиатуры вернула NULL "+

// abbTmp+" "+Lv0+" "+Lv1+" "+Lv2+" "+Lv3+" "+Lv4+" "+Lv5+" " +reportId);

}

return a.ToString();

}

public static string deleteSpaces(string input)

{

string tmpStr = input;

tmpStr = tmpStr.Replace(" ","");

return tmpStr;

}

public static List<int> GetBasicIdList(string input)

{

KPIWebDataContext kpiweb = new KPIWebDataContext();

List<int> tmpList = new List<int>();

string tmpStr;

tmpStr = input;

deleteSpaces(tmpStr);

tmpStr = tmpStr.Replace("\r", "");

tmpStr = tmpStr.Replace("\n", "");

string[] abbArray = splitString(tmpStr);

foreach (string str in abbArray)

{

if ((str != null) && (str != " ") && (!str.IsEmpty()))

{

if (!str.IsFloat())

{

int result = (from a in kpiweb.BasicParametersTable

where a.AbbreviationEN == str

select a.BasicParametersTableID).FirstOrDefault();

if ((result != null)&&(result > 0))

{

tmpList.Add(result);

}

}

}

}

// LogHandler.LogWriter.WriteLog(LogCategory.ERROR, errorList.ToString());

return tmpList;

}

public static List<int> GetCollectIdList(string input)

{

KPIWebDataContext kpiweb = new KPIWebDataContext();

List<int> tmpList = new List<int>();

string tmpStr;

tmpStr = input;

deleteSpaces(tmpStr);

tmpStr = tmpStr.Replace("\r", "");

tmpStr = tmpStr.Replace("\n", "");

string[] abbArray = splitString(tmpStr);

foreach (string str in abbArray)

{

if ((str != null) && (str != " ") && (!str.IsEmpty()))

{

if (!str.IsFloat())

{

int result = (from a in kpiweb.CalculatedParametrs

where a.AbbreviationEN == str

select a.CalculatedParametrsID).FirstOrDefault();

if ((result != null) && (result > 0))

{

tmpList.Add(result);

}

}

}

}

// LogHandler.LogWriter.WriteLog(LogCategory.ERROR, errorList.ToString());

return tmpList;

}

public static string[] splitString(string input)

{

string tmpStr = input;

string[] abbArray = tmpStr.Split('/','^','+','-','(',')','\*',' ');

return abbArray;

}

public static string CheckAbbString(string input)

{

string tmpStr;

tmpStr = input;

if (tmpStr=="")

{

return "Поле формулы не заполнена";

}

deleteSpaces(tmpStr);

string[] abbArray = splitString(tmpStr);

KPIWebDataContext KPIWebDataContext = new KPIWebDataContext();

tmpStr = "0";

foreach (string str in abbArray)

{

if ((str != null) && (str != " ") && (!str.IsEmpty()))

{

if (!str.IsFloat())

{

string strtmp;

char[] charsToTrim = {'\n','\r',' '};

strtmp=str.TrimEnd(charsToTrim);

strtmp = strtmp.TrimStart(charsToTrim);

strtmp=strtmp.Replace("\r\n", "");

int a = (from basic in KPIWebDataContext.BasicParametersTable

where

basic.AbbreviationEN == strtmp

select basic).Count();

if (a>1)

{

tmpStr += "\r\n" + a.ToString() + " включений аббревиатуры " + str;

}

else if(a<1)

{

tmpStr +="\r\n"+str + " такой аббревиатеры не существует" ;

}

}

}

}

return tmpStr;

}

public static double CalculateForLevel(int type, string input, int report, int spec, int Lv0, int Lv1, int Lv2, int Lv3, int Lv4, int Lv5, int param1)

{

string tmpStr;

tmpStr = input;

deleteSpaces(tmpStr);

tmpStr = tmpStr.Replace("\r", "");

tmpStr = tmpStr.Replace("\n", "");

string[] abbArray = splitString(tmpStr);

//tmpStr = "";

foreach (string str in abbArray)

{

if ((str != null) && (str != " ") && (!str.IsEmpty()))

{

if (!str.IsFloat())

{

int idx = tmpStr.IndexOf(str);

if (idx != -1)

tmpStr = tmpStr.Remove(idx, str.Length).Insert(idx, replaseAbbWithValueForLevel(type, str, report, spec, Lv0, Lv1, Lv2, Lv3, Lv4, Lv5));

}

}

}

// LogHandler.LogWriter.WriteLog(LogCategory.ERROR, errorList.ToString());

return Polish.Calculate(tmpStr);

}

public static double? SumForLevel(int BasicId, int report,int SpecID, int Lv0, int Lv1, int Lv2, int Lv3, int Lv4, int Lv5)

{

KPIWebDataContext KPIWebDataContext = new KPIWebDataContext();

double? a = 0;

if (SpecID == 0)

{

a = (from collect in KPIWebDataContext.CollectedBasicParametersTable

where

(collect.FK\_ZeroLevelSubdivisionTable == Lv0 || Lv0 == 0)

&& (collect.FK\_FirstLevelSubdivisionTable == Lv1 || Lv1 == 0)

&& (collect.FK\_SecondLevelSubdivisionTable == Lv2 || Lv2 == 0)

&& (collect.FK\_ThirdLevelSubdivisionTable == Lv3 || Lv3 == 0)

&& (collect.FK\_FourthLevelSubdivisionTable == Lv4 || Lv4 == 0)

&& (collect.FK\_FifthLevelSubdivisionTable == Lv5 || Lv5 == 0)

&& collect.FK\_BasicParametersTable == BasicId

select collect.CollectedValue).Sum();

}

else

{

a = (from collect in KPIWebDataContext.CollectedBasicParametersTable

join Fourth in KPIWebDataContext.FourthLevelSubdivisionTable

on collect.FK\_FourthLevelSubdivisionTable equals Fourth.FourthLevelSubdivisionTableID

where

Fourth.FK\_Specialization == SpecID

&& (collect.FK\_ZeroLevelSubdivisionTable == Lv0 || Lv0 == 0)

&& (collect.FK\_FirstLevelSubdivisionTable == Lv1 || Lv1 == 0)

&& (collect.FK\_SecondLevelSubdivisionTable == Lv2 || Lv2 == 0)

&& (collect.FK\_ThirdLevelSubdivisionTable == Lv3 || Lv3 == 0)

&& (collect.FK\_FourthLevelSubdivisionTable == Lv4 || Lv4 == 0)

&& (collect.FK\_FifthLevelSubdivisionTable == Lv5 || Lv5 == 0)

&& collect.FK\_BasicParametersTable == BasicId

select collect.CollectedValue).Sum();

}

if (a == null)

{

a = 0;

}

return a;

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Filename: CalculateIndicator.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

namespace KPIWeb

{

public class CalculateIndicator

{

public static double Calculate(int IndicatorsTableID, int ReportArchiveTableID)

{

double? returnValue = -1;

KPIWebDataContext KPIWebDataContext = new KPIWebDataContext();

switch (IndicatorsTableID)

{

case 1:

double? a = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 15

select item.CollectedValue).Sum();

double? b = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 21

select item.CollectedValue).Sum();

double? c = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 28

select item.CollectedValue).Sum();

double? d = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 35

select item.CollectedValue).Sum();

if (a == null) a = 0;

if (b == null) b = 0;

if (c == null) c = 0;

if (d == null) d = 0;

double? M\_pk = (double)a + ((double)b \* 0.25) + (((double)c + (double)d) \* 0.1);

//---------------------------------------------------------------------------------------------------------

a = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 2

select item.CollectedValue).Sum();

b = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 5

select item.CollectedValue).Sum();

c = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 8

select item.CollectedValue).Sum();

d = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 11

select item.CollectedValue).Sum();

if (a == null) a = 0;

if (b == null) b = 0;

if (c == null) c = 0;

if (d == null) d = 0;

double? A\_pk = (double)a + ((double)b \* 0.25) + (((double)c + (double)d) \* 0.1);

//---------------------------------------------------------------------------------------------------------

a = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 63

select item.CollectedValue).Sum();

b = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 69

select item.CollectedValue).Sum();

c = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 75

select item.CollectedValue).Sum();

d = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 81

select item.CollectedValue).Sum();

if (a == null) a = 0;

if (b == null) b = 0;

if (c == null) c = 0;

if (d == null) d = 0;

double? B\_pk = (double)a + ((double)b \* 0.25) + (((double)c + (double)d) \* 0.1);

//---------------------------------------------------------------------------------------------------------

a = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 42

select item.CollectedValue).Sum();

b = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 47

select item.CollectedValue).Sum();

c = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 52

select item.CollectedValue).Sum();

d = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 57

select item.CollectedValue).Sum();

if (a == null) a = 0;

if (b == null) b = 0;

if (c == null) c = 0;

if (d == null) d = 0;

double? S\_pk = (double)a + ((double)b \* 0.25) + (((double)c + (double)d) \* 0.1);

returnValue = (M\_pk + A\_pk) / (B\_pk + S\_pk + M\_pk + A\_pk) \* 100;

break;

case 2:

double? SBB\_EG = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 108

select item.CollectedValue).Sum();

double? SBB\_EG\_DI = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 109

select item.CollectedValue).Sum();

double? ChB\_EG\_DI = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 113

select item.CollectedValue).Sum();

double? ChB\_EG = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 112

select item.CollectedValue).Sum();

double? ChS\_EG = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 119

select item.CollectedValue).Sum();

double? ChS\_EG\_DI = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 120

select item.CollectedValue).Sum();

if (SBB\_EG == null) SBB\_EG = 0;

if (SBB\_EG\_DI == null) SBB\_EG\_DI = 0;

if (ChB\_EG\_DI == null) ChB\_EG\_DI = 0;

if (ChB\_EG == null) ChB\_EG = 0;

if (ChS\_EG == null) ChS\_EG = 0;

if (ChS\_EG\_DI == null) ChS\_EG\_DI = 0;

returnValue = ((SBB\_EG \* (ChB\_EG + ChS\_EG) + SBB\_EG\_DI \* (ChB\_EG + ChS\_EG\_DI)) / (ChB\_EG + ChS\_EG + ChB\_EG\_DI + ChS\_EG\_DI));

break;

case 3:

double? a\_M\_dvo\_pk = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 16

select item.CollectedValue).Sum();

double? b\_M\_dvo\_pk = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 23

select item.CollectedValue).Sum();

double? c\_M\_dvo\_pk = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 30

select item.CollectedValue).Sum();

double? d\_M\_dvo\_pk = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 37

select item.CollectedValue).Sum();

if (a\_M\_dvo\_pk == null) a\_M\_dvo\_pk = 0;

if (b\_M\_dvo\_pk == null) b\_M\_dvo\_pk = 0;

if (c\_M\_dvo\_pk == null) c\_M\_dvo\_pk = 0;

if (d\_M\_dvo\_pk == null) d\_M\_dvo\_pk = 0;

double M\_dvo\_pk = (double)a\_M\_dvo\_pk + ((double)b\_M\_dvo\_pk \* 0.25) + (((double)c\_M\_dvo\_pk + (double)d\_M\_dvo\_pk) \* 0.1);

//---------------------------------------------------------------------------------------------------------

a\_M\_dvo\_pk = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 2

select item.CollectedValue).Sum();

b\_M\_dvo\_pk = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 5

select item.CollectedValue).Sum();

c\_M\_dvo\_pk = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 8

select item.CollectedValue).Sum();

d\_M\_dvo\_pk = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 22

select item.CollectedValue).Sum();

if (a\_M\_dvo\_pk == null) a\_M\_dvo\_pk = 0;

if (b\_M\_dvo\_pk == null) b\_M\_dvo\_pk = 0;

if (c\_M\_dvo\_pk == null) c\_M\_dvo\_pk = 0;

if (d\_M\_dvo\_pk == null) d\_M\_dvo\_pk = 0;

double А\_pk = (double)a\_M\_dvo\_pk + ((double)b\_M\_dvo\_pk \* 0.25) + (((double)c\_M\_dvo\_pk + (double)d\_M\_dvo\_pk) \* 0.1);

//---------------------------------------------------------------------------------------------------------

a\_M\_dvo\_pk = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 4

select item.CollectedValue).Sum();

b\_M\_dvo\_pk = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 7

select item.CollectedValue).Sum();

c\_M\_dvo\_pk = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 10

select item.CollectedValue).Sum();

d\_M\_dvo\_pk = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 13

select item.CollectedValue).Sum();

if (a\_M\_dvo\_pk == null) a\_M\_dvo\_pk = 0;

if (b\_M\_dvo\_pk == null) b\_M\_dvo\_pk = 0;

if (c\_M\_dvo\_pk == null) c\_M\_dvo\_pk = 0;

if (d\_M\_dvo\_pk == null) d\_M\_dvo\_pk = 0;

double А\_dvo\_pk = (double)a\_M\_dvo\_pk + ((double)b\_M\_dvo\_pk \* 0.25) + (((double)c\_M\_dvo\_pk + (double)d\_M\_dvo\_pk) \* 0.1);

a = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 2

select item.CollectedValue).Sum();

b = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 5

select item.CollectedValue).Sum();

c = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 8

select item.CollectedValue).Sum();

d = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 11

select item.CollectedValue).Sum();

if (a == null) a = 0;

if (b == null) b = 0;

if (c == null) c = 0;

if (d == null) d = 0;

A\_pk = (double)a + ((double)b \* 0.25) + (((double)c + (double)d) \* 0.1);

a = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 15

select item.CollectedValue).Sum();

b = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 21

select item.CollectedValue).Sum();

c = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 28

select item.CollectedValue).Sum();

d = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 35

select item.CollectedValue).Sum();

if (a == null) a = 0;

if (b == null) b = 0;

if (c == null) c = 0;

if (d == null) d = 0;

M\_pk = (double)a + ((double)b \* 0.25) + (((double)c + (double)d) \* 0.1);

returnValue = (M\_dvo\_pk + А\_dvo\_pk) / (M\_pk + A\_pk) \* 100;

break;

case 4:

double? a\_B\_dco\_pk\_oz = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 65

select item.CollectedValue).Sum();

double? b\_B\_dco\_pk\_oz = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 71

select item.CollectedValue).Sum();

double? c\_B\_dco\_pk\_oz = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 77

select item.CollectedValue).Sum();

double? d\_B\_dco\_pk\_oz = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 83

select item.CollectedValue).Sum();

if (a\_B\_dco\_pk\_oz == null) a\_B\_dco\_pk\_oz = 0;

if (b\_B\_dco\_pk\_oz == null) b\_B\_dco\_pk\_oz = 0;

if (c\_B\_dco\_pk\_oz == null) c\_B\_dco\_pk\_oz = 0;

if (d\_B\_dco\_pk\_oz == null) d\_B\_dco\_pk\_oz = 0;

double? B\_dco\_pk\_oz = (double)a\_B\_dco\_pk\_oz + ((double)b\_B\_dco\_pk\_oz \* 0.25) + (((double)c\_B\_dco\_pk\_oz + (double)d\_B\_dco\_pk\_oz) \* 0.1);

//---------------------------------------------------------------------------------------------------------

a\_B\_dco\_pk\_oz = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 43

select item.CollectedValue).Sum();

b\_B\_dco\_pk\_oz = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 48

select item.CollectedValue).Sum();

c\_B\_dco\_pk\_oz = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 53

select item.CollectedValue).Sum();

d\_B\_dco\_pk\_oz = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 58

select item.CollectedValue).Sum();

if (a\_B\_dco\_pk\_oz == null) a\_B\_dco\_pk\_oz = 0;

if (b\_B\_dco\_pk\_oz == null) b\_B\_dco\_pk\_oz = 0;

if (c\_B\_dco\_pk\_oz == null) c\_B\_dco\_pk\_oz = 0;

if (d\_B\_dco\_pk\_oz == null) d\_B\_dco\_pk\_oz = 0;

double? S\_dco\_pk\_oz = (double)a\_B\_dco\_pk\_oz + ((double)b\_B\_dco\_pk\_oz \* 0.25) + (((double)c\_B\_dco\_pk\_oz + (double)d\_B\_dco\_pk\_oz) \* 0.1);

//---------------------------------------------------------------------------------------------------------

a\_B\_dco\_pk\_oz = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 17

select item.CollectedValue).Sum();

b\_B\_dco\_pk\_oz = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 24

select item.CollectedValue).Sum();

c\_B\_dco\_pk\_oz = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 31

select item.CollectedValue).Sum();

d\_B\_dco\_pk\_oz = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 38

select item.CollectedValue).Sum();

if (a\_B\_dco\_pk\_oz == null) a\_B\_dco\_pk\_oz = 0;

if (b\_B\_dco\_pk\_oz == null) b\_B\_dco\_pk\_oz = 0;

if (c\_B\_dco\_pk\_oz == null) c\_B\_dco\_pk\_oz = 0;

if (d\_B\_dco\_pk\_oz == null) d\_B\_dco\_pk\_oz = 0;

double? M\_dco\_pk\_oz = (double)a\_B\_dco\_pk\_oz + ((double)b\_B\_dco\_pk\_oz \* 0.25) + (((double)c\_B\_dco\_pk\_oz + (double)d\_B\_dco\_pk\_oz) \* 0.1);

//---------------------------------------------------------------------------------------------------------

a\_B\_dco\_pk\_oz = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 66

select item.CollectedValue).Sum();

b\_B\_dco\_pk\_oz = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 72

select item.CollectedValue).Sum();

c\_B\_dco\_pk\_oz = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 78

select item.CollectedValue).Sum();

d\_B\_dco\_pk\_oz = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 84

select item.CollectedValue).Sum();

if (a\_B\_dco\_pk\_oz == null) a\_B\_dco\_pk\_oz = 0;

if (b\_B\_dco\_pk\_oz == null) b\_B\_dco\_pk\_oz = 0;

if (c\_B\_dco\_pk\_oz == null) c\_B\_dco\_pk\_oz = 0;

if (d\_B\_dco\_pk\_oz == null) d\_B\_dco\_pk\_oz = 0;

double? B\_pk\_oz = (double)a\_B\_dco\_pk\_oz + ((double)b\_B\_dco\_pk\_oz \* 0.25) + (((double)c\_B\_dco\_pk\_oz + (double)d\_B\_dco\_pk\_oz) \* 0.1);

//---------------------------------------------------------------------------------------------------------

a\_B\_dco\_pk\_oz = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 44

select item.CollectedValue).Sum();

b\_B\_dco\_pk\_oz = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 49

select item.CollectedValue).Sum();

c\_B\_dco\_pk\_oz = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 54

select item.CollectedValue).Sum();

d\_B\_dco\_pk\_oz = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 59

select item.CollectedValue).Sum();

if (a\_B\_dco\_pk\_oz == null) a\_B\_dco\_pk\_oz = 0;

if (b\_B\_dco\_pk\_oz == null) b\_B\_dco\_pk\_oz = 0;

if (c\_B\_dco\_pk\_oz == null) c\_B\_dco\_pk\_oz = 0;

if (d\_B\_dco\_pk\_oz == null) d\_B\_dco\_pk\_oz = 0;

double? S\_pk\_oz = (double)a\_B\_dco\_pk\_oz + ((double)b\_B\_dco\_pk\_oz \* 0.25) + (((double)c\_B\_dco\_pk\_oz + (double)d\_B\_dco\_pk\_oz) \* 0.1);

//---------------------------------------------------------------------------------------------------------

a\_B\_dco\_pk\_oz = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 18

select item.CollectedValue).Sum();

b\_B\_dco\_pk\_oz = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 25

select item.CollectedValue).Sum();

c\_B\_dco\_pk\_oz = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 32

select item.CollectedValue).Sum();

d\_B\_dco\_pk\_oz = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 39

select item.CollectedValue).Sum();

if (a\_B\_dco\_pk\_oz == null) a\_B\_dco\_pk\_oz = 0;

if (b\_B\_dco\_pk\_oz == null) b\_B\_dco\_pk\_oz = 0;

if (c\_B\_dco\_pk\_oz == null) c\_B\_dco\_pk\_oz = 0;

if (d\_B\_dco\_pk\_oz == null) d\_B\_dco\_pk\_oz = 0;

double? M\_pk\_oz = (double)a\_B\_dco\_pk\_oz + ((double)b\_B\_dco\_pk\_oz \* 0.25) + (((double)c\_B\_dco\_pk\_oz + (double)d\_B\_dco\_pk\_oz) \* 0.1);

returnValue = (B\_dco\_pk\_oz + S\_dco\_pk\_oz + M\_dco\_pk\_oz) / (B\_pk\_oz + S\_pk\_oz + M\_pk\_oz) \* 100;

break;

case 5:

double? SCHNPR = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 1

select item.CollectedValue).Sum();

double? CHS\_PPS\_sh = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 124

select item.CollectedValue).Sum();

double? CHS\_PPS\_vnsm = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 123

select item.CollectedValue).Sum();

double? CHS\_NR\_sh = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 122

select item.CollectedValue).Sum();

double? CHS\_NR\_vnsm = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 121

select item.CollectedValue).Sum();

if (SCHNPR == null) SCHNPR = 0;

if (CHS\_PPS\_sh == null) CHS\_PPS\_sh = 0;

if (CHS\_PPS\_vnsm == null) CHS\_PPS\_vnsm = 0;

if (CHS\_NR\_sh == null) CHS\_NR\_sh = 0;

if (CHS\_NR\_vnsm == null) CHS\_NR\_vnsm = 0;

// КП / СЧНПР \* (ЧС\_ППС\_ш+ЧС\_ППС\_внсм+ЧС\_НР\_ш+ЧС\_НР\_внсм)

returnValue = (double)SCHNPR / ((double)CHS\_PPS\_sh + (double)CHS\_PPS\_vnsm + (double)CHS\_NR\_sh + (double)CHS\_NR\_vnsm);

break;

case 6:

double? KP = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 96

select item.CollectedValue).Sum();

SCHNPR = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 1

select item.CollectedValue).Sum();

CHS\_PPS\_sh = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 124

select item.CollectedValue).Sum();

CHS\_PPS\_vnsm = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 123

select item.CollectedValue).Sum();

CHS\_NR\_sh = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 122

select item.CollectedValue).Sum();

CHS\_NR\_vnsm = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 121

select item.CollectedValue).Sum();

if (KP == null) KP = 0;

if (SCHNPR == null) SCHNPR = 0;

if (CHS\_PPS\_sh == null) CHS\_PPS\_sh = 0;

if (CHS\_PPS\_vnsm == null) CHS\_PPS\_vnsm = 0;

if (CHS\_NR\_sh == null) CHS\_NR\_sh = 0;

if (CHS\_NR\_vnsm == null) CHS\_NR\_vnsm = 0;

// КП / СЧНПР \* (ЧС\_ППС\_ш+ЧС\_ППС\_внсм+ЧС\_НР\_ш+ЧС\_НР\_внсм)

returnValue = (double)KP / (double)SCHNPR \* ((double)CHS\_PPS\_sh + (double)CHS\_PPS\_vnsm + (double)CHS\_NR\_sh + (double)CHS\_NR\_vnsm);

break;

case 7:

double? KPS5 = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 98

select item.CollectedValue).Sum();

SCHNPR = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 1

select item.CollectedValue).Sum();

CHS\_PPS\_sh = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 124

select item.CollectedValue).Sum();

CHS\_PPS\_vnsm = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 123

select item.CollectedValue).Sum();

CHS\_NR\_sh = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 122

select item.CollectedValue).Sum();

CHS\_NR\_vnsm = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 121

select item.CollectedValue).Sum();

if (KPS5 == null) KPS5 = 0;

if (SCHNPR == null) SCHNPR = 0;

if (CHS\_PPS\_sh == null) CHS\_PPS\_sh = 0;

if (CHS\_PPS\_vnsm == null) CHS\_PPS\_vnsm = 0;

if (CHS\_NR\_sh == null) CHS\_NR\_sh = 0;

if (CHS\_NR\_vnsm == null) CHS\_NR\_vnsm = 0;

// КПС5 / СЧНПР \* (ЧС\_ППС\_ш+ЧС\_ППС\_внсм+ЧС\_НР\_ш+ЧС\_НР\_внсм)

returnValue = KPS5 / SCHNPR \* (CHS\_PPS\_sh + CHS\_PPS\_vnsm + CHS\_NR\_sh + CHS\_NR\_vnsm);

break;

case 8:

double? NYOKR = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 99

select item.CollectedValue).Sum();

SCHNPR = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 1

select item.CollectedValue).Sum();

CHS\_PPS\_sh = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 124

select item.CollectedValue).Sum();

CHS\_PPS\_vnsm = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 123

select item.CollectedValue).Sum();

CHS\_NR\_sh = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 122

select item.CollectedValue).Sum();

CHS\_NR\_vnsm = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 121

select item.CollectedValue).Sum();

if (NYOKR == null) NYOKR = 0;

if (SCHNPR == null) SCHNPR = 0;

if (CHS\_PPS\_sh == null) CHS\_PPS\_sh = 0;

if (CHS\_PPS\_vnsm == null) CHS\_PPS\_vnsm = 0;

if (CHS\_NR\_sh == null) CHS\_NR\_sh = 0;

if (CHS\_NR\_vnsm == null) CHS\_NR\_vnsm = 0;

// КП / СЧНПР \* (ЧС\_ППС\_ш+ЧС\_ППС\_внсм+ЧС\_НР\_ш+ЧС\_НР\_внсм)

returnValue = (double)NYOKR / (double)SCHNPR \* ((double)CHS\_PPS\_sh + (double)CHS\_PPS\_vnsm + (double)CHS\_NR\_sh + (double)CHS\_NR\_vnsm);

break;

case 9:

double? a\_YSB\_pr = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 62

select item.CollectedValue).Sum();

double? b\_YSB\_pr = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 68

select item.CollectedValue).Sum();

double? c\_YSB\_pr = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 74

select item.CollectedValue).Sum();

double? d\_YSB\_pr = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 80

select item.CollectedValue).Sum();

if (a\_YSB\_pr == null) a\_YSB\_pr = 0;

if (b\_YSB\_pr == null) b\_YSB\_pr = 0;

if (c\_YSB\_pr == null) c\_YSB\_pr = 0;

if (d\_YSB\_pr == null) d\_YSB\_pr = 0;

double? YSB\_pr = (double)a\_YSB\_pr + ((double)b\_YSB\_pr \* 0.25) + (((double)c\_YSB\_pr + (double)d\_YSB\_pr) \* 0.1);

//---------------------------------------------------------------------------------------------------------

a\_YSB\_pr = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 45

select item.CollectedValue).Sum();

b\_YSB\_pr = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 50

select item.CollectedValue).Sum();

c\_YSB\_pr = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 55

select item.CollectedValue).Sum();

d\_YSB\_pr = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 60

select item.CollectedValue).Sum();

if (a\_YSB\_pr == null) a\_YSB\_pr = 0;

if (b\_YSB\_pr == null) b\_YSB\_pr = 0;

if (c\_YSB\_pr == null) c\_YSB\_pr = 0;

if (d\_YSB\_pr == null) d\_YSB\_pr = 0;

double? YYS\_pr = (double)a\_YSB\_pr + ((double)b\_YSB\_pr \* 0.25) + (((double)c\_YSB\_pr + (double)d\_YSB\_pr) \* 0.1);

//---------------------------------------------------------------------------------------------------------

a\_YSB\_pr = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 14

select item.CollectedValue).Sum();

b\_YSB\_pr = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 22

select item.CollectedValue).Sum();

c\_YSB\_pr = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 29

select item.CollectedValue).Sum();

d\_YSB\_pr = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 36

select item.CollectedValue).Sum();

if (a\_YSB\_pr == null) a\_YSB\_pr = 0;

if (b\_YSB\_pr == null) b\_YSB\_pr = 0;

if (c\_YSB\_pr == null) c\_YSB\_pr = 0;

if (d\_YSB\_pr == null) d\_YSB\_pr = 0;

double? YSM\_pr = (double)a\_YSB\_pr + ((double)b\_YSB\_pr \* 0.25) + (((double)c\_YSB\_pr + (double)d\_YSB\_pr) \* 0.1);

//---------------------------------------------------------------------------------------------------------

a\_YSB\_pr = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 67

select item.CollectedValue).Sum();

b\_YSB\_pr = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 73

select item.CollectedValue).Sum();

c\_YSB\_pr = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 79

select item.CollectedValue).Sum();

d\_YSB\_pr = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 85

select item.CollectedValue).Sum();

if (a\_YSB\_pr == null) a\_YSB\_pr = 0;

if (b\_YSB\_pr == null) b\_YSB\_pr = 0;

if (c\_YSB\_pr == null) c\_YSB\_pr = 0;

if (d\_YSB\_pr == null) d\_YSB\_pr = 0;

double? B\_pk\_ys = (double)a\_YSB\_pr + ((double)b\_YSB\_pr \* 0.25) + (((double)c\_YSB\_pr + (double)d\_YSB\_pr) \* 0.1);

//---------------------------------------------------------------------------------------------------------

a\_YSB\_pr = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 46

select item.CollectedValue).Sum();

b\_YSB\_pr = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 51

select item.CollectedValue).Sum();

c\_YSB\_pr = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 56

select item.CollectedValue).Sum();

d\_YSB\_pr = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 61

select item.CollectedValue).Sum();

if (a\_YSB\_pr == null) a\_YSB\_pr = 0;

if (b\_YSB\_pr == null) b\_YSB\_pr = 0;

if (c\_YSB\_pr == null) c\_YSB\_pr = 0;

if (d\_YSB\_pr == null) d\_YSB\_pr = 0;

double? S\_pk\_ys = (double)a\_YSB\_pr + ((double)b\_YSB\_pr \* 0.25) + (((double)c\_YSB\_pr + (double)d\_YSB\_pr) \* 0.1);

//---------------------------------------------------------------------------------------------------------

a\_YSB\_pr = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 19

select item.CollectedValue).Sum();

b\_YSB\_pr = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 26

select item.CollectedValue).Sum();

c\_YSB\_pr = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 33

select item.CollectedValue).Sum();

d\_YSB\_pr = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 40

select item.CollectedValue).Sum();

if (a\_YSB\_pr == null) a\_YSB\_pr = 0;

if (b\_YSB\_pr == null) b\_YSB\_pr = 0;

if (c\_YSB\_pr == null) c\_YSB\_pr = 0;

if (d\_YSB\_pr == null) d\_YSB\_pr = 0;

double? M\_pk\_ys = (double)a\_YSB\_pr + ((double)b\_YSB\_pr \* 0.25) + (((double)c\_YSB\_pr + (double)d\_YSB\_pr) \* 0.1);

//(ИСБ\_пр+ИСС\_пр+ИСМ\_пр) / (Б\_пк\_ис+С\_пк\_ис+М\_пк\_ис) \* 100

returnValue = (YSM\_pr + YYS\_pr + YSB\_pr) / (B\_pk\_ys + S\_pk\_ys + M\_pk\_ys) \* 100;

break;

case 10:

// "=" (Численность зарубежных ведущих профессоров, преподавателей и исследователей, работающих в университете не менее 1 семестра)

returnValue = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 114

select item.CollectedValue).Sum();

break;

case 11:

double? OOS = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 104

select item.CollectedValue).Sum();

double? OS\_VNB = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 105

select item.CollectedValue).Sum();

if (OOS == null) OOS = 0;

if (OS\_VNB == null) OS\_VNB = 0;

// (ОС\_ВНБ / ООС) \* 100

returnValue = ((OS\_VNB / OOS) \* 100);

break;

case 12:

OOS = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 104

select item.CollectedValue).Sum();

SCHNPR = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 1

select item.CollectedValue).Sum();

CHS\_PPS\_sh = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 124

select item.CollectedValue).Sum();

CHS\_PPS\_vnsm = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 123

select item.CollectedValue).Sum();

CHS\_NR\_sh = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 122

select item.CollectedValue).Sum();

CHS\_NR\_vnsm = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 121

select item.CollectedValue).Sum();

if (OOS == null) OOS = 0;

if (SCHNPR == null) SCHNPR = 0;

if (CHS\_PPS\_sh == null) CHS\_PPS\_sh = 0;

if (CHS\_PPS\_vnsm == null) CHS\_PPS\_vnsm = 0;

if (CHS\_NR\_sh == null) CHS\_NR\_sh = 0;

if (CHS\_NR\_vnsm == null) CHS\_NR\_vnsm = 0;

//ООС / СЧНПР \* (ЧС\_ППС\_ш+ЧС\_ППС\_внсм+ЧС\_НР\_ш+ЧС\_НР\_внсм)

returnValue = OOS / SCHNPR \* (CHS\_PPS\_sh + CHS\_PPS\_vnsm + CHS\_NR\_sh + CHS\_NR\_vnsm);

break;

case 13:

OS\_VNB = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 111

select item.CollectedValue).Sum();

SCHNPR = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 1

select item.CollectedValue).Sum();

double? SZP = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 124

select item.CollectedValue).Sum();

if (OS\_VNB == null) OS\_VNB = 0;

if (SCHNPR == null) SCHNPR = 0;

if (SZP == null) SZP = 0;

// (ФзпНПР(OS\_VNB) / СЧНПР) / 12 / СЗП \* 100

returnValue = (OS\_VNB / SCHNPR) / 12 / SZP \* 100;

break;

case 14:

double? b\_OOP = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 100

select item.CollectedValue).Sum();

double? d\_OOP = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 101

select item.CollectedValue).Sum();

double? m\_OOP = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 102

select item.CollectedValue).Sum();

double? s\_OOP = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 103

select item.CollectedValue).Sum();

if (b\_OOP == null) b\_OOP = 0;

if (d\_OOP == null) d\_OOP = 0;

if (m\_OOP == null) m\_OOP = 0;

if (s\_OOP == null) s\_OOP = 0;

double? OOP = b\_OOP + d\_OOP + m\_OOP + s\_OOP;

//---------------------------------------------------------------------------------------------------------

double? b\_CHOOP = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 115

select item.CollectedValue).Sum();

double? d\_CHOOP = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 116

select item.CollectedValue).Sum();

double? m\_CHOOP = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 117

select item.CollectedValue).Sum();

double? s\_CHOOP = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 118

select item.CollectedValue).Sum();

if (b\_CHOOP == null) b\_CHOOP = 0;

if (d\_CHOOP == null) d\_OOP = 0;

if (m\_CHOOP == null) m\_CHOOP = 0;

if (s\_CHOOP == null) s\_CHOOP = 0;

double? CHOOP = b\_CHOOP + d\_CHOOP + m\_CHOOP + s\_CHOOP;

// (ООП\_ОП/ ЧООП)\*100

returnValue = (OOP / CHOOP) \* 100;

break;

case 15:

double? b\_OP = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 89

select item.CollectedValue).Sum();

double? s\_OP = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 91

select item.CollectedValue).Sum();

double? m\_OP = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 90

select item.CollectedValue).Sum();

double? d\_OP = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 87

select item.CollectedValue).Sum();

if (b\_OP == null) b\_OP = 0;

if (s\_OP == null) s\_OP = 0;

if (m\_OP == null) m\_OP = 0;

if (d\_OP == null) d\_OP = 0;

// "сумма"

returnValue = b\_OP + s\_OP + m\_OP + d\_OP;

break;

case 16:

double? OOP\_100 = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 100

select item.CollectedValue).Sum();

double? OOP\_103 = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 103

select item.CollectedValue).Sum();

double? OOP\_102 = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 102

select item.CollectedValue).Sum();

double? OOP\_101 = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 101

select item.CollectedValue).Sum();

if (OOP\_100 == null) OOP\_100 = 0;

if (OOP\_101 == null) OOP\_101 = 0;

if (OOP\_102 == null) OOP\_102 = 0;

if (OOP\_103 == null) OOP\_103 = 0;

double? buf\_OOPSUM = (OOP\_101 + OOP\_102 + OOP\_103 + OOP\_100);

OOP\_100 = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 115

select item.CollectedValue).Sum();

OOP\_103 = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 118

select item.CollectedValue).Sum();

OOP\_102 = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 117

select item.CollectedValue).Sum();

OOP\_101 = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 116

select item.CollectedValue).Sum();

if (OOP\_100 == null) OOP\_100 = 0;

if (OOP\_101 == null) OOP\_101 = 0;

if (OOP\_102 == null) OOP\_102 = 0;

if (OOP\_103 == null) OOP\_103 = 0;

double? buf\_OOPSUM\_2 = (OOP\_101 + OOP\_102 + OOP\_103 + OOP\_100);

//(ООП\_СОТ/ ЧООП)\*100

returnValue = ((buf\_OOPSUM / buf\_OOPSUM\_2) \* 100);

break;

case 17:

double? KP\_65 = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 97

select item.CollectedValue).Sum();

SCHNPR = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 1

select item.CollectedValue).Sum();

CHS\_PPS\_sh = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 124

select item.CollectedValue).Sum();

CHS\_PPS\_vnsm = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 123

select item.CollectedValue).Sum();

CHS\_NR\_sh = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 122

select item.CollectedValue).Sum();

CHS\_NR\_vnsm = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 121

select item.CollectedValue).Sum();

if (KP\_65 == null) KP\_65 = 0;

if (SCHNPR == null) SCHNPR = 0;

if (CHS\_PPS\_sh == null) CHS\_PPS\_sh = 0;

if (CHS\_PPS\_vnsm == null) CHS\_PPS\_vnsm = 0;

if (CHS\_NR\_sh == null) CHS\_NR\_sh = 0;

if (CHS\_NR\_vnsm == null) CHS\_NR\_vnsm = 0;

// КП / СЧНПР \* (ЧС\_ППС\_ш+ЧС\_ППС\_внсм+ЧС\_НР\_ш+ЧС\_НР\_внсм)

returnValue = KP\_65 / SCHNPR \* (CHS\_PPS\_sh + CHS\_PPS\_vnsm + CHS\_NR\_sh + CHS\_NR\_vnsm);

break;

case 18:

double? kol\_MM = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 88

select item.CollectedValue).Sum();

if (kol\_MM == null) kol\_MM = 0;

// "="

returnValue = kol\_MM;

break;

case 19:

SCHNPR = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 1

select item.CollectedValue).Sum();

double? PK\_NPR = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 106

select item.CollectedValue).Sum();

if (PK\_NPR == null) PK\_NPR = 0;

if (SCHNPR == null) SCHNPR = 0;

// (ПК\_НПР / СЧНПР) \*100

returnValue = (PK\_NPR / SCHNPR) \* 100;

break;

case 20:

//Б\_Р\_пк

double? a\_B\_R\_pk = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 64

select item.CollectedValue).Sum();

double? b\_B\_R\_pk = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 70

select item.CollectedValue).Sum();

double? c\_B\_R\_pk = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 76

select item.CollectedValue).Sum();

double? d\_B\_R\_pk = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 82

select item.CollectedValue).Sum();

if (a\_B\_R\_pk == null) a\_B\_R\_pk = 0;

if (b\_B\_R\_pk == null) b\_B\_R\_pk = 0;

if (c\_B\_R\_pk == null) c\_B\_R\_pk = 0;

if (d\_B\_R\_pk == null) d\_B\_R\_pk = 0;

double? B\_R\_pk = (double)a\_B\_R\_pk + ((double)b\_B\_R\_pk \* 0.25) + (((double)c\_B\_R\_pk + (double)d\_B\_R\_pk) \* 0.1);

//М\_Р\_пк

double? a\_M\_R\_pk = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 20

select item.CollectedValue).Sum();

double? b\_M\_R\_pk = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 20

select item.CollectedValue).Sum();

double? c\_M\_R\_pk = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 20

select item.CollectedValue).Sum();

double? d\_M\_R\_pk = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 20

select item.CollectedValue).Sum();

if (a\_M\_R\_pk == null) a\_M\_R\_pk = 0;

if (b\_M\_R\_pk == null) b\_M\_R\_pk = 0;

if (c\_M\_R\_pk == null) c\_M\_R\_pk = 0;

if (d\_M\_R\_pk == null) d\_M\_R\_pk = 0;

double? M\_R\_pk = (double)a\_M\_R\_pk + ((double)b\_M\_R\_pk \* 0.25) + (((double)c\_M\_R\_pk + (double)d\_M\_R\_pk) \* 0.1);

//А\_Р\_пк

double? a\_A\_P\_pk = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 3

select item.CollectedValue).Sum();

double? b\_A\_P\_pk = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 6

select item.CollectedValue).Sum();

double? c\_A\_P\_pk = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 9

select item.CollectedValue).Sum();

double? d\_A\_P\_pk = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 12

select item.CollectedValue).Sum();

if (a\_A\_P\_pk == null) a\_A\_P\_pk = 0;

if (b\_A\_P\_pk == null) b\_A\_P\_pk = 0;

if (c\_A\_P\_pk == null) c\_A\_P\_pk = 0;

if (d\_A\_P\_pk == null) d\_A\_P\_pk = 0;

double? A\_R\_pk = (double)a\_A\_P\_pk + ((double)b\_A\_P\_pk \* 0.25) + (((double)c\_A\_P\_pk + (double)d\_A\_P\_pk) \* 0.1);

//M\_пк

a = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 15

select item.CollectedValue).Sum();

b = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 21

select item.CollectedValue).Sum();

c = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 28

select item.CollectedValue).Sum();

d = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 35

select item.CollectedValue).Sum();

if (a == null) a = 0;

if (b == null) b = 0;

if (c == null) c = 0;

if (d == null) d = 0;

M\_pk = (double)a + ((double)b \* 0.25) + (((double)c + (double)d) \* 0.1);

//Б\_пк

a = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 63

select item.CollectedValue).Sum();

b = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 69

select item.CollectedValue).Sum();

c = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 75

select item.CollectedValue).Sum();

d = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 81

select item.CollectedValue).Sum();

if (a == null) a = 0;

if (b == null) b = 0;

if (c == null) c = 0;

if (d == null) d = 0;

B\_pk = (double)a + ((double)b \* 0.25) + (((double)c + (double)d) \* 0.1);

//А\_пк

a = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 2

select item.CollectedValue).Sum();

b = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 5

select item.CollectedValue).Sum();

c = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 8

select item.CollectedValue).Sum();

d = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 11

select item.CollectedValue).Sum();

if (a == null) a = 0;

if (b == null) b = 0;

if (c == null) c = 0;

if (d == null) d = 0;

A\_pk = (double)a + ((double)b \* 0.25) + (((double)c + (double)d) \* 0.1);

//((Б\_Р\_пк+М\_Р\_пк+А\_Р\_пк) / (Б\_пк+М\_пк+А\_пк)) \* 100

returnValue = (B\_R\_pk + M\_R\_pk + A\_R\_pk) / (B\_pk + M\_pk + A\_pk);

break;

case 21:

// Количество базовых кафедр, открытых на предприятиях и в организациях реального сектора экономики

returnValue = (from item in KPIWebDataContext.CollectedBasicParametersTable

where item.FK\_ReportArchiveTable == ReportArchiveTableID &&

item.FK\_BasicParametersTable == 86

select item.CollectedValue).Sum();

break;

default:

returnValue = -1;

break;

}

return (double)returnValue;

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Filename: ChartItems.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using System;

using System.Collections.Generic;

using System.Data;

using System.Linq;

using System.Web;

using KPIWeb.Rector;

namespace KPIWeb

{

public class ChartItems

{

public class DataItem

{

public string Name { get; set; }

public double Value { get; set; }

}

private List<DataItem> dataItems;

public ChartItems()

{

dataItems = new List<DataItem>();

}

public List<DataItem> GetDataSource()

{

return dataItems;

}

public void AddChartItem(string name, double value)

{

KPIWebDataContext kPiDataContext = new KPIWebDataContext();

IndicatorsTable Indicator = (from a in kPiDataContext.IndicatorsTable

where a.Name == name

select a).FirstOrDefault();

dataItems.Add(new DataItem() { Name = name, Value = Math.Round(value, 1) });

}

public List<ChartOneValue> ReturnTopFive(List<ChartOneValue> collection)

{

collection.Sort(delegate(ChartOneValue value1, ChartOneValue value2)

{ return value1.value.CompareTo(value2.value); });

collection.Reverse();

var sort = collection.Take(5);

List<ChartOneValue> newList = sort.ToList();

newList.Reverse();

return newList;

}

public List<ChartOneValue> Sort(List<ChartOneValue> collection)

{

collection.Sort(delegate(ChartOneValue value1, ChartOneValue value2)

{ return value1.value.CompareTo(value2.value); });

return collection;

}

public List<ChartOneValue> SortReverse(List<ChartOneValue> collection)

{

collection.Sort(delegate(ChartOneValue value1, ChartOneValue value2)

{ return value1.value.CompareTo(value2.value); });

collection.Reverse();

return collection;

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Filename: CommonCode.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

namespace KPIWeb

{

public class CommonCode

{

public static String GetUserById(int UserID)

{

KPIWebDataContext kpiWebDataContext = new KPIWebDataContext();

if (UserID == null)

return "0";

UsersTable User = (from a in kpiWebDataContext.UsersTable

where a.UsersTableID == UserID

select a).FirstOrDefault();

if (User == null)

return "0";

if (User.Position == null)

return User.Email;

if (User.Position.Length > 2)

return User.Position;

return User.Email;

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Filename: Contact.aspx.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

namespace KPIWeb

{

public partial class Contact : Page

{

protected void Page\_Load(object sender, EventArgs e)

{

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Filename: Contact.aspx.designer.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//------------------------------------------------------------------------------

// <auto-generated>

// This code was generated by a tool.

//

// Changes to this file may cause incorrect behavior and will be lost if

// the code is regenerated.

// </auto-generated>

//------------------------------------------------------------------------------

namespace KPIWeb {

public partial class Contact {

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Filename: Default.aspx.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.Security;

using System.Web.UI;

using System.Web.UI.WebControls;

namespace KPIWeb

{

public partial class \_Default : Page

{

UsersTable user;

public void Directions(UsersTable user)

{

if (user.Position == null)

{

FormsAuthentication.SetAuthCookie(user.Email, true);

}

else if (user.Position.Length > 2)

{

FormsAuthentication.SetAuthCookie(user.Position, true);

}

else

{

FormsAuthentication.SetAuthCookie(user.Email, true);

}

int accessLevel = (int)user.AccessLevel;

if (accessLevel == 10)

{

Response.Redirect("~/AutomationDepartment/Main.aspx");

}

else if (accessLevel == 9)

{

Response.Redirect("~/StatisticsDepartment/MonitoringMain.aspx");

}

else if (accessLevel == 5)

{

Response.Redirect("~/Rector/RectorMain.aspx");

}

else if (accessLevel == 3)

{

Response.Redirect("~/FinKadr/OtdelChooseReport.aspx");

}

else if (accessLevel == 4)

{

Response.Redirect("~/Director/DMain.aspx");

}

else if (accessLevel == 7)

{

Response.Redirect("~/Rector/RMain.aspx");

}

else if (accessLevel == 0)

{

Response.Redirect("~/Reports\_/ChooseReport.aspx");

}

else //если входим сюда то что то не так) скорей всего пользователю не присвоен уровень в UsersTable

{

FormsAuthentication.SignOut();

Session.Abandon();

Response.Redirect("~/Account/UserLogin.aspx");

}

}

protected void Page\_Load(object sender, EventArgs e)

{

Serialization UserSer = (Serialization)Session["UserID"];

if (UserSer == null)

{

FormsAuthentication.SignOut();

Session.Abandon();

Response.Redirect("~/Account/UserLogin.aspx");

}

KPIWebDataContext KPIWebDataContext = new KPIWebDataContext();

UsersTable user = (from usersTables in KPIWebDataContext.UsersTable

where usersTables.UsersTableID == UserSer.Id

select usersTables).FirstOrDefault();

if (user != null)

{

List<MultiUser> MultiuserList = (from a in KPIWebDataContext.MultiUser

where a.Active == true

&& a.FK\_UserCanAccess == user.UsersTableID

select a).ToList();

if (MultiuserList.Count()>0)

{

Response.Redirect("~/MultiUser.aspx");

}

Directions(user);

}

else

{

FormsAuthentication.SignOut();

Session.Abandon();

Response.Redirect("~/Account/UserLogin.aspx");

}

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Filename: Default.aspx.designer.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//------------------------------------------------------------------------------

// <auto-generated>

// This code was generated by a tool.

//

// Changes to this file may cause incorrect behavior and will be lost if

// the code is regenerated.

// </auto-generated>

//------------------------------------------------------------------------------

namespace KPIWeb

{

public partial class \_Default

{

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Filename: Encode.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text.RegularExpressions;

using System.Web;

namespace KPIWeb

{

public class Encode

{

public static string getType(string code) // "31.05.02"

{

return code.Remove(0, 2).Remove(4, 2).Replace(".", string.Empty);

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Filename: FillingProcessInfo.aspx.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Data;

using System.Web.UI;

using System.Web.UI.WebControls;

namespace KPIWeb

{

public partial class FillingProcessInfo : System.Web.UI.Page

{

public class MyObject

{

public int Id;

public int ParentId;

public string Name;

//public string UrlAddr;

//public int Active;

}

private void BindTree(IEnumerable<MyObject> list, TreeNode parentNode)

{

var nodes = list.Where(x => parentNode == null ? x.ParentId == 0 : x.ParentId == int.Parse(parentNode.Value));

foreach (var node in nodes)

{

TreeNode newNode = new TreeNode(node.Name, node.Id.ToString());

/\* if (node.Active == 1)

{

//newNode.NavigateUrl = node.UrlAddr;

}

else\*/

newNode.SelectAction = TreeNodeSelectAction.None;

if (parentNode == null)

{

TreeView1.Nodes.Add(newNode);

}

else

{

parentNode.ChildNodes.Add(newNode);

}

BindTree(list, newNode);

}

}

protected void Page\_Load(object sender, EventArgs e)

{

}

protected void Button1\_Click(object sender, EventArgs e)

{

if (TextBox1.Text == "123\_")

{

KPIWebDataContext kPiDataContext = new KPIWebDataContext();

TextBox2.Text = "";

TextBox2.Text += "Всего структурных подразделений вносящих данные: " + (from a in kPiDataContext.ThirdLevelParametrs

where a.Active == true

select a).Count().ToString();

TextBox2.Text += Environment.NewLine;

TextBox2.Text += "Структурных подразделений вносящих данные, где есть хоть один пользователь: " +

(from a in kPiDataContext.ThirdLevelParametrs

join b in kPiDataContext.UsersTable

on a.ThirdLevelParametrsID equals b.FK\_ThirdLevelSubdivisionTable

where a.Active == true

&& b.Active == true

&& b.AccessLevel == 0

select a).Distinct().Count().ToString();

TextBox2.Text += Environment.NewLine;

TextBox2.Text += "Всего пользователей: " + (from b in kPiDataContext.UsersTable

where

b.AccessLevel == 0

&& b.Active == true

select b).Count().ToString();

TextBox2.Text += Environment.NewLine;

TextBox2.Text += "Всего пользователей активировавших аккаунты: " + (from b in kPiDataContext.UsersTable

where

b.AccessLevel == 0

&& b.Active == true

&& b.Confirmed == true

select b).Count().ToString();

TextBox2.Text += Environment.NewLine;

TextBox2.Text += Environment.NewLine;

TextBox2.Text += "Структурных подразделений утвердивших отчет " + (from a in kPiDataContext.ThirdLevelParametrs

join b in kPiDataContext.CollectedBasicParametersTable

on a.ThirdLevelParametrsID equals b.FK\_ThirdLevelSubdivisionTable

where b.Status == 4

select a).Distinct().Count().ToString();

TextBox2.Text += Environment.NewLine;

TextBox2.Text += "Структурных подразделений c отправленным на утверждение отчетом " + (from a in kPiDataContext.ThirdLevelParametrs

join b in kPiDataContext.CollectedBasicParametersTable

on a.ThirdLevelParametrsID equals b.FK\_ThirdLevelSubdivisionTable

where b.Status == 3

select a).Distinct().Count().ToString();

TextBox2.Text += Environment.NewLine;

TextBox2.Text += "Структурных подразделений отчетом возвращенным на доработку " + (from a in kPiDataContext.ThirdLevelParametrs

join b in kPiDataContext.CollectedBasicParametersTable

on a.ThirdLevelParametrsID equals b.FK\_ThirdLevelSubdivisionTable

where b.Status == 1

select a).Distinct().Count().ToString();

TextBox2.Text += Environment.NewLine;

TextBox2.Text += "Структурных подразделений внесших 1 и более показателей (учитываются утвержденные и отправленные на утверждение) " +

(from a in kPiDataContext.ThirdLevelParametrs

join b in kPiDataContext.CollectedBasicParametersTable

on a.ThirdLevelParametrsID equals b.FK\_ThirdLevelSubdivisionTable

where

b.CollectedValue != null

select a).Distinct().Count().ToString();

TextBox2.Text += Environment.NewLine;

TextBox2.Text += Environment.NewLine;

TextBox2.Text += "Кол-во верифицирующих которые должны быть в системе для данной компании: ";

TextBox2.Text += (from a in kPiDataContext.ThirdLevelSubdivisionTable

join b in kPiDataContext.SecondLevelSubdivisionTable

on a.FK\_SecondLevelSubdivisionTable equals b.SecondLevelSubdivisionTableID

join c in kPiDataContext.ReportArchiveAndLevelMappingTable

on b.FK\_FirstLevelSubdivisionTable equals c.FK\_FirstLevelSubmisionTableId

where a.Active == true

&& b.Active == true

&& c.Active == true

&& c.FK\_ReportArchiveTableId == 1

select a).Count().ToString();

TextBox2.Text += Environment.NewLine;

TextBox2.Text += "Кол-во пользователей с правами утверждающего ";

TextBox2.Text += (from a in kPiDataContext.UsersTable

where

a.Active == true

join b in kPiDataContext.BasicParametrsAndUsersMapping

on a.UsersTableID equals b.FK\_UsersTable

where b.CanConfirm == true

&& b.CanEdit == false

&& b.Active == true

select a).Distinct().Count().ToString();

TextBox2.Text += Environment.NewLine;

TextBox2.Text += "Кол-во пользователей с правами заполняющего ";

TextBox2.Text += (from a in kPiDataContext.UsersTable

where

a.Active == true

join b in kPiDataContext.BasicParametrsAndUsersMapping

on a.UsersTableID equals b.FK\_UsersTable

where b.CanConfirm == false

&& b.CanEdit == true

&& b.Active == true

select a).Distinct().Count().ToString();

TextBox2.Text += Environment.NewLine;

TextBox2.Text += "Кол-во пользователей с правами заполняющего и утверждающего ";

TextBox2.Text += (from a in kPiDataContext.UsersTable

where

a.Active == true

join b in kPiDataContext.BasicParametrsAndUsersMapping

on a.UsersTableID equals b.FK\_UsersTable

where b.CanConfirm == true

&& b.CanEdit == true

&& b.Active == true

select a).Distinct().Count().ToString();

TextBox2.Text += Environment.NewLine;

TextBox2.Text += "Кол-во верифицирующих с email адресами внесенными в систему: ";

TextBox2.Text += (from a in kPiDataContext.UsersTable

where

a.Active == true

join b in kPiDataContext.BasicParametrsAndUsersMapping

on a.UsersTableID equals b.FK\_UsersTable

where b.CanConfirm == true

&& b.Active == true

select a).Distinct().Count().ToString();

TextBox2.Text += Environment.NewLine;

TextBox2.Text += "Кол-во верифицирующих не завершивших регистрацию: ";

TextBox2.Text += (from a in kPiDataContext.UsersTable

where

a.Active == true

&& a.Confirmed == false

join b in kPiDataContext.BasicParametrsAndUsersMapping

on a.UsersTableID equals b.FK\_UsersTable

where b.CanConfirm == true

&& b.Active == true

select a).Distinct().Count().ToString();

TextBox2.Text += Environment.NewLine;

TextBox2.Text += "Кол-во верифицирующих завершивших регистрацию: ";

TextBox2.Text += (from a in kPiDataContext.UsersTable

where

a.Active == true

&& a.Confirmed == true

join b in kPiDataContext.BasicParametrsAndUsersMapping

on a.UsersTableID equals b.FK\_UsersTable

where b.CanConfirm == true

&& b.Active == true

select a).Distinct().Count().ToString();

TextBox2.Text += Environment.NewLine;

TextBox2.Text += "Кол-во верифицирующих, которые утвердили данные: ";

TextBox2.Text += (from a in kPiDataContext.ThirdLevelParametrs

join b in kPiDataContext.CollectedBasicParametersTable

on a.ThirdLevelParametrsID equals b.FK\_ThirdLevelSubdivisionTable

where b.Status == 4

select a).Distinct().Count().ToString();

TextBox2.Text += Environment.NewLine;

//////////////////////////////////////////////////////////////////////////////////////////

DataTable dataTable = new DataTable();

dataTable.Columns.Add(new DataColumn("First", typeof(string)));

dataTable.Columns.Add(new DataColumn("Second", typeof(string)));

dataTable.Columns.Add(new DataColumn("Third", typeof(string)));

DataTable dataTable1 = new DataTable();

dataTable1.Columns.Add(new DataColumn("First", typeof(string)));

dataTable1.Columns.Add(new DataColumn("Second", typeof(string)));

dataTable1.Columns.Add(new DataColumn("Third", typeof(string)));

List<ThirdLevelSubdivisionTable> ThirdLevelWithConfirmList = (from a in kPiDataContext.ThirdLevelSubdivisionTable

join b in kPiDataContext.UsersTable

on a.ThirdLevelSubdivisionTableID equals b.FK\_ThirdLevelSubdivisionTable

join c in kPiDataContext.BasicParametrsAndUsersMapping

on b.UsersTableID equals c.FK\_UsersTable

where a.Active == true

&& b.Active == true

&& c.Active == true

&& c.CanConfirm == true

select a).Distinct().ToList();

List<ThirdLevelSubdivisionTable> ThirdLevelAll = (from a in kPiDataContext.ThirdLevelSubdivisionTable

where a.Active == true

select a).ToList();

foreach (ThirdLevelSubdivisionTable cur in ThirdLevelWithConfirmList)

{

ThirdLevelAll.Remove(cur);

}

foreach (ThirdLevelSubdivisionTable CurrentThirdLevel in ThirdLevelAll )

{

DataRow dataRow = dataTable.NewRow();

dataRow["First"] = (from a in kPiDataContext.SecondLevelSubdivisionTable

join b in kPiDataContext.FirstLevelSubdivisionTable

on a.FK\_FirstLevelSubdivisionTable equals b.FirstLevelSubdivisionTableID

where a.SecondLevelSubdivisionTableID == CurrentThirdLevel.FK\_SecondLevelSubdivisionTable

select b.Name).FirstOrDefault();

dataRow["Second"] = (from a in kPiDataContext.SecondLevelSubdivisionTable

where a.SecondLevelSubdivisionTableID == CurrentThirdLevel.FK\_SecondLevelSubdivisionTable

select a.Name).FirstOrDefault();

dataRow["Third"] = CurrentThirdLevel.Name;

dataTable.Rows.Add(dataRow);

}

///////////////////////////////////////////////////////////////////////////////////

//////////////////////////////////////////////////////////////////////////////////////////

List<ThirdLevelSubdivisionTable> ThirdLevelWithEditList = (from a in kPiDataContext.ThirdLevelSubdivisionTable

join b in kPiDataContext.UsersTable

on a.ThirdLevelSubdivisionTableID equals b.FK\_ThirdLevelSubdivisionTable

join c in kPiDataContext.BasicParametrsAndUsersMapping

on b.UsersTableID equals c.FK\_UsersTable

where a.Active == true

&& b.Active == true

&& c.Active == true

&& c.CanEdit == true

select a).Distinct().ToList();

List<ThirdLevelSubdivisionTable> ThirdLevelAllForEdit = (from a in kPiDataContext.ThirdLevelSubdivisionTable

where a.Active == true

select a).ToList();

foreach (ThirdLevelSubdivisionTable cur in ThirdLevelWithEditList)

{

ThirdLevelAllForEdit.Remove(cur);

}

foreach (ThirdLevelSubdivisionTable CurrentThirdLevel in ThirdLevelAllForEdit)

{

DataRow dataRow2 = dataTable1.NewRow();

dataRow2["First"] = (from a in kPiDataContext.SecondLevelSubdivisionTable

join b in kPiDataContext.FirstLevelSubdivisionTable

on a.FK\_FirstLevelSubdivisionTable equals b.FirstLevelSubdivisionTableID

where a.SecondLevelSubdivisionTableID == CurrentThirdLevel.FK\_SecondLevelSubdivisionTable

select b.Name).FirstOrDefault();

dataRow2["Second"] = (from a in kPiDataContext.SecondLevelSubdivisionTable

where a.SecondLevelSubdivisionTableID == CurrentThirdLevel.FK\_SecondLevelSubdivisionTable

select a.Name).FirstOrDefault();

dataRow2["Third"] = CurrentThirdLevel.Name;

dataTable1.Rows.Add(dataRow2);

}

GridView1.DataSource = dataTable;

GridView1.DataBind();

GridView2.DataSource = dataTable1;

GridView2.DataBind();

///////////////////////////////////////////////////////////////////////////////////

TreeView1.DataSource = null;

TreeView1.DataBind();

Label1.Visible = true;

List<MyObject> list = new List<MyObject>();

int allZeroLevel = 0;

int insertZeroLevel = 0;

int confirmZeroLevel = 0;

int i = 1;

int UsrCnt = 0;

List<UsersTable> Users;

string tmp;

string tmp2;

#region get zero leve list

List<ZeroLevelSubdivisionTable> zeroLevelList = (from a in kPiDataContext.ZeroLevelSubdivisionTable

where a.Active == true

select a).ToList();

#endregion

// колво утвердивших (% ) // % ожидающих утверждения // % начавших заполнение

foreach (ZeroLevelSubdivisionTable zeroLevelItem in zeroLevelList)//по каждому университету

{

#region get first level list

List<FirstLevelSubdivisionTable> firstLevelList = (from b in kPiDataContext.FirstLevelSubdivisionTable

where b.FK\_ZeroLevelSubvisionTable == zeroLevelItem.ZeroLevelSubdivisionTableID

&& b.Active == true

select b).ToList();

#endregion

#region

int all0 = (from a in kPiDataContext.ThirdLevelParametrs where a.Active == true select a).Count();

int conf0 = (from a in kPiDataContext.ThirdLevelParametrs

join b in kPiDataContext.CollectedBasicParametersTable

on a.ThirdLevelParametrsID equals b.FK\_ThirdLevelSubdivisionTable

where b.Status == 4

select a).Distinct().Count();

int toconf0 = (from a in kPiDataContext.ThirdLevelParametrs

join b in kPiDataContext.CollectedBasicParametersTable

on a.ThirdLevelParametrsID equals b.FK\_ThirdLevelSubdivisionTable

where b.Status == 3

select a).Distinct().Count();

int started0 = (from a in kPiDataContext.ThirdLevelParametrs

join b in kPiDataContext.CollectedBasicParametersTable

on a.ThirdLevelParametrsID equals b.FK\_ThirdLevelSubdivisionTable

where

b.CollectedValue != null

select a).Distinct().Count();

#endregion

i++;

int par0 = i;

tmp2 = zeroLevelItem.Name;

list.Add(new MyObject() { Id = i, ParentId = 0, Name = tmp2 +" : "+ all0.ToString() +"/"+ conf0.ToString() +"/"+ toconf0.ToString() +"/"+ started0.ToString() });

foreach (FirstLevelSubdivisionTable firstLevelItem in firstLevelList)//по каждой академии

{

#region get second level list

List<SecondLevelSubdivisionTable> secondLevelList =

(from d in kPiDataContext.SecondLevelSubdivisionTable

where d.FK\_FirstLevelSubdivisionTable == firstLevelItem.FirstLevelSubdivisionTableID

&& d.Active == true

select d).ToList();

#endregion

#region

all0 = (from a in kPiDataContext.ThirdLevelSubdivisionTable

join b in kPiDataContext.SecondLevelSubdivisionTable

on a.FK\_SecondLevelSubdivisionTable equals b.SecondLevelSubdivisionTableID

where a.Active == true

&& b.FK\_FirstLevelSubdivisionTable == firstLevelItem.FirstLevelSubdivisionTableID

select a).Count();

conf0 = (from a in kPiDataContext.ThirdLevelParametrs

join b in kPiDataContext.CollectedBasicParametersTable

on a.ThirdLevelParametrsID equals b.FK\_ThirdLevelSubdivisionTable

where b.Status == 4

&& b.FK\_FirstLevelSubdivisionTable == firstLevelItem.FirstLevelSubdivisionTableID

select a).Distinct().Count();

toconf0 = (from a in kPiDataContext.ThirdLevelParametrs

join b in kPiDataContext.CollectedBasicParametersTable

on a.ThirdLevelParametrsID equals b.FK\_ThirdLevelSubdivisionTable

where b.Status == 3

&& b.FK\_FirstLevelSubdivisionTable == firstLevelItem.FirstLevelSubdivisionTableID

select a).Distinct().Count();

started0 = (from a in kPiDataContext.ThirdLevelParametrs

join b in kPiDataContext.CollectedBasicParametersTable

on a.ThirdLevelParametrsID equals b.FK\_ThirdLevelSubdivisionTable

where

b.FK\_FirstLevelSubdivisionTable == firstLevelItem.FirstLevelSubdivisionTableID

&& b.CollectedValue != null

select a).Distinct().Count();

#endregion

i++;

int par1 = i;

tmp2 = firstLevelItem.Name;

list.Add(new MyObject() { Id = i, ParentId = par0, Name = tmp2 + " : " + all0.ToString() + "/" + conf0.ToString() + "/" + toconf0.ToString() + "/" + started0.ToString() });

foreach (SecondLevelSubdivisionTable secondLevelItem in secondLevelList)//по каждому факультету

{

#region get third level list

List<ThirdLevelSubdivisionTable> thirdLevelList =

(from f in kPiDataContext.ThirdLevelSubdivisionTable

where f.FK\_SecondLevelSubdivisionTable == secondLevelItem.SecondLevelSubdivisionTableID

&& f.Active == true

select f).ToList();

#endregion

#region

all0 = (from a in kPiDataContext.ThirdLevelSubdivisionTable

where a.Active == true

&& a.FK\_SecondLevelSubdivisionTable == secondLevelItem.SecondLevelSubdivisionTableID

select a).Count();

conf0 = (from a in kPiDataContext.ThirdLevelParametrs

join b in kPiDataContext.CollectedBasicParametersTable

on a.ThirdLevelParametrsID equals b.FK\_ThirdLevelSubdivisionTable

where b.Status == 4

&& b.FK\_SecondLevelSubdivisionTable == secondLevelItem.SecondLevelSubdivisionTableID

select a).Distinct().Count();

toconf0 = (from a in kPiDataContext.ThirdLevelParametrs

join b in kPiDataContext.CollectedBasicParametersTable

on a.ThirdLevelParametrsID equals b.FK\_ThirdLevelSubdivisionTable

where b.Status == 3

&& b.FK\_SecondLevelSubdivisionTable == secondLevelItem.SecondLevelSubdivisionTableID

select a).Distinct().Count();

started0 = (from a in kPiDataContext.ThirdLevelParametrs

join b in kPiDataContext.CollectedBasicParametersTable

on a.ThirdLevelParametrsID equals b.FK\_ThirdLevelSubdivisionTable

where

b.FK\_SecondLevelSubdivisionTable == secondLevelItem.SecondLevelSubdivisionTableID

&& b.CollectedValue != null

select a).Distinct().Count();

#endregion

i++;

int par2 = i;

tmp2 = secondLevelItem.Name;

list.Add(new MyObject() { Id = i, ParentId = par1, Name = tmp2 + " : " + all0.ToString() + "/" + conf0.ToString() + "/" + toconf0.ToString() + "/" + started0.ToString() });

foreach (ThirdLevelSubdivisionTable thirdLevelItem in thirdLevelList)//по кафедре

{

#region

all0 = (from a in kPiDataContext.ThirdLevelSubdivisionTable

where a.Active == true

&& a.ThirdLevelSubdivisionTableID == thirdLevelItem.ThirdLevelSubdivisionTableID

select a).Count();

conf0 = (from a in kPiDataContext.ThirdLevelParametrs

join b in kPiDataContext.CollectedBasicParametersTable

on a.ThirdLevelParametrsID equals b.FK\_ThirdLevelSubdivisionTable

where b.Status == 4

&& b.FK\_ThirdLevelSubdivisionTable == thirdLevelItem.ThirdLevelSubdivisionTableID

select a).Distinct().Count();

toconf0 = (from a in kPiDataContext.ThirdLevelParametrs

join b in kPiDataContext.CollectedBasicParametersTable

on a.ThirdLevelParametrsID equals b.FK\_ThirdLevelSubdivisionTable

where b.Status == 3

&& b.FK\_ThirdLevelSubdivisionTable == thirdLevelItem.ThirdLevelSubdivisionTableID

select a).Distinct().Count();

started0 = (from a in kPiDataContext.ThirdLevelParametrs

join b in kPiDataContext.CollectedBasicParametersTable

on a.ThirdLevelParametrsID equals b.FK\_ThirdLevelSubdivisionTable

where

b.FK\_ThirdLevelSubdivisionTable == thirdLevelItem.ThirdLevelSubdivisionTableID

&& b.CollectedValue != null

select a).Distinct().Count();

#endregion

i++;

int par3 = i;

tmp2 = thirdLevelItem.Name;

list.Add(new MyObject() { Id = i, ParentId = par2, Name = tmp2 + " : " + all0.ToString() + "/" + conf0.ToString() + "/" + toconf0.ToString() + "/" + started0.ToString() });

}

}

}

}

BindTree(list, null);

TreeView1.CollapseAll();

}

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Filename: FillingProcessInfo.aspx.designer.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//------------------------------------------------------------------------------

// <auto-generated>

// This code was generated by a tool.

//

// Changes to this file may cause incorrect behavior and will be lost if

// the code is regenerated.

// </auto-generated>

//------------------------------------------------------------------------------

namespace KPIWeb {

public partial class FillingProcessInfo {

/// <summary>

/// TextBox1 control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.TextBox TextBox1;

/// <summary>

/// Button1 control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.Button Button1;

/// <summary>

/// TextBox2 control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.TextBox TextBox2;

/// <summary>

/// GridView1 control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.GridView GridView1;

/// <summary>

/// GridView2 control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.GridView GridView2;

/// <summary>

/// Label1 control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.Label Label1;

/// <summary>

/// TreeView1 control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.TreeView TreeView1;

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Filename: Global.asax.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.Optimization;

using System.Web.Routing;

using System.Web.Security;

using System.Web.SessionState;

namespace KPIWeb

{

public class Global : HttpApplication

{

void Application\_Start(object sender, EventArgs e)

{

// Code that runs on application startup

RouteConfig.RegisterRoutes(RouteTable.Routes);

BundleConfig.RegisterBundles(BundleTable.Bundles);

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Filename: KPIWeb.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

namespace KPIWeb

{

partial class KPIWebDataContext

{

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Filename: KPIWeb.designer.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#pragma warning disable 1591

//------------------------------------------------------------------------------

// <auto-generated>

// This code was generated by a tool.

// Runtime Version:4.0.30319.18444

//

// Changes to this file may cause incorrect behavior and will be lost if

// the code is regenerated.

// </auto-generated>

//------------------------------------------------------------------------------

namespace KPIWeb

{

using System.Data.Linq;

using System.Data.Linq.Mapping;

using System.Data;

using System.Collections.Generic;

using System.Reflection;

using System.Linq;

using System.Linq.Expressions;

using System.ComponentModel;

using System;

[global::System.Data.Linq.Mapping.DatabaseAttribute(Name="Release\_")]

public partial class KPIWebDataContext : System.Data.Linq.DataContext

{

private static System.Data.Linq.Mapping.MappingSource mappingSource = new AttributeMappingSource();

#region Extensibility Method Definitions

partial void OnCreated();

partial void InsertBasicParametersAndRolesMappingTable(BasicParametersAndRolesMappingTable instance);

partial void UpdateBasicParametersAndRolesMappingTable(BasicParametersAndRolesMappingTable instance);

partial void DeleteBasicParametersAndRolesMappingTable(BasicParametersAndRolesMappingTable instance);

partial void InsertZeroLevelSubdivisionTable(ZeroLevelSubdivisionTable instance);

partial void UpdateZeroLevelSubdivisionTable(ZeroLevelSubdivisionTable instance);

partial void DeleteZeroLevelSubdivisionTable(ZeroLevelSubdivisionTable instance);

partial void InsertBasicParametersTable(BasicParametersTable instance);

partial void UpdateBasicParametersTable(BasicParametersTable instance);

partial void DeleteBasicParametersTable(BasicParametersTable instance);

partial void InsertBasicParametrAdditional(BasicParametrAdditional instance);

partial void UpdateBasicParametrAdditional(BasicParametrAdditional instance);

partial void DeleteBasicParametrAdditional(BasicParametrAdditional instance);

partial void InsertBasicParametrsAndUsersMapping(BasicParametrsAndUsersMapping instance);

partial void UpdateBasicParametrsAndUsersMapping(BasicParametrsAndUsersMapping instance);

partial void DeleteBasicParametrsAndUsersMapping(BasicParametrsAndUsersMapping instance);

partial void InsertCalculatedParametrs(CalculatedParametrs instance);

partial void UpdateCalculatedParametrs(CalculatedParametrs instance);

partial void DeleteCalculatedParametrs(CalculatedParametrs instance);

partial void InsertCalculatedParametrsAndRolesMappingTable(CalculatedParametrsAndRolesMappingTable instance);

partial void UpdateCalculatedParametrsAndRolesMappingTable(CalculatedParametrsAndRolesMappingTable instance);

partial void DeleteCalculatedParametrsAndRolesMappingTable(CalculatedParametrsAndRolesMappingTable instance);

partial void InsertCalculatedParametrsAndUsersMapping(CalculatedParametrsAndUsersMapping instance);

partial void UpdateCalculatedParametrsAndUsersMapping(CalculatedParametrsAndUsersMapping instance);

partial void DeleteCalculatedParametrsAndUsersMapping(CalculatedParametrsAndUsersMapping instance);

partial void InsertCollectedBasicParametersTable(CollectedBasicParametersTable instance);

partial void UpdateCollectedBasicParametersTable(CollectedBasicParametersTable instance);

partial void DeleteCollectedBasicParametersTable(CollectedBasicParametersTable instance);

partial void InsertCollectedCalculatedParametrs(CollectedCalculatedParametrs instance);

partial void UpdateCollectedCalculatedParametrs(CollectedCalculatedParametrs instance);

partial void DeleteCollectedCalculatedParametrs(CollectedCalculatedParametrs instance);

partial void InsertCollectedIndicatorsForR(CollectedIndicatorsForR instance);

partial void UpdateCollectedIndicatorsForR(CollectedIndicatorsForR instance);

partial void DeleteCollectedIndicatorsForR(CollectedIndicatorsForR instance);

partial void InsertCollectedIndocators(CollectedIndocators instance);

partial void UpdateCollectedIndocators(CollectedIndocators instance);

partial void DeleteCollectedIndocators(CollectedIndocators instance);

partial void InsertCommetntForBasicInReport(CommetntForBasicInReport instance);

partial void UpdateCommetntForBasicInReport(CommetntForBasicInReport instance);

partial void DeleteCommetntForBasicInReport(CommetntForBasicInReport instance);

partial void InsertConfirmationHistory(ConfirmationHistory instance);

partial void UpdateConfirmationHistory(ConfirmationHistory instance);

partial void DeleteConfirmationHistory(ConfirmationHistory instance);

partial void InsertDocumentTable(DocumentTable instance);

partial void UpdateDocumentTable(DocumentTable instance);

partial void DeleteDocumentTable(DocumentTable instance);

partial void InsertEmailSendHistory(EmailSendHistory instance);

partial void UpdateEmailSendHistory(EmailSendHistory instance);

partial void DeleteEmailSendHistory(EmailSendHistory instance);

partial void InsertEmailSendTable(EmailSendTable instance);

partial void UpdateEmailSendTable(EmailSendTable instance);

partial void DeleteEmailSendTable(EmailSendTable instance);

partial void InsertEmailTemplate(EmailTemplate instance);

partial void UpdateEmailTemplate(EmailTemplate instance);

partial void DeleteEmailTemplate(EmailTemplate instance);

partial void InsertFieldOfExpertise(FieldOfExpertise instance);

partial void UpdateFieldOfExpertise(FieldOfExpertise instance);

partial void DeleteFieldOfExpertise(FieldOfExpertise instance);

partial void InsertFifthLevelSubdivisionTable(FifthLevelSubdivisionTable instance);

partial void UpdateFifthLevelSubdivisionTable(FifthLevelSubdivisionTable instance);

partial void DeleteFifthLevelSubdivisionTable(FifthLevelSubdivisionTable instance);

partial void InsertFirstLevelSubdivisionTable(FirstLevelSubdivisionTable instance);

partial void UpdateFirstLevelSubdivisionTable(FirstLevelSubdivisionTable instance);

partial void DeleteFirstLevelSubdivisionTable(FirstLevelSubdivisionTable instance);

partial void InsertFourthLevelParametrs(FourthLevelParametrs instance);

partial void UpdateFourthLevelParametrs(FourthLevelParametrs instance);

partial void DeleteFourthLevelParametrs(FourthLevelParametrs instance);

partial void InsertFourthLevelSubdivisionTable(FourthLevelSubdivisionTable instance);

partial void UpdateFourthLevelSubdivisionTable(FourthLevelSubdivisionTable instance);

partial void DeleteFourthLevelSubdivisionTable(FourthLevelSubdivisionTable instance);

partial void InsertIndicatorClass(IndicatorClass instance);

partial void UpdateIndicatorClass(IndicatorClass instance);

partial void DeleteIndicatorClass(IndicatorClass instance);

partial void InsertIndicatorsAndRolesMappingTable(IndicatorsAndRolesMappingTable instance);

partial void UpdateIndicatorsAndRolesMappingTable(IndicatorsAndRolesMappingTable instance);

partial void DeleteIndicatorsAndRolesMappingTable(IndicatorsAndRolesMappingTable instance);

partial void InsertIndicatorsAndUsersMapping(IndicatorsAndUsersMapping instance);

partial void UpdateIndicatorsAndUsersMapping(IndicatorsAndUsersMapping instance);

partial void DeleteIndicatorsAndUsersMapping(IndicatorsAndUsersMapping instance);

partial void InsertManualTable(ManualTable instance);

partial void UpdateManualTable(ManualTable instance);

partial void DeleteManualTable(ManualTable instance);

partial void InsertMultiUser(MultiUser instance);

partial void UpdateMultiUser(MultiUser instance);

partial void DeleteMultiUser(MultiUser instance);

partial void InsertPlannedIndicator(PlannedIndicator instance);

partial void UpdatePlannedIndicator(PlannedIndicator instance);

partial void DeletePlannedIndicator(PlannedIndicator instance);

partial void InsertReportArchiveAndBasicParametrsMappingTable(ReportArchiveAndBasicParametrsMappingTable instance);

partial void UpdateReportArchiveAndBasicParametrsMappingTable(ReportArchiveAndBasicParametrsMappingTable instance);

partial void DeleteReportArchiveAndBasicParametrsMappingTable(ReportArchiveAndBasicParametrsMappingTable instance);

partial void InsertReportArchiveAndCalculatedParametrsMappingTable(ReportArchiveAndCalculatedParametrsMappingTable instance);

partial void UpdateReportArchiveAndCalculatedParametrsMappingTable(ReportArchiveAndCalculatedParametrsMappingTable instance);

partial void DeleteReportArchiveAndCalculatedParametrsMappingTable(ReportArchiveAndCalculatedParametrsMappingTable instance);

partial void InsertReportArchiveAndIndicatorsMappingTable(ReportArchiveAndIndicatorsMappingTable instance);

partial void UpdateReportArchiveAndIndicatorsMappingTable(ReportArchiveAndIndicatorsMappingTable instance);

partial void DeleteReportArchiveAndIndicatorsMappingTable(ReportArchiveAndIndicatorsMappingTable instance);

partial void InsertReportArchiveAndLevelMappingTable(ReportArchiveAndLevelMappingTable instance);

partial void UpdateReportArchiveAndLevelMappingTable(ReportArchiveAndLevelMappingTable instance);

partial void DeleteReportArchiveAndLevelMappingTable(ReportArchiveAndLevelMappingTable instance);

partial void InsertReportArchiveTable(ReportArchiveTable instance);

partial void UpdateReportArchiveTable(ReportArchiveTable instance);

partial void DeleteReportArchiveTable(ReportArchiveTable instance);

partial void InsertRolesTable(RolesTable instance);

partial void UpdateRolesTable(RolesTable instance);

partial void DeleteRolesTable(RolesTable instance);

partial void InsertSecondLevelSubdivisionTable(SecondLevelSubdivisionTable instance);

partial void UpdateSecondLevelSubdivisionTable(SecondLevelSubdivisionTable instance);

partial void DeleteSecondLevelSubdivisionTable(SecondLevelSubdivisionTable instance);

partial void InsertSpecializationTable(SpecializationTable instance);

partial void UpdateSpecializationTable(SpecializationTable instance);

partial void DeleteSpecializationTable(SpecializationTable instance);

partial void InsertThirdLevelParametrs(ThirdLevelParametrs instance);

partial void UpdateThirdLevelParametrs(ThirdLevelParametrs instance);

partial void DeleteThirdLevelParametrs(ThirdLevelParametrs instance);

partial void InsertThirdLevelSubdivisionTable(ThirdLevelSubdivisionTable instance);

partial void UpdateThirdLevelSubdivisionTable(ThirdLevelSubdivisionTable instance);

partial void DeleteThirdLevelSubdivisionTable(ThirdLevelSubdivisionTable instance);

partial void InsertUsersTable(UsersTable instance);

partial void UpdateUsersTable(UsersTable instance);

partial void DeleteUsersTable(UsersTable instance);

partial void InsertEducationCostTable(EducationCostTable instance);

partial void UpdateEducationCostTable(EducationCostTable instance);

partial void DeleteEducationCostTable(EducationCostTable instance);

partial void InsertIndicatorsTable(IndicatorsTable instance);

partial void UpdateIndicatorsTable(IndicatorsTable instance);

partial void DeleteIndicatorsTable(IndicatorsTable instance);

#endregion

public KPIWebDataContext() :

base(global::System.Configuration.ConfigurationManager.ConnectionStrings["Release\_ConnectionString1"].ConnectionString, mappingSource)

{

OnCreated();

}

public KPIWebDataContext(string connection) :

base(connection, mappingSource)

{

OnCreated();

}

public KPIWebDataContext(System.Data.IDbConnection connection) :

base(connection, mappingSource)

{

OnCreated();

}

public KPIWebDataContext(string connection, System.Data.Linq.Mapping.MappingSource mappingSource) :

base(connection, mappingSource)

{

OnCreated();

}

public KPIWebDataContext(System.Data.IDbConnection connection, System.Data.Linq.Mapping.MappingSource mappingSource) :

base(connection, mappingSource)

{

OnCreated();

}

public System.Data.Linq.Table<BasicParametersAndRolesMappingTable> BasicParametersAndRolesMappingTable

{

get

{

return this.GetTable<BasicParametersAndRolesMappingTable>();

}

}

public System.Data.Linq.Table<ZeroLevelSubdivisionTable> ZeroLevelSubdivisionTable

{

get

{

return this.GetTable<ZeroLevelSubdivisionTable>();

}

}

public System.Data.Linq.Table<BasicParametersTable> BasicParametersTable

{

get

{

return this.GetTable<BasicParametersTable>();

}

}

public System.Data.Linq.Table<BasicParametrAdditional> BasicParametrAdditional

{

get

{

return this.GetTable<BasicParametrAdditional>();

}

}

public System.Data.Linq.Table<BasicParametrsAndUsersMapping> BasicParametrsAndUsersMapping

{

get

{

return this.GetTable<BasicParametrsAndUsersMapping>();

}

}

public System.Data.Linq.Table<CalculatedParametrs> CalculatedParametrs

{

get

{

return this.GetTable<CalculatedParametrs>();

}

}

public System.Data.Linq.Table<CalculatedParametrsAndRolesMappingTable> CalculatedParametrsAndRolesMappingTable

{

get

{

return this.GetTable<CalculatedParametrsAndRolesMappingTable>();

}

}

public System.Data.Linq.Table<CalculatedParametrsAndUsersMapping> CalculatedParametrsAndUsersMapping

{

get

{

return this.GetTable<CalculatedParametrsAndUsersMapping>();

}

}

public System.Data.Linq.Table<CollectedBasicParametersTable> CollectedBasicParametersTable

{

get

{

return this.GetTable<CollectedBasicParametersTable>();

}

}

public System.Data.Linq.Table<CollectedCalculatedParametrs> CollectedCalculatedParametrs

{

get

{

return this.GetTable<CollectedCalculatedParametrs>();

}

}

public System.Data.Linq.Table<CollectedIndicatorsForR> CollectedIndicatorsForR

{

get

{

return this.GetTable<CollectedIndicatorsForR>();

}

}

public System.Data.Linq.Table<CollectedIndocators> CollectedIndocators

{

get

{

return this.GetTable<CollectedIndocators>();

}

}

public System.Data.Linq.Table<CommetntForBasicInReport> CommetntForBasicInReport

{

get

{

return this.GetTable<CommetntForBasicInReport>();

}

}

public System.Data.Linq.Table<ConfirmationHistory> ConfirmationHistory

{

get

{

return this.GetTable<ConfirmationHistory>();

}

}

public System.Data.Linq.Table<DocumentTable> DocumentTable

{

get

{

return this.GetTable<DocumentTable>();

}

}

public System.Data.Linq.Table<EmailSendHistory> EmailSendHistory

{

get

{

return this.GetTable<EmailSendHistory>();

}

}

public System.Data.Linq.Table<EmailSendTable> EmailSendTable

{

get

{

return this.GetTable<EmailSendTable>();

}

}

public System.Data.Linq.Table<EmailTemplate> EmailTemplate

{

get

{

return this.GetTable<EmailTemplate>();

}

}

public System.Data.Linq.Table<FieldOfExpertise> FieldOfExpertise

{

get

{

return this.GetTable<FieldOfExpertise>();

}

}

public System.Data.Linq.Table<FifthLevelSubdivisionTable> FifthLevelSubdivisionTable

{

get

{

return this.GetTable<FifthLevelSubdivisionTable>();

}

}

public System.Data.Linq.Table<FirstLevelSubdivisionTable> FirstLevelSubdivisionTable

{

get

{

return this.GetTable<FirstLevelSubdivisionTable>();

}

}

public System.Data.Linq.Table<FourthLevelParametrs> FourthLevelParametrs

{

get

{

return this.GetTable<FourthLevelParametrs>();

}

}

public System.Data.Linq.Table<FourthLevelSubdivisionTable> FourthLevelSubdivisionTable

{

get

{

return this.GetTable<FourthLevelSubdivisionTable>();

}

}

public System.Data.Linq.Table<IndicatorClass> IndicatorClass

{

get

{

return this.GetTable<IndicatorClass>();

}

}

public System.Data.Linq.Table<IndicatorsAndRolesMappingTable> IndicatorsAndRolesMappingTable

{

get

{

return this.GetTable<IndicatorsAndRolesMappingTable>();

}

}

public System.Data.Linq.Table<IndicatorsAndUsersMapping> IndicatorsAndUsersMapping

{

get

{

return this.GetTable<IndicatorsAndUsersMapping>();

}

}

public System.Data.Linq.Table<ManualTable> ManualTable

{

get

{

return this.GetTable<ManualTable>();

}

}

public System.Data.Linq.Table<MultiUser> MultiUser

{

get

{

return this.GetTable<MultiUser>();

}

}

public System.Data.Linq.Table<PlannedIndicator> PlannedIndicator

{

get

{

return this.GetTable<PlannedIndicator>();

}

}

public System.Data.Linq.Table<ReportArchiveAndBasicParametrsMappingTable> ReportArchiveAndBasicParametrsMappingTable

{

get

{

return this.GetTable<ReportArchiveAndBasicParametrsMappingTable>();

}

}

public System.Data.Linq.Table<ReportArchiveAndCalculatedParametrsMappingTable> ReportArchiveAndCalculatedParametrsMappingTable

{

get

{

return this.GetTable<ReportArchiveAndCalculatedParametrsMappingTable>();

}

}

public System.Data.Linq.Table<ReportArchiveAndIndicatorsMappingTable> ReportArchiveAndIndicatorsMappingTable

{

get

{

return this.GetTable<ReportArchiveAndIndicatorsMappingTable>();

}

}

public System.Data.Linq.Table<ReportArchiveAndLevelMappingTable> ReportArchiveAndLevelMappingTable

{

get

{

return this.GetTable<ReportArchiveAndLevelMappingTable>();

}

}

public System.Data.Linq.Table<ReportArchiveTable> ReportArchiveTable

{

get

{

return this.GetTable<ReportArchiveTable>();

}

}

public System.Data.Linq.Table<RolesTable> RolesTable

{

get

{

return this.GetTable<RolesTable>();

}

}

public System.Data.Linq.Table<SecondLevelSubdivisionTable> SecondLevelSubdivisionTable

{

get

{

return this.GetTable<SecondLevelSubdivisionTable>();

}

}

public System.Data.Linq.Table<SpecializationTable> SpecializationTable

{

get

{

return this.GetTable<SpecializationTable>();

}

}

public System.Data.Linq.Table<ThirdLevelParametrs> ThirdLevelParametrs

{

get

{

return this.GetTable<ThirdLevelParametrs>();

}

}

public System.Data.Linq.Table<ThirdLevelSubdivisionTable> ThirdLevelSubdivisionTable

{

get

{

return this.GetTable<ThirdLevelSubdivisionTable>();

}

}

public System.Data.Linq.Table<UsersTable> UsersTable

{

get

{

return this.GetTable<UsersTable>();

}

}

public System.Data.Linq.Table<EducationCostTable> EducationCostTable

{

get

{

return this.GetTable<EducationCostTable>();

}

}

public System.Data.Linq.Table<IndicatorsTable> IndicatorsTable

{

get

{

return this.GetTable<IndicatorsTable>();

}

}

}

[global::System.Data.Linq.Mapping.TableAttribute(Name="dbo.BasicParametersAndRolesMappingTable")]

public partial class BasicParametersAndRolesMappingTable : INotifyPropertyChanging, INotifyPropertyChanged

{

private static PropertyChangingEventArgs emptyChangingEventArgs = new PropertyChangingEventArgs(String.Empty);

private int \_BasicParametersAndRolesMappingTableID;

private bool \_Active;

private int \_FK\_RolesTable;

private int \_FK\_BasicParametersTable;

private System.Nullable<bool> \_CanEdit;

private System.Nullable<bool> \_CanView;

private System.Nullable<bool> \_CanConfirm;

private EntityRef<BasicParametersTable> \_BasicParametersTable;

private EntityRef<RolesTable> \_RolesTable;

#region Extensibility Method Definitions

partial void OnLoaded();

partial void OnValidate(System.Data.Linq.ChangeAction action);

partial void OnCreated();

partial void OnBasicParametersAndRolesMappingTableIDChanging(int value);

partial void OnBasicParametersAndRolesMappingTableIDChanged();

partial void OnActiveChanging(bool value);

partial void OnActiveChanged();

partial void OnFK\_RolesTableChanging(int value);

partial void OnFK\_RolesTableChanged();

partial void OnFK\_BasicParametersTableChanging(int value);

partial void OnFK\_BasicParametersTableChanged();

partial void OnCanEditChanging(System.Nullable<bool> value);

partial void OnCanEditChanged();

partial void OnCanViewChanging(System.Nullable<bool> value);

partial void OnCanViewChanged();

partial void OnCanConfirmChanging(System.Nullable<bool> value);

partial void OnCanConfirmChanged();

#endregion

public BasicParametersAndRolesMappingTable()

{

this.\_BasicParametersTable = default(EntityRef<BasicParametersTable>);

this.\_RolesTable = default(EntityRef<RolesTable>);

OnCreated();

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_BasicParametersAndRolesMappingTableID", AutoSync=AutoSync.OnInsert, DbType="Int NOT NULL IDENTITY", IsPrimaryKey=true, IsDbGenerated=true)]

public int BasicParametersAndRolesMappingTableID

{

get

{

return this.\_BasicParametersAndRolesMappingTableID;

}

set

{

if ((this.\_BasicParametersAndRolesMappingTableID != value))

{

this.OnBasicParametersAndRolesMappingTableIDChanging(value);

this.SendPropertyChanging();

this.\_BasicParametersAndRolesMappingTableID = value;

this.SendPropertyChanged("BasicParametersAndRolesMappingTableID");

this.OnBasicParametersAndRolesMappingTableIDChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_Active", DbType="Bit NOT NULL")]

public bool Active

{

get

{

return this.\_Active;

}

set

{

if ((this.\_Active != value))

{

this.OnActiveChanging(value);

this.SendPropertyChanging();

this.\_Active = value;

this.SendPropertyChanged("Active");

this.OnActiveChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_FK\_RolesTable", DbType="Int NOT NULL")]

public int FK\_RolesTable

{

get

{

return this.\_FK\_RolesTable;

}

set

{

if ((this.\_FK\_RolesTable != value))

{

if (this.\_RolesTable.HasLoadedOrAssignedValue)

{

throw new System.Data.Linq.ForeignKeyReferenceAlreadyHasValueException();

}

this.OnFK\_RolesTableChanging(value);

this.SendPropertyChanging();

this.\_FK\_RolesTable = value;

this.SendPropertyChanged("FK\_RolesTable");

this.OnFK\_RolesTableChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_FK\_BasicParametersTable", DbType="Int NOT NULL")]

public int FK\_BasicParametersTable

{

get

{

return this.\_FK\_BasicParametersTable;

}

set

{

if ((this.\_FK\_BasicParametersTable != value))

{

if (this.\_BasicParametersTable.HasLoadedOrAssignedValue)

{

throw new System.Data.Linq.ForeignKeyReferenceAlreadyHasValueException();

}

this.OnFK\_BasicParametersTableChanging(value);

this.SendPropertyChanging();

this.\_FK\_BasicParametersTable = value;

this.SendPropertyChanged("FK\_BasicParametersTable");

this.OnFK\_BasicParametersTableChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_CanEdit", DbType="Bit")]

public System.Nullable<bool> CanEdit

{

get

{

return this.\_CanEdit;

}

set

{

if ((this.\_CanEdit != value))

{

this.OnCanEditChanging(value);

this.SendPropertyChanging();

this.\_CanEdit = value;

this.SendPropertyChanged("CanEdit");

this.OnCanEditChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_CanView", DbType="Bit")]

public System.Nullable<bool> CanView

{

get

{

return this.\_CanView;

}

set

{

if ((this.\_CanView != value))

{

this.OnCanViewChanging(value);

this.SendPropertyChanging();

this.\_CanView = value;

this.SendPropertyChanged("CanView");

this.OnCanViewChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_CanConfirm", DbType="Bit")]

public System.Nullable<bool> CanConfirm

{

get

{

return this.\_CanConfirm;

}

set

{

if ((this.\_CanConfirm != value))

{

this.OnCanConfirmChanging(value);

this.SendPropertyChanging();

this.\_CanConfirm = value;

this.SendPropertyChanged("CanConfirm");

this.OnCanConfirmChanged();

}

}

}

[global::System.Data.Linq.Mapping.AssociationAttribute(Name="BasicParametersTable\_BasicParametersAndRolesMappingTable", Storage="\_BasicParametersTable", ThisKey="FK\_BasicParametersTable", OtherKey="BasicParametersTableID", IsForeignKey=true)]

public BasicParametersTable BasicParametersTable

{

get

{

return this.\_BasicParametersTable.Entity;

}

set

{

BasicParametersTable previousValue = this.\_BasicParametersTable.Entity;

if (((previousValue != value)

|| (this.\_BasicParametersTable.HasLoadedOrAssignedValue == false)))

{

this.SendPropertyChanging();

if ((previousValue != null))

{

this.\_BasicParametersTable.Entity = null;

previousValue.BasicParametersAndRolesMappingTable.Remove(this);

}

this.\_BasicParametersTable.Entity = value;

if ((value != null))

{

value.BasicParametersAndRolesMappingTable.Add(this);

this.\_FK\_BasicParametersTable = value.BasicParametersTableID;

}

else

{

this.\_FK\_BasicParametersTable = default(int);

}

this.SendPropertyChanged("BasicParametersTable");

}

}

}

[global::System.Data.Linq.Mapping.AssociationAttribute(Name="RolesTable\_BasicParametersAndRolesMappingTable", Storage="\_RolesTable", ThisKey="FK\_RolesTable", OtherKey="RolesTableID", IsForeignKey=true)]

public RolesTable RolesTable

{

get

{

return this.\_RolesTable.Entity;

}

set

{

RolesTable previousValue = this.\_RolesTable.Entity;

if (((previousValue != value)

|| (this.\_RolesTable.HasLoadedOrAssignedValue == false)))

{

this.SendPropertyChanging();

if ((previousValue != null))

{

this.\_RolesTable.Entity = null;

previousValue.BasicParametersAndRolesMappingTable.Remove(this);

}

this.\_RolesTable.Entity = value;

if ((value != null))

{

value.BasicParametersAndRolesMappingTable.Add(this);

this.\_FK\_RolesTable = value.RolesTableID;

}

else

{

this.\_FK\_RolesTable = default(int);

}

this.SendPropertyChanged("RolesTable");

}

}

}

public event PropertyChangingEventHandler PropertyChanging;

public event PropertyChangedEventHandler PropertyChanged;

protected virtual void SendPropertyChanging()

{

if ((this.PropertyChanging != null))

{

this.PropertyChanging(this, emptyChangingEventArgs);

}

}

protected virtual void SendPropertyChanged(String propertyName)

{

if ((this.PropertyChanged != null))

{

this.PropertyChanged(this, new PropertyChangedEventArgs(propertyName));

}

}

}

[global::System.Data.Linq.Mapping.TableAttribute(Name="dbo.ZeroLevelSubdivisionTable")]

public partial class ZeroLevelSubdivisionTable : INotifyPropertyChanging, INotifyPropertyChanged

{

private static PropertyChangingEventArgs emptyChangingEventArgs = new PropertyChangingEventArgs(String.Empty);

private int \_ZeroLevelSubdivisionTableID;

private bool \_Active;

private string \_Name;

private EntitySet<CollectedBasicParametersTable> \_CollectedBasicParametersTable;

private EntitySet<FirstLevelSubdivisionTable> \_FirstLevelSubdivisionTable;

private EntitySet<UsersTable> \_UsersTable;

#region Extensibility Method Definitions

partial void OnLoaded();

partial void OnValidate(System.Data.Linq.ChangeAction action);

partial void OnCreated();

partial void OnZeroLevelSubdivisionTableIDChanging(int value);

partial void OnZeroLevelSubdivisionTableIDChanged();

partial void OnActiveChanging(bool value);

partial void OnActiveChanged();

partial void OnNameChanging(string value);

partial void OnNameChanged();

#endregion

public ZeroLevelSubdivisionTable()

{

this.\_CollectedBasicParametersTable = new EntitySet<CollectedBasicParametersTable>(new Action<CollectedBasicParametersTable>(this.attach\_CollectedBasicParametersTable), new Action<CollectedBasicParametersTable>(this.detach\_CollectedBasicParametersTable));

this.\_FirstLevelSubdivisionTable = new EntitySet<FirstLevelSubdivisionTable>(new Action<FirstLevelSubdivisionTable>(this.attach\_FirstLevelSubdivisionTable), new Action<FirstLevelSubdivisionTable>(this.detach\_FirstLevelSubdivisionTable));

this.\_UsersTable = new EntitySet<UsersTable>(new Action<UsersTable>(this.attach\_UsersTable), new Action<UsersTable>(this.detach\_UsersTable));

OnCreated();

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_ZeroLevelSubdivisionTableID", AutoSync=AutoSync.OnInsert, DbType="Int NOT NULL IDENTITY", IsPrimaryKey=true, IsDbGenerated=true)]

public int ZeroLevelSubdivisionTableID

{

get

{

return this.\_ZeroLevelSubdivisionTableID;

}

set

{

if ((this.\_ZeroLevelSubdivisionTableID != value))

{

this.OnZeroLevelSubdivisionTableIDChanging(value);

this.SendPropertyChanging();

this.\_ZeroLevelSubdivisionTableID = value;

this.SendPropertyChanged("ZeroLevelSubdivisionTableID");

this.OnZeroLevelSubdivisionTableIDChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_Active", DbType="Bit NOT NULL")]

public bool Active

{

get

{

return this.\_Active;

}

set

{

if ((this.\_Active != value))

{

this.OnActiveChanging(value);

this.SendPropertyChanging();

this.\_Active = value;

this.SendPropertyChanged("Active");

this.OnActiveChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_Name", DbType="VarChar(500) NOT NULL", CanBeNull=false)]

public string Name

{

get

{

return this.\_Name;

}

set

{

if ((this.\_Name != value))

{

this.OnNameChanging(value);

this.SendPropertyChanging();

this.\_Name = value;

this.SendPropertyChanged("Name");

this.OnNameChanged();

}

}

}

[global::System.Data.Linq.Mapping.AssociationAttribute(Name="ZeroLevelSubdivisionTable\_CollectedBasicParametersTable", Storage="\_CollectedBasicParametersTable", ThisKey="ZeroLevelSubdivisionTableID", OtherKey="FK\_ZeroLevelSubdivisionTable")]

public EntitySet<CollectedBasicParametersTable> CollectedBasicParametersTable

{

get

{

return this.\_CollectedBasicParametersTable;

}

set

{

this.\_CollectedBasicParametersTable.Assign(value);

}

}

[global::System.Data.Linq.Mapping.AssociationAttribute(Name="ZeroLevelSubdivisionTable\_FirstLevelSubdivisionTable", Storage="\_FirstLevelSubdivisionTable", ThisKey="ZeroLevelSubdivisionTableID", OtherKey="FK\_ZeroLevelSubvisionTable")]

public EntitySet<FirstLevelSubdivisionTable> FirstLevelSubdivisionTable

{

get

{

return this.\_FirstLevelSubdivisionTable;

}

set

{

this.\_FirstLevelSubdivisionTable.Assign(value);

}

}

[global::System.Data.Linq.Mapping.AssociationAttribute(Name="ZeroLevelSubdivisionTable\_UsersTable", Storage="\_UsersTable", ThisKey="ZeroLevelSubdivisionTableID", OtherKey="FK\_ZeroLevelSubdivisionTable")]

public EntitySet<UsersTable> UsersTable

{

get

{

return this.\_UsersTable;

}

set

{

this.\_UsersTable.Assign(value);

}

}

public event PropertyChangingEventHandler PropertyChanging;

public event PropertyChangedEventHandler PropertyChanged;

protected virtual void SendPropertyChanging()

{

if ((this.PropertyChanging != null))

{

this.PropertyChanging(this, emptyChangingEventArgs);

}

}

protected virtual void SendPropertyChanged(String propertyName)

{

if ((this.PropertyChanged != null))

{

this.PropertyChanged(this, new PropertyChangedEventArgs(propertyName));

}

}

private void attach\_CollectedBasicParametersTable(CollectedBasicParametersTable entity)

{

this.SendPropertyChanging();

entity.ZeroLevelSubdivisionTable = this;

}

private void detach\_CollectedBasicParametersTable(CollectedBasicParametersTable entity)

{

this.SendPropertyChanging();

entity.ZeroLevelSubdivisionTable = null;

}

private void attach\_FirstLevelSubdivisionTable(FirstLevelSubdivisionTable entity)

{

this.SendPropertyChanging();

entity.ZeroLevelSubdivisionTable = this;

}

private void detach\_FirstLevelSubdivisionTable(FirstLevelSubdivisionTable entity)

{

this.SendPropertyChanging();

entity.ZeroLevelSubdivisionTable = null;

}

private void attach\_UsersTable(UsersTable entity)

{

this.SendPropertyChanging();

entity.ZeroLevelSubdivisionTable = this;

}

private void detach\_UsersTable(UsersTable entity)

{

this.SendPropertyChanging();

entity.ZeroLevelSubdivisionTable = null;

}

}

[global::System.Data.Linq.Mapping.TableAttribute(Name="dbo.BasicParametersTable")]

public partial class BasicParametersTable : INotifyPropertyChanging, INotifyPropertyChanged

{

private static PropertyChangingEventArgs emptyChangingEventArgs = new PropertyChangingEventArgs(String.Empty);

private int \_BasicParametersTableID;

private bool \_Active;

private string \_Name;

private string \_AbbreviationEN;

private string \_AbbreviationRU;

private string \_Measure;

private EntitySet<BasicParametersAndRolesMappingTable> \_BasicParametersAndRolesMappingTable;

private EntityRef<BasicParametrAdditional> \_BasicParametrAdditional;

private EntitySet<BasicParametrsAndUsersMapping> \_BasicParametrsAndUsersMapping;

private EntitySet<CollectedBasicParametersTable> \_CollectedBasicParametersTable;

private EntitySet<ConfirmationHistory> \_ConfirmationHistory;

private EntitySet<ReportArchiveAndBasicParametrsMappingTable> \_ReportArchiveAndBasicParametrsMappingTable;

#region Extensibility Method Definitions

partial void OnLoaded();

partial void OnValidate(System.Data.Linq.ChangeAction action);

partial void OnCreated();

partial void OnBasicParametersTableIDChanging(int value);

partial void OnBasicParametersTableIDChanged();

partial void OnActiveChanging(bool value);

partial void OnActiveChanged();

partial void OnNameChanging(string value);

partial void OnNameChanged();

partial void OnAbbreviationENChanging(string value);

partial void OnAbbreviationENChanged();

partial void OnAbbreviationRUChanging(string value);

partial void OnAbbreviationRUChanged();

partial void OnMeasureChanging(string value);

partial void OnMeasureChanged();

#endregion

public BasicParametersTable()

{

this.\_BasicParametersAndRolesMappingTable = new EntitySet<BasicParametersAndRolesMappingTable>(new Action<BasicParametersAndRolesMappingTable>(this.attach\_BasicParametersAndRolesMappingTable), new Action<BasicParametersAndRolesMappingTable>(this.detach\_BasicParametersAndRolesMappingTable));

this.\_BasicParametrAdditional = default(EntityRef<BasicParametrAdditional>);

this.\_BasicParametrsAndUsersMapping = new EntitySet<BasicParametrsAndUsersMapping>(new Action<BasicParametrsAndUsersMapping>(this.attach\_BasicParametrsAndUsersMapping), new Action<BasicParametrsAndUsersMapping>(this.detach\_BasicParametrsAndUsersMapping));

this.\_CollectedBasicParametersTable = new EntitySet<CollectedBasicParametersTable>(new Action<CollectedBasicParametersTable>(this.attach\_CollectedBasicParametersTable), new Action<CollectedBasicParametersTable>(this.detach\_CollectedBasicParametersTable));

this.\_ConfirmationHistory = new EntitySet<ConfirmationHistory>(new Action<ConfirmationHistory>(this.attach\_ConfirmationHistory), new Action<ConfirmationHistory>(this.detach\_ConfirmationHistory));

this.\_ReportArchiveAndBasicParametrsMappingTable = new EntitySet<ReportArchiveAndBasicParametrsMappingTable>(new Action<ReportArchiveAndBasicParametrsMappingTable>(this.attach\_ReportArchiveAndBasicParametrsMappingTable), new Action<ReportArchiveAndBasicParametrsMappingTable>(this.detach\_ReportArchiveAndBasicParametrsMappingTable));

OnCreated();

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_BasicParametersTableID", AutoSync=AutoSync.OnInsert, DbType="Int NOT NULL IDENTITY", IsPrimaryKey=true, IsDbGenerated=true)]

public int BasicParametersTableID

{

get

{

return this.\_BasicParametersTableID;

}

set

{

if ((this.\_BasicParametersTableID != value))

{

this.OnBasicParametersTableIDChanging(value);

this.SendPropertyChanging();

this.\_BasicParametersTableID = value;

this.SendPropertyChanged("BasicParametersTableID");

this.OnBasicParametersTableIDChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_Active", DbType="Bit NOT NULL")]

public bool Active

{

get

{

return this.\_Active;

}

set

{

if ((this.\_Active != value))

{

this.OnActiveChanging(value);

this.SendPropertyChanging();

this.\_Active = value;

this.SendPropertyChanged("Active");

this.OnActiveChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_Name", DbType="VarChar(1000) NOT NULL", CanBeNull=false)]

public string Name

{

get

{

return this.\_Name;

}

set

{

if ((this.\_Name != value))

{

this.OnNameChanging(value);

this.SendPropertyChanging();

this.\_Name = value;

this.SendPropertyChanged("Name");

this.OnNameChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_AbbreviationEN", DbType="VarChar(20) NOT NULL", CanBeNull=false)]

public string AbbreviationEN

{

get

{

return this.\_AbbreviationEN;

}

set

{

if ((this.\_AbbreviationEN != value))

{

this.OnAbbreviationENChanging(value);

this.SendPropertyChanging();

this.\_AbbreviationEN = value;

this.SendPropertyChanged("AbbreviationEN");

this.OnAbbreviationENChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_AbbreviationRU", DbType="VarChar(20) NOT NULL", CanBeNull=false)]

public string AbbreviationRU

{

get

{

return this.\_AbbreviationRU;

}

set

{

if ((this.\_AbbreviationRU != value))

{

this.OnAbbreviationRUChanging(value);

this.SendPropertyChanging();

this.\_AbbreviationRU = value;

this.SendPropertyChanged("AbbreviationRU");

this.OnAbbreviationRUChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_Measure", DbType="VarChar(10) NOT NULL", CanBeNull=false)]

public string Measure

{

get

{

return this.\_Measure;

}

set

{

if ((this.\_Measure != value))

{

this.OnMeasureChanging(value);

this.SendPropertyChanging();

this.\_Measure = value;

this.SendPropertyChanged("Measure");

this.OnMeasureChanged();

}

}

}

[global::System.Data.Linq.Mapping.AssociationAttribute(Name="BasicParametersTable\_BasicParametersAndRolesMappingTable", Storage="\_BasicParametersAndRolesMappingTable", ThisKey="BasicParametersTableID", OtherKey="FK\_BasicParametersTable")]

public EntitySet<BasicParametersAndRolesMappingTable> BasicParametersAndRolesMappingTable

{

get

{

return this.\_BasicParametersAndRolesMappingTable;

}

set

{

this.\_BasicParametersAndRolesMappingTable.Assign(value);

}

}

[global::System.Data.Linq.Mapping.AssociationAttribute(Name="BasicParametersTable\_BasicParametrAdditional", Storage="\_BasicParametrAdditional", ThisKey="BasicParametersTableID", OtherKey="BasicParametrAdditionalID", IsUnique=true, IsForeignKey=false)]

public BasicParametrAdditional BasicParametrAdditional

{

get

{

return this.\_BasicParametrAdditional.Entity;

}

set

{

BasicParametrAdditional previousValue = this.\_BasicParametrAdditional.Entity;

if (((previousValue != value)

|| (this.\_BasicParametrAdditional.HasLoadedOrAssignedValue == false)))

{

this.SendPropertyChanging();

if ((previousValue != null))

{

this.\_BasicParametrAdditional.Entity = null;

previousValue.BasicParametersTable = null;

}

this.\_BasicParametrAdditional.Entity = value;

if ((value != null))

{

value.BasicParametersTable = this;

}

this.SendPropertyChanged("BasicParametrAdditional");

}

}

}

[global::System.Data.Linq.Mapping.AssociationAttribute(Name="BasicParametersTable\_BasicParametrsAndUsersMapping", Storage="\_BasicParametrsAndUsersMapping", ThisKey="BasicParametersTableID", OtherKey="FK\_ParametrsTable")]

public EntitySet<BasicParametrsAndUsersMapping> BasicParametrsAndUsersMapping

{

get

{

return this.\_BasicParametrsAndUsersMapping;

}

set

{

this.\_BasicParametrsAndUsersMapping.Assign(value);

}

}

[global::System.Data.Linq.Mapping.AssociationAttribute(Name="BasicParametersTable\_CollectedBasicParametersTable", Storage="\_CollectedBasicParametersTable", ThisKey="BasicParametersTableID", OtherKey="FK\_BasicParametersTable")]

public EntitySet<CollectedBasicParametersTable> CollectedBasicParametersTable

{

get

{

return this.\_CollectedBasicParametersTable;

}

set

{

this.\_CollectedBasicParametersTable.Assign(value);

}

}

[global::System.Data.Linq.Mapping.AssociationAttribute(Name="BasicParametersTable\_ConfirmationHistory", Storage="\_ConfirmationHistory", ThisKey="BasicParametersTableID", OtherKey="FK\_BasicParamTable")]

public EntitySet<ConfirmationHistory> ConfirmationHistory

{

get

{

return this.\_ConfirmationHistory;

}

set

{

this.\_ConfirmationHistory.Assign(value);

}

}

[global::System.Data.Linq.Mapping.AssociationAttribute(Name="BasicParametersTable\_ReportArchiveAndBasicParametrsMappingTable", Storage="\_ReportArchiveAndBasicParametrsMappingTable", ThisKey="BasicParametersTableID", OtherKey="FK\_BasicParametrsTable")]

public EntitySet<ReportArchiveAndBasicParametrsMappingTable> ReportArchiveAndBasicParametrsMappingTable

{

get

{

return this.\_ReportArchiveAndBasicParametrsMappingTable;

}

set

{

this.\_ReportArchiveAndBasicParametrsMappingTable.Assign(value);

}

}

public event PropertyChangingEventHandler PropertyChanging;

public event PropertyChangedEventHandler PropertyChanged;

protected virtual void SendPropertyChanging()

{

if ((this.PropertyChanging != null))

{

this.PropertyChanging(this, emptyChangingEventArgs);

}

}

protected virtual void SendPropertyChanged(String propertyName)

{

if ((this.PropertyChanged != null))

{

this.PropertyChanged(this, new PropertyChangedEventArgs(propertyName));

}

}

private void attach\_BasicParametersAndRolesMappingTable(BasicParametersAndRolesMappingTable entity)

{

this.SendPropertyChanging();

entity.BasicParametersTable = this;

}

private void detach\_BasicParametersAndRolesMappingTable(BasicParametersAndRolesMappingTable entity)

{

this.SendPropertyChanging();

entity.BasicParametersTable = null;

}

private void attach\_BasicParametrsAndUsersMapping(BasicParametrsAndUsersMapping entity)

{

this.SendPropertyChanging();

entity.BasicParametersTable = this;

}

private void detach\_BasicParametrsAndUsersMapping(BasicParametrsAndUsersMapping entity)

{

this.SendPropertyChanging();

entity.BasicParametersTable = null;

}

private void attach\_CollectedBasicParametersTable(CollectedBasicParametersTable entity)

{

this.SendPropertyChanging();

entity.BasicParametersTable = this;

}

private void detach\_CollectedBasicParametersTable(CollectedBasicParametersTable entity)

{

this.SendPropertyChanging();

entity.BasicParametersTable = null;

}

private void attach\_ConfirmationHistory(ConfirmationHistory entity)

{

this.SendPropertyChanging();

entity.BasicParametersTable = this;

}

private void detach\_ConfirmationHistory(ConfirmationHistory entity)

{

this.SendPropertyChanging();

entity.BasicParametersTable = null;

}

private void attach\_ReportArchiveAndBasicParametrsMappingTable(ReportArchiveAndBasicParametrsMappingTable entity)

{

this.SendPropertyChanging();

entity.BasicParametersTable = this;

}

private void detach\_ReportArchiveAndBasicParametrsMappingTable(ReportArchiveAndBasicParametrsMappingTable entity)

{

this.SendPropertyChanging();

entity.BasicParametersTable = null;

}

}

[global::System.Data.Linq.Mapping.TableAttribute(Name="dbo.BasicParametrAdditional")]

public partial class BasicParametrAdditional : INotifyPropertyChanging, INotifyPropertyChanged

{

private static PropertyChangingEventArgs emptyChangingEventArgs = new PropertyChangingEventArgs(String.Empty);

private int \_BasicParametrAdditionalID;

private System.Nullable<bool> \_Active;

private System.Nullable<bool> \_ForForeignStudents;

private System.Nullable<int> \_FK\_FieldOfExpertise;

private System.Nullable<int> \_SubvisionLevel;

private System.Nullable<bool> \_IsGraduating;

private System.Nullable<bool> \_Calculated;

private System.Nullable<int> \_SpecType;

private System.Nullable<int> \_DataType;

private string \_Comment;

private EntityRef<BasicParametersTable> \_BasicParametersTable;

private EntityRef<FieldOfExpertise> \_FieldOfExpertise;

#region Extensibility Method Definitions

partial void OnLoaded();

partial void OnValidate(System.Data.Linq.ChangeAction action);

partial void OnCreated();

partial void OnBasicParametrAdditionalIDChanging(int value);

partial void OnBasicParametrAdditionalIDChanged();

partial void OnActiveChanging(System.Nullable<bool> value);

partial void OnActiveChanged();

partial void OnForForeignStudentsChanging(System.Nullable<bool> value);

partial void OnForForeignStudentsChanged();

partial void OnFK\_FieldOfExpertiseChanging(System.Nullable<int> value);

partial void OnFK\_FieldOfExpertiseChanged();

partial void OnSubvisionLevelChanging(System.Nullable<int> value);

partial void OnSubvisionLevelChanged();

partial void OnIsGraduatingChanging(System.Nullable<bool> value);

partial void OnIsGraduatingChanged();

partial void OnCalculatedChanging(System.Nullable<bool> value);

partial void OnCalculatedChanged();

partial void OnSpecTypeChanging(System.Nullable<int> value);

partial void OnSpecTypeChanged();

partial void OnDataTypeChanging(System.Nullable<int> value);

partial void OnDataTypeChanged();

partial void OnCommentChanging(string value);

partial void OnCommentChanged();

#endregion

public BasicParametrAdditional()

{

this.\_BasicParametersTable = default(EntityRef<BasicParametersTable>);

this.\_FieldOfExpertise = default(EntityRef<FieldOfExpertise>);

OnCreated();

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_BasicParametrAdditionalID", DbType="Int NOT NULL", IsPrimaryKey=true)]

public int BasicParametrAdditionalID

{

get

{

return this.\_BasicParametrAdditionalID;

}

set

{

if ((this.\_BasicParametrAdditionalID != value))

{

if (this.\_BasicParametersTable.HasLoadedOrAssignedValue)

{

throw new System.Data.Linq.ForeignKeyReferenceAlreadyHasValueException();

}

this.OnBasicParametrAdditionalIDChanging(value);

this.SendPropertyChanging();

this.\_BasicParametrAdditionalID = value;

this.SendPropertyChanged("BasicParametrAdditionalID");

this.OnBasicParametrAdditionalIDChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_Active", DbType="Bit")]

public System.Nullable<bool> Active

{

get

{

return this.\_Active;

}

set

{

if ((this.\_Active != value))

{

this.OnActiveChanging(value);

this.SendPropertyChanging();

this.\_Active = value;

this.SendPropertyChanged("Active");

this.OnActiveChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_ForForeignStudents", DbType="Bit")]

public System.Nullable<bool> ForForeignStudents

{

get

{

return this.\_ForForeignStudents;

}

set

{

if ((this.\_ForForeignStudents != value))

{

this.OnForForeignStudentsChanging(value);

this.SendPropertyChanging();

this.\_ForForeignStudents = value;

this.SendPropertyChanged("ForForeignStudents");

this.OnForForeignStudentsChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_FK\_FieldOfExpertise", DbType="Int")]

public System.Nullable<int> FK\_FieldOfExpertise

{

get

{

return this.\_FK\_FieldOfExpertise;

}

set

{

if ((this.\_FK\_FieldOfExpertise != value))

{

if (this.\_FieldOfExpertise.HasLoadedOrAssignedValue)

{

throw new System.Data.Linq.ForeignKeyReferenceAlreadyHasValueException();

}

this.OnFK\_FieldOfExpertiseChanging(value);

this.SendPropertyChanging();

this.\_FK\_FieldOfExpertise = value;

this.SendPropertyChanged("FK\_FieldOfExpertise");

this.OnFK\_FieldOfExpertiseChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_SubvisionLevel", DbType="Int")]

public System.Nullable<int> SubvisionLevel

{

get

{

return this.\_SubvisionLevel;

}

set

{

if ((this.\_SubvisionLevel != value))

{

this.OnSubvisionLevelChanging(value);

this.SendPropertyChanging();

this.\_SubvisionLevel = value;

this.SendPropertyChanged("SubvisionLevel");

this.OnSubvisionLevelChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_IsGraduating", DbType="Bit")]

public System.Nullable<bool> IsGraduating

{

get

{

return this.\_IsGraduating;

}

set

{

if ((this.\_IsGraduating != value))

{

this.OnIsGraduatingChanging(value);

this.SendPropertyChanging();

this.\_IsGraduating = value;

this.SendPropertyChanged("IsGraduating");

this.OnIsGraduatingChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_Calculated", DbType="Bit")]

public System.Nullable<bool> Calculated

{

get

{

return this.\_Calculated;

}

set

{

if ((this.\_Calculated != value))

{

this.OnCalculatedChanging(value);

this.SendPropertyChanging();

this.\_Calculated = value;

this.SendPropertyChanged("Calculated");

this.OnCalculatedChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_SpecType", DbType="Int")]

public System.Nullable<int> SpecType

{

get

{

return this.\_SpecType;

}

set

{

if ((this.\_SpecType != value))

{

this.OnSpecTypeChanging(value);

this.SendPropertyChanging();

this.\_SpecType = value;

this.SendPropertyChanged("SpecType");

this.OnSpecTypeChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_DataType", DbType="Int")]

public System.Nullable<int> DataType

{

get

{

return this.\_DataType;

}

set

{

if ((this.\_DataType != value))

{

this.OnDataTypeChanging(value);

this.SendPropertyChanging();

this.\_DataType = value;

this.SendPropertyChanged("DataType");

this.OnDataTypeChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_Comment", DbType="Text", UpdateCheck=UpdateCheck.Never)]

public string Comment

{

get

{

return this.\_Comment;

}

set

{

if ((this.\_Comment != value))

{

this.OnCommentChanging(value);

this.SendPropertyChanging();

this.\_Comment = value;

this.SendPropertyChanged("Comment");

this.OnCommentChanged();

}

}

}

[global::System.Data.Linq.Mapping.AssociationAttribute(Name="BasicParametersTable\_BasicParametrAdditional", Storage="\_BasicParametersTable", ThisKey="BasicParametrAdditionalID", OtherKey="BasicParametersTableID", IsForeignKey=true)]

public BasicParametersTable BasicParametersTable

{

get

{

return this.\_BasicParametersTable.Entity;

}

set

{

BasicParametersTable previousValue = this.\_BasicParametersTable.Entity;

if (((previousValue != value)

|| (this.\_BasicParametersTable.HasLoadedOrAssignedValue == false)))

{

this.SendPropertyChanging();

if ((previousValue != null))

{

this.\_BasicParametersTable.Entity = null;

previousValue.BasicParametrAdditional = null;

}

this.\_BasicParametersTable.Entity = value;

if ((value != null))

{

value.BasicParametrAdditional = this;

this.\_BasicParametrAdditionalID = value.BasicParametersTableID;

}

else

{

this.\_BasicParametrAdditionalID = default(int);

}

this.SendPropertyChanged("BasicParametersTable");

}

}

}

[global::System.Data.Linq.Mapping.AssociationAttribute(Name="FieldOfExpertise\_BasicParametrAdditional", Storage="\_FieldOfExpertise", ThisKey="FK\_FieldOfExpertise", OtherKey="FieldOfExpertiseID", IsForeignKey=true)]

public FieldOfExpertise FieldOfExpertise

{

get

{

return this.\_FieldOfExpertise.Entity;

}

set

{

FieldOfExpertise previousValue = this.\_FieldOfExpertise.Entity;

if (((previousValue != value)

|| (this.\_FieldOfExpertise.HasLoadedOrAssignedValue == false)))

{

this.SendPropertyChanging();

if ((previousValue != null))

{

this.\_FieldOfExpertise.Entity = null;

previousValue.BasicParametrAdditional.Remove(this);

}

this.\_FieldOfExpertise.Entity = value;

if ((value != null))

{

value.BasicParametrAdditional.Add(this);

this.\_FK\_FieldOfExpertise = value.FieldOfExpertiseID;

}

else

{

this.\_FK\_FieldOfExpertise = default(Nullable<int>);

}

this.SendPropertyChanged("FieldOfExpertise");

}

}

}

public event PropertyChangingEventHandler PropertyChanging;

public event PropertyChangedEventHandler PropertyChanged;

protected virtual void SendPropertyChanging()

{

if ((this.PropertyChanging != null))

{

this.PropertyChanging(this, emptyChangingEventArgs);

}

}

protected virtual void SendPropertyChanged(String propertyName)

{

if ((this.PropertyChanged != null))

{

this.PropertyChanged(this, new PropertyChangedEventArgs(propertyName));

}

}

}

[global::System.Data.Linq.Mapping.TableAttribute(Name="dbo.BasicParametrsAndUsersMapping")]

public partial class BasicParametrsAndUsersMapping : INotifyPropertyChanging, INotifyPropertyChanged

{

private static PropertyChangingEventArgs emptyChangingEventArgs = new PropertyChangingEventArgs(String.Empty);

private int \_BasicParametrsAndUsersMappingID;

private bool \_Active;

private int \_FK\_ParametrsTable;

private int \_FK\_UsersTable;

private bool \_CanEdit;

private bool \_CanView;

private bool \_CanConfirm;

private EntityRef<BasicParametersTable> \_BasicParametersTable;

private EntityRef<UsersTable> \_UsersTable;

#region Extensibility Method Definitions

partial void OnLoaded();

partial void OnValidate(System.Data.Linq.ChangeAction action);

partial void OnCreated();

partial void OnBasicParametrsAndUsersMappingIDChanging(int value);

partial void OnBasicParametrsAndUsersMappingIDChanged();

partial void OnActiveChanging(bool value);

partial void OnActiveChanged();

partial void OnFK\_ParametrsTableChanging(int value);

partial void OnFK\_ParametrsTableChanged();

partial void OnFK\_UsersTableChanging(int value);

partial void OnFK\_UsersTableChanged();

partial void OnCanEditChanging(bool value);

partial void OnCanEditChanged();

partial void OnCanViewChanging(bool value);

partial void OnCanViewChanged();

partial void OnCanConfirmChanging(bool value);

partial void OnCanConfirmChanged();

#endregion

public BasicParametrsAndUsersMapping()

{

this.\_BasicParametersTable = default(EntityRef<BasicParametersTable>);

this.\_UsersTable = default(EntityRef<UsersTable>);

OnCreated();

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_BasicParametrsAndUsersMappingID", AutoSync=AutoSync.OnInsert, DbType="Int NOT NULL IDENTITY", IsPrimaryKey=true, IsDbGenerated=true)]

public int BasicParametrsAndUsersMappingID

{

get

{

return this.\_BasicParametrsAndUsersMappingID;

}

set

{

if ((this.\_BasicParametrsAndUsersMappingID != value))

{

this.OnBasicParametrsAndUsersMappingIDChanging(value);

this.SendPropertyChanging();

this.\_BasicParametrsAndUsersMappingID = value;

this.SendPropertyChanged("BasicParametrsAndUsersMappingID");

this.OnBasicParametrsAndUsersMappingIDChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_Active", DbType="Bit NOT NULL")]

public bool Active

{

get

{

return this.\_Active;

}

set

{

if ((this.\_Active != value))

{

this.OnActiveChanging(value);

this.SendPropertyChanging();

this.\_Active = value;

this.SendPropertyChanged("Active");

this.OnActiveChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_FK\_ParametrsTable", DbType="Int NOT NULL")]

public int FK\_ParametrsTable

{

get

{

return this.\_FK\_ParametrsTable;

}

set

{

if ((this.\_FK\_ParametrsTable != value))

{

if (this.\_BasicParametersTable.HasLoadedOrAssignedValue)

{

throw new System.Data.Linq.ForeignKeyReferenceAlreadyHasValueException();

}

this.OnFK\_ParametrsTableChanging(value);

this.SendPropertyChanging();

this.\_FK\_ParametrsTable = value;

this.SendPropertyChanged("FK\_ParametrsTable");

this.OnFK\_ParametrsTableChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_FK\_UsersTable", DbType="Int NOT NULL")]

public int FK\_UsersTable

{

get

{

return this.\_FK\_UsersTable;

}

set

{

if ((this.\_FK\_UsersTable != value))

{

if (this.\_UsersTable.HasLoadedOrAssignedValue)

{

throw new System.Data.Linq.ForeignKeyReferenceAlreadyHasValueException();

}

this.OnFK\_UsersTableChanging(value);

this.SendPropertyChanging();

this.\_FK\_UsersTable = value;

this.SendPropertyChanged("FK\_UsersTable");

this.OnFK\_UsersTableChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_CanEdit", DbType="Bit NOT NULL")]

public bool CanEdit

{

get

{

return this.\_CanEdit;

}

set

{

if ((this.\_CanEdit != value))

{

this.OnCanEditChanging(value);

this.SendPropertyChanging();

this.\_CanEdit = value;

this.SendPropertyChanged("CanEdit");

this.OnCanEditChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_CanView", DbType="Bit NOT NULL")]

public bool CanView

{

get

{

return this.\_CanView;

}

set

{

if ((this.\_CanView != value))

{

this.OnCanViewChanging(value);

this.SendPropertyChanging();

this.\_CanView = value;

this.SendPropertyChanged("CanView");

this.OnCanViewChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_CanConfirm", DbType="Bit NOT NULL")]

public bool CanConfirm

{

get

{

return this.\_CanConfirm;

}

set

{

if ((this.\_CanConfirm != value))

{

this.OnCanConfirmChanging(value);

this.SendPropertyChanging();

this.\_CanConfirm = value;

this.SendPropertyChanged("CanConfirm");

this.OnCanConfirmChanged();

}

}

}

[global::System.Data.Linq.Mapping.AssociationAttribute(Name="BasicParametersTable\_BasicParametrsAndUsersMapping", Storage="\_BasicParametersTable", ThisKey="FK\_ParametrsTable", OtherKey="BasicParametersTableID", IsForeignKey=true)]

public BasicParametersTable BasicParametersTable

{

get

{

return this.\_BasicParametersTable.Entity;

}

set

{

BasicParametersTable previousValue = this.\_BasicParametersTable.Entity;

if (((previousValue != value)

|| (this.\_BasicParametersTable.HasLoadedOrAssignedValue == false)))

{

this.SendPropertyChanging();

if ((previousValue != null))

{

this.\_BasicParametersTable.Entity = null;

previousValue.BasicParametrsAndUsersMapping.Remove(this);

}

this.\_BasicParametersTable.Entity = value;

if ((value != null))

{

value.BasicParametrsAndUsersMapping.Add(this);

this.\_FK\_ParametrsTable = value.BasicParametersTableID;

}

else

{

this.\_FK\_ParametrsTable = default(int);

}

this.SendPropertyChanged("BasicParametersTable");

}

}

}

[global::System.Data.Linq.Mapping.AssociationAttribute(Name="UsersTable\_BasicParametrsAndUsersMapping", Storage="\_UsersTable", ThisKey="FK\_UsersTable", OtherKey="UsersTableID", IsForeignKey=true)]

public UsersTable UsersTable

{

get

{

return this.\_UsersTable.Entity;

}

set

{

UsersTable previousValue = this.\_UsersTable.Entity;

if (((previousValue != value)

|| (this.\_UsersTable.HasLoadedOrAssignedValue == false)))

{

this.SendPropertyChanging();

if ((previousValue != null))

{

this.\_UsersTable.Entity = null;

previousValue.BasicParametrsAndUsersMapping.Remove(this);

}

this.\_UsersTable.Entity = value;

if ((value != null))

{

value.BasicParametrsAndUsersMapping.Add(this);

this.\_FK\_UsersTable = value.UsersTableID;

}

else

{

this.\_FK\_UsersTable = default(int);

}

this.SendPropertyChanged("UsersTable");

}

}

}

public event PropertyChangingEventHandler PropertyChanging;

public event PropertyChangedEventHandler PropertyChanged;

protected virtual void SendPropertyChanging()

{

if ((this.PropertyChanging != null))

{

this.PropertyChanging(this, emptyChangingEventArgs);

}

}

protected virtual void SendPropertyChanged(String propertyName)

{

if ((this.PropertyChanged != null))

{

this.PropertyChanged(this, new PropertyChangedEventArgs(propertyName));

}

}

}

[global::System.Data.Linq.Mapping.TableAttribute(Name="dbo.CalculatedParametrs")]

public partial class CalculatedParametrs : INotifyPropertyChanging, INotifyPropertyChanged

{

private static PropertyChangingEventArgs emptyChangingEventArgs = new PropertyChangingEventArgs(String.Empty);

private int \_CalculatedParametrsID;

private bool \_Active;

private string \_Name;

private string \_Measure;

private string \_Formula;

private string \_AbbreviationEN;

private string \_AbbreviationRU;

private EntitySet<CalculatedParametrsAndRolesMappingTable> \_CalculatedParametrsAndRolesMappingTable;

private EntitySet<CalculatedParametrsAndUsersMapping> \_CalculatedParametrsAndUsersMapping;

private EntitySet<CollectedCalculatedParametrs> \_CollectedCalculatedParametrs;

private EntitySet<ConfirmationHistory> \_ConfirmationHistory;

private EntitySet<ReportArchiveAndCalculatedParametrsMappingTable> \_ReportArchiveAndCalculatedParametrsMappingTable;

#region Extensibility Method Definitions

partial void OnLoaded();

partial void OnValidate(System.Data.Linq.ChangeAction action);

partial void OnCreated();

partial void OnCalculatedParametrsIDChanging(int value);

partial void OnCalculatedParametrsIDChanged();

partial void OnActiveChanging(bool value);

partial void OnActiveChanged();

partial void OnNameChanging(string value);

partial void OnNameChanged();

partial void OnMeasureChanging(string value);

partial void OnMeasureChanged();

partial void OnFormulaChanging(string value);

partial void OnFormulaChanged();

partial void OnAbbreviationENChanging(string value);

partial void OnAbbreviationENChanged();

partial void OnAbbreviationRUChanging(string value);

partial void OnAbbreviationRUChanged();

#endregion

public CalculatedParametrs()

{

this.\_CalculatedParametrsAndRolesMappingTable = new EntitySet<CalculatedParametrsAndRolesMappingTable>(new Action<CalculatedParametrsAndRolesMappingTable>(this.attach\_CalculatedParametrsAndRolesMappingTable), new Action<CalculatedParametrsAndRolesMappingTable>(this.detach\_CalculatedParametrsAndRolesMappingTable));

this.\_CalculatedParametrsAndUsersMapping = new EntitySet<CalculatedParametrsAndUsersMapping>(new Action<CalculatedParametrsAndUsersMapping>(this.attach\_CalculatedParametrsAndUsersMapping), new Action<CalculatedParametrsAndUsersMapping>(this.detach\_CalculatedParametrsAndUsersMapping));

this.\_CollectedCalculatedParametrs = new EntitySet<CollectedCalculatedParametrs>(new Action<CollectedCalculatedParametrs>(this.attach\_CollectedCalculatedParametrs), new Action<CollectedCalculatedParametrs>(this.detach\_CollectedCalculatedParametrs));

this.\_ConfirmationHistory = new EntitySet<ConfirmationHistory>(new Action<ConfirmationHistory>(this.attach\_ConfirmationHistory), new Action<ConfirmationHistory>(this.detach\_ConfirmationHistory));

this.\_ReportArchiveAndCalculatedParametrsMappingTable = new EntitySet<ReportArchiveAndCalculatedParametrsMappingTable>(new Action<ReportArchiveAndCalculatedParametrsMappingTable>(this.attach\_ReportArchiveAndCalculatedParametrsMappingTable), new Action<ReportArchiveAndCalculatedParametrsMappingTable>(this.detach\_ReportArchiveAndCalculatedParametrsMappingTable));

OnCreated();

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_CalculatedParametrsID", AutoSync=AutoSync.OnInsert, DbType="Int NOT NULL IDENTITY", IsPrimaryKey=true, IsDbGenerated=true)]

public int CalculatedParametrsID

{

get

{

return this.\_CalculatedParametrsID;

}

set

{

if ((this.\_CalculatedParametrsID != value))

{

this.OnCalculatedParametrsIDChanging(value);

this.SendPropertyChanging();

this.\_CalculatedParametrsID = value;

this.SendPropertyChanged("CalculatedParametrsID");

this.OnCalculatedParametrsIDChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_Active", DbType="Bit NOT NULL")]

public bool Active

{

get

{

return this.\_Active;

}

set

{

if ((this.\_Active != value))

{

this.OnActiveChanging(value);

this.SendPropertyChanging();

this.\_Active = value;

this.SendPropertyChanged("Active");

this.OnActiveChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_Name", DbType="VarChar(1000)")]

public string Name

{

get

{

return this.\_Name;

}

set

{

if ((this.\_Name != value))

{

this.OnNameChanging(value);

this.SendPropertyChanging();

this.\_Name = value;

this.SendPropertyChanged("Name");

this.OnNameChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_Measure", DbType="VarChar(10)")]

public string Measure

{

get

{

return this.\_Measure;

}

set

{

if ((this.\_Measure != value))

{

this.OnMeasureChanging(value);

this.SendPropertyChanging();

this.\_Measure = value;

this.SendPropertyChanged("Measure");

this.OnMeasureChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_Formula", DbType="VarChar(5000)")]

public string Formula

{

get

{

return this.\_Formula;

}

set

{

if ((this.\_Formula != value))

{

this.OnFormulaChanging(value);

this.SendPropertyChanging();

this.\_Formula = value;

this.SendPropertyChanged("Formula");

this.OnFormulaChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_AbbreviationEN", DbType="VarChar(50)")]

public string AbbreviationEN

{

get

{

return this.\_AbbreviationEN;

}

set

{

if ((this.\_AbbreviationEN != value))

{

this.OnAbbreviationENChanging(value);

this.SendPropertyChanging();

this.\_AbbreviationEN = value;

this.SendPropertyChanged("AbbreviationEN");

this.OnAbbreviationENChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_AbbreviationRU", DbType="VarChar(50)")]

public string AbbreviationRU

{

get

{

return this.\_AbbreviationRU;

}

set

{

if ((this.\_AbbreviationRU != value))

{

this.OnAbbreviationRUChanging(value);

this.SendPropertyChanging();

this.\_AbbreviationRU = value;

this.SendPropertyChanged("AbbreviationRU");

this.OnAbbreviationRUChanged();

}

}

}

[global::System.Data.Linq.Mapping.AssociationAttribute(Name="CalculatedParametrs\_CalculatedParametrsAndRolesMappingTable", Storage="\_CalculatedParametrsAndRolesMappingTable", ThisKey="CalculatedParametrsID", OtherKey="FK\_CalculatedParametrs")]

public EntitySet<CalculatedParametrsAndRolesMappingTable> CalculatedParametrsAndRolesMappingTable

{

get

{

return this.\_CalculatedParametrsAndRolesMappingTable;

}

set

{

this.\_CalculatedParametrsAndRolesMappingTable.Assign(value);

}

}

[global::System.Data.Linq.Mapping.AssociationAttribute(Name="CalculatedParametrs\_CalculatedParametrsAndUsersMapping", Storage="\_CalculatedParametrsAndUsersMapping", ThisKey="CalculatedParametrsID", OtherKey="FK\_CalculatedParametrsTable")]

public EntitySet<CalculatedParametrsAndUsersMapping> CalculatedParametrsAndUsersMapping

{

get

{

return this.\_CalculatedParametrsAndUsersMapping;

}

set

{

this.\_CalculatedParametrsAndUsersMapping.Assign(value);

}

}

[global::System.Data.Linq.Mapping.AssociationAttribute(Name="CalculatedParametrs\_CollectedCalculatedParametrs", Storage="\_CollectedCalculatedParametrs", ThisKey="CalculatedParametrsID", OtherKey="FK\_CalculatedParametrs")]

public EntitySet<CollectedCalculatedParametrs> CollectedCalculatedParametrs

{

get

{

return this.\_CollectedCalculatedParametrs;

}

set

{

this.\_CollectedCalculatedParametrs.Assign(value);

}

}

[global::System.Data.Linq.Mapping.AssociationAttribute(Name="CalculatedParametrs\_ConfirmationHistory", Storage="\_ConfirmationHistory", ThisKey="CalculatedParametrsID", OtherKey="FK\_CalculatedParamTable")]

public EntitySet<ConfirmationHistory> ConfirmationHistory

{

get

{

return this.\_ConfirmationHistory;

}

set

{

this.\_ConfirmationHistory.Assign(value);

}

}

[global::System.Data.Linq.Mapping.AssociationAttribute(Name="CalculatedParametrs\_ReportArchiveAndCalculatedParametrsMappingTable", Storage="\_ReportArchiveAndCalculatedParametrsMappingTable", ThisKey="CalculatedParametrsID", OtherKey="FK\_CalculatedParametrsTable")]

public EntitySet<ReportArchiveAndCalculatedParametrsMappingTable> ReportArchiveAndCalculatedParametrsMappingTable

{

get

{

return this.\_ReportArchiveAndCalculatedParametrsMappingTable;

}

set

{

this.\_ReportArchiveAndCalculatedParametrsMappingTable.Assign(value);

}

}

public event PropertyChangingEventHandler PropertyChanging;

public event PropertyChangedEventHandler PropertyChanged;

protected virtual void SendPropertyChanging()

{

if ((this.PropertyChanging != null))

{

this.PropertyChanging(this, emptyChangingEventArgs);

}

}

protected virtual void SendPropertyChanged(String propertyName)

{

if ((this.PropertyChanged != null))

{

this.PropertyChanged(this, new PropertyChangedEventArgs(propertyName));

}

}

private void attach\_CalculatedParametrsAndRolesMappingTable(CalculatedParametrsAndRolesMappingTable entity)

{

this.SendPropertyChanging();

entity.CalculatedParametrs = this;

}

private void detach\_CalculatedParametrsAndRolesMappingTable(CalculatedParametrsAndRolesMappingTable entity)

{

this.SendPropertyChanging();

entity.CalculatedParametrs = null;

}

private void attach\_CalculatedParametrsAndUsersMapping(CalculatedParametrsAndUsersMapping entity)

{

this.SendPropertyChanging();

entity.CalculatedParametrs = this;

}

private void detach\_CalculatedParametrsAndUsersMapping(CalculatedParametrsAndUsersMapping entity)

{

this.SendPropertyChanging();

entity.CalculatedParametrs = null;

}

private void attach\_CollectedCalculatedParametrs(CollectedCalculatedParametrs entity)

{

this.SendPropertyChanging();

entity.CalculatedParametrs = this;

}

private void detach\_CollectedCalculatedParametrs(CollectedCalculatedParametrs entity)

{

this.SendPropertyChanging();

entity.CalculatedParametrs = null;

}

private void attach\_ConfirmationHistory(ConfirmationHistory entity)

{

this.SendPropertyChanging();

entity.CalculatedParametrs = this;

}

private void detach\_ConfirmationHistory(ConfirmationHistory entity)

{

this.SendPropertyChanging();

entity.CalculatedParametrs = null;

}

private void attach\_ReportArchiveAndCalculatedParametrsMappingTable(ReportArchiveAndCalculatedParametrsMappingTable entity)

{

this.SendPropertyChanging();

entity.CalculatedParametrs = this;

}

private void detach\_ReportArchiveAndCalculatedParametrsMappingTable(ReportArchiveAndCalculatedParametrsMappingTable entity)

{

this.SendPropertyChanging();

entity.CalculatedParametrs = null;

}

}

[global::System.Data.Linq.Mapping.TableAttribute(Name="dbo.CalculatedParametrsAndRolesMappingTable")]

public partial class CalculatedParametrsAndRolesMappingTable : INotifyPropertyChanging, INotifyPropertyChanged

{

private static PropertyChangingEventArgs emptyChangingEventArgs = new PropertyChangingEventArgs(String.Empty);

private int \_BasicParametersAndRolesWithLeadersID;

private bool \_Active;

private System.Nullable<int> \_FK\_RolesTable;

private System.Nullable<int> \_FK\_CalculatedParametrs;

private System.Nullable<bool> \_CanEdit;

private System.Nullable<bool> \_CanView;

private System.Nullable<bool> \_CanConfirm;

private EntityRef<CalculatedParametrs> \_CalculatedParametrs;

private EntityRef<RolesTable> \_RolesTable;

#region Extensibility Method Definitions

partial void OnLoaded();

partial void OnValidate(System.Data.Linq.ChangeAction action);

partial void OnCreated();

partial void OnBasicParametersAndRolesWithLeadersIDChanging(int value);

partial void OnBasicParametersAndRolesWithLeadersIDChanged();

partial void OnActiveChanging(bool value);

partial void OnActiveChanged();

partial void OnFK\_RolesTableChanging(System.Nullable<int> value);

partial void OnFK\_RolesTableChanged();

partial void OnFK\_CalculatedParametrsChanging(System.Nullable<int> value);

partial void OnFK\_CalculatedParametrsChanged();

partial void OnCanEditChanging(System.Nullable<bool> value);

partial void OnCanEditChanged();

partial void OnCanViewChanging(System.Nullable<bool> value);

partial void OnCanViewChanged();

partial void OnCanConfirmChanging(System.Nullable<bool> value);

partial void OnCanConfirmChanged();

#endregion

public CalculatedParametrsAndRolesMappingTable()

{

this.\_CalculatedParametrs = default(EntityRef<CalculatedParametrs>);

this.\_RolesTable = default(EntityRef<RolesTable>);

OnCreated();

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_BasicParametersAndRolesWithLeadersID", AutoSync=AutoSync.OnInsert, DbType="Int NOT NULL IDENTITY", IsPrimaryKey=true, IsDbGenerated=true)]

public int BasicParametersAndRolesWithLeadersID

{

get

{

return this.\_BasicParametersAndRolesWithLeadersID;

}

set

{

if ((this.\_BasicParametersAndRolesWithLeadersID != value))

{

this.OnBasicParametersAndRolesWithLeadersIDChanging(value);

this.SendPropertyChanging();

this.\_BasicParametersAndRolesWithLeadersID = value;

this.SendPropertyChanged("BasicParametersAndRolesWithLeadersID");

this.OnBasicParametersAndRolesWithLeadersIDChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_Active", DbType="Bit NOT NULL")]

public bool Active

{

get

{

return this.\_Active;

}

set

{

if ((this.\_Active != value))

{

this.OnActiveChanging(value);

this.SendPropertyChanging();

this.\_Active = value;

this.SendPropertyChanged("Active");

this.OnActiveChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_FK\_RolesTable", DbType="Int")]

public System.Nullable<int> FK\_RolesTable

{

get

{

return this.\_FK\_RolesTable;

}

set

{

if ((this.\_FK\_RolesTable != value))

{

if (this.\_RolesTable.HasLoadedOrAssignedValue)

{

throw new System.Data.Linq.ForeignKeyReferenceAlreadyHasValueException();

}

this.OnFK\_RolesTableChanging(value);

this.SendPropertyChanging();

this.\_FK\_RolesTable = value;

this.SendPropertyChanged("FK\_RolesTable");

this.OnFK\_RolesTableChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_FK\_CalculatedParametrs", DbType="Int")]

public System.Nullable<int> FK\_CalculatedParametrs

{

get

{

return this.\_FK\_CalculatedParametrs;

}

set

{

if ((this.\_FK\_CalculatedParametrs != value))

{

if (this.\_CalculatedParametrs.HasLoadedOrAssignedValue)

{

throw new System.Data.Linq.ForeignKeyReferenceAlreadyHasValueException();

}

this.OnFK\_CalculatedParametrsChanging(value);

this.SendPropertyChanging();

this.\_FK\_CalculatedParametrs = value;

this.SendPropertyChanged("FK\_CalculatedParametrs");

this.OnFK\_CalculatedParametrsChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_CanEdit", DbType="Bit")]

public System.Nullable<bool> CanEdit

{

get

{

return this.\_CanEdit;

}

set

{

if ((this.\_CanEdit != value))

{

this.OnCanEditChanging(value);

this.SendPropertyChanging();

this.\_CanEdit = value;

this.SendPropertyChanged("CanEdit");

this.OnCanEditChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_CanView", DbType="Bit")]

public System.Nullable<bool> CanView

{

get

{

return this.\_CanView;

}

set

{

if ((this.\_CanView != value))

{

this.OnCanViewChanging(value);

this.SendPropertyChanging();

this.\_CanView = value;

this.SendPropertyChanged("CanView");

this.OnCanViewChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_CanConfirm", DbType="Bit")]

public System.Nullable<bool> CanConfirm

{

get

{

return this.\_CanConfirm;

}

set

{

if ((this.\_CanConfirm != value))

{

this.OnCanConfirmChanging(value);

this.SendPropertyChanging();

this.\_CanConfirm = value;

this.SendPropertyChanged("CanConfirm");

this.OnCanConfirmChanged();

}

}

}

[global::System.Data.Linq.Mapping.AssociationAttribute(Name="CalculatedParametrs\_CalculatedParametrsAndRolesMappingTable", Storage="\_CalculatedParametrs", ThisKey="FK\_CalculatedParametrs", OtherKey="CalculatedParametrsID", IsForeignKey=true)]

public CalculatedParametrs CalculatedParametrs

{

get

{

return this.\_CalculatedParametrs.Entity;

}

set

{

CalculatedParametrs previousValue = this.\_CalculatedParametrs.Entity;

if (((previousValue != value)

|| (this.\_CalculatedParametrs.HasLoadedOrAssignedValue == false)))

{

this.SendPropertyChanging();

if ((previousValue != null))

{

this.\_CalculatedParametrs.Entity = null;

previousValue.CalculatedParametrsAndRolesMappingTable.Remove(this);

}

this.\_CalculatedParametrs.Entity = value;

if ((value != null))

{

value.CalculatedParametrsAndRolesMappingTable.Add(this);

this.\_FK\_CalculatedParametrs = value.CalculatedParametrsID;

}

else

{

this.\_FK\_CalculatedParametrs = default(Nullable<int>);

}

this.SendPropertyChanged("CalculatedParametrs");

}

}

}

[global::System.Data.Linq.Mapping.AssociationAttribute(Name="RolesTable\_CalculatedParametrsAndRolesMappingTable", Storage="\_RolesTable", ThisKey="FK\_RolesTable", OtherKey="RolesTableID", IsForeignKey=true)]

public RolesTable RolesTable

{

get

{

return this.\_RolesTable.Entity;

}

set

{

RolesTable previousValue = this.\_RolesTable.Entity;

if (((previousValue != value)

|| (this.\_RolesTable.HasLoadedOrAssignedValue == false)))

{

this.SendPropertyChanging();

if ((previousValue != null))

{

this.\_RolesTable.Entity = null;

previousValue.CalculatedParametrsAndRolesMappingTable.Remove(this);

}

this.\_RolesTable.Entity = value;

if ((value != null))

{

value.CalculatedParametrsAndRolesMappingTable.Add(this);

this.\_FK\_RolesTable = value.RolesTableID;

}

else

{

this.\_FK\_RolesTable = default(Nullable<int>);

}

this.SendPropertyChanged("RolesTable");

}

}

}

public event PropertyChangingEventHandler PropertyChanging;

public event PropertyChangedEventHandler PropertyChanged;

protected virtual void SendPropertyChanging()

{

if ((this.PropertyChanging != null))

{

this.PropertyChanging(this, emptyChangingEventArgs);

}

}

protected virtual void SendPropertyChanged(String propertyName)

{

if ((this.PropertyChanged != null))

{

this.PropertyChanged(this, new PropertyChangedEventArgs(propertyName));

}

}

}

[global::System.Data.Linq.Mapping.TableAttribute(Name="dbo.CalculatedParametrsAndUsersMapping")]

public partial class CalculatedParametrsAndUsersMapping : INotifyPropertyChanging, INotifyPropertyChanged

{

private static PropertyChangingEventArgs emptyChangingEventArgs = new PropertyChangingEventArgs(String.Empty);

private int \_CalculatedParametrsAndUsersMappingID;

private bool \_Active;

private System.Nullable<bool> \_CanEdit;

private System.Nullable<bool> \_CanView;

private System.Nullable<bool> \_CanConfirm;

private System.Nullable<int> \_FK\_UsersTable;

private System.Nullable<int> \_FK\_CalculatedParametrsTable;

private EntityRef<CalculatedParametrs> \_CalculatedParametrs;

private EntityRef<UsersTable> \_UsersTable;

#region Extensibility Method Definitions

partial void OnLoaded();

partial void OnValidate(System.Data.Linq.ChangeAction action);

partial void OnCreated();

partial void OnCalculatedParametrsAndUsersMappingIDChanging(int value);

partial void OnCalculatedParametrsAndUsersMappingIDChanged();

partial void OnActiveChanging(bool value);

partial void OnActiveChanged();

partial void OnCanEditChanging(System.Nullable<bool> value);

partial void OnCanEditChanged();

partial void OnCanViewChanging(System.Nullable<bool> value);

partial void OnCanViewChanged();

partial void OnCanConfirmChanging(System.Nullable<bool> value);

partial void OnCanConfirmChanged();

partial void OnFK\_UsersTableChanging(System.Nullable<int> value);

partial void OnFK\_UsersTableChanged();

partial void OnFK\_CalculatedParametrsTableChanging(System.Nullable<int> value);

partial void OnFK\_CalculatedParametrsTableChanged();

#endregion

public CalculatedParametrsAndUsersMapping()

{

this.\_CalculatedParametrs = default(EntityRef<CalculatedParametrs>);

this.\_UsersTable = default(EntityRef<UsersTable>);

OnCreated();

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_CalculatedParametrsAndUsersMappingID", AutoSync=AutoSync.OnInsert, DbType="Int NOT NULL IDENTITY", IsPrimaryKey=true, IsDbGenerated=true)]

public int CalculatedParametrsAndUsersMappingID

{

get

{

return this.\_CalculatedParametrsAndUsersMappingID;

}

set

{

if ((this.\_CalculatedParametrsAndUsersMappingID != value))

{

this.OnCalculatedParametrsAndUsersMappingIDChanging(value);

this.SendPropertyChanging();

this.\_CalculatedParametrsAndUsersMappingID = value;

this.SendPropertyChanged("CalculatedParametrsAndUsersMappingID");

this.OnCalculatedParametrsAndUsersMappingIDChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_Active", DbType="Bit NOT NULL")]

public bool Active

{

get

{

return this.\_Active;

}

set

{

if ((this.\_Active != value))

{

this.OnActiveChanging(value);

this.SendPropertyChanging();

this.\_Active = value;

this.SendPropertyChanged("Active");

this.OnActiveChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_CanEdit", DbType="Bit")]

public System.Nullable<bool> CanEdit

{

get

{

return this.\_CanEdit;

}

set

{

if ((this.\_CanEdit != value))

{

this.OnCanEditChanging(value);

this.SendPropertyChanging();

this.\_CanEdit = value;

this.SendPropertyChanged("CanEdit");

this.OnCanEditChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_CanView", DbType="Bit")]

public System.Nullable<bool> CanView

{

get

{

return this.\_CanView;

}

set

{

if ((this.\_CanView != value))

{

this.OnCanViewChanging(value);

this.SendPropertyChanging();

this.\_CanView = value;

this.SendPropertyChanged("CanView");

this.OnCanViewChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_CanConfirm", DbType="Bit")]

public System.Nullable<bool> CanConfirm

{

get

{

return this.\_CanConfirm;

}

set

{

if ((this.\_CanConfirm != value))

{

this.OnCanConfirmChanging(value);

this.SendPropertyChanging();

this.\_CanConfirm = value;

this.SendPropertyChanged("CanConfirm");

this.OnCanConfirmChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_FK\_UsersTable", DbType="Int")]

public System.Nullable<int> FK\_UsersTable

{

get

{

return this.\_FK\_UsersTable;

}

set

{

if ((this.\_FK\_UsersTable != value))

{

if (this.\_UsersTable.HasLoadedOrAssignedValue)

{

throw new System.Data.Linq.ForeignKeyReferenceAlreadyHasValueException();

}

this.OnFK\_UsersTableChanging(value);

this.SendPropertyChanging();

this.\_FK\_UsersTable = value;

this.SendPropertyChanged("FK\_UsersTable");

this.OnFK\_UsersTableChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_FK\_CalculatedParametrsTable", DbType="Int")]

public System.Nullable<int> FK\_CalculatedParametrsTable

{

get

{

return this.\_FK\_CalculatedParametrsTable;

}

set

{

if ((this.\_FK\_CalculatedParametrsTable != value))

{

if (this.\_CalculatedParametrs.HasLoadedOrAssignedValue)

{

throw new System.Data.Linq.ForeignKeyReferenceAlreadyHasValueException();

}

this.OnFK\_CalculatedParametrsTableChanging(value);

this.SendPropertyChanging();

this.\_FK\_CalculatedParametrsTable = value;

this.SendPropertyChanged("FK\_CalculatedParametrsTable");

this.OnFK\_CalculatedParametrsTableChanged();

}

}

}

[global::System.Data.Linq.Mapping.AssociationAttribute(Name="CalculatedParametrs\_CalculatedParametrsAndUsersMapping", Storage="\_CalculatedParametrs", ThisKey="FK\_CalculatedParametrsTable", OtherKey="CalculatedParametrsID", IsForeignKey=true)]

public CalculatedParametrs CalculatedParametrs

{

get

{

return this.\_CalculatedParametrs.Entity;

}

set

{

CalculatedParametrs previousValue = this.\_CalculatedParametrs.Entity;

if (((previousValue != value)

|| (this.\_CalculatedParametrs.HasLoadedOrAssignedValue == false)))

{

this.SendPropertyChanging();

if ((previousValue != null))

{

this.\_CalculatedParametrs.Entity = null;

previousValue.CalculatedParametrsAndUsersMapping.Remove(this);

}

this.\_CalculatedParametrs.Entity = value;

if ((value != null))

{

value.CalculatedParametrsAndUsersMapping.Add(this);

this.\_FK\_CalculatedParametrsTable = value.CalculatedParametrsID;

}

else

{

this.\_FK\_CalculatedParametrsTable = default(Nullable<int>);

}

this.SendPropertyChanged("CalculatedParametrs");

}

}

}

[global::System.Data.Linq.Mapping.AssociationAttribute(Name="UsersTable\_CalculatedParametrsAndUsersMapping", Storage="\_UsersTable", ThisKey="FK\_UsersTable", OtherKey="UsersTableID", IsForeignKey=true)]

public UsersTable UsersTable

{

get

{

return this.\_UsersTable.Entity;

}

set

{

UsersTable previousValue = this.\_UsersTable.Entity;

if (((previousValue != value)

|| (this.\_UsersTable.HasLoadedOrAssignedValue == false)))

{

this.SendPropertyChanging();

if ((previousValue != null))

{

this.\_UsersTable.Entity = null;

previousValue.CalculatedParametrsAndUsersMapping.Remove(this);

}

this.\_UsersTable.Entity = value;

if ((value != null))

{

value.CalculatedParametrsAndUsersMapping.Add(this);

this.\_FK\_UsersTable = value.UsersTableID;

}

else

{

this.\_FK\_UsersTable = default(Nullable<int>);

}

this.SendPropertyChanged("UsersTable");

}

}

}

public event PropertyChangingEventHandler PropertyChanging;

public event PropertyChangedEventHandler PropertyChanged;

protected virtual void SendPropertyChanging()

{

if ((this.PropertyChanging != null))

{

this.PropertyChanging(this, emptyChangingEventArgs);

}

}

protected virtual void SendPropertyChanged(String propertyName)

{

if ((this.PropertyChanged != null))

{

this.PropertyChanged(this, new PropertyChangedEventArgs(propertyName));

}

}

}

[global::System.Data.Linq.Mapping.TableAttribute(Name="dbo.CollectedBasicParametersTable")]

public partial class CollectedBasicParametersTable : INotifyPropertyChanging, INotifyPropertyChanged

{

private static PropertyChangingEventArgs emptyChangingEventArgs = new PropertyChangingEventArgs(String.Empty);

private int \_CollectedBasicParametersTableID;

private bool \_Active;

private int \_FK\_UsersTable;

private int \_FK\_ReportArchiveTable;

private System.DateTime \_LastChangeDateTime;

private System.Nullable<System.DateTime> \_SavedDateTime;

private string \_UserIP;

private System.Nullable<double> \_CollectedValue;

private System.Nullable<int> \_FK\_BasicParametersTable;

private System.Nullable<int> \_FK\_ZeroLevelSubdivisionTable;

private System.Nullable<int> \_FK\_FirstLevelSubdivisionTable;

private System.Nullable<int> \_FK\_SecondLevelSubdivisionTable;

private System.Nullable<int> \_FK\_ThirdLevelSubdivisionTable;

private System.Nullable<int> \_FK\_FourthLevelSubdivisionTable;

private System.Nullable<int> \_FK\_FifthLevelSubdivisionTable;

private System.Nullable<int> \_Status;

private EntityRef<BasicParametersTable> \_BasicParametersTable;

private EntityRef<ZeroLevelSubdivisionTable> \_ZeroLevelSubdivisionTable;

private EntityRef<FifthLevelSubdivisionTable> \_FifthLevelSubdivisionTable;

private EntityRef<FirstLevelSubdivisionTable> \_FirstLevelSubdivisionTable;

private EntityRef<FourthLevelSubdivisionTable> \_FourthLevelSubdivisionTable;

private EntityRef<ReportArchiveTable> \_ReportArchiveTable;

private EntityRef<SecondLevelSubdivisionTable> \_SecondLevelSubdivisionTable;

private EntityRef<ThirdLevelSubdivisionTable> \_ThirdLevelSubdivisionTable;

private EntityRef<UsersTable> \_UsersTable;

#region Extensibility Method Definitions

partial void OnLoaded();

partial void OnValidate(System.Data.Linq.ChangeAction action);

partial void OnCreated();

partial void OnCollectedBasicParametersTableIDChanging(int value);

partial void OnCollectedBasicParametersTableIDChanged();

partial void OnActiveChanging(bool value);

partial void OnActiveChanged();

partial void OnFK\_UsersTableChanging(int value);

partial void OnFK\_UsersTableChanged();

partial void OnFK\_ReportArchiveTableChanging(int value);

partial void OnFK\_ReportArchiveTableChanged();

partial void OnLastChangeDateTimeChanging(System.DateTime value);

partial void OnLastChangeDateTimeChanged();

partial void OnSavedDateTimeChanging(System.Nullable<System.DateTime> value);

partial void OnSavedDateTimeChanged();

partial void OnUserIPChanging(string value);

partial void OnUserIPChanged();

partial void OnCollectedValueChanging(System.Nullable<double> value);

partial void OnCollectedValueChanged();

partial void OnFK\_BasicParametersTableChanging(System.Nullable<int> value);

partial void OnFK\_BasicParametersTableChanged();

partial void OnFK\_ZeroLevelSubdivisionTableChanging(System.Nullable<int> value);

partial void OnFK\_ZeroLevelSubdivisionTableChanged();

partial void OnFK\_FirstLevelSubdivisionTableChanging(System.Nullable<int> value);

partial void OnFK\_FirstLevelSubdivisionTableChanged();

partial void OnFK\_SecondLevelSubdivisionTableChanging(System.Nullable<int> value);

partial void OnFK\_SecondLevelSubdivisionTableChanged();

partial void OnFK\_ThirdLevelSubdivisionTableChanging(System.Nullable<int> value);

partial void OnFK\_ThirdLevelSubdivisionTableChanged();

partial void OnFK\_FourthLevelSubdivisionTableChanging(System.Nullable<int> value);

partial void OnFK\_FourthLevelSubdivisionTableChanged();

partial void OnFK\_FifthLevelSubdivisionTableChanging(System.Nullable<int> value);

partial void OnFK\_FifthLevelSubdivisionTableChanged();

partial void OnStatusChanging(System.Nullable<int> value);

partial void OnStatusChanged();

#endregion

public CollectedBasicParametersTable()

{

this.\_BasicParametersTable = default(EntityRef<BasicParametersTable>);

this.\_ZeroLevelSubdivisionTable = default(EntityRef<ZeroLevelSubdivisionTable>);

this.\_FifthLevelSubdivisionTable = default(EntityRef<FifthLevelSubdivisionTable>);

this.\_FirstLevelSubdivisionTable = default(EntityRef<FirstLevelSubdivisionTable>);

this.\_FourthLevelSubdivisionTable = default(EntityRef<FourthLevelSubdivisionTable>);

this.\_ReportArchiveTable = default(EntityRef<ReportArchiveTable>);

this.\_SecondLevelSubdivisionTable = default(EntityRef<SecondLevelSubdivisionTable>);

this.\_ThirdLevelSubdivisionTable = default(EntityRef<ThirdLevelSubdivisionTable>);

this.\_UsersTable = default(EntityRef<UsersTable>);

OnCreated();

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_CollectedBasicParametersTableID", AutoSync=AutoSync.OnInsert, DbType="Int NOT NULL IDENTITY", IsPrimaryKey=true, IsDbGenerated=true)]

public int CollectedBasicParametersTableID

{

get

{

return this.\_CollectedBasicParametersTableID;

}

set

{

if ((this.\_CollectedBasicParametersTableID != value))

{

this.OnCollectedBasicParametersTableIDChanging(value);

this.SendPropertyChanging();

this.\_CollectedBasicParametersTableID = value;

this.SendPropertyChanged("CollectedBasicParametersTableID");

this.OnCollectedBasicParametersTableIDChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_Active", DbType="Bit NOT NULL")]

public bool Active

{

get

{

return this.\_Active;

}

set

{

if ((this.\_Active != value))

{

this.OnActiveChanging(value);

this.SendPropertyChanging();

this.\_Active = value;

this.SendPropertyChanged("Active");

this.OnActiveChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_FK\_UsersTable", DbType="Int NOT NULL")]

public int FK\_UsersTable

{

get

{

return this.\_FK\_UsersTable;

}

set

{

if ((this.\_FK\_UsersTable != value))

{

if (this.\_UsersTable.HasLoadedOrAssignedValue)

{

throw new System.Data.Linq.ForeignKeyReferenceAlreadyHasValueException();

}

this.OnFK\_UsersTableChanging(value);

this.SendPropertyChanging();

this.\_FK\_UsersTable = value;

this.SendPropertyChanged("FK\_UsersTable");

this.OnFK\_UsersTableChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_FK\_ReportArchiveTable", DbType="Int NOT NULL")]

public int FK\_ReportArchiveTable

{

get

{

return this.\_FK\_ReportArchiveTable;

}

set

{

if ((this.\_FK\_ReportArchiveTable != value))

{

if (this.\_ReportArchiveTable.HasLoadedOrAssignedValue)

{

throw new System.Data.Linq.ForeignKeyReferenceAlreadyHasValueException();

}

this.OnFK\_ReportArchiveTableChanging(value);

this.SendPropertyChanging();

this.\_FK\_ReportArchiveTable = value;

this.SendPropertyChanged("FK\_ReportArchiveTable");

this.OnFK\_ReportArchiveTableChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_LastChangeDateTime", DbType="DateTime NOT NULL")]

public System.DateTime LastChangeDateTime

{

get

{

return this.\_LastChangeDateTime;

}

set

{

if ((this.\_LastChangeDateTime != value))

{

this.OnLastChangeDateTimeChanging(value);

this.SendPropertyChanging();

this.\_LastChangeDateTime = value;

this.SendPropertyChanged("LastChangeDateTime");

this.OnLastChangeDateTimeChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_SavedDateTime", DbType="DateTime")]

public System.Nullable<System.DateTime> SavedDateTime

{

get

{

return this.\_SavedDateTime;

}

set

{

if ((this.\_SavedDateTime != value))

{

this.OnSavedDateTimeChanging(value);

this.SendPropertyChanging();

this.\_SavedDateTime = value;

this.SendPropertyChanged("SavedDateTime");

this.OnSavedDateTimeChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_UserIP", DbType="VarChar(15)")]

public string UserIP

{

get

{

return this.\_UserIP;

}

set

{

if ((this.\_UserIP != value))

{

this.OnUserIPChanging(value);

this.SendPropertyChanging();

this.\_UserIP = value;

this.SendPropertyChanged("UserIP");

this.OnUserIPChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_CollectedValue", DbType="Float")]

public System.Nullable<double> CollectedValue

{

get

{

return this.\_CollectedValue;

}

set

{

if ((this.\_CollectedValue != value))

{

this.OnCollectedValueChanging(value);

this.SendPropertyChanging();

this.\_CollectedValue = value;

this.SendPropertyChanged("CollectedValue");

this.OnCollectedValueChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_FK\_BasicParametersTable", DbType="Int")]

public System.Nullable<int> FK\_BasicParametersTable

{

get

{

return this.\_FK\_BasicParametersTable;

}

set

{

if ((this.\_FK\_BasicParametersTable != value))

{

if (this.\_BasicParametersTable.HasLoadedOrAssignedValue)

{

throw new System.Data.Linq.ForeignKeyReferenceAlreadyHasValueException();

}

this.OnFK\_BasicParametersTableChanging(value);

this.SendPropertyChanging();

this.\_FK\_BasicParametersTable = value;

this.SendPropertyChanged("FK\_BasicParametersTable");

this.OnFK\_BasicParametersTableChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_FK\_ZeroLevelSubdivisionTable", DbType="Int")]

public System.Nullable<int> FK\_ZeroLevelSubdivisionTable

{

get

{

return this.\_FK\_ZeroLevelSubdivisionTable;

}

set

{

if ((this.\_FK\_ZeroLevelSubdivisionTable != value))

{

if (this.\_ZeroLevelSubdivisionTable.HasLoadedOrAssignedValue)

{

throw new System.Data.Linq.ForeignKeyReferenceAlreadyHasValueException();

}

this.OnFK\_ZeroLevelSubdivisionTableChanging(value);

this.SendPropertyChanging();

this.\_FK\_ZeroLevelSubdivisionTable = value;

this.SendPropertyChanged("FK\_ZeroLevelSubdivisionTable");

this.OnFK\_ZeroLevelSubdivisionTableChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_FK\_FirstLevelSubdivisionTable", DbType="Int")]

public System.Nullable<int> FK\_FirstLevelSubdivisionTable

{

get

{

return this.\_FK\_FirstLevelSubdivisionTable;

}

set

{

if ((this.\_FK\_FirstLevelSubdivisionTable != value))

{

if (this.\_FirstLevelSubdivisionTable.HasLoadedOrAssignedValue)

{

throw new System.Data.Linq.ForeignKeyReferenceAlreadyHasValueException();

}

this.OnFK\_FirstLevelSubdivisionTableChanging(value);

this.SendPropertyChanging();

this.\_FK\_FirstLevelSubdivisionTable = value;

this.SendPropertyChanged("FK\_FirstLevelSubdivisionTable");

this.OnFK\_FirstLevelSubdivisionTableChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_FK\_SecondLevelSubdivisionTable", DbType="Int")]

public System.Nullable<int> FK\_SecondLevelSubdivisionTable

{

get

{

return this.\_FK\_SecondLevelSubdivisionTable;

}

set

{

if ((this.\_FK\_SecondLevelSubdivisionTable != value))

{

if (this.\_SecondLevelSubdivisionTable.HasLoadedOrAssignedValue)

{

throw new System.Data.Linq.ForeignKeyReferenceAlreadyHasValueException();

}

this.OnFK\_SecondLevelSubdivisionTableChanging(value);

this.SendPropertyChanging();

this.\_FK\_SecondLevelSubdivisionTable = value;

this.SendPropertyChanged("FK\_SecondLevelSubdivisionTable");

this.OnFK\_SecondLevelSubdivisionTableChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_FK\_ThirdLevelSubdivisionTable", DbType="Int")]

public System.Nullable<int> FK\_ThirdLevelSubdivisionTable

{

get

{

return this.\_FK\_ThirdLevelSubdivisionTable;

}

set

{

if ((this.\_FK\_ThirdLevelSubdivisionTable != value))

{

if (this.\_ThirdLevelSubdivisionTable.HasLoadedOrAssignedValue)

{

throw new System.Data.Linq.ForeignKeyReferenceAlreadyHasValueException();

}

this.OnFK\_ThirdLevelSubdivisionTableChanging(value);

this.SendPropertyChanging();

this.\_FK\_ThirdLevelSubdivisionTable = value;

this.SendPropertyChanged("FK\_ThirdLevelSubdivisionTable");

this.OnFK\_ThirdLevelSubdivisionTableChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_FK\_FourthLevelSubdivisionTable", DbType="Int")]

public System.Nullable<int> FK\_FourthLevelSubdivisionTable

{

get

{

return this.\_FK\_FourthLevelSubdivisionTable;

}

set

{

if ((this.\_FK\_FourthLevelSubdivisionTable != value))

{

if (this.\_FourthLevelSubdivisionTable.HasLoadedOrAssignedValue)

{

throw new System.Data.Linq.ForeignKeyReferenceAlreadyHasValueException();

}

this.OnFK\_FourthLevelSubdivisionTableChanging(value);

this.SendPropertyChanging();

this.\_FK\_FourthLevelSubdivisionTable = value;

this.SendPropertyChanged("FK\_FourthLevelSubdivisionTable");

this.OnFK\_FourthLevelSubdivisionTableChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_FK\_FifthLevelSubdivisionTable", DbType="Int")]

public System.Nullable<int> FK\_FifthLevelSubdivisionTable

{

get

{

return this.\_FK\_FifthLevelSubdivisionTable;

}

set

{

if ((this.\_FK\_FifthLevelSubdivisionTable != value))

{

if (this.\_FifthLevelSubdivisionTable.HasLoadedOrAssignedValue)

{

throw new System.Data.Linq.ForeignKeyReferenceAlreadyHasValueException();

}

this.OnFK\_FifthLevelSubdivisionTableChanging(value);

this.SendPropertyChanging();

this.\_FK\_FifthLevelSubdivisionTable = value;

this.SendPropertyChanged("FK\_FifthLevelSubdivisionTable");

this.OnFK\_FifthLevelSubdivisionTableChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_Status", DbType="Int")]

public System.Nullable<int> Status

{

get

{

return this.\_Status;

}

set

{

if ((this.\_Status != value))

{

this.OnStatusChanging(value);

this.SendPropertyChanging();

this.\_Status = value;

this.SendPropertyChanged("Status");

this.OnStatusChanged();

}

}

}

[global::System.Data.Linq.Mapping.AssociationAttribute(Name="BasicParametersTable\_CollectedBasicParametersTable", Storage="\_BasicParametersTable", ThisKey="FK\_BasicParametersTable", OtherKey="BasicParametersTableID", IsForeignKey=true)]

public BasicParametersTable BasicParametersTable

{

get

{

return this.\_BasicParametersTable.Entity;

}

set

{

BasicParametersTable previousValue = this.\_BasicParametersTable.Entity;

if (((previousValue != value)

|| (this.\_BasicParametersTable.HasLoadedOrAssignedValue == false)))

{

this.SendPropertyChanging();

if ((previousValue != null))

{

this.\_BasicParametersTable.Entity = null;

previousValue.CollectedBasicParametersTable.Remove(this);

}

this.\_BasicParametersTable.Entity = value;

if ((value != null))

{

value.CollectedBasicParametersTable.Add(this);

this.\_FK\_BasicParametersTable = value.BasicParametersTableID;

}

else

{

this.\_FK\_BasicParametersTable = default(Nullable<int>);

}

this.SendPropertyChanged("BasicParametersTable");

}

}

}

[global::System.Data.Linq.Mapping.AssociationAttribute(Name="ZeroLevelSubdivisionTable\_CollectedBasicParametersTable", Storage="\_ZeroLevelSubdivisionTable", ThisKey="FK\_ZeroLevelSubdivisionTable", OtherKey="ZeroLevelSubdivisionTableID", IsForeignKey=true)]

public ZeroLevelSubdivisionTable ZeroLevelSubdivisionTable

{

get

{

return this.\_ZeroLevelSubdivisionTable.Entity;

}

set

{

ZeroLevelSubdivisionTable previousValue = this.\_ZeroLevelSubdivisionTable.Entity;

if (((previousValue != value)

|| (this.\_ZeroLevelSubdivisionTable.HasLoadedOrAssignedValue == false)))

{

this.SendPropertyChanging();

if ((previousValue != null))

{

this.\_ZeroLevelSubdivisionTable.Entity = null;

previousValue.CollectedBasicParametersTable.Remove(this);

}

this.\_ZeroLevelSubdivisionTable.Entity = value;

if ((value != null))

{

value.CollectedBasicParametersTable.Add(this);

this.\_FK\_ZeroLevelSubdivisionTable = value.ZeroLevelSubdivisionTableID;

}

else

{

this.\_FK\_ZeroLevelSubdivisionTable = default(Nullable<int>);

}

this.SendPropertyChanged("ZeroLevelSubdivisionTable");

}

}

}

[global::System.Data.Linq.Mapping.AssociationAttribute(Name="FifthLevelSubdivisionTable\_CollectedBasicParametersTable", Storage="\_FifthLevelSubdivisionTable", ThisKey="FK\_FifthLevelSubdivisionTable", OtherKey="FifthLevelSubdivisionTableID", IsForeignKey=true)]

public FifthLevelSubdivisionTable FifthLevelSubdivisionTable

{

get

{

return this.\_FifthLevelSubdivisionTable.Entity;

}

set

{

FifthLevelSubdivisionTable previousValue = this.\_FifthLevelSubdivisionTable.Entity;

if (((previousValue != value)

|| (this.\_FifthLevelSubdivisionTable.HasLoadedOrAssignedValue == false)))

{

this.SendPropertyChanging();

if ((previousValue != null))

{

this.\_FifthLevelSubdivisionTable.Entity = null;

previousValue.CollectedBasicParametersTable.Remove(this);

}

this.\_FifthLevelSubdivisionTable.Entity = value;

if ((value != null))

{

value.CollectedBasicParametersTable.Add(this);

this.\_FK\_FifthLevelSubdivisionTable = value.FifthLevelSubdivisionTableID;

}

else

{

this.\_FK\_FifthLevelSubdivisionTable = default(Nullable<int>);

}

this.SendPropertyChanged("FifthLevelSubdivisionTable");

}

}

}

[global::System.Data.Linq.Mapping.AssociationAttribute(Name="FirstLevelSubdivisionTable\_CollectedBasicParametersTable", Storage="\_FirstLevelSubdivisionTable", ThisKey="FK\_FirstLevelSubdivisionTable", OtherKey="FirstLevelSubdivisionTableID", IsForeignKey=true)]

public FirstLevelSubdivisionTable FirstLevelSubdivisionTable

{

get

{

return this.\_FirstLevelSubdivisionTable.Entity;

}

set

{

FirstLevelSubdivisionTable previousValue = this.\_FirstLevelSubdivisionTable.Entity;

if (((previousValue != value)

|| (this.\_FirstLevelSubdivisionTable.HasLoadedOrAssignedValue == false)))

{

this.SendPropertyChanging();

if ((previousValue != null))

{

this.\_FirstLevelSubdivisionTable.Entity = null;

previousValue.CollectedBasicParametersTable.Remove(this);

}

this.\_FirstLevelSubdivisionTable.Entity = value;

if ((value != null))

{

value.CollectedBasicParametersTable.Add(this);

this.\_FK\_FirstLevelSubdivisionTable = value.FirstLevelSubdivisionTableID;

}

else

{

this.\_FK\_FirstLevelSubdivisionTable = default(Nullable<int>);

}

this.SendPropertyChanged("FirstLevelSubdivisionTable");

}

}

}

[global::System.Data.Linq.Mapping.AssociationAttribute(Name="FourthLevelSubdivisionTable\_CollectedBasicParametersTable", Storage="\_FourthLevelSubdivisionTable", ThisKey="FK\_FourthLevelSubdivisionTable", OtherKey="FourthLevelSubdivisionTableID", IsForeignKey=true)]

public FourthLevelSubdivisionTable FourthLevelSubdivisionTable

{

get

{

return this.\_FourthLevelSubdivisionTable.Entity;

}

set

{

FourthLevelSubdivisionTable previousValue = this.\_FourthLevelSubdivisionTable.Entity;

if (((previousValue != value)

|| (this.\_FourthLevelSubdivisionTable.HasLoadedOrAssignedValue == false)))

{

this.SendPropertyChanging();

if ((previousValue != null))

{

this.\_FourthLevelSubdivisionTable.Entity = null;

previousValue.CollectedBasicParametersTable.Remove(this);

}

this.\_FourthLevelSubdivisionTable.Entity = value;

if ((value != null))

{

value.CollectedBasicParametersTable.Add(this);

this.\_FK\_FourthLevelSubdivisionTable = value.FourthLevelSubdivisionTableID;

}

else

{

this.\_FK\_FourthLevelSubdivisionTable = default(Nullable<int>);

}

this.SendPropertyChanged("FourthLevelSubdivisionTable");

}

}

}

[global::System.Data.Linq.Mapping.AssociationAttribute(Name="ReportArchiveTable\_CollectedBasicParametersTable", Storage="\_ReportArchiveTable", ThisKey="FK\_ReportArchiveTable", OtherKey="ReportArchiveTableID", IsForeignKey=true)]

public ReportArchiveTable ReportArchiveTable

{

get

{

return this.\_ReportArchiveTable.Entity;

}

set

{

ReportArchiveTable previousValue = this.\_ReportArchiveTable.Entity;

if (((previousValue != value)

|| (this.\_ReportArchiveTable.HasLoadedOrAssignedValue == false)))

{

this.SendPropertyChanging();

if ((previousValue != null))

{

this.\_ReportArchiveTable.Entity = null;

previousValue.CollectedBasicParametersTable.Remove(this);

}

this.\_ReportArchiveTable.Entity = value;

if ((value != null))

{

value.CollectedBasicParametersTable.Add(this);

this.\_FK\_ReportArchiveTable = value.ReportArchiveTableID;

}

else

{

this.\_FK\_ReportArchiveTable = default(int);

}

this.SendPropertyChanged("ReportArchiveTable");

}

}

}

[global::System.Data.Linq.Mapping.AssociationAttribute(Name="SecondLevelSubdivisionTable\_CollectedBasicParametersTable", Storage="\_SecondLevelSubdivisionTable", ThisKey="FK\_SecondLevelSubdivisionTable", OtherKey="SecondLevelSubdivisionTableID", IsForeignKey=true)]

public SecondLevelSubdivisionTable SecondLevelSubdivisionTable

{

get

{

return this.\_SecondLevelSubdivisionTable.Entity;

}

set

{

SecondLevelSubdivisionTable previousValue = this.\_SecondLevelSubdivisionTable.Entity;

if (((previousValue != value)

|| (this.\_SecondLevelSubdivisionTable.HasLoadedOrAssignedValue == false)))

{

this.SendPropertyChanging();

if ((previousValue != null))

{

this.\_SecondLevelSubdivisionTable.Entity = null;

previousValue.CollectedBasicParametersTable.Remove(this);

}

this.\_SecondLevelSubdivisionTable.Entity = value;

if ((value != null))

{

value.CollectedBasicParametersTable.Add(this);

this.\_FK\_SecondLevelSubdivisionTable = value.SecondLevelSubdivisionTableID;

}

else

{

this.\_FK\_SecondLevelSubdivisionTable = default(Nullable<int>);

}

this.SendPropertyChanged("SecondLevelSubdivisionTable");

}

}

}

[global::System.Data.Linq.Mapping.AssociationAttribute(Name="ThirdLevelSubdivisionTable\_CollectedBasicParametersTable", Storage="\_ThirdLevelSubdivisionTable", ThisKey="FK\_ThirdLevelSubdivisionTable", OtherKey="ThirdLevelSubdivisionTableID", IsForeignKey=true)]

public ThirdLevelSubdivisionTable ThirdLevelSubdivisionTable

{

get

{

return this.\_ThirdLevelSubdivisionTable.Entity;

}

set

{

ThirdLevelSubdivisionTable previousValue = this.\_ThirdLevelSubdivisionTable.Entity;

if (((previousValue != value)

|| (this.\_ThirdLevelSubdivisionTable.HasLoadedOrAssignedValue == false)))

{

this.SendPropertyChanging();

if ((previousValue != null))

{

this.\_ThirdLevelSubdivisionTable.Entity = null;

previousValue.CollectedBasicParametersTable.Remove(this);

}

this.\_ThirdLevelSubdivisionTable.Entity = value;

if ((value != null))

{

value.CollectedBasicParametersTable.Add(this);

this.\_FK\_ThirdLevelSubdivisionTable = value.ThirdLevelSubdivisionTableID;

}

else

{

this.\_FK\_ThirdLevelSubdivisionTable = default(Nullable<int>);

}

this.SendPropertyChanged("ThirdLevelSubdivisionTable");

}

}

}

[global::System.Data.Linq.Mapping.AssociationAttribute(Name="UsersTable\_CollectedBasicParametersTable", Storage="\_UsersTable", ThisKey="FK\_UsersTable", OtherKey="UsersTableID", IsForeignKey=true)]

public UsersTable UsersTable

{

get

{

return this.\_UsersTable.Entity;

}

set

{

UsersTable previousValue = this.\_UsersTable.Entity;

if (((previousValue != value)

|| (this.\_UsersTable.HasLoadedOrAssignedValue == false)))

{

this.SendPropertyChanging();

if ((previousValue != null))

{

this.\_UsersTable.Entity = null;

previousValue.CollectedBasicParametersTable.Remove(this);

}

this.\_UsersTable.Entity = value;

if ((value != null))

{

value.CollectedBasicParametersTable.Add(this);

this.\_FK\_UsersTable = value.UsersTableID;

}

else

{

this.\_FK\_UsersTable = default(int);

}

this.SendPropertyChanged("UsersTable");

}

}

}

public event PropertyChangingEventHandler PropertyChanging;

public event PropertyChangedEventHandler PropertyChanged;

protected virtual void SendPropertyChanging()

{

if ((this.PropertyChanging != null))

{

this.PropertyChanging(this, emptyChangingEventArgs);

}

}

protected virtual void SendPropertyChanged(String propertyName)

{

if ((this.PropertyChanged != null))

{

this.PropertyChanged(this, new PropertyChangedEventArgs(propertyName));

}

}

}

[global::System.Data.Linq.Mapping.TableAttribute(Name="dbo.CollectedCalculatedParametrs")]

public partial class CollectedCalculatedParametrs : INotifyPropertyChanging, INotifyPropertyChanged

{

private static PropertyChangingEventArgs emptyChangingEventArgs = new PropertyChangingEventArgs(String.Empty);

private int \_CollectedCalculatedParametrsID;

private bool \_Active;

private System.Nullable<int> \_FK\_CalculatedParametrs;

private System.Nullable<double> \_CollectedValue;

private System.Nullable<System.DateTime> \_LastChangeDateTime;

private System.Nullable<System.DateTime> \_SavedDateTime;

private string \_UserIP;

private System.Nullable<int> \_FK\_UsersTable;

private System.Nullable<bool> \_Confirmed;

private System.Nullable<int> \_FK\_ReportArchiveTable;

private EntityRef<CalculatedParametrs> \_CalculatedParametrs;

private EntityRef<ReportArchiveTable> \_ReportArchiveTable;

private EntityRef<UsersTable> \_UsersTable;

#region Extensibility Method Definitions

partial void OnLoaded();

partial void OnValidate(System.Data.Linq.ChangeAction action);

partial void OnCreated();

partial void OnCollectedCalculatedParametrsIDChanging(int value);

partial void OnCollectedCalculatedParametrsIDChanged();

partial void OnActiveChanging(bool value);

partial void OnActiveChanged();

partial void OnFK\_CalculatedParametrsChanging(System.Nullable<int> value);

partial void OnFK\_CalculatedParametrsChanged();

partial void OnCollectedValueChanging(System.Nullable<double> value);

partial void OnCollectedValueChanged();

partial void OnLastChangeDateTimeChanging(System.Nullable<System.DateTime> value);

partial void OnLastChangeDateTimeChanged();

partial void OnSavedDateTimeChanging(System.Nullable<System.DateTime> value);

partial void OnSavedDateTimeChanged();

partial void OnUserIPChanging(string value);

partial void OnUserIPChanged();

partial void OnFK\_UsersTableChanging(System.Nullable<int> value);

partial void OnFK\_UsersTableChanged();

partial void OnConfirmedChanging(System.Nullable<bool> value);

partial void OnConfirmedChanged();

partial void OnFK\_ReportArchiveTableChanging(System.Nullable<int> value);

partial void OnFK\_ReportArchiveTableChanged();

#endregion

public CollectedCalculatedParametrs()

{

this.\_CalculatedParametrs = default(EntityRef<CalculatedParametrs>);

this.\_ReportArchiveTable = default(EntityRef<ReportArchiveTable>);

this.\_UsersTable = default(EntityRef<UsersTable>);

OnCreated();

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_CollectedCalculatedParametrsID", AutoSync=AutoSync.OnInsert, DbType="Int NOT NULL IDENTITY", IsPrimaryKey=true, IsDbGenerated=true)]

public int CollectedCalculatedParametrsID

{

get

{

return this.\_CollectedCalculatedParametrsID;

}

set

{

if ((this.\_CollectedCalculatedParametrsID != value))

{

this.OnCollectedCalculatedParametrsIDChanging(value);

this.SendPropertyChanging();

this.\_CollectedCalculatedParametrsID = value;

this.SendPropertyChanged("CollectedCalculatedParametrsID");

this.OnCollectedCalculatedParametrsIDChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_Active", DbType="Bit NOT NULL")]

public bool Active

{

get

{

return this.\_Active;

}

set

{

if ((this.\_Active != value))

{

this.OnActiveChanging(value);

this.SendPropertyChanging();

this.\_Active = value;

this.SendPropertyChanged("Active");

this.OnActiveChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_FK\_CalculatedParametrs", DbType="Int")]

public System.Nullable<int> FK\_CalculatedParametrs

{

get

{

return this.\_FK\_CalculatedParametrs;

}

set

{

if ((this.\_FK\_CalculatedParametrs != value))

{

if (this.\_CalculatedParametrs.HasLoadedOrAssignedValue)

{

throw new System.Data.Linq.ForeignKeyReferenceAlreadyHasValueException();

}

this.OnFK\_CalculatedParametrsChanging(value);

this.SendPropertyChanging();

this.\_FK\_CalculatedParametrs = value;

this.SendPropertyChanged("FK\_CalculatedParametrs");

this.OnFK\_CalculatedParametrsChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_CollectedValue", DbType="Float")]

public System.Nullable<double> CollectedValue

{

get

{

return this.\_CollectedValue;

}

set

{

if ((this.\_CollectedValue != value))

{

this.OnCollectedValueChanging(value);

this.SendPropertyChanging();

this.\_CollectedValue = value;

this.SendPropertyChanged("CollectedValue");

this.OnCollectedValueChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_LastChangeDateTime", DbType="DateTime")]

public System.Nullable<System.DateTime> LastChangeDateTime

{

get

{

return this.\_LastChangeDateTime;

}

set

{

if ((this.\_LastChangeDateTime != value))

{

this.OnLastChangeDateTimeChanging(value);

this.SendPropertyChanging();

this.\_LastChangeDateTime = value;

this.SendPropertyChanged("LastChangeDateTime");

this.OnLastChangeDateTimeChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_SavedDateTime", DbType="DateTime")]

public System.Nullable<System.DateTime> SavedDateTime

{

get

{

return this.\_SavedDateTime;

}

set

{

if ((this.\_SavedDateTime != value))

{

this.OnSavedDateTimeChanging(value);

this.SendPropertyChanging();

this.\_SavedDateTime = value;

this.SendPropertyChanged("SavedDateTime");

this.OnSavedDateTimeChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_UserIP", DbType="VarChar(15)")]

public string UserIP

{

get

{

return this.\_UserIP;

}

set

{

if ((this.\_UserIP != value))

{

this.OnUserIPChanging(value);

this.SendPropertyChanging();

this.\_UserIP = value;

this.SendPropertyChanged("UserIP");

this.OnUserIPChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_FK\_UsersTable", DbType="Int")]

public System.Nullable<int> FK\_UsersTable

{

get

{

return this.\_FK\_UsersTable;

}

set

{

if ((this.\_FK\_UsersTable != value))

{

if (this.\_UsersTable.HasLoadedOrAssignedValue)

{

throw new System.Data.Linq.ForeignKeyReferenceAlreadyHasValueException();

}

this.OnFK\_UsersTableChanging(value);

this.SendPropertyChanging();

this.\_FK\_UsersTable = value;

this.SendPropertyChanged("FK\_UsersTable");

this.OnFK\_UsersTableChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_Confirmed", DbType="Bit")]

public System.Nullable<bool> Confirmed

{

get

{

return this.\_Confirmed;

}

set

{

if ((this.\_Confirmed != value))

{

this.OnConfirmedChanging(value);

this.SendPropertyChanging();

this.\_Confirmed = value;

this.SendPropertyChanged("Confirmed");

this.OnConfirmedChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_FK\_ReportArchiveTable", DbType="Int")]

public System.Nullable<int> FK\_ReportArchiveTable

{

get

{

return this.\_FK\_ReportArchiveTable;

}

set

{

if ((this.\_FK\_ReportArchiveTable != value))

{

if (this.\_ReportArchiveTable.HasLoadedOrAssignedValue)

{

throw new System.Data.Linq.ForeignKeyReferenceAlreadyHasValueException();

}

this.OnFK\_ReportArchiveTableChanging(value);

this.SendPropertyChanging();

this.\_FK\_ReportArchiveTable = value;

this.SendPropertyChanged("FK\_ReportArchiveTable");

this.OnFK\_ReportArchiveTableChanged();

}

}

}

[global::System.Data.Linq.Mapping.AssociationAttribute(Name="CalculatedParametrs\_CollectedCalculatedParametrs", Storage="\_CalculatedParametrs", ThisKey="FK\_CalculatedParametrs", OtherKey="CalculatedParametrsID", IsForeignKey=true)]

public CalculatedParametrs CalculatedParametrs

{

get

{

return this.\_CalculatedParametrs.Entity;

}

set

{

CalculatedParametrs previousValue = this.\_CalculatedParametrs.Entity;

if (((previousValue != value)

|| (this.\_CalculatedParametrs.HasLoadedOrAssignedValue == false)))

{

this.SendPropertyChanging();

if ((previousValue != null))

{

this.\_CalculatedParametrs.Entity = null;

previousValue.CollectedCalculatedParametrs.Remove(this);

}

this.\_CalculatedParametrs.Entity = value;

if ((value != null))

{

value.CollectedCalculatedParametrs.Add(this);

this.\_FK\_CalculatedParametrs = value.CalculatedParametrsID;

}

else

{

this.\_FK\_CalculatedParametrs = default(Nullable<int>);

}

this.SendPropertyChanged("CalculatedParametrs");

}

}

}

[global::System.Data.Linq.Mapping.AssociationAttribute(Name="ReportArchiveTable\_CollectedCalculatedParametrs", Storage="\_ReportArchiveTable", ThisKey="FK\_ReportArchiveTable", OtherKey="ReportArchiveTableID", IsForeignKey=true)]

public ReportArchiveTable ReportArchiveTable

{

get

{

return this.\_ReportArchiveTable.Entity;

}

set

{

ReportArchiveTable previousValue = this.\_ReportArchiveTable.Entity;

if (((previousValue != value)

|| (this.\_ReportArchiveTable.HasLoadedOrAssignedValue == false)))

{

this.SendPropertyChanging();

if ((previousValue != null))

{

this.\_ReportArchiveTable.Entity = null;

previousValue.CollectedCalculatedParametrs.Remove(this);

}

this.\_ReportArchiveTable.Entity = value;

if ((value != null))

{

value.CollectedCalculatedParametrs.Add(this);

this.\_FK\_ReportArchiveTable = value.ReportArchiveTableID;

}

else

{

this.\_FK\_ReportArchiveTable = default(Nullable<int>);

}

this.SendPropertyChanged("ReportArchiveTable");

}

}

}

[global::System.Data.Linq.Mapping.AssociationAttribute(Name="UsersTable\_CollectedCalculatedParametrs", Storage="\_UsersTable", ThisKey="FK\_UsersTable", OtherKey="UsersTableID", IsForeignKey=true)]

public UsersTable UsersTable

{

get

{

return this.\_UsersTable.Entity;

}

set

{

UsersTable previousValue = this.\_UsersTable.Entity;

if (((previousValue != value)

|| (this.\_UsersTable.HasLoadedOrAssignedValue == false)))

{

this.SendPropertyChanging();

if ((previousValue != null))

{

this.\_UsersTable.Entity = null;

previousValue.CollectedCalculatedParametrs.Remove(this);

}

this.\_UsersTable.Entity = value;

if ((value != null))

{

value.CollectedCalculatedParametrs.Add(this);

this.\_FK\_UsersTable = value.UsersTableID;

}

else

{

this.\_FK\_UsersTable = default(Nullable<int>);

}

this.SendPropertyChanged("UsersTable");

}

}

}

public event PropertyChangingEventHandler PropertyChanging;

public event PropertyChangedEventHandler PropertyChanged;

protected virtual void SendPropertyChanging()

{

if ((this.PropertyChanging != null))

{

this.PropertyChanging(this, emptyChangingEventArgs);

}

}

protected virtual void SendPropertyChanged(String propertyName)

{

if ((this.PropertyChanged != null))

{

this.PropertyChanged(this, new PropertyChangedEventArgs(propertyName));

}

}

}

[global::System.Data.Linq.Mapping.TableAttribute(Name="dbo.CollectedIndicatorsForR")]

public partial class CollectedIndicatorsForR : INotifyPropertyChanging, INotifyPropertyChanged

{

private static PropertyChangingEventArgs emptyChangingEventArgs = new PropertyChangingEventArgs(String.Empty);

private int \_CollectedIndicatorsForRID;

private System.Nullable<double> \_Value;

private System.Nullable<bool> \_Active;

private System.Nullable<int> \_FK\_IndicatorsTable;

private System.Nullable<int> \_FK\_FirstLevelSubdivisionTable;

private System.Nullable<int> \_FK\_SecondLevelSubdivisionTable;

private System.Nullable<int> \_FK\_ThirdLevelSubdivisionTable;

private System.Nullable<int> \_FK\_FourthLelevlSubdivisionTable;

private System.Nullable<int> \_FK\_FifthLevelSubdivisionTable;

private System.Nullable<int> \_FK\_ReportArchiveTable;

private System.Nullable<System.DateTime> \_CreatedDateTime;

private EntityRef<FifthLevelSubdivisionTable> \_FifthLevelSubdivisionTable;

private EntityRef<FirstLevelSubdivisionTable> \_FirstLevelSubdivisionTable;

private EntityRef<FourthLevelSubdivisionTable> \_FourthLevelSubdivisionTable;

private EntityRef<ReportArchiveTable> \_ReportArchiveTable;

private EntityRef<SecondLevelSubdivisionTable> \_SecondLevelSubdivisionTable;

private EntityRef<ThirdLevelSubdivisionTable> \_ThirdLevelSubdivisionTable;

private EntityRef<IndicatorsTable> \_IndicatorsTable;

#region Extensibility Method Definitions

partial void OnLoaded();

partial void OnValidate(System.Data.Linq.ChangeAction action);

partial void OnCreated();

partial void OnCollectedIndicatorsForRIDChanging(int value);

partial void OnCollectedIndicatorsForRIDChanged();

partial void OnValueChanging(System.Nullable<double> value);

partial void OnValueChanged();

partial void OnActiveChanging(System.Nullable<bool> value);

partial void OnActiveChanged();

partial void OnFK\_IndicatorsTableChanging(System.Nullable<int> value);

partial void OnFK\_IndicatorsTableChanged();

partial void OnFK\_FirstLevelSubdivisionTableChanging(System.Nullable<int> value);

partial void OnFK\_FirstLevelSubdivisionTableChanged();

partial void OnFK\_SecondLevelSubdivisionTableChanging(System.Nullable<int> value);

partial void OnFK\_SecondLevelSubdivisionTableChanged();

partial void OnFK\_ThirdLevelSubdivisionTableChanging(System.Nullable<int> value);

partial void OnFK\_ThirdLevelSubdivisionTableChanged();

partial void OnFK\_FourthLelevlSubdivisionTableChanging(System.Nullable<int> value);

partial void OnFK\_FourthLelevlSubdivisionTableChanged();

partial void OnFK\_FifthLevelSubdivisionTableChanging(System.Nullable<int> value);

partial void OnFK\_FifthLevelSubdivisionTableChanged();

partial void OnFK\_ReportArchiveTableChanging(System.Nullable<int> value);

partial void OnFK\_ReportArchiveTableChanged();

partial void OnCreatedDateTimeChanging(System.Nullable<System.DateTime> value);

partial void OnCreatedDateTimeChanged();

#endregion

public CollectedIndicatorsForR()

{

this.\_FifthLevelSubdivisionTable = default(EntityRef<FifthLevelSubdivisionTable>);

this.\_FirstLevelSubdivisionTable = default(EntityRef<FirstLevelSubdivisionTable>);

this.\_FourthLevelSubdivisionTable = default(EntityRef<FourthLevelSubdivisionTable>);

this.\_ReportArchiveTable = default(EntityRef<ReportArchiveTable>);

this.\_SecondLevelSubdivisionTable = default(EntityRef<SecondLevelSubdivisionTable>);

this.\_ThirdLevelSubdivisionTable = default(EntityRef<ThirdLevelSubdivisionTable>);

this.\_IndicatorsTable = default(EntityRef<IndicatorsTable>);

OnCreated();

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_CollectedIndicatorsForRID", AutoSync=AutoSync.OnInsert, DbType="Int NOT NULL IDENTITY", IsPrimaryKey=true, IsDbGenerated=true)]

public int CollectedIndicatorsForRID

{

get

{

return this.\_CollectedIndicatorsForRID;

}

set

{

if ((this.\_CollectedIndicatorsForRID != value))

{

this.OnCollectedIndicatorsForRIDChanging(value);

this.SendPropertyChanging();

this.\_CollectedIndicatorsForRID = value;

this.SendPropertyChanged("CollectedIndicatorsForRID");

this.OnCollectedIndicatorsForRIDChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_Value", DbType="Float")]

public System.Nullable<double> Value

{

get

{

return this.\_Value;

}

set

{

if ((this.\_Value != value))

{

this.OnValueChanging(value);

this.SendPropertyChanging();

this.\_Value = value;

this.SendPropertyChanged("Value");

this.OnValueChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_Active", DbType="Bit")]

public System.Nullable<bool> Active

{

get

{

return this.\_Active;

}

set

{

if ((this.\_Active != value))

{

this.OnActiveChanging(value);

this.SendPropertyChanging();

this.\_Active = value;

this.SendPropertyChanged("Active");

this.OnActiveChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_FK\_IndicatorsTable", DbType="Int")]

public System.Nullable<int> FK\_IndicatorsTable

{

get

{

return this.\_FK\_IndicatorsTable;

}

set

{

if ((this.\_FK\_IndicatorsTable != value))

{

if (this.\_IndicatorsTable.HasLoadedOrAssignedValue)

{

throw new System.Data.Linq.ForeignKeyReferenceAlreadyHasValueException();

}

this.OnFK\_IndicatorsTableChanging(value);

this.SendPropertyChanging();

this.\_FK\_IndicatorsTable = value;

this.SendPropertyChanged("FK\_IndicatorsTable");

this.OnFK\_IndicatorsTableChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_FK\_FirstLevelSubdivisionTable", DbType="Int")]

public System.Nullable<int> FK\_FirstLevelSubdivisionTable

{

get

{

return this.\_FK\_FirstLevelSubdivisionTable;

}

set

{

if ((this.\_FK\_FirstLevelSubdivisionTable != value))

{

if (this.\_FirstLevelSubdivisionTable.HasLoadedOrAssignedValue)

{

throw new System.Data.Linq.ForeignKeyReferenceAlreadyHasValueException();

}

this.OnFK\_FirstLevelSubdivisionTableChanging(value);

this.SendPropertyChanging();

this.\_FK\_FirstLevelSubdivisionTable = value;

this.SendPropertyChanged("FK\_FirstLevelSubdivisionTable");

this.OnFK\_FirstLevelSubdivisionTableChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_FK\_SecondLevelSubdivisionTable", DbType="Int")]

public System.Nullable<int> FK\_SecondLevelSubdivisionTable

{

get

{

return this.\_FK\_SecondLevelSubdivisionTable;

}

set

{

if ((this.\_FK\_SecondLevelSubdivisionTable != value))

{

if (this.\_SecondLevelSubdivisionTable.HasLoadedOrAssignedValue)

{

throw new System.Data.Linq.ForeignKeyReferenceAlreadyHasValueException();

}

this.OnFK\_SecondLevelSubdivisionTableChanging(value);

this.SendPropertyChanging();

this.\_FK\_SecondLevelSubdivisionTable = value;

this.SendPropertyChanged("FK\_SecondLevelSubdivisionTable");

this.OnFK\_SecondLevelSubdivisionTableChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_FK\_ThirdLevelSubdivisionTable", DbType="Int")]

public System.Nullable<int> FK\_ThirdLevelSubdivisionTable

{

get

{

return this.\_FK\_ThirdLevelSubdivisionTable;

}

set

{

if ((this.\_FK\_ThirdLevelSubdivisionTable != value))

{

if (this.\_ThirdLevelSubdivisionTable.HasLoadedOrAssignedValue)

{

throw new System.Data.Linq.ForeignKeyReferenceAlreadyHasValueException();

}

this.OnFK\_ThirdLevelSubdivisionTableChanging(value);

this.SendPropertyChanging();

this.\_FK\_ThirdLevelSubdivisionTable = value;

this.SendPropertyChanged("FK\_ThirdLevelSubdivisionTable");

this.OnFK\_ThirdLevelSubdivisionTableChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_FK\_FourthLelevlSubdivisionTable", DbType="Int")]

public System.Nullable<int> FK\_FourthLelevlSubdivisionTable

{

get

{

return this.\_FK\_FourthLelevlSubdivisionTable;

}

set

{

if ((this.\_FK\_FourthLelevlSubdivisionTable != value))

{

if (this.\_FourthLevelSubdivisionTable.HasLoadedOrAssignedValue)

{

throw new System.Data.Linq.ForeignKeyReferenceAlreadyHasValueException();

}

this.OnFK\_FourthLelevlSubdivisionTableChanging(value);

this.SendPropertyChanging();

this.\_FK\_FourthLelevlSubdivisionTable = value;

this.SendPropertyChanged("FK\_FourthLelevlSubdivisionTable");

this.OnFK\_FourthLelevlSubdivisionTableChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_FK\_FifthLevelSubdivisionTable", DbType="Int")]

public System.Nullable<int> FK\_FifthLevelSubdivisionTable

{

get

{

return this.\_FK\_FifthLevelSubdivisionTable;

}

set

{

if ((this.\_FK\_FifthLevelSubdivisionTable != value))

{

if (this.\_FifthLevelSubdivisionTable.HasLoadedOrAssignedValue)

{

throw new System.Data.Linq.ForeignKeyReferenceAlreadyHasValueException();

}

this.OnFK\_FifthLevelSubdivisionTableChanging(value);

this.SendPropertyChanging();

this.\_FK\_FifthLevelSubdivisionTable = value;

this.SendPropertyChanged("FK\_FifthLevelSubdivisionTable");

this.OnFK\_FifthLevelSubdivisionTableChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_FK\_ReportArchiveTable", DbType="Int")]

public System.Nullable<int> FK\_ReportArchiveTable

{

get

{

return this.\_FK\_ReportArchiveTable;

}

set

{

if ((this.\_FK\_ReportArchiveTable != value))

{

if (this.\_ReportArchiveTable.HasLoadedOrAssignedValue)

{

throw new System.Data.Linq.ForeignKeyReferenceAlreadyHasValueException();

}

this.OnFK\_ReportArchiveTableChanging(value);

this.SendPropertyChanging();

this.\_FK\_ReportArchiveTable = value;

this.SendPropertyChanged("FK\_ReportArchiveTable");

this.OnFK\_ReportArchiveTableChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_CreatedDateTime", DbType="DateTime")]

public System.Nullable<System.DateTime> CreatedDateTime

{

get

{

return this.\_CreatedDateTime;

}

set

{

if ((this.\_CreatedDateTime != value))

{

this.OnCreatedDateTimeChanging(value);

this.SendPropertyChanging();

this.\_CreatedDateTime = value;

this.SendPropertyChanged("CreatedDateTime");

this.OnCreatedDateTimeChanged();

}

}

}

[global::System.Data.Linq.Mapping.AssociationAttribute(Name="FifthLevelSubdivisionTable\_CollectedIndicatorsForR", Storage="\_FifthLevelSubdivisionTable", ThisKey="FK\_FifthLevelSubdivisionTable", OtherKey="FifthLevelSubdivisionTableID", IsForeignKey=true)]

public FifthLevelSubdivisionTable FifthLevelSubdivisionTable

{

get

{

return this.\_FifthLevelSubdivisionTable.Entity;

}

set

{

FifthLevelSubdivisionTable previousValue = this.\_FifthLevelSubdivisionTable.Entity;

if (((previousValue != value)

|| (this.\_FifthLevelSubdivisionTable.HasLoadedOrAssignedValue == false)))

{

this.SendPropertyChanging();

if ((previousValue != null))

{

this.\_FifthLevelSubdivisionTable.Entity = null;

previousValue.CollectedIndicatorsForR.Remove(this);

}

this.\_FifthLevelSubdivisionTable.Entity = value;

if ((value != null))

{

value.CollectedIndicatorsForR.Add(this);

this.\_FK\_FifthLevelSubdivisionTable = value.FifthLevelSubdivisionTableID;

}

else

{

this.\_FK\_FifthLevelSubdivisionTable = default(Nullable<int>);

}

this.SendPropertyChanged("FifthLevelSubdivisionTable");

}

}

}

[global::System.Data.Linq.Mapping.AssociationAttribute(Name="FirstLevelSubdivisionTable\_CollectedIndicatorsForR", Storage="\_FirstLevelSubdivisionTable", ThisKey="FK\_FirstLevelSubdivisionTable", OtherKey="FirstLevelSubdivisionTableID", IsForeignKey=true)]

public FirstLevelSubdivisionTable FirstLevelSubdivisionTable

{

get

{

return this.\_FirstLevelSubdivisionTable.Entity;

}

set

{

FirstLevelSubdivisionTable previousValue = this.\_FirstLevelSubdivisionTable.Entity;

if (((previousValue != value)

|| (this.\_FirstLevelSubdivisionTable.HasLoadedOrAssignedValue == false)))

{

this.SendPropertyChanging();

if ((previousValue != null))

{

this.\_FirstLevelSubdivisionTable.Entity = null;

previousValue.CollectedIndicatorsForR.Remove(this);

}

this.\_FirstLevelSubdivisionTable.Entity = value;

if ((value != null))

{

value.CollectedIndicatorsForR.Add(this);

this.\_FK\_FirstLevelSubdivisionTable = value.FirstLevelSubdivisionTableID;

}

else

{

this.\_FK\_FirstLevelSubdivisionTable = default(Nullable<int>);

}

this.SendPropertyChanged("FirstLevelSubdivisionTable");

}

}

}

[global::System.Data.Linq.Mapping.AssociationAttribute(Name="FourthLevelSubdivisionTable\_CollectedIndicatorsForR", Storage="\_FourthLevelSubdivisionTable", ThisKey="FK\_FourthLelevlSubdivisionTable", OtherKey="FourthLevelSubdivisionTableID", IsForeignKey=true)]

public FourthLevelSubdivisionTable FourthLevelSubdivisionTable

{

get

{

return this.\_FourthLevelSubdivisionTable.Entity;

}

set

{

FourthLevelSubdivisionTable previousValue = this.\_FourthLevelSubdivisionTable.Entity;

if (((previousValue != value)

|| (this.\_FourthLevelSubdivisionTable.HasLoadedOrAssignedValue == false)))

{

this.SendPropertyChanging();

if ((previousValue != null))

{

this.\_FourthLevelSubdivisionTable.Entity = null;

previousValue.CollectedIndicatorsForR.Remove(this);

}

this.\_FourthLevelSubdivisionTable.Entity = value;

if ((value != null))

{

value.CollectedIndicatorsForR.Add(this);

this.\_FK\_FourthLelevlSubdivisionTable = value.FourthLevelSubdivisionTableID;

}

else

{

this.\_FK\_FourthLelevlSubdivisionTable = default(Nullable<int>);

}

this.SendPropertyChanged("FourthLevelSubdivisionTable");

}

}

}

[global::System.Data.Linq.Mapping.AssociationAttribute(Name="ReportArchiveTable\_CollectedIndicatorsForR", Storage="\_ReportArchiveTable", ThisKey="FK\_ReportArchiveTable", OtherKey="ReportArchiveTableID", IsForeignKey=true)]

public ReportArchiveTable ReportArchiveTable

{

get

{

return this.\_ReportArchiveTable.Entity;

}

set

{

ReportArchiveTable previousValue = this.\_ReportArchiveTable.Entity;

if (((previousValue != value)

|| (this.\_ReportArchiveTable.HasLoadedOrAssignedValue == false)))

{

this.SendPropertyChanging();

if ((previousValue != null))

{

this.\_ReportArchiveTable.Entity = null;

previousValue.CollectedIndicatorsForR.Remove(this);

}

this.\_ReportArchiveTable.Entity = value;

if ((value != null))

{

value.CollectedIndicatorsForR.Add(this);

this.\_FK\_ReportArchiveTable = value.ReportArchiveTableID;

}

else

{

this.\_FK\_ReportArchiveTable = default(Nullable<int>);

}

this.SendPropertyChanged("ReportArchiveTable");

}

}

}

[global::System.Data.Linq.Mapping.AssociationAttribute(Name="SecondLevelSubdivisionTable\_CollectedIndicatorsForR", Storage="\_SecondLevelSubdivisionTable", ThisKey="FK\_SecondLevelSubdivisionTable", OtherKey="SecondLevelSubdivisionTableID", IsForeignKey=true)]

public SecondLevelSubdivisionTable SecondLevelSubdivisionTable

{

get

{

return this.\_SecondLevelSubdivisionTable.Entity;

}

set

{

SecondLevelSubdivisionTable previousValue = this.\_SecondLevelSubdivisionTable.Entity;

if (((previousValue != value)

|| (this.\_SecondLevelSubdivisionTable.HasLoadedOrAssignedValue == false)))

{

this.SendPropertyChanging();

if ((previousValue != null))

{

this.\_SecondLevelSubdivisionTable.Entity = null;

previousValue.CollectedIndicatorsForR.Remove(this);

}

this.\_SecondLevelSubdivisionTable.Entity = value;

if ((value != null))

{

value.CollectedIndicatorsForR.Add(this);

this.\_FK\_SecondLevelSubdivisionTable = value.SecondLevelSubdivisionTableID;

}

else

{

this.\_FK\_SecondLevelSubdivisionTable = default(Nullable<int>);

}

this.SendPropertyChanged("SecondLevelSubdivisionTable");

}

}

}

[global::System.Data.Linq.Mapping.AssociationAttribute(Name="ThirdLevelSubdivisionTable\_CollectedIndicatorsForR", Storage="\_ThirdLevelSubdivisionTable", ThisKey="FK\_ThirdLevelSubdivisionTable", OtherKey="ThirdLevelSubdivisionTableID", IsForeignKey=true)]

public ThirdLevelSubdivisionTable ThirdLevelSubdivisionTable

{

get

{

return this.\_ThirdLevelSubdivisionTable.Entity;

}

set

{

ThirdLevelSubdivisionTable previousValue = this.\_ThirdLevelSubdivisionTable.Entity;

if (((previousValue != value)

|| (this.\_ThirdLevelSubdivisionTable.HasLoadedOrAssignedValue == false)))

{

this.SendPropertyChanging();

if ((previousValue != null))

{

this.\_ThirdLevelSubdivisionTable.Entity = null;

previousValue.CollectedIndicatorsForR.Remove(this);

}

this.\_ThirdLevelSubdivisionTable.Entity = value;

if ((value != null))

{

value.CollectedIndicatorsForR.Add(this);

this.\_FK\_ThirdLevelSubdivisionTable = value.ThirdLevelSubdivisionTableID;

}

else

{

this.\_FK\_ThirdLevelSubdivisionTable = default(Nullable<int>);

}

this.SendPropertyChanged("ThirdLevelSubdivisionTable");

}

}

}

[global::System.Data.Linq.Mapping.AssociationAttribute(Name="IndicatorsTable\_CollectedIndicatorsForR", Storage="\_IndicatorsTable", ThisKey="FK\_IndicatorsTable", OtherKey="IndicatorsTableID", IsForeignKey=true)]

public IndicatorsTable IndicatorsTable

{

get

{

return this.\_IndicatorsTable.Entity;

}

set

{

IndicatorsTable previousValue = this.\_IndicatorsTable.Entity;

if (((previousValue != value)

|| (this.\_IndicatorsTable.HasLoadedOrAssignedValue == false)))

{

this.SendPropertyChanging();

if ((previousValue != null))

{

this.\_IndicatorsTable.Entity = null;

previousValue.CollectedIndicatorsForR.Remove(this);

}

this.\_IndicatorsTable.Entity = value;

if ((value != null))

{

value.CollectedIndicatorsForR.Add(this);

this.\_FK\_IndicatorsTable = value.IndicatorsTableID;

}

else

{

this.\_FK\_IndicatorsTable = default(Nullable<int>);

}

this.SendPropertyChanged("IndicatorsTable");

}

}

}

public event PropertyChangingEventHandler PropertyChanging;

public event PropertyChangedEventHandler PropertyChanged;

protected virtual void SendPropertyChanging()

{

if ((this.PropertyChanging != null))

{

this.PropertyChanging(this, emptyChangingEventArgs);

}

}

protected virtual void SendPropertyChanged(String propertyName)

{

if ((this.PropertyChanged != null))

{

this.PropertyChanged(this, new PropertyChangedEventArgs(propertyName));

}

}

}

[global::System.Data.Linq.Mapping.TableAttribute(Name="dbo.CollectedIndocators")]

public partial class CollectedIndocators : INotifyPropertyChanging, INotifyPropertyChanged

{

private static PropertyChangingEventArgs emptyChangingEventArgs = new PropertyChangingEventArgs(String.Empty);

private int \_CollectedIndocatorsID;

private bool \_Active;

private System.Nullable<int> \_FK\_Indicators;

private System.Nullable<double> \_CollectedValue;

private System.Nullable<System.DateTime> \_LastChangeDateTime;

private System.Nullable<System.DateTime> \_SavedDateTime;

private string \_UserIP;

private System.Nullable<int> \_FK\_UsersTable;

private System.Nullable<bool> \_Confirmed;

private System.Nullable<int> \_FK\_ReportArchiveTable;

private EntityRef<ReportArchiveTable> \_ReportArchiveTable;

private EntityRef<UsersTable> \_UsersTable;

private EntityRef<IndicatorsTable> \_IndicatorsTable;

#region Extensibility Method Definitions

partial void OnLoaded();

partial void OnValidate(System.Data.Linq.ChangeAction action);

partial void OnCreated();

partial void OnCollectedIndocatorsIDChanging(int value);

partial void OnCollectedIndocatorsIDChanged();

partial void OnActiveChanging(bool value);

partial void OnActiveChanged();

partial void OnFK\_IndicatorsChanging(System.Nullable<int> value);

partial void OnFK\_IndicatorsChanged();

partial void OnCollectedValueChanging(System.Nullable<double> value);

partial void OnCollectedValueChanged();

partial void OnLastChangeDateTimeChanging(System.Nullable<System.DateTime> value);

partial void OnLastChangeDateTimeChanged();

partial void OnSavedDateTimeChanging(System.Nullable<System.DateTime> value);

partial void OnSavedDateTimeChanged();

partial void OnUserIPChanging(string value);

partial void OnUserIPChanged();

partial void OnFK\_UsersTableChanging(System.Nullable<int> value);

partial void OnFK\_UsersTableChanged();

partial void OnConfirmedChanging(System.Nullable<bool> value);

partial void OnConfirmedChanged();

partial void OnFK\_ReportArchiveTableChanging(System.Nullable<int> value);

partial void OnFK\_ReportArchiveTableChanged();

#endregion

public CollectedIndocators()

{

this.\_ReportArchiveTable = default(EntityRef<ReportArchiveTable>);

this.\_UsersTable = default(EntityRef<UsersTable>);

this.\_IndicatorsTable = default(EntityRef<IndicatorsTable>);

OnCreated();

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_CollectedIndocatorsID", AutoSync=AutoSync.OnInsert, DbType="Int NOT NULL IDENTITY", IsPrimaryKey=true, IsDbGenerated=true)]

public int CollectedIndocatorsID

{

get

{

return this.\_CollectedIndocatorsID;

}

set

{

if ((this.\_CollectedIndocatorsID != value))

{

this.OnCollectedIndocatorsIDChanging(value);

this.SendPropertyChanging();

this.\_CollectedIndocatorsID = value;

this.SendPropertyChanged("CollectedIndocatorsID");

this.OnCollectedIndocatorsIDChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_Active", DbType="Bit NOT NULL")]

public bool Active

{

get

{

return this.\_Active;

}

set

{

if ((this.\_Active != value))

{

this.OnActiveChanging(value);

this.SendPropertyChanging();

this.\_Active = value;

this.SendPropertyChanged("Active");

this.OnActiveChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_FK\_Indicators", DbType="Int")]

public System.Nullable<int> FK\_Indicators

{

get

{

return this.\_FK\_Indicators;

}

set

{

if ((this.\_FK\_Indicators != value))

{

if (this.\_IndicatorsTable.HasLoadedOrAssignedValue)

{

throw new System.Data.Linq.ForeignKeyReferenceAlreadyHasValueException();

}

this.OnFK\_IndicatorsChanging(value);

this.SendPropertyChanging();

this.\_FK\_Indicators = value;

this.SendPropertyChanged("FK\_Indicators");

this.OnFK\_IndicatorsChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_CollectedValue", DbType="Float")]

public System.Nullable<double> CollectedValue

{

get

{

return this.\_CollectedValue;

}

set

{

if ((this.\_CollectedValue != value))

{

this.OnCollectedValueChanging(value);

this.SendPropertyChanging();

this.\_CollectedValue = value;

this.SendPropertyChanged("CollectedValue");

this.OnCollectedValueChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_LastChangeDateTime", DbType="DateTime")]

public System.Nullable<System.DateTime> LastChangeDateTime

{

get

{

return this.\_LastChangeDateTime;

}

set

{

if ((this.\_LastChangeDateTime != value))

{

this.OnLastChangeDateTimeChanging(value);

this.SendPropertyChanging();

this.\_LastChangeDateTime = value;

this.SendPropertyChanged("LastChangeDateTime");

this.OnLastChangeDateTimeChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_SavedDateTime", DbType="DateTime")]

public System.Nullable<System.DateTime> SavedDateTime

{

get

{

return this.\_SavedDateTime;

}

set

{

if ((this.\_SavedDateTime != value))

{

this.OnSavedDateTimeChanging(value);

this.SendPropertyChanging();

this.\_SavedDateTime = value;

this.SendPropertyChanged("SavedDateTime");

this.OnSavedDateTimeChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_UserIP", DbType="VarChar(15)")]

public string UserIP

{

get

{

return this.\_UserIP;

}

set

{

if ((this.\_UserIP != value))

{

this.OnUserIPChanging(value);

this.SendPropertyChanging();

this.\_UserIP = value;

this.SendPropertyChanged("UserIP");

this.OnUserIPChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_FK\_UsersTable", DbType="Int")]

public System.Nullable<int> FK\_UsersTable

{

get

{

return this.\_FK\_UsersTable;

}

set

{

if ((this.\_FK\_UsersTable != value))

{

if (this.\_UsersTable.HasLoadedOrAssignedValue)

{

throw new System.Data.Linq.ForeignKeyReferenceAlreadyHasValueException();

}

this.OnFK\_UsersTableChanging(value);

this.SendPropertyChanging();

this.\_FK\_UsersTable = value;

this.SendPropertyChanged("FK\_UsersTable");

this.OnFK\_UsersTableChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_Confirmed", DbType="Bit")]

public System.Nullable<bool> Confirmed

{

get

{

return this.\_Confirmed;

}

set

{

if ((this.\_Confirmed != value))

{

this.OnConfirmedChanging(value);

this.SendPropertyChanging();

this.\_Confirmed = value;

this.SendPropertyChanged("Confirmed");

this.OnConfirmedChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_FK\_ReportArchiveTable", DbType="Int")]

public System.Nullable<int> FK\_ReportArchiveTable

{

get

{

return this.\_FK\_ReportArchiveTable;

}

set

{

if ((this.\_FK\_ReportArchiveTable != value))

{

if (this.\_ReportArchiveTable.HasLoadedOrAssignedValue)

{

throw new System.Data.Linq.ForeignKeyReferenceAlreadyHasValueException();

}

this.OnFK\_ReportArchiveTableChanging(value);

this.SendPropertyChanging();

this.\_FK\_ReportArchiveTable = value;

this.SendPropertyChanged("FK\_ReportArchiveTable");

this.OnFK\_ReportArchiveTableChanged();

}

}

}

[global::System.Data.Linq.Mapping.AssociationAttribute(Name="ReportArchiveTable\_CollectedIndocators", Storage="\_ReportArchiveTable", ThisKey="FK\_ReportArchiveTable", OtherKey="ReportArchiveTableID", IsForeignKey=true)]

public ReportArchiveTable ReportArchiveTable

{

get

{

return this.\_ReportArchiveTable.Entity;

}

set

{

ReportArchiveTable previousValue = this.\_ReportArchiveTable.Entity;

if (((previousValue != value)

|| (this.\_ReportArchiveTable.HasLoadedOrAssignedValue == false)))

{

this.SendPropertyChanging();

if ((previousValue != null))

{

this.\_ReportArchiveTable.Entity = null;

previousValue.CollectedIndocators.Remove(this);

}

this.\_ReportArchiveTable.Entity = value;

if ((value != null))

{

value.CollectedIndocators.Add(this);

this.\_FK\_ReportArchiveTable = value.ReportArchiveTableID;

}

else

{

this.\_FK\_ReportArchiveTable = default(Nullable<int>);

}

this.SendPropertyChanged("ReportArchiveTable");

}

}

}

[global::System.Data.Linq.Mapping.AssociationAttribute(Name="UsersTable\_CollectedIndocators", Storage="\_UsersTable", ThisKey="FK\_UsersTable", OtherKey="UsersTableID", IsForeignKey=true)]

public UsersTable UsersTable

{

get

{

return this.\_UsersTable.Entity;

}

set

{

UsersTable previousValue = this.\_UsersTable.Entity;

if (((previousValue != value)

|| (this.\_UsersTable.HasLoadedOrAssignedValue == false)))

{

this.SendPropertyChanging();

if ((previousValue != null))

{

this.\_UsersTable.Entity = null;

previousValue.CollectedIndocators.Remove(this);

}

this.\_UsersTable.Entity = value;

if ((value != null))

{

value.CollectedIndocators.Add(this);

this.\_FK\_UsersTable = value.UsersTableID;

}

else

{

this.\_FK\_UsersTable = default(Nullable<int>);

}

this.SendPropertyChanged("UsersTable");

}

}

}

[global::System.Data.Linq.Mapping.AssociationAttribute(Name="IndicatorsTable\_CollectedIndocators", Storage="\_IndicatorsTable", ThisKey="FK\_Indicators", OtherKey="IndicatorsTableID", IsForeignKey=true)]

public IndicatorsTable IndicatorsTable

{

get

{

return this.\_IndicatorsTable.Entity;

}

set

{

IndicatorsTable previousValue = this.\_IndicatorsTable.Entity;

if (((previousValue != value)

|| (this.\_IndicatorsTable.HasLoadedOrAssignedValue == false)))

{

this.SendPropertyChanging();

if ((previousValue != null))

{

this.\_IndicatorsTable.Entity = null;

previousValue.CollectedIndocators.Remove(this);

}

this.\_IndicatorsTable.Entity = value;

if ((value != null))

{

value.CollectedIndocators.Add(this);

this.\_FK\_Indicators = value.IndicatorsTableID;

}

else

{

this.\_FK\_Indicators = default(Nullable<int>);

}

this.SendPropertyChanged("IndicatorsTable");

}

}

}

public event PropertyChangingEventHandler PropertyChanging;

public event PropertyChangedEventHandler PropertyChanged;

protected virtual void SendPropertyChanging()

{

if ((this.PropertyChanging != null))

{

this.PropertyChanging(this, emptyChangingEventArgs);

}

}

protected virtual void SendPropertyChanged(String propertyName)

{

if ((this.PropertyChanged != null))

{

this.PropertyChanged(this, new PropertyChangedEventArgs(propertyName));

}

}

}

[global::System.Data.Linq.Mapping.TableAttribute(Name="dbo.CommetntForBasicInReport")]

public partial class CommetntForBasicInReport : INotifyPropertyChanging, INotifyPropertyChanged

{

private static PropertyChangingEventArgs emptyChangingEventArgs = new PropertyChangingEventArgs(String.Empty);

private int \_ID;

private System.Nullable<int> \_FK\_BasickParamets;

private System.Nullable<int> \_FK\_Report;

private string \_Comment;

#region Extensibility Method Definitions

partial void OnLoaded();

partial void OnValidate(System.Data.Linq.ChangeAction action);

partial void OnCreated();

partial void OnIDChanging(int value);

partial void OnIDChanged();

partial void OnFK\_BasickParametsChanging(System.Nullable<int> value);

partial void OnFK\_BasickParametsChanged();

partial void OnFK\_ReportChanging(System.Nullable<int> value);

partial void OnFK\_ReportChanged();

partial void OnCommentChanging(string value);

partial void OnCommentChanged();

#endregion

public CommetntForBasicInReport()

{

OnCreated();

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_ID", DbType="Int NOT NULL", IsPrimaryKey=true)]

public int ID

{

get

{

return this.\_ID;

}

set

{

if ((this.\_ID != value))

{

this.OnIDChanging(value);

this.SendPropertyChanging();

this.\_ID = value;

this.SendPropertyChanged("ID");

this.OnIDChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_FK\_BasickParamets", DbType="Int")]

public System.Nullable<int> FK\_BasickParamets

{

get

{

return this.\_FK\_BasickParamets;

}

set

{

if ((this.\_FK\_BasickParamets != value))

{

this.OnFK\_BasickParametsChanging(value);

this.SendPropertyChanging();

this.\_FK\_BasickParamets = value;

this.SendPropertyChanged("FK\_BasickParamets");

this.OnFK\_BasickParametsChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_FK\_Report", DbType="Int")]

public System.Nullable<int> FK\_Report

{

get

{

return this.\_FK\_Report;

}

set

{

if ((this.\_FK\_Report != value))

{

this.OnFK\_ReportChanging(value);

this.SendPropertyChanging();

this.\_FK\_Report = value;

this.SendPropertyChanged("FK\_Report");

this.OnFK\_ReportChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_Comment", DbType="Text", UpdateCheck=UpdateCheck.Never)]

public string Comment

{

get

{

return this.\_Comment;

}

set

{

if ((this.\_Comment != value))

{

this.OnCommentChanging(value);

this.SendPropertyChanging();

this.\_Comment = value;

this.SendPropertyChanged("Comment");

this.OnCommentChanged();

}

}

}

public event PropertyChangingEventHandler PropertyChanging;

public event PropertyChangedEventHandler PropertyChanged;

protected virtual void SendPropertyChanging()

{

if ((this.PropertyChanging != null))

{

this.PropertyChanging(this, emptyChangingEventArgs);

}

}

protected virtual void SendPropertyChanged(String propertyName)

{

if ((this.PropertyChanged != null))

{

this.PropertyChanged(this, new PropertyChangedEventArgs(propertyName));

}

}

}

[global::System.Data.Linq.Mapping.TableAttribute(Name="dbo.ConfirmationHistory")]

public partial class ConfirmationHistory : INotifyPropertyChanging, INotifyPropertyChanged

{

private static PropertyChangingEventArgs emptyChangingEventArgs = new PropertyChangingEventArgs(String.Empty);

private int \_ID;

private string \_Name;

private System.Nullable<System.DateTime> \_Date;

private System.Nullable<int> \_FK\_IndicatorsTable;

private System.Nullable<int> \_FK\_CalculatedParamTable;

private System.Nullable<int> \_FK\_BasicParamTable;

private System.Nullable<int> \_FK\_UsersTable;

private System.Nullable<int> \_FK\_ReportTable;

private string \_Comment;

private EntityRef<BasicParametersTable> \_BasicParametersTable;

private EntityRef<CalculatedParametrs> \_CalculatedParametrs;

private EntityRef<ReportArchiveTable> \_ReportArchiveTable;

private EntityRef<UsersTable> \_UsersTable;

private EntityRef<IndicatorsTable> \_IndicatorsTable;

#region Extensibility Method Definitions

partial void OnLoaded();

partial void OnValidate(System.Data.Linq.ChangeAction action);

partial void OnCreated();

partial void OnIDChanging(int value);

partial void OnIDChanged();

partial void OnNameChanging(string value);

partial void OnNameChanged();

partial void OnDateChanging(System.Nullable<System.DateTime> value);

partial void OnDateChanged();

partial void OnFK\_IndicatorsTableChanging(System.Nullable<int> value);

partial void OnFK\_IndicatorsTableChanged();

partial void OnFK\_CalculatedParamTableChanging(System.Nullable<int> value);

partial void OnFK\_CalculatedParamTableChanged();

partial void OnFK\_BasicParamTableChanging(System.Nullable<int> value);

partial void OnFK\_BasicParamTableChanged();

partial void OnFK\_UsersTableChanging(System.Nullable<int> value);

partial void OnFK\_UsersTableChanged();

partial void OnFK\_ReportTableChanging(System.Nullable<int> value);

partial void OnFK\_ReportTableChanged();

partial void OnCommentChanging(string value);

partial void OnCommentChanged();

#endregion

public ConfirmationHistory()

{

this.\_BasicParametersTable = default(EntityRef<BasicParametersTable>);

this.\_CalculatedParametrs = default(EntityRef<CalculatedParametrs>);

this.\_ReportArchiveTable = default(EntityRef<ReportArchiveTable>);

this.\_UsersTable = default(EntityRef<UsersTable>);

this.\_IndicatorsTable = default(EntityRef<IndicatorsTable>);

OnCreated();

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_ID", AutoSync=AutoSync.OnInsert, DbType="Int NOT NULL IDENTITY", IsPrimaryKey=true, IsDbGenerated=true)]

public int ID

{

get

{

return this.\_ID;

}

set

{

if ((this.\_ID != value))

{

this.OnIDChanging(value);

this.SendPropertyChanging();

this.\_ID = value;

this.SendPropertyChanged("ID");

this.OnIDChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_Name", DbType="VarChar(50) NOT NULL", CanBeNull=false)]

public string Name

{

get

{

return this.\_Name;

}

set

{

if ((this.\_Name != value))

{

this.OnNameChanging(value);

this.SendPropertyChanging();

this.\_Name = value;

this.SendPropertyChanged("Name");

this.OnNameChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_Date", DbType="DateTime")]

public System.Nullable<System.DateTime> Date

{

get

{

return this.\_Date;

}

set

{

if ((this.\_Date != value))

{

this.OnDateChanging(value);

this.SendPropertyChanging();

this.\_Date = value;

this.SendPropertyChanged("Date");

this.OnDateChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_FK\_IndicatorsTable", DbType="Int")]

public System.Nullable<int> FK\_IndicatorsTable

{

get

{

return this.\_FK\_IndicatorsTable;

}

set

{

if ((this.\_FK\_IndicatorsTable != value))

{

if (this.\_IndicatorsTable.HasLoadedOrAssignedValue)

{

throw new System.Data.Linq.ForeignKeyReferenceAlreadyHasValueException();

}

this.OnFK\_IndicatorsTableChanging(value);

this.SendPropertyChanging();

this.\_FK\_IndicatorsTable = value;

this.SendPropertyChanged("FK\_IndicatorsTable");

this.OnFK\_IndicatorsTableChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_FK\_CalculatedParamTable", DbType="Int")]

public System.Nullable<int> FK\_CalculatedParamTable

{

get

{

return this.\_FK\_CalculatedParamTable;

}

set

{

if ((this.\_FK\_CalculatedParamTable != value))

{

if (this.\_CalculatedParametrs.HasLoadedOrAssignedValue)

{

throw new System.Data.Linq.ForeignKeyReferenceAlreadyHasValueException();

}

this.OnFK\_CalculatedParamTableChanging(value);

this.SendPropertyChanging();

this.\_FK\_CalculatedParamTable = value;

this.SendPropertyChanged("FK\_CalculatedParamTable");

this.OnFK\_CalculatedParamTableChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_FK\_BasicParamTable", DbType="Int")]

public System.Nullable<int> FK\_BasicParamTable

{

get

{

return this.\_FK\_BasicParamTable;

}

set

{

if ((this.\_FK\_BasicParamTable != value))

{

if (this.\_BasicParametersTable.HasLoadedOrAssignedValue)

{

throw new System.Data.Linq.ForeignKeyReferenceAlreadyHasValueException();

}

this.OnFK\_BasicParamTableChanging(value);

this.SendPropertyChanging();

this.\_FK\_BasicParamTable = value;

this.SendPropertyChanged("FK\_BasicParamTable");

this.OnFK\_BasicParamTableChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_FK\_UsersTable", DbType="Int")]

public System.Nullable<int> FK\_UsersTable

{

get

{

return this.\_FK\_UsersTable;

}

set

{

if ((this.\_FK\_UsersTable != value))

{

if (this.\_UsersTable.HasLoadedOrAssignedValue)

{

throw new System.Data.Linq.ForeignKeyReferenceAlreadyHasValueException();

}

this.OnFK\_UsersTableChanging(value);

this.SendPropertyChanging();

this.\_FK\_UsersTable = value;

this.SendPropertyChanged("FK\_UsersTable");

this.OnFK\_UsersTableChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_FK\_ReportTable", DbType="Int")]

public System.Nullable<int> FK\_ReportTable

{

get

{

return this.\_FK\_ReportTable;

}

set

{

if ((this.\_FK\_ReportTable != value))

{

if (this.\_ReportArchiveTable.HasLoadedOrAssignedValue)

{

throw new System.Data.Linq.ForeignKeyReferenceAlreadyHasValueException();

}

this.OnFK\_ReportTableChanging(value);

this.SendPropertyChanging();

this.\_FK\_ReportTable = value;

this.SendPropertyChanged("FK\_ReportTable");

this.OnFK\_ReportTableChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_Comment", DbType="Text", UpdateCheck=UpdateCheck.Never)]

public string Comment

{

get

{

return this.\_Comment;

}

set

{

if ((this.\_Comment != value))

{

this.OnCommentChanging(value);

this.SendPropertyChanging();

this.\_Comment = value;

this.SendPropertyChanged("Comment");

this.OnCommentChanged();

}

}

}

[global::System.Data.Linq.Mapping.AssociationAttribute(Name="BasicParametersTable\_ConfirmationHistory", Storage="\_BasicParametersTable", ThisKey="FK\_BasicParamTable", OtherKey="BasicParametersTableID", IsForeignKey=true)]

public BasicParametersTable BasicParametersTable

{

get

{

return this.\_BasicParametersTable.Entity;

}

set

{

BasicParametersTable previousValue = this.\_BasicParametersTable.Entity;

if (((previousValue != value)

|| (this.\_BasicParametersTable.HasLoadedOrAssignedValue == false)))

{

this.SendPropertyChanging();

if ((previousValue != null))

{

this.\_BasicParametersTable.Entity = null;

previousValue.ConfirmationHistory.Remove(this);

}

this.\_BasicParametersTable.Entity = value;

if ((value != null))

{

value.ConfirmationHistory.Add(this);

this.\_FK\_BasicParamTable = value.BasicParametersTableID;

}

else

{

this.\_FK\_BasicParamTable = default(Nullable<int>);

}

this.SendPropertyChanged("BasicParametersTable");

}

}

}

[global::System.Data.Linq.Mapping.AssociationAttribute(Name="CalculatedParametrs\_ConfirmationHistory", Storage="\_CalculatedParametrs", ThisKey="FK\_CalculatedParamTable", OtherKey="CalculatedParametrsID", IsForeignKey=true)]

public CalculatedParametrs CalculatedParametrs

{

get

{

return this.\_CalculatedParametrs.Entity;

}

set

{

CalculatedParametrs previousValue = this.\_CalculatedParametrs.Entity;

if (((previousValue != value)

|| (this.\_CalculatedParametrs.HasLoadedOrAssignedValue == false)))

{

this.SendPropertyChanging();

if ((previousValue != null))

{

this.\_CalculatedParametrs.Entity = null;

previousValue.ConfirmationHistory.Remove(this);

}

this.\_CalculatedParametrs.Entity = value;

if ((value != null))

{

value.ConfirmationHistory.Add(this);

this.\_FK\_CalculatedParamTable = value.CalculatedParametrsID;

}

else

{

this.\_FK\_CalculatedParamTable = default(Nullable<int>);

}

this.SendPropertyChanged("CalculatedParametrs");

}

}

}

[global::System.Data.Linq.Mapping.AssociationAttribute(Name="ReportArchiveTable\_ConfirmationHistory", Storage="\_ReportArchiveTable", ThisKey="FK\_ReportTable", OtherKey="ReportArchiveTableID", IsForeignKey=true)]

public ReportArchiveTable ReportArchiveTable

{

get

{

return this.\_ReportArchiveTable.Entity;

}

set

{

ReportArchiveTable previousValue = this.\_ReportArchiveTable.Entity;

if (((previousValue != value)

|| (this.\_ReportArchiveTable.HasLoadedOrAssignedValue == false)))

{

this.SendPropertyChanging();

if ((previousValue != null))

{

this.\_ReportArchiveTable.Entity = null;

previousValue.ConfirmationHistory.Remove(this);

}

this.\_ReportArchiveTable.Entity = value;

if ((value != null))

{

value.ConfirmationHistory.Add(this);

this.\_FK\_ReportTable = value.ReportArchiveTableID;

}

else

{

this.\_FK\_ReportTable = default(Nullable<int>);

}

this.SendPropertyChanged("ReportArchiveTable");

}

}

}

[global::System.Data.Linq.Mapping.AssociationAttribute(Name="UsersTable\_ConfirmationHistory", Storage="\_UsersTable", ThisKey="FK\_UsersTable", OtherKey="UsersTableID", IsForeignKey=true)]

public UsersTable UsersTable

{

get

{

return this.\_UsersTable.Entity;

}

set

{

UsersTable previousValue = this.\_UsersTable.Entity;

if (((previousValue != value)

|| (this.\_UsersTable.HasLoadedOrAssignedValue == false)))

{

this.SendPropertyChanging();

if ((previousValue != null))

{

this.\_UsersTable.Entity = null;

previousValue.ConfirmationHistory.Remove(this);

}

this.\_UsersTable.Entity = value;

if ((value != null))

{

value.ConfirmationHistory.Add(this);

this.\_FK\_UsersTable = value.UsersTableID;

}

else

{

this.\_FK\_UsersTable = default(Nullable<int>);

}

this.SendPropertyChanged("UsersTable");

}

}

}

[global::System.Data.Linq.Mapping.AssociationAttribute(Name="IndicatorsTable\_ConfirmationHistory", Storage="\_IndicatorsTable", ThisKey="FK\_IndicatorsTable", OtherKey="IndicatorsTableID", IsForeignKey=true)]

public IndicatorsTable IndicatorsTable

{

get

{

return this.\_IndicatorsTable.Entity;

}

set

{

IndicatorsTable previousValue = this.\_IndicatorsTable.Entity;

if (((previousValue != value)

|| (this.\_IndicatorsTable.HasLoadedOrAssignedValue == false)))

{

this.SendPropertyChanging();

if ((previousValue != null))

{

this.\_IndicatorsTable.Entity = null;

previousValue.ConfirmationHistory.Remove(this);

}

this.\_IndicatorsTable.Entity = value;

if ((value != null))

{

value.ConfirmationHistory.Add(this);

this.\_FK\_IndicatorsTable = value.IndicatorsTableID;

}

else

{

this.\_FK\_IndicatorsTable = default(Nullable<int>);

}

this.SendPropertyChanged("IndicatorsTable");

}

}

}

public event PropertyChangingEventHandler PropertyChanging;

public event PropertyChangedEventHandler PropertyChanged;

protected virtual void SendPropertyChanging()

{

if ((this.PropertyChanging != null))

{

this.PropertyChanging(this, emptyChangingEventArgs);

}

}

protected virtual void SendPropertyChanged(String propertyName)

{

if ((this.PropertyChanged != null))

{

this.PropertyChanged(this, new PropertyChangedEventArgs(propertyName));

}

}

}

[global::System.Data.Linq.Mapping.TableAttribute(Name="dbo.DocumentTable")]

public partial class DocumentTable : INotifyPropertyChanging, INotifyPropertyChanged

{

private static PropertyChangingEventArgs emptyChangingEventArgs = new PropertyChangingEventArgs(String.Empty);

private int \_DocumentID;

private string \_DocumentName;

private string \_DocumentLink;

private System.Nullable<bool> \_Active;

#region Extensibility Method Definitions

partial void OnLoaded();

partial void OnValidate(System.Data.Linq.ChangeAction action);

partial void OnCreated();

partial void OnDocumentIDChanging(int value);

partial void OnDocumentIDChanged();

partial void OnDocumentNameChanging(string value);

partial void OnDocumentNameChanged();

partial void OnDocumentLinkChanging(string value);

partial void OnDocumentLinkChanged();

partial void OnActiveChanging(System.Nullable<bool> value);

partial void OnActiveChanged();

#endregion

public DocumentTable()

{

OnCreated();

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_DocumentID", AutoSync=AutoSync.OnInsert, DbType="Int NOT NULL IDENTITY", IsPrimaryKey=true, IsDbGenerated=true)]

public int DocumentID

{

get

{

return this.\_DocumentID;

}

set

{

if ((this.\_DocumentID != value))

{

this.OnDocumentIDChanging(value);

this.SendPropertyChanging();

this.\_DocumentID = value;

this.SendPropertyChanged("DocumentID");

this.OnDocumentIDChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_DocumentName", DbType="VarChar(200)")]

public string DocumentName

{

get

{

return this.\_DocumentName;

}

set

{

if ((this.\_DocumentName != value))

{

this.OnDocumentNameChanging(value);

this.SendPropertyChanging();

this.\_DocumentName = value;

this.SendPropertyChanged("DocumentName");

this.OnDocumentNameChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_DocumentLink", DbType="VarChar(200)")]

public string DocumentLink

{

get

{

return this.\_DocumentLink;

}

set

{

if ((this.\_DocumentLink != value))

{

this.OnDocumentLinkChanging(value);

this.SendPropertyChanging();

this.\_DocumentLink = value;

this.SendPropertyChanged("DocumentLink");

this.OnDocumentLinkChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_Active", DbType="Bit")]

public System.Nullable<bool> Active

{

get

{

return this.\_Active;

}

set

{

if ((this.\_Active != value))

{

this.OnActiveChanging(value);

this.SendPropertyChanging();

this.\_Active = value;

this.SendPropertyChanged("Active");

this.OnActiveChanged();

}

}

}

public event PropertyChangingEventHandler PropertyChanging;

public event PropertyChangedEventHandler PropertyChanged;

protected virtual void SendPropertyChanging()

{

if ((this.PropertyChanging != null))

{

this.PropertyChanging(this, emptyChangingEventArgs);

}

}

protected virtual void SendPropertyChanged(String propertyName)

{

if ((this.PropertyChanged != null))

{

this.PropertyChanged(this, new PropertyChangedEventArgs(propertyName));

}

}

}

[global::System.Data.Linq.Mapping.TableAttribute(Name="dbo.EmailSendHistory")]

public partial class EmailSendHistory : INotifyPropertyChanging, INotifyPropertyChanged

{

private static PropertyChangingEventArgs emptyChangingEventArgs = new PropertyChangingEventArgs(String.Empty);

private int \_ID;

private System.Nullable<bool> \_Active;

private System.Nullable<int> \_FK\_ReportsArchiveTable;

private System.Nullable<System.DateTime> \_Date;

private string \_Value;

private System.Nullable<int> \_Count;

private EntityRef<ReportArchiveTable> \_ReportArchiveTable;

#region Extensibility Method Definitions

partial void OnLoaded();

partial void OnValidate(System.Data.Linq.ChangeAction action);

partial void OnCreated();

partial void OnIDChanging(int value);

partial void OnIDChanged();

partial void OnActiveChanging(System.Nullable<bool> value);

partial void OnActiveChanged();

partial void OnFK\_ReportsArchiveTableChanging(System.Nullable<int> value);

partial void OnFK\_ReportsArchiveTableChanged();

partial void OnDateChanging(System.Nullable<System.DateTime> value);

partial void OnDateChanged();

partial void OnValueChanging(string value);

partial void OnValueChanged();

partial void OnCountChanging(System.Nullable<int> value);

partial void OnCountChanged();

#endregion

public EmailSendHistory()

{

this.\_ReportArchiveTable = default(EntityRef<ReportArchiveTable>);

OnCreated();

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_ID", AutoSync=AutoSync.OnInsert, DbType="Int NOT NULL IDENTITY", IsPrimaryKey=true, IsDbGenerated=true)]

public int ID

{

get

{

return this.\_ID;

}

set

{

if ((this.\_ID != value))

{

this.OnIDChanging(value);

this.SendPropertyChanging();

this.\_ID = value;

this.SendPropertyChanged("ID");

this.OnIDChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_Active", DbType="Bit")]

public System.Nullable<bool> Active

{

get

{

return this.\_Active;

}

set

{

if ((this.\_Active != value))

{

this.OnActiveChanging(value);

this.SendPropertyChanging();

this.\_Active = value;

this.SendPropertyChanged("Active");

this.OnActiveChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_FK\_ReportsArchiveTable", DbType="Int")]

public System.Nullable<int> FK\_ReportsArchiveTable

{

get

{

return this.\_FK\_ReportsArchiveTable;

}

set

{

if ((this.\_FK\_ReportsArchiveTable != value))

{

if (this.\_ReportArchiveTable.HasLoadedOrAssignedValue)

{

throw new System.Data.Linq.ForeignKeyReferenceAlreadyHasValueException();

}

this.OnFK\_ReportsArchiveTableChanging(value);

this.SendPropertyChanging();

this.\_FK\_ReportsArchiveTable = value;

this.SendPropertyChanged("FK\_ReportsArchiveTable");

this.OnFK\_ReportsArchiveTableChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_Date", DbType="DateTime2")]

public System.Nullable<System.DateTime> Date

{

get

{

return this.\_Date;

}

set

{

if ((this.\_Date != value))

{

this.OnDateChanging(value);

this.SendPropertyChanging();

this.\_Date = value;

this.SendPropertyChanged("Date");

this.OnDateChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_Value", DbType="VarChar(50)")]

public string Value

{

get

{

return this.\_Value;

}

set

{

if ((this.\_Value != value))

{

this.OnValueChanging(value);

this.SendPropertyChanging();

this.\_Value = value;

this.SendPropertyChanged("Value");

this.OnValueChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_Count", DbType="Int")]

public System.Nullable<int> Count

{

get

{

return this.\_Count;

}

set

{

if ((this.\_Count != value))

{

this.OnCountChanging(value);

this.SendPropertyChanging();

this.\_Count = value;

this.SendPropertyChanged("Count");

this.OnCountChanged();

}

}

}

[global::System.Data.Linq.Mapping.AssociationAttribute(Name="ReportArchiveTable\_EmailSendHistory", Storage="\_ReportArchiveTable", ThisKey="FK\_ReportsArchiveTable", OtherKey="ReportArchiveTableID", IsForeignKey=true)]

public ReportArchiveTable ReportArchiveTable

{

get

{

return this.\_ReportArchiveTable.Entity;

}

set

{

ReportArchiveTable previousValue = this.\_ReportArchiveTable.Entity;

if (((previousValue != value)

|| (this.\_ReportArchiveTable.HasLoadedOrAssignedValue == false)))

{

this.SendPropertyChanging();

if ((previousValue != null))

{

this.\_ReportArchiveTable.Entity = null;

previousValue.EmailSendHistory.Remove(this);

}

this.\_ReportArchiveTable.Entity = value;

if ((value != null))

{

value.EmailSendHistory.Add(this);

this.\_FK\_ReportsArchiveTable = value.ReportArchiveTableID;

}

else

{

this.\_FK\_ReportsArchiveTable = default(Nullable<int>);

}

this.SendPropertyChanged("ReportArchiveTable");

}

}

}

public event PropertyChangingEventHandler PropertyChanging;

public event PropertyChangedEventHandler PropertyChanged;

protected virtual void SendPropertyChanging()

{

if ((this.PropertyChanging != null))

{

this.PropertyChanging(this, emptyChangingEventArgs);

}

}

protected virtual void SendPropertyChanged(String propertyName)

{

if ((this.PropertyChanged != null))

{

this.PropertyChanged(this, new PropertyChangedEventArgs(propertyName));

}

}

}

[global::System.Data.Linq.Mapping.TableAttribute(Name="dbo.EmailSendTable")]

public partial class EmailSendTable : INotifyPropertyChanging, INotifyPropertyChanged

{

private static PropertyChangingEventArgs emptyChangingEventArgs = new PropertyChangingEventArgs(String.Empty);

private int \_ID;

private string \_Email;

private string \_SMTPName;

private System.Nullable<int> \_Port;

private string \_Password;

private System.Nullable<long> \_SendOk;

private System.Nullable<long> \_SendError;

#region Extensibility Method Definitions

partial void OnLoaded();

partial void OnValidate(System.Data.Linq.ChangeAction action);

partial void OnCreated();

partial void OnIDChanging(int value);

partial void OnIDChanged();

partial void OnEmailChanging(string value);

partial void OnEmailChanged();

partial void OnSMTPNameChanging(string value);

partial void OnSMTPNameChanged();

partial void OnPortChanging(System.Nullable<int> value);

partial void OnPortChanged();

partial void OnPasswordChanging(string value);

partial void OnPasswordChanged();

partial void OnSendOkChanging(System.Nullable<long> value);

partial void OnSendOkChanged();

partial void OnSendErrorChanging(System.Nullable<long> value);

partial void OnSendErrorChanged();

#endregion

public EmailSendTable()

{

OnCreated();

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_ID", AutoSync=AutoSync.OnInsert, DbType="Int NOT NULL IDENTITY", IsPrimaryKey=true, IsDbGenerated=true)]

public int ID

{

get

{

return this.\_ID;

}

set

{

if ((this.\_ID != value))

{

this.OnIDChanging(value);

this.SendPropertyChanging();

this.\_ID = value;

this.SendPropertyChanged("ID");

this.OnIDChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_Email", DbType="VarChar(500)")]

public string Email

{

get

{

return this.\_Email;

}

set

{

if ((this.\_Email != value))

{

this.OnEmailChanging(value);

this.SendPropertyChanging();

this.\_Email = value;

this.SendPropertyChanged("Email");

this.OnEmailChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_SMTPName", DbType="VarChar(500)")]

public string SMTPName

{

get

{

return this.\_SMTPName;

}

set

{

if ((this.\_SMTPName != value))

{

this.OnSMTPNameChanging(value);

this.SendPropertyChanging();

this.\_SMTPName = value;

this.SendPropertyChanged("SMTPName");

this.OnSMTPNameChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_Port", DbType="Int")]

public System.Nullable<int> Port

{

get

{

return this.\_Port;

}

set

{

if ((this.\_Port != value))

{

this.OnPortChanging(value);

this.SendPropertyChanging();

this.\_Port = value;

this.SendPropertyChanged("Port");

this.OnPortChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_Password", DbType="VarChar(500)")]

public string Password

{

get

{

return this.\_Password;

}

set

{

if ((this.\_Password != value))

{

this.OnPasswordChanging(value);

this.SendPropertyChanging();

this.\_Password = value;

this.SendPropertyChanged("Password");

this.OnPasswordChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_SendOk", DbType="BigInt")]

public System.Nullable<long> SendOk

{

get

{

return this.\_SendOk;

}

set

{

if ((this.\_SendOk != value))

{

this.OnSendOkChanging(value);

this.SendPropertyChanging();

this.\_SendOk = value;

this.SendPropertyChanged("SendOk");

this.OnSendOkChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_SendError", DbType="BigInt")]

public System.Nullable<long> SendError

{

get

{

return this.\_SendError;

}

set

{

if ((this.\_SendError != value))

{

this.OnSendErrorChanging(value);

this.SendPropertyChanging();

this.\_SendError = value;

this.SendPropertyChanged("SendError");

this.OnSendErrorChanged();

}

}

}

public event PropertyChangingEventHandler PropertyChanging;

public event PropertyChangedEventHandler PropertyChanged;

protected virtual void SendPropertyChanging()

{

if ((this.PropertyChanging != null))

{

this.PropertyChanging(this, emptyChangingEventArgs);

}

}

protected virtual void SendPropertyChanged(String propertyName)

{

if ((this.PropertyChanged != null))

{

this.PropertyChanged(this, new PropertyChangedEventArgs(propertyName));

}

}

}

[global::System.Data.Linq.Mapping.TableAttribute(Name="dbo.EmailTemplate")]

public partial class EmailTemplate : INotifyPropertyChanging, INotifyPropertyChanged

{

private static PropertyChangingEventArgs emptyChangingEventArgs = new PropertyChangingEventArgs(String.Empty);

private int \_ID;

private string \_Name;

private string \_EmailTitle;

private string \_EmailContent;

private System.Nullable<bool> \_Active;

private string \_Comment;

private string \_NameRus;

#region Extensibility Method Definitions

partial void OnLoaded();

partial void OnValidate(System.Data.Linq.ChangeAction action);

partial void OnCreated();

partial void OnIDChanging(int value);

partial void OnIDChanged();

partial void OnNameChanging(string value);

partial void OnNameChanged();

partial void OnEmailTitleChanging(string value);

partial void OnEmailTitleChanged();

partial void OnEmailContentChanging(string value);

partial void OnEmailContentChanged();

partial void OnActiveChanging(System.Nullable<bool> value);

partial void OnActiveChanged();

partial void OnCommentChanging(string value);

partial void OnCommentChanged();

partial void OnNameRusChanging(string value);

partial void OnNameRusChanged();

#endregion

public EmailTemplate()

{

OnCreated();

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_ID", DbType="Int NOT NULL", IsPrimaryKey=true)]

public int ID

{

get

{

return this.\_ID;

}

set

{

if ((this.\_ID != value))

{

this.OnIDChanging(value);

this.SendPropertyChanging();

this.\_ID = value;

this.SendPropertyChanged("ID");

this.OnIDChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_Name", DbType="VarChar(500)")]

public string Name

{

get

{

return this.\_Name;

}

set

{

if ((this.\_Name != value))

{

this.OnNameChanging(value);

this.SendPropertyChanging();

this.\_Name = value;

this.SendPropertyChanged("Name");

this.OnNameChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_EmailTitle", DbType="Text", UpdateCheck=UpdateCheck.Never)]

public string EmailTitle

{

get

{

return this.\_EmailTitle;

}

set

{

if ((this.\_EmailTitle != value))

{

this.OnEmailTitleChanging(value);

this.SendPropertyChanging();

this.\_EmailTitle = value;

this.SendPropertyChanged("EmailTitle");

this.OnEmailTitleChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_EmailContent", DbType="Text", UpdateCheck=UpdateCheck.Never)]

public string EmailContent

{

get

{

return this.\_EmailContent;

}

set

{

if ((this.\_EmailContent != value))

{

this.OnEmailContentChanging(value);

this.SendPropertyChanging();

this.\_EmailContent = value;

this.SendPropertyChanged("EmailContent");

this.OnEmailContentChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_Active", DbType="Bit")]

public System.Nullable<bool> Active

{

get

{

return this.\_Active;

}

set

{

if ((this.\_Active != value))

{

this.OnActiveChanging(value);

this.SendPropertyChanging();

this.\_Active = value;

this.SendPropertyChanged("Active");

this.OnActiveChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_Comment", DbType="Text", UpdateCheck=UpdateCheck.Never)]

public string Comment

{

get

{

return this.\_Comment;

}

set

{

if ((this.\_Comment != value))

{

this.OnCommentChanging(value);

this.SendPropertyChanging();

this.\_Comment = value;

this.SendPropertyChanged("Comment");

this.OnCommentChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_NameRus", DbType="VarChar(500)")]

public string NameRus

{

get

{

return this.\_NameRus;

}

set

{

if ((this.\_NameRus != value))

{

this.OnNameRusChanging(value);

this.SendPropertyChanging();

this.\_NameRus = value;

this.SendPropertyChanged("NameRus");

this.OnNameRusChanged();

}

}

}

public event PropertyChangingEventHandler PropertyChanging;

public event PropertyChangedEventHandler PropertyChanged;

protected virtual void SendPropertyChanging()

{

if ((this.PropertyChanging != null))

{

this.PropertyChanging(this, emptyChangingEventArgs);

}

}

protected virtual void SendPropertyChanged(String propertyName)

{

if ((this.PropertyChanged != null))

{

this.PropertyChanged(this, new PropertyChangedEventArgs(propertyName));

}

}

}

[global::System.Data.Linq.Mapping.TableAttribute(Name="dbo.FieldOfExpertise")]

public partial class FieldOfExpertise : INotifyPropertyChanging, INotifyPropertyChanged

{

private static PropertyChangingEventArgs emptyChangingEventArgs = new PropertyChangingEventArgs(String.Empty);

private int \_FieldOfExpertiseID;

private bool \_Active;

private string \_Name;

private EntitySet<BasicParametrAdditional> \_BasicParametrAdditional;

private EntitySet<SpecializationTable> \_SpecializationTable;

#region Extensibility Method Definitions

partial void OnLoaded();

partial void OnValidate(System.Data.Linq.ChangeAction action);

partial void OnCreated();

partial void OnFieldOfExpertiseIDChanging(int value);

partial void OnFieldOfExpertiseIDChanged();

partial void OnActiveChanging(bool value);

partial void OnActiveChanged();

partial void OnNameChanging(string value);

partial void OnNameChanged();

#endregion

public FieldOfExpertise()

{

this.\_BasicParametrAdditional = new EntitySet<BasicParametrAdditional>(new Action<BasicParametrAdditional>(this.attach\_BasicParametrAdditional), new Action<BasicParametrAdditional>(this.detach\_BasicParametrAdditional));

this.\_SpecializationTable = new EntitySet<SpecializationTable>(new Action<SpecializationTable>(this.attach\_SpecializationTable), new Action<SpecializationTable>(this.detach\_SpecializationTable));

OnCreated();

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_FieldOfExpertiseID", AutoSync=AutoSync.OnInsert, DbType="Int NOT NULL IDENTITY", IsPrimaryKey=true, IsDbGenerated=true)]

public int FieldOfExpertiseID

{

get

{

return this.\_FieldOfExpertiseID;

}

set

{

if ((this.\_FieldOfExpertiseID != value))

{

this.OnFieldOfExpertiseIDChanging(value);

this.SendPropertyChanging();

this.\_FieldOfExpertiseID = value;

this.SendPropertyChanged("FieldOfExpertiseID");

this.OnFieldOfExpertiseIDChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_Active", DbType="Bit NOT NULL")]

public bool Active

{

get

{

return this.\_Active;

}

set

{

if ((this.\_Active != value))

{

this.OnActiveChanging(value);

this.SendPropertyChanging();

this.\_Active = value;

this.SendPropertyChanged("Active");

this.OnActiveChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_Name", DbType="VarChar(1000) NOT NULL", CanBeNull=false)]

public string Name

{

get

{

return this.\_Name;

}

set

{

if ((this.\_Name != value))

{

this.OnNameChanging(value);

this.SendPropertyChanging();

this.\_Name = value;

this.SendPropertyChanged("Name");

this.OnNameChanged();

}

}

}

[global::System.Data.Linq.Mapping.AssociationAttribute(Name="FieldOfExpertise\_BasicParametrAdditional", Storage="\_BasicParametrAdditional", ThisKey="FieldOfExpertiseID", OtherKey="FK\_FieldOfExpertise")]

public EntitySet<BasicParametrAdditional> BasicParametrAdditional

{

get

{

return this.\_BasicParametrAdditional;

}

set

{

this.\_BasicParametrAdditional.Assign(value);

}

}

[global::System.Data.Linq.Mapping.AssociationAttribute(Name="FieldOfExpertise\_SpecializationTable", Storage="\_SpecializationTable", ThisKey="FieldOfExpertiseID", OtherKey="FK\_FieldOfExpertise")]

public EntitySet<SpecializationTable> SpecializationTable

{

get

{

return this.\_SpecializationTable;

}

set

{

this.\_SpecializationTable.Assign(value);

}

}

public event PropertyChangingEventHandler PropertyChanging;

public event PropertyChangedEventHandler PropertyChanged;

protected virtual void SendPropertyChanging()

{

if ((this.PropertyChanging != null))

{

this.PropertyChanging(this, emptyChangingEventArgs);

}

}

protected virtual void SendPropertyChanged(String propertyName)

{

if ((this.PropertyChanged != null))

{

this.PropertyChanged(this, new PropertyChangedEventArgs(propertyName));

}

}

private void attach\_BasicParametrAdditional(BasicParametrAdditional entity)

{

this.SendPropertyChanging();

entity.FieldOfExpertise = this;

}

private void detach\_BasicParametrAdditional(BasicParametrAdditional entity)

{

this.SendPropertyChanging();

entity.FieldOfExpertise = null;

}

private void attach\_SpecializationTable(SpecializationTable entity)

{

this.SendPropertyChanging();

entity.FieldOfExpertise = this;

}

private void detach\_SpecializationTable(SpecializationTable entity)

{

this.SendPropertyChanging();

entity.FieldOfExpertise = null;

}

}

[global::System.Data.Linq.Mapping.TableAttribute(Name="dbo.FifthLevelSubdivisionTable")]

public partial class FifthLevelSubdivisionTable : INotifyPropertyChanging, INotifyPropertyChanged

{

private static PropertyChangingEventArgs emptyChangingEventArgs = new PropertyChangingEventArgs(String.Empty);

private int \_FifthLevelSubdivisionTableID;

private bool \_Active;

private string \_Name;

private int \_FK\_FourthLevelSubdivisionTable;

private EntitySet<CollectedBasicParametersTable> \_CollectedBasicParametersTable;

private EntitySet<CollectedIndicatorsForR> \_CollectedIndicatorsForR;

private EntitySet<UsersTable> \_UsersTable;

private EntityRef<FourthLevelSubdivisionTable> \_FourthLevelSubdivisionTable;

#region Extensibility Method Definitions

partial void OnLoaded();

partial void OnValidate(System.Data.Linq.ChangeAction action);

partial void OnCreated();

partial void OnFifthLevelSubdivisionTableIDChanging(int value);

partial void OnFifthLevelSubdivisionTableIDChanged();

partial void OnActiveChanging(bool value);

partial void OnActiveChanged();

partial void OnNameChanging(string value);

partial void OnNameChanged();

partial void OnFK\_FourthLevelSubdivisionTableChanging(int value);

partial void OnFK\_FourthLevelSubdivisionTableChanged();

#endregion

public FifthLevelSubdivisionTable()

{

this.\_CollectedBasicParametersTable = new EntitySet<CollectedBasicParametersTable>(new Action<CollectedBasicParametersTable>(this.attach\_CollectedBasicParametersTable), new Action<CollectedBasicParametersTable>(this.detach\_CollectedBasicParametersTable));

this.\_CollectedIndicatorsForR = new EntitySet<CollectedIndicatorsForR>(new Action<CollectedIndicatorsForR>(this.attach\_CollectedIndicatorsForR), new Action<CollectedIndicatorsForR>(this.detach\_CollectedIndicatorsForR));

this.\_UsersTable = new EntitySet<UsersTable>(new Action<UsersTable>(this.attach\_UsersTable), new Action<UsersTable>(this.detach\_UsersTable));

this.\_FourthLevelSubdivisionTable = default(EntityRef<FourthLevelSubdivisionTable>);

OnCreated();

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_FifthLevelSubdivisionTableID", AutoSync=AutoSync.OnInsert, DbType="Int NOT NULL IDENTITY", IsPrimaryKey=true, IsDbGenerated=true)]

public int FifthLevelSubdivisionTableID

{

get

{

return this.\_FifthLevelSubdivisionTableID;

}

set

{

if ((this.\_FifthLevelSubdivisionTableID != value))

{

this.OnFifthLevelSubdivisionTableIDChanging(value);

this.SendPropertyChanging();

this.\_FifthLevelSubdivisionTableID = value;

this.SendPropertyChanged("FifthLevelSubdivisionTableID");

this.OnFifthLevelSubdivisionTableIDChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_Active", DbType="Bit NOT NULL")]

public bool Active

{

get

{

return this.\_Active;

}

set

{

if ((this.\_Active != value))

{

this.OnActiveChanging(value);

this.SendPropertyChanging();

this.\_Active = value;

this.SendPropertyChanged("Active");

this.OnActiveChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_Name", DbType="VarChar(500) NOT NULL", CanBeNull=false)]

public string Name

{

get

{

return this.\_Name;

}

set

{

if ((this.\_Name != value))

{

this.OnNameChanging(value);

this.SendPropertyChanging();

this.\_Name = value;

this.SendPropertyChanged("Name");

this.OnNameChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_FK\_FourthLevelSubdivisionTable", DbType="Int NOT NULL")]

public int FK\_FourthLevelSubdivisionTable

{

get

{

return this.\_FK\_FourthLevelSubdivisionTable;

}

set

{

if ((this.\_FK\_FourthLevelSubdivisionTable != value))

{

if (this.\_FourthLevelSubdivisionTable.HasLoadedOrAssignedValue)

{

throw new System.Data.Linq.ForeignKeyReferenceAlreadyHasValueException();

}

this.OnFK\_FourthLevelSubdivisionTableChanging(value);

this.SendPropertyChanging();

this.\_FK\_FourthLevelSubdivisionTable = value;

this.SendPropertyChanged("FK\_FourthLevelSubdivisionTable");

this.OnFK\_FourthLevelSubdivisionTableChanged();

}

}

}

[global::System.Data.Linq.Mapping.AssociationAttribute(Name="FifthLevelSubdivisionTable\_CollectedBasicParametersTable", Storage="\_CollectedBasicParametersTable", ThisKey="FifthLevelSubdivisionTableID", OtherKey="FK\_FifthLevelSubdivisionTable")]

public EntitySet<CollectedBasicParametersTable> CollectedBasicParametersTable

{

get

{

return this.\_CollectedBasicParametersTable;

}

set

{

this.\_CollectedBasicParametersTable.Assign(value);

}

}

[global::System.Data.Linq.Mapping.AssociationAttribute(Name="FifthLevelSubdivisionTable\_CollectedIndicatorsForR", Storage="\_CollectedIndicatorsForR", ThisKey="FifthLevelSubdivisionTableID", OtherKey="FK\_FifthLevelSubdivisionTable")]

public EntitySet<CollectedIndicatorsForR> CollectedIndicatorsForR

{

get

{

return this.\_CollectedIndicatorsForR;

}

set

{

this.\_CollectedIndicatorsForR.Assign(value);

}

}

[global::System.Data.Linq.Mapping.AssociationAttribute(Name="FifthLevelSubdivisionTable\_UsersTable", Storage="\_UsersTable", ThisKey="FifthLevelSubdivisionTableID", OtherKey="FK\_FifthLevelSubdivisionTable")]

public EntitySet<UsersTable> UsersTable

{

get

{

return this.\_UsersTable;

}

set

{

this.\_UsersTable.Assign(value);

}

}

[global::System.Data.Linq.Mapping.AssociationAttribute(Name="FourthLevelSubdivisionTable\_FifthLevelSubdivisionTable", Storage="\_FourthLevelSubdivisionTable", ThisKey="FK\_FourthLevelSubdivisionTable", OtherKey="FourthLevelSubdivisionTableID", IsForeignKey=true)]

public FourthLevelSubdivisionTable FourthLevelSubdivisionTable

{

get

{

return this.\_FourthLevelSubdivisionTable.Entity;

}

set

{

FourthLevelSubdivisionTable previousValue = this.\_FourthLevelSubdivisionTable.Entity;

if (((previousValue != value)

|| (this.\_FourthLevelSubdivisionTable.HasLoadedOrAssignedValue == false)))

{

this.SendPropertyChanging();

if ((previousValue != null))

{

this.\_FourthLevelSubdivisionTable.Entity = null;

previousValue.FifthLevelSubdivisionTable.Remove(this);

}

this.\_FourthLevelSubdivisionTable.Entity = value;

if ((value != null))

{

value.FifthLevelSubdivisionTable.Add(this);

this.\_FK\_FourthLevelSubdivisionTable = value.FourthLevelSubdivisionTableID;

}

else

{

this.\_FK\_FourthLevelSubdivisionTable = default(int);

}

this.SendPropertyChanged("FourthLevelSubdivisionTable");

}

}

}

public event PropertyChangingEventHandler PropertyChanging;

public event PropertyChangedEventHandler PropertyChanged;

protected virtual void SendPropertyChanging()

{

if ((this.PropertyChanging != null))

{

this.PropertyChanging(this, emptyChangingEventArgs);

}

}

protected virtual void SendPropertyChanged(String propertyName)

{

if ((this.PropertyChanged != null))

{

this.PropertyChanged(this, new PropertyChangedEventArgs(propertyName));

}

}

private void attach\_CollectedBasicParametersTable(CollectedBasicParametersTable entity)

{

this.SendPropertyChanging();

entity.FifthLevelSubdivisionTable = this;

}

private void detach\_CollectedBasicParametersTable(CollectedBasicParametersTable entity)

{

this.SendPropertyChanging();

entity.FifthLevelSubdivisionTable = null;

}

private void attach\_CollectedIndicatorsForR(CollectedIndicatorsForR entity)

{

this.SendPropertyChanging();

entity.FifthLevelSubdivisionTable = this;

}

private void detach\_CollectedIndicatorsForR(CollectedIndicatorsForR entity)

{

this.SendPropertyChanging();

entity.FifthLevelSubdivisionTable = null;

}

private void attach\_UsersTable(UsersTable entity)

{

this.SendPropertyChanging();

entity.FifthLevelSubdivisionTable = this;

}

private void detach\_UsersTable(UsersTable entity)

{

this.SendPropertyChanging();

entity.FifthLevelSubdivisionTable = null;

}

}

[global::System.Data.Linq.Mapping.TableAttribute(Name="dbo.FirstLevelSubdivisionTable")]

public partial class FirstLevelSubdivisionTable : INotifyPropertyChanging, INotifyPropertyChanged

{

private static PropertyChangingEventArgs emptyChangingEventArgs = new PropertyChangingEventArgs(String.Empty);

private int \_FirstLevelSubdivisionTableID;

private bool \_Active;

private string \_Name;

private System.Nullable<int> \_FK\_ZeroLevelSubvisionTable;

private EntitySet<CollectedBasicParametersTable> \_CollectedBasicParametersTable;

private EntitySet<CollectedIndicatorsForR> \_CollectedIndicatorsForR;

private EntitySet<ReportArchiveAndLevelMappingTable> \_ReportArchiveAndLevelMappingTable;

private EntitySet<SecondLevelSubdivisionTable> \_SecondLevelSubdivisionTable;

private EntitySet<UsersTable> \_UsersTable;

private EntityRef<ZeroLevelSubdivisionTable> \_ZeroLevelSubdivisionTable;

#region Extensibility Method Definitions

partial void OnLoaded();

partial void OnValidate(System.Data.Linq.ChangeAction action);

partial void OnCreated();

partial void OnFirstLevelSubdivisionTableIDChanging(int value);

partial void OnFirstLevelSubdivisionTableIDChanged();

partial void OnActiveChanging(bool value);

partial void OnActiveChanged();

partial void OnNameChanging(string value);

partial void OnNameChanged();

partial void OnFK\_ZeroLevelSubvisionTableChanging(System.Nullable<int> value);

partial void OnFK\_ZeroLevelSubvisionTableChanged();

#endregion

public FirstLevelSubdivisionTable()

{

this.\_CollectedBasicParametersTable = new EntitySet<CollectedBasicParametersTable>(new Action<CollectedBasicParametersTable>(this.attach\_CollectedBasicParametersTable), new Action<CollectedBasicParametersTable>(this.detach\_CollectedBasicParametersTable));

this.\_CollectedIndicatorsForR = new EntitySet<CollectedIndicatorsForR>(new Action<CollectedIndicatorsForR>(this.attach\_CollectedIndicatorsForR), new Action<CollectedIndicatorsForR>(this.detach\_CollectedIndicatorsForR));

this.\_ReportArchiveAndLevelMappingTable = new EntitySet<ReportArchiveAndLevelMappingTable>(new Action<ReportArchiveAndLevelMappingTable>(this.attach\_ReportArchiveAndLevelMappingTable), new Action<ReportArchiveAndLevelMappingTable>(this.detach\_ReportArchiveAndLevelMappingTable));

this.\_SecondLevelSubdivisionTable = new EntitySet<SecondLevelSubdivisionTable>(new Action<SecondLevelSubdivisionTable>(this.attach\_SecondLevelSubdivisionTable), new Action<SecondLevelSubdivisionTable>(this.detach\_SecondLevelSubdivisionTable));

this.\_UsersTable = new EntitySet<UsersTable>(new Action<UsersTable>(this.attach\_UsersTable), new Action<UsersTable>(this.detach\_UsersTable));

this.\_ZeroLevelSubdivisionTable = default(EntityRef<ZeroLevelSubdivisionTable>);

OnCreated();

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_FirstLevelSubdivisionTableID", AutoSync=AutoSync.OnInsert, DbType="Int NOT NULL IDENTITY", IsPrimaryKey=true, IsDbGenerated=true)]

public int FirstLevelSubdivisionTableID

{

get

{

return this.\_FirstLevelSubdivisionTableID;

}

set

{

if ((this.\_FirstLevelSubdivisionTableID != value))

{

this.OnFirstLevelSubdivisionTableIDChanging(value);

this.SendPropertyChanging();

this.\_FirstLevelSubdivisionTableID = value;

this.SendPropertyChanged("FirstLevelSubdivisionTableID");

this.OnFirstLevelSubdivisionTableIDChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_Active", DbType="Bit NOT NULL")]

public bool Active

{

get

{

return this.\_Active;

}

set

{

if ((this.\_Active != value))

{

this.OnActiveChanging(value);

this.SendPropertyChanging();

this.\_Active = value;

this.SendPropertyChanged("Active");

this.OnActiveChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_Name", DbType="VarChar(500) NOT NULL", CanBeNull=false)]

public string Name

{

get

{

return this.\_Name;

}

set

{

if ((this.\_Name != value))

{

this.OnNameChanging(value);

this.SendPropertyChanging();

this.\_Name = value;

this.SendPropertyChanged("Name");

this.OnNameChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_FK\_ZeroLevelSubvisionTable", DbType="Int")]

public System.Nullable<int> FK\_ZeroLevelSubvisionTable

{

get

{

return this.\_FK\_ZeroLevelSubvisionTable;

}

set

{

if ((this.\_FK\_ZeroLevelSubvisionTable != value))

{

if (this.\_ZeroLevelSubdivisionTable.HasLoadedOrAssignedValue)

{

throw new System.Data.Linq.ForeignKeyReferenceAlreadyHasValueException();

}

this.OnFK\_ZeroLevelSubvisionTableChanging(value);

this.SendPropertyChanging();

this.\_FK\_ZeroLevelSubvisionTable = value;

this.SendPropertyChanged("FK\_ZeroLevelSubvisionTable");

this.OnFK\_ZeroLevelSubvisionTableChanged();

}

}

}

[global::System.Data.Linq.Mapping.AssociationAttribute(Name="FirstLevelSubdivisionTable\_CollectedBasicParametersTable", Storage="\_CollectedBasicParametersTable", ThisKey="FirstLevelSubdivisionTableID", OtherKey="FK\_FirstLevelSubdivisionTable")]

public EntitySet<CollectedBasicParametersTable> CollectedBasicParametersTable

{

get

{

return this.\_CollectedBasicParametersTable;

}

set

{

this.\_CollectedBasicParametersTable.Assign(value);

}

}

[global::System.Data.Linq.Mapping.AssociationAttribute(Name="FirstLevelSubdivisionTable\_CollectedIndicatorsForR", Storage="\_CollectedIndicatorsForR", ThisKey="FirstLevelSubdivisionTableID", OtherKey="FK\_FirstLevelSubdivisionTable")]

public EntitySet<CollectedIndicatorsForR> CollectedIndicatorsForR

{

get

{

return this.\_CollectedIndicatorsForR;

}

set

{

this.\_CollectedIndicatorsForR.Assign(value);

}

}

[global::System.Data.Linq.Mapping.AssociationAttribute(Name="FirstLevelSubdivisionTable\_ReportArchiveAndLevelMappingTable", Storage="\_ReportArchiveAndLevelMappingTable", ThisKey="FirstLevelSubdivisionTableID", OtherKey="FK\_FirstLevelSubmisionTableId")]

public EntitySet<ReportArchiveAndLevelMappingTable> ReportArchiveAndLevelMappingTable

{

get

{

return this.\_ReportArchiveAndLevelMappingTable;

}

set

{

this.\_ReportArchiveAndLevelMappingTable.Assign(value);

}

}

[global::System.Data.Linq.Mapping.AssociationAttribute(Name="FirstLevelSubdivisionTable\_SecondLevelSubdivisionTable", Storage="\_SecondLevelSubdivisionTable", ThisKey="FirstLevelSubdivisionTableID", OtherKey="FK\_FirstLevelSubdivisionTable")]

public EntitySet<SecondLevelSubdivisionTable> SecondLevelSubdivisionTable

{

get

{

return this.\_SecondLevelSubdivisionTable;

}

set

{

this.\_SecondLevelSubdivisionTable.Assign(value);

}

}

[global::System.Data.Linq.Mapping.AssociationAttribute(Name="FirstLevelSubdivisionTable\_UsersTable", Storage="\_UsersTable", ThisKey="FirstLevelSubdivisionTableID", OtherKey="FK\_FirstLevelSubdivisionTable")]

public EntitySet<UsersTable> UsersTable

{

get

{

return this.\_UsersTable;

}

set

{

this.\_UsersTable.Assign(value);

}

}

[global::System.Data.Linq.Mapping.AssociationAttribute(Name="ZeroLevelSubdivisionTable\_FirstLevelSubdivisionTable", Storage="\_ZeroLevelSubdivisionTable", ThisKey="FK\_ZeroLevelSubvisionTable", OtherKey="ZeroLevelSubdivisionTableID", IsForeignKey=true)]

public ZeroLevelSubdivisionTable ZeroLevelSubdivisionTable

{

get

{

return this.\_ZeroLevelSubdivisionTable.Entity;

}

set

{

ZeroLevelSubdivisionTable previousValue = this.\_ZeroLevelSubdivisionTable.Entity;

if (((previousValue != value)

|| (this.\_ZeroLevelSubdivisionTable.HasLoadedOrAssignedValue == false)))

{

this.SendPropertyChanging();

if ((previousValue != null))

{

this.\_ZeroLevelSubdivisionTable.Entity = null;

previousValue.FirstLevelSubdivisionTable.Remove(this);

}

this.\_ZeroLevelSubdivisionTable.Entity = value;

if ((value != null))

{

value.FirstLevelSubdivisionTable.Add(this);

this.\_FK\_ZeroLevelSubvisionTable = value.ZeroLevelSubdivisionTableID;

}

else

{

this.\_FK\_ZeroLevelSubvisionTable = default(Nullable<int>);

}

this.SendPropertyChanged("ZeroLevelSubdivisionTable");

}

}

}

public event PropertyChangingEventHandler PropertyChanging;

public event PropertyChangedEventHandler PropertyChanged;

protected virtual void SendPropertyChanging()

{

if ((this.PropertyChanging != null))

{

this.PropertyChanging(this, emptyChangingEventArgs);

}

}

protected virtual void SendPropertyChanged(String propertyName)

{

if ((this.PropertyChanged != null))

{

this.PropertyChanged(this, new PropertyChangedEventArgs(propertyName));

}

}

private void attach\_CollectedBasicParametersTable(CollectedBasicParametersTable entity)

{

this.SendPropertyChanging();

entity.FirstLevelSubdivisionTable = this;

}

private void detach\_CollectedBasicParametersTable(CollectedBasicParametersTable entity)

{

this.SendPropertyChanging();

entity.FirstLevelSubdivisionTable = null;

}

private void attach\_CollectedIndicatorsForR(CollectedIndicatorsForR entity)

{

this.SendPropertyChanging();

entity.FirstLevelSubdivisionTable = this;

}

private void detach\_CollectedIndicatorsForR(CollectedIndicatorsForR entity)

{

this.SendPropertyChanging();

entity.FirstLevelSubdivisionTable = null;

}

private void attach\_ReportArchiveAndLevelMappingTable(ReportArchiveAndLevelMappingTable entity)

{

this.SendPropertyChanging();

entity.FirstLevelSubdivisionTable = this;

}

private void detach\_ReportArchiveAndLevelMappingTable(ReportArchiveAndLevelMappingTable entity)

{

this.SendPropertyChanging();

entity.FirstLevelSubdivisionTable = null;

}

private void attach\_SecondLevelSubdivisionTable(SecondLevelSubdivisionTable entity)

{

this.SendPropertyChanging();

entity.FirstLevelSubdivisionTable = this;

}

private void detach\_SecondLevelSubdivisionTable(SecondLevelSubdivisionTable entity)

{

this.SendPropertyChanging();

entity.FirstLevelSubdivisionTable = null;

}

private void attach\_UsersTable(UsersTable entity)

{

this.SendPropertyChanging();

entity.FirstLevelSubdivisionTable = this;

}

private void detach\_UsersTable(UsersTable entity)

{

this.SendPropertyChanging();

entity.FirstLevelSubdivisionTable = null;

}

}

[global::System.Data.Linq.Mapping.TableAttribute(Name="dbo.FourthLevelParametrs")]

public partial class FourthLevelParametrs : INotifyPropertyChanging, INotifyPropertyChanged

{

private static PropertyChangingEventArgs emptyChangingEventArgs = new PropertyChangingEventArgs(String.Empty);

private int \_FourthLevelParametrsID;

private System.Nullable<bool> \_Active;

private System.Nullable<bool> \_IsModernEducationTechnologies;

private System.Nullable<bool> \_IsNetworkComunication;

private System.Nullable<bool> \_IsInvalidStudentsFacilities;

private System.Nullable<bool> \_IsForeignStudentsAccept;

private System.Nullable<int> \_SpecType;

private EntityRef<FourthLevelSubdivisionTable> \_FourthLevelSubdivisionTable;

#region Extensibility Method Definitions

partial void OnLoaded();

partial void OnValidate(System.Data.Linq.ChangeAction action);

partial void OnCreated();

partial void OnFourthLevelParametrsIDChanging(int value);

partial void OnFourthLevelParametrsIDChanged();

partial void OnActiveChanging(System.Nullable<bool> value);

partial void OnActiveChanged();

partial void OnIsModernEducationTechnologiesChanging(System.Nullable<bool> value);

partial void OnIsModernEducationTechnologiesChanged();

partial void OnIsNetworkComunicationChanging(System.Nullable<bool> value);

partial void OnIsNetworkComunicationChanged();

partial void OnIsInvalidStudentsFacilitiesChanging(System.Nullable<bool> value);

partial void OnIsInvalidStudentsFacilitiesChanged();

partial void OnIsForeignStudentsAcceptChanging(System.Nullable<bool> value);

partial void OnIsForeignStudentsAcceptChanged();

partial void OnSpecTypeChanging(System.Nullable<int> value);

partial void OnSpecTypeChanged();

#endregion

public FourthLevelParametrs()

{

this.\_FourthLevelSubdivisionTable = default(EntityRef<FourthLevelSubdivisionTable>);

OnCreated();

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_FourthLevelParametrsID", DbType="Int NOT NULL", IsPrimaryKey=true)]

public int FourthLevelParametrsID

{

get

{

return this.\_FourthLevelParametrsID;

}

set

{

if ((this.\_FourthLevelParametrsID != value))

{

if (this.\_FourthLevelSubdivisionTable.HasLoadedOrAssignedValue)

{

throw new System.Data.Linq.ForeignKeyReferenceAlreadyHasValueException();

}

this.OnFourthLevelParametrsIDChanging(value);

this.SendPropertyChanging();

this.\_FourthLevelParametrsID = value;

this.SendPropertyChanged("FourthLevelParametrsID");

this.OnFourthLevelParametrsIDChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_Active", DbType="Bit")]

public System.Nullable<bool> Active

{

get

{

return this.\_Active;

}

set

{

if ((this.\_Active != value))

{

this.OnActiveChanging(value);

this.SendPropertyChanging();

this.\_Active = value;

this.SendPropertyChanged("Active");

this.OnActiveChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_IsModernEducationTechnologies", DbType="Bit")]

public System.Nullable<bool> IsModernEducationTechnologies

{

get

{

return this.\_IsModernEducationTechnologies;

}

set

{

if ((this.\_IsModernEducationTechnologies != value))

{

this.OnIsModernEducationTechnologiesChanging(value);

this.SendPropertyChanging();

this.\_IsModernEducationTechnologies = value;

this.SendPropertyChanged("IsModernEducationTechnologies");

this.OnIsModernEducationTechnologiesChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_IsNetworkComunication", DbType="Bit")]

public System.Nullable<bool> IsNetworkComunication

{

get

{

return this.\_IsNetworkComunication;

}

set

{

if ((this.\_IsNetworkComunication != value))

{

this.OnIsNetworkComunicationChanging(value);

this.SendPropertyChanging();

this.\_IsNetworkComunication = value;

this.SendPropertyChanged("IsNetworkComunication");

this.OnIsNetworkComunicationChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_IsInvalidStudentsFacilities", DbType="Bit")]

public System.Nullable<bool> IsInvalidStudentsFacilities

{

get

{

return this.\_IsInvalidStudentsFacilities;

}

set

{

if ((this.\_IsInvalidStudentsFacilities != value))

{

this.OnIsInvalidStudentsFacilitiesChanging(value);

this.SendPropertyChanging();

this.\_IsInvalidStudentsFacilities = value;

this.SendPropertyChanged("IsInvalidStudentsFacilities");

this.OnIsInvalidStudentsFacilitiesChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_IsForeignStudentsAccept", DbType="Bit")]

public System.Nullable<bool> IsForeignStudentsAccept

{

get

{

return this.\_IsForeignStudentsAccept;

}

set

{

if ((this.\_IsForeignStudentsAccept != value))

{

this.OnIsForeignStudentsAcceptChanging(value);

this.SendPropertyChanging();

this.\_IsForeignStudentsAccept = value;

this.SendPropertyChanged("IsForeignStudentsAccept");

this.OnIsForeignStudentsAcceptChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_SpecType", DbType="Int")]

public System.Nullable<int> SpecType

{

get

{

return this.\_SpecType;

}

set

{

if ((this.\_SpecType != value))

{

this.OnSpecTypeChanging(value);

this.SendPropertyChanging();

this.\_SpecType = value;

this.SendPropertyChanged("SpecType");

this.OnSpecTypeChanged();

}

}

}

[global::System.Data.Linq.Mapping.AssociationAttribute(Name="FourthLevelSubdivisionTable\_FourthLevelParametrs", Storage="\_FourthLevelSubdivisionTable", ThisKey="FourthLevelParametrsID", OtherKey="FourthLevelSubdivisionTableID", IsForeignKey=true)]

public FourthLevelSubdivisionTable FourthLevelSubdivisionTable

{

get

{

return this.\_FourthLevelSubdivisionTable.Entity;

}

set

{

FourthLevelSubdivisionTable previousValue = this.\_FourthLevelSubdivisionTable.Entity;

if (((previousValue != value)

|| (this.\_FourthLevelSubdivisionTable.HasLoadedOrAssignedValue == false)))

{

this.SendPropertyChanging();

if ((previousValue != null))

{

this.\_FourthLevelSubdivisionTable.Entity = null;

previousValue.FourthLevelParametrs = null;

}

this.\_FourthLevelSubdivisionTable.Entity = value;

if ((value != null))

{

value.FourthLevelParametrs = this;

this.\_FourthLevelParametrsID = value.FourthLevelSubdivisionTableID;

}

else

{

this.\_FourthLevelParametrsID = default(int);

}

this.SendPropertyChanged("FourthLevelSubdivisionTable");

}

}

}

public event PropertyChangingEventHandler PropertyChanging;

public event PropertyChangedEventHandler PropertyChanged;

protected virtual void SendPropertyChanging()

{

if ((this.PropertyChanging != null))

{

this.PropertyChanging(this, emptyChangingEventArgs);

}

}

protected virtual void SendPropertyChanged(String propertyName)

{

if ((this.PropertyChanged != null))

{

this.PropertyChanged(this, new PropertyChangedEventArgs(propertyName));

}

}

}

[global::System.Data.Linq.Mapping.TableAttribute(Name="dbo.FourthLevelSubdivisionTable")]

public partial class FourthLevelSubdivisionTable : INotifyPropertyChanging, INotifyPropertyChanged

{

private static PropertyChangingEventArgs emptyChangingEventArgs = new PropertyChangingEventArgs(String.Empty);

private int \_FourthLevelSubdivisionTableID;

private bool \_Active;

private string \_Name;

private int \_FK\_ThirdLevelSubdivisionTable;

private int \_FK\_Specialization;

private EntitySet<CollectedBasicParametersTable> \_CollectedBasicParametersTable;

private EntitySet<CollectedIndicatorsForR> \_CollectedIndicatorsForR;

private EntitySet<FifthLevelSubdivisionTable> \_FifthLevelSubdivisionTable;

private EntityRef<FourthLevelParametrs> \_FourthLevelParametrs;

private EntitySet<UsersTable> \_UsersTable;

private EntityRef<SpecializationTable> \_SpecializationTable;

private EntityRef<ThirdLevelSubdivisionTable> \_ThirdLevelSubdivisionTable;

#region Extensibility Method Definitions

partial void OnLoaded();

partial void OnValidate(System.Data.Linq.ChangeAction action);

partial void OnCreated();

partial void OnFourthLevelSubdivisionTableIDChanging(int value);

partial void OnFourthLevelSubdivisionTableIDChanged();

partial void OnActiveChanging(bool value);

partial void OnActiveChanged();

partial void OnNameChanging(string value);

partial void OnNameChanged();

partial void OnFK\_ThirdLevelSubdivisionTableChanging(int value);

partial void OnFK\_ThirdLevelSubdivisionTableChanged();

partial void OnFK\_SpecializationChanging(int value);

partial void OnFK\_SpecializationChanged();

#endregion

public FourthLevelSubdivisionTable()

{

this.\_CollectedBasicParametersTable = new EntitySet<CollectedBasicParametersTable>(new Action<CollectedBasicParametersTable>(this.attach\_CollectedBasicParametersTable), new Action<CollectedBasicParametersTable>(this.detach\_CollectedBasicParametersTable));

this.\_CollectedIndicatorsForR = new EntitySet<CollectedIndicatorsForR>(new Action<CollectedIndicatorsForR>(this.attach\_CollectedIndicatorsForR), new Action<CollectedIndicatorsForR>(this.detach\_CollectedIndicatorsForR));

this.\_FifthLevelSubdivisionTable = new EntitySet<FifthLevelSubdivisionTable>(new Action<FifthLevelSubdivisionTable>(this.attach\_FifthLevelSubdivisionTable), new Action<FifthLevelSubdivisionTable>(this.detach\_FifthLevelSubdivisionTable));

this.\_FourthLevelParametrs = default(EntityRef<FourthLevelParametrs>);

this.\_UsersTable = new EntitySet<UsersTable>(new Action<UsersTable>(this.attach\_UsersTable), new Action<UsersTable>(this.detach\_UsersTable));

this.\_SpecializationTable = default(EntityRef<SpecializationTable>);

this.\_ThirdLevelSubdivisionTable = default(EntityRef<ThirdLevelSubdivisionTable>);

OnCreated();

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_FourthLevelSubdivisionTableID", AutoSync=AutoSync.OnInsert, DbType="Int NOT NULL IDENTITY", IsPrimaryKey=true, IsDbGenerated=true)]

public int FourthLevelSubdivisionTableID

{

get

{

return this.\_FourthLevelSubdivisionTableID;

}

set

{

if ((this.\_FourthLevelSubdivisionTableID != value))

{

this.OnFourthLevelSubdivisionTableIDChanging(value);

this.SendPropertyChanging();

this.\_FourthLevelSubdivisionTableID = value;

this.SendPropertyChanged("FourthLevelSubdivisionTableID");

this.OnFourthLevelSubdivisionTableIDChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_Active", DbType="Bit NOT NULL")]

public bool Active

{

get

{

return this.\_Active;

}

set

{

if ((this.\_Active != value))

{

this.OnActiveChanging(value);

this.SendPropertyChanging();

this.\_Active = value;

this.SendPropertyChanged("Active");

this.OnActiveChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_Name", DbType="VarChar(500) NOT NULL", CanBeNull=false)]

public string Name

{

get

{

return this.\_Name;

}

set

{

if ((this.\_Name != value))

{

this.OnNameChanging(value);

this.SendPropertyChanging();

this.\_Name = value;

this.SendPropertyChanged("Name");

this.OnNameChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_FK\_ThirdLevelSubdivisionTable", DbType="Int NOT NULL")]

public int FK\_ThirdLevelSubdivisionTable

{

get

{

return this.\_FK\_ThirdLevelSubdivisionTable;

}

set

{

if ((this.\_FK\_ThirdLevelSubdivisionTable != value))

{

if (this.\_ThirdLevelSubdivisionTable.HasLoadedOrAssignedValue)

{

throw new System.Data.Linq.ForeignKeyReferenceAlreadyHasValueException();

}

this.OnFK\_ThirdLevelSubdivisionTableChanging(value);

this.SendPropertyChanging();

this.\_FK\_ThirdLevelSubdivisionTable = value;

this.SendPropertyChanged("FK\_ThirdLevelSubdivisionTable");

this.OnFK\_ThirdLevelSubdivisionTableChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_FK\_Specialization", DbType="Int NOT NULL")]

public int FK\_Specialization

{

get

{

return this.\_FK\_Specialization;

}

set

{

if ((this.\_FK\_Specialization != value))

{

if (this.\_SpecializationTable.HasLoadedOrAssignedValue)

{

throw new System.Data.Linq.ForeignKeyReferenceAlreadyHasValueException();

}

this.OnFK\_SpecializationChanging(value);

this.SendPropertyChanging();

this.\_FK\_Specialization = value;

this.SendPropertyChanged("FK\_Specialization");

this.OnFK\_SpecializationChanged();

}

}

}

[global::System.Data.Linq.Mapping.AssociationAttribute(Name="FourthLevelSubdivisionTable\_CollectedBasicParametersTable", Storage="\_CollectedBasicParametersTable", ThisKey="FourthLevelSubdivisionTableID", OtherKey="FK\_FourthLevelSubdivisionTable")]

public EntitySet<CollectedBasicParametersTable> CollectedBasicParametersTable

{

get

{

return this.\_CollectedBasicParametersTable;

}

set

{

this.\_CollectedBasicParametersTable.Assign(value);

}

}

[global::System.Data.Linq.Mapping.AssociationAttribute(Name="FourthLevelSubdivisionTable\_CollectedIndicatorsForR", Storage="\_CollectedIndicatorsForR", ThisKey="FourthLevelSubdivisionTableID", OtherKey="FK\_FourthLelevlSubdivisionTable")]

public EntitySet<CollectedIndicatorsForR> CollectedIndicatorsForR

{

get

{

return this.\_CollectedIndicatorsForR;

}

set

{

this.\_CollectedIndicatorsForR.Assign(value);

}

}

[global::System.Data.Linq.Mapping.AssociationAttribute(Name="FourthLevelSubdivisionTable\_FifthLevelSubdivisionTable", Storage="\_FifthLevelSubdivisionTable", ThisKey="FourthLevelSubdivisionTableID", OtherKey="FK\_FourthLevelSubdivisionTable")]

public EntitySet<FifthLevelSubdivisionTable> FifthLevelSubdivisionTable

{

get

{

return this.\_FifthLevelSubdivisionTable;

}

set

{

this.\_FifthLevelSubdivisionTable.Assign(value);

}

}

[global::System.Data.Linq.Mapping.AssociationAttribute(Name="FourthLevelSubdivisionTable\_FourthLevelParametrs", Storage="\_FourthLevelParametrs", ThisKey="FourthLevelSubdivisionTableID", OtherKey="FourthLevelParametrsID", IsUnique=true, IsForeignKey=false)]

public FourthLevelParametrs FourthLevelParametrs

{

get

{

return this.\_FourthLevelParametrs.Entity;

}

set

{

FourthLevelParametrs previousValue = this.\_FourthLevelParametrs.Entity;

if (((previousValue != value)

|| (this.\_FourthLevelParametrs.HasLoadedOrAssignedValue == false)))

{

this.SendPropertyChanging();

if ((previousValue != null))

{

this.\_FourthLevelParametrs.Entity = null;

previousValue.FourthLevelSubdivisionTable = null;

}

this.\_FourthLevelParametrs.Entity = value;

if ((value != null))

{

value.FourthLevelSubdivisionTable = this;

}

this.SendPropertyChanged("FourthLevelParametrs");

}

}

}

[global::System.Data.Linq.Mapping.AssociationAttribute(Name="FourthLevelSubdivisionTable\_UsersTable", Storage="\_UsersTable", ThisKey="FourthLevelSubdivisionTableID", OtherKey="FK\_FourthLevelSubdivisionTable")]

public EntitySet<UsersTable> UsersTable

{

get

{

return this.\_UsersTable;

}

set

{

this.\_UsersTable.Assign(value);

}

}

[global::System.Data.Linq.Mapping.AssociationAttribute(Name="SpecializationTable\_FourthLevelSubdivisionTable", Storage="\_SpecializationTable", ThisKey="FK\_Specialization", OtherKey="SpecializationTableID", IsForeignKey=true)]

public SpecializationTable SpecializationTable

{

get

{

return this.\_SpecializationTable.Entity;

}

set

{

SpecializationTable previousValue = this.\_SpecializationTable.Entity;

if (((previousValue != value)

|| (this.\_SpecializationTable.HasLoadedOrAssignedValue == false)))

{

this.SendPropertyChanging();

if ((previousValue != null))

{

this.\_SpecializationTable.Entity = null;

previousValue.FourthLevelSubdivisionTable.Remove(this);

}

this.\_SpecializationTable.Entity = value;

if ((value != null))

{

value.FourthLevelSubdivisionTable.Add(this);

this.\_FK\_Specialization = value.SpecializationTableID;

}

else

{

this.\_FK\_Specialization = default(int);

}

this.SendPropertyChanged("SpecializationTable");

}

}

}

[global::System.Data.Linq.Mapping.AssociationAttribute(Name="ThirdLevelSubdivisionTable\_FourthLevelSubdivisionTable", Storage="\_ThirdLevelSubdivisionTable", ThisKey="FK\_ThirdLevelSubdivisionTable", OtherKey="ThirdLevelSubdivisionTableID", IsForeignKey=true)]

public ThirdLevelSubdivisionTable ThirdLevelSubdivisionTable

{

get

{

return this.\_ThirdLevelSubdivisionTable.Entity;

}

set

{

ThirdLevelSubdivisionTable previousValue = this.\_ThirdLevelSubdivisionTable.Entity;

if (((previousValue != value)

|| (this.\_ThirdLevelSubdivisionTable.HasLoadedOrAssignedValue == false)))

{

this.SendPropertyChanging();

if ((previousValue != null))

{

this.\_ThirdLevelSubdivisionTable.Entity = null;

previousValue.FourthLevelSubdivisionTable.Remove(this);

}

this.\_ThirdLevelSubdivisionTable.Entity = value;

if ((value != null))

{

value.FourthLevelSubdivisionTable.Add(this);

this.\_FK\_ThirdLevelSubdivisionTable = value.ThirdLevelSubdivisionTableID;

}

else

{

this.\_FK\_ThirdLevelSubdivisionTable = default(int);

}

this.SendPropertyChanged("ThirdLevelSubdivisionTable");

}

}

}

public event PropertyChangingEventHandler PropertyChanging;

public event PropertyChangedEventHandler PropertyChanged;

protected virtual void SendPropertyChanging()

{

if ((this.PropertyChanging != null))

{

this.PropertyChanging(this, emptyChangingEventArgs);

}

}

protected virtual void SendPropertyChanged(String propertyName)

{

if ((this.PropertyChanged != null))

{

this.PropertyChanged(this, new PropertyChangedEventArgs(propertyName));

}

}

private void attach\_CollectedBasicParametersTable(CollectedBasicParametersTable entity)

{

this.SendPropertyChanging();

entity.FourthLevelSubdivisionTable = this;

}

private void detach\_CollectedBasicParametersTable(CollectedBasicParametersTable entity)

{

this.SendPropertyChanging();

entity.FourthLevelSubdivisionTable = null;

}

private void attach\_CollectedIndicatorsForR(CollectedIndicatorsForR entity)

{

this.SendPropertyChanging();

entity.FourthLevelSubdivisionTable = this;

}

private void detach\_CollectedIndicatorsForR(CollectedIndicatorsForR entity)

{

this.SendPropertyChanging();

entity.FourthLevelSubdivisionTable = null;

}

private void attach\_FifthLevelSubdivisionTable(FifthLevelSubdivisionTable entity)

{

this.SendPropertyChanging();

entity.FourthLevelSubdivisionTable = this;

}

private void detach\_FifthLevelSubdivisionTable(FifthLevelSubdivisionTable entity)

{

this.SendPropertyChanging();

entity.FourthLevelSubdivisionTable = null;

}

private void attach\_UsersTable(UsersTable entity)

{

this.SendPropertyChanging();

entity.FourthLevelSubdivisionTable = this;

}

private void detach\_UsersTable(UsersTable entity)

{

this.SendPropertyChanging();

entity.FourthLevelSubdivisionTable = null;

}

}

[global::System.Data.Linq.Mapping.TableAttribute(Name="dbo.IndicatorClass")]

public partial class IndicatorClass : INotifyPropertyChanging, INotifyPropertyChanged

{

private static PropertyChangingEventArgs emptyChangingEventArgs = new PropertyChangingEventArgs(String.Empty);

private int \_IndicatorClassID;

private string \_ClassName;

private EntityRef<IndicatorClass> \_IndicatorClass2;

private EntitySet<IndicatorsTable> \_IndicatorsTable;

private EntityRef<IndicatorClass> \_IndicatorClass1;

#region Extensibility Method Definitions

partial void OnLoaded();

partial void OnValidate(System.Data.Linq.ChangeAction action);

partial void OnCreated();

partial void OnIndicatorClassIDChanging(int value);

partial void OnIndicatorClassIDChanged();

partial void OnClassNameChanging(string value);

partial void OnClassNameChanged();

#endregion

public IndicatorClass()

{

this.\_IndicatorClass2 = default(EntityRef<IndicatorClass>);

this.\_IndicatorsTable = new EntitySet<IndicatorsTable>(new Action<IndicatorsTable>(this.attach\_IndicatorsTable), new Action<IndicatorsTable>(this.detach\_IndicatorsTable));

this.\_IndicatorClass1 = default(EntityRef<IndicatorClass>);

OnCreated();

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_IndicatorClassID", AutoSync=AutoSync.OnInsert, DbType="Int NOT NULL IDENTITY", IsPrimaryKey=true, IsDbGenerated=true)]

public int IndicatorClassID

{

get

{

return this.\_IndicatorClassID;

}

set

{

if ((this.\_IndicatorClassID != value))

{

if (this.\_IndicatorClass1.HasLoadedOrAssignedValue)

{

throw new System.Data.Linq.ForeignKeyReferenceAlreadyHasValueException();

}

this.OnIndicatorClassIDChanging(value);

this.SendPropertyChanging();

this.\_IndicatorClassID = value;

this.SendPropertyChanged("IndicatorClassID");

this.OnIndicatorClassIDChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_ClassName", DbType="VarChar(500)")]

public string ClassName

{

get

{

return this.\_ClassName;

}

set

{

if ((this.\_ClassName != value))

{

this.OnClassNameChanging(value);

this.SendPropertyChanging();

this.\_ClassName = value;

this.SendPropertyChanged("ClassName");

this.OnClassNameChanged();

}

}

}

[global::System.Data.Linq.Mapping.AssociationAttribute(Name="IndicatorClass\_IndicatorClass", Storage="\_IndicatorClass2", ThisKey="IndicatorClassID", OtherKey="IndicatorClassID", IsUnique=true, IsForeignKey=false)]

public IndicatorClass IndicatorClass2

{

get

{

return this.\_IndicatorClass2.Entity;

}

set

{

IndicatorClass previousValue = this.\_IndicatorClass2.Entity;

if (((previousValue != value)

|| (this.\_IndicatorClass2.HasLoadedOrAssignedValue == false)))

{

this.SendPropertyChanging();

if ((previousValue != null))

{

this.\_IndicatorClass2.Entity = null;

previousValue.IndicatorClass1 = null;

}

this.\_IndicatorClass2.Entity = value;

if ((value != null))

{

value.IndicatorClass1 = this;

}

this.SendPropertyChanged("IndicatorClass2");

}

}

}

[global::System.Data.Linq.Mapping.AssociationAttribute(Name="IndicatorClass\_IndicatorsTable", Storage="\_IndicatorsTable", ThisKey="IndicatorClassID", OtherKey="FK\_IndicatorClass")]

public EntitySet<IndicatorsTable> IndicatorsTable

{

get

{

return this.\_IndicatorsTable;

}

set

{

this.\_IndicatorsTable.Assign(value);

}

}

[global::System.Data.Linq.Mapping.AssociationAttribute(Name="IndicatorClass\_IndicatorClass", Storage="\_IndicatorClass1", ThisKey="IndicatorClassID", OtherKey="IndicatorClassID", IsForeignKey=true)]

public IndicatorClass IndicatorClass1

{

get

{

return this.\_IndicatorClass1.Entity;

}

set

{

IndicatorClass previousValue = this.\_IndicatorClass1.Entity;

if (((previousValue != value)

|| (this.\_IndicatorClass1.HasLoadedOrAssignedValue == false)))

{

this.SendPropertyChanging();

if ((previousValue != null))

{

this.\_IndicatorClass1.Entity = null;

previousValue.IndicatorClass2 = null;

}

this.\_IndicatorClass1.Entity = value;

if ((value != null))

{

value.IndicatorClass2 = this;

this.\_IndicatorClassID = value.IndicatorClassID;

}

else

{

this.\_IndicatorClassID = default(int);

}

this.SendPropertyChanged("IndicatorClass1");

}

}

}

public event PropertyChangingEventHandler PropertyChanging;

public event PropertyChangedEventHandler PropertyChanged;

protected virtual void SendPropertyChanging()

{

if ((this.PropertyChanging != null))

{

this.PropertyChanging(this, emptyChangingEventArgs);

}

}

protected virtual void SendPropertyChanged(String propertyName)

{

if ((this.PropertyChanged != null))

{

this.PropertyChanged(this, new PropertyChangedEventArgs(propertyName));

}

}

private void attach\_IndicatorsTable(IndicatorsTable entity)

{

this.SendPropertyChanging();

entity.IndicatorClass = this;

}

private void detach\_IndicatorsTable(IndicatorsTable entity)

{

this.SendPropertyChanging();

entity.IndicatorClass = null;

}

}

[global::System.Data.Linq.Mapping.TableAttribute(Name="dbo.IndicatorsAndRolesMappingTable")]

public partial class IndicatorsAndRolesMappingTable : INotifyPropertyChanging, INotifyPropertyChanged

{

private static PropertyChangingEventArgs emptyChangingEventArgs = new PropertyChangingEventArgs(String.Empty);

private int \_CalculatedIndicatorsAndRolesWithLeadersID;

private System.Nullable<bool> \_Active;

private System.Nullable<int> \_FK\_RolesTable;

private System.Nullable<int> \_FK\_Indicators;

private System.Nullable<bool> \_CanEdit;

private System.Nullable<bool> \_CanView;

private System.Nullable<bool> \_CanConfirm;

private EntityRef<RolesTable> \_RolesTable;

private EntityRef<IndicatorsTable> \_IndicatorsTable;

#region Extensibility Method Definitions

partial void OnLoaded();

partial void OnValidate(System.Data.Linq.ChangeAction action);

partial void OnCreated();

partial void OnCalculatedIndicatorsAndRolesWithLeadersIDChanging(int value);

partial void OnCalculatedIndicatorsAndRolesWithLeadersIDChanged();

partial void OnActiveChanging(System.Nullable<bool> value);

partial void OnActiveChanged();

partial void OnFK\_RolesTableChanging(System.Nullable<int> value);

partial void OnFK\_RolesTableChanged();

partial void OnFK\_IndicatorsChanging(System.Nullable<int> value);

partial void OnFK\_IndicatorsChanged();

partial void OnCanEditChanging(System.Nullable<bool> value);

partial void OnCanEditChanged();

partial void OnCanViewChanging(System.Nullable<bool> value);

partial void OnCanViewChanged();

partial void OnCanConfirmChanging(System.Nullable<bool> value);

partial void OnCanConfirmChanged();

#endregion

public IndicatorsAndRolesMappingTable()

{

this.\_RolesTable = default(EntityRef<RolesTable>);

this.\_IndicatorsTable = default(EntityRef<IndicatorsTable>);

OnCreated();

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_CalculatedIndicatorsAndRolesWithLeadersID", AutoSync=AutoSync.OnInsert, DbType="Int NOT NULL IDENTITY", IsPrimaryKey=true, IsDbGenerated=true)]

public int CalculatedIndicatorsAndRolesWithLeadersID

{

get

{

return this.\_CalculatedIndicatorsAndRolesWithLeadersID;

}

set

{

if ((this.\_CalculatedIndicatorsAndRolesWithLeadersID != value))

{

this.OnCalculatedIndicatorsAndRolesWithLeadersIDChanging(value);

this.SendPropertyChanging();

this.\_CalculatedIndicatorsAndRolesWithLeadersID = value;

this.SendPropertyChanged("CalculatedIndicatorsAndRolesWithLeadersID");

this.OnCalculatedIndicatorsAndRolesWithLeadersIDChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_Active", DbType="Bit")]

public System.Nullable<bool> Active

{

get

{

return this.\_Active;

}

set

{

if ((this.\_Active != value))

{

this.OnActiveChanging(value);

this.SendPropertyChanging();

this.\_Active = value;

this.SendPropertyChanged("Active");

this.OnActiveChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_FK\_RolesTable", DbType="Int")]

public System.Nullable<int> FK\_RolesTable

{

get

{

return this.\_FK\_RolesTable;

}

set

{

if ((this.\_FK\_RolesTable != value))

{

if (this.\_RolesTable.HasLoadedOrAssignedValue)

{

throw new System.Data.Linq.ForeignKeyReferenceAlreadyHasValueException();

}

this.OnFK\_RolesTableChanging(value);

this.SendPropertyChanging();

this.\_FK\_RolesTable = value;

this.SendPropertyChanged("FK\_RolesTable");

this.OnFK\_RolesTableChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_FK\_Indicators", DbType="Int")]

public System.Nullable<int> FK\_Indicators

{

get

{

return this.\_FK\_Indicators;

}

set

{

if ((this.\_FK\_Indicators != value))

{

if (this.\_IndicatorsTable.HasLoadedOrAssignedValue)

{

throw new System.Data.Linq.ForeignKeyReferenceAlreadyHasValueException();

}

this.OnFK\_IndicatorsChanging(value);

this.SendPropertyChanging();

this.\_FK\_Indicators = value;

this.SendPropertyChanged("FK\_Indicators");

this.OnFK\_IndicatorsChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_CanEdit", DbType="Bit")]

public System.Nullable<bool> CanEdit

{

get

{

return this.\_CanEdit;

}

set

{

if ((this.\_CanEdit != value))

{

this.OnCanEditChanging(value);

this.SendPropertyChanging();

this.\_CanEdit = value;

this.SendPropertyChanged("CanEdit");

this.OnCanEditChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_CanView", DbType="Bit")]

public System.Nullable<bool> CanView

{

get

{

return this.\_CanView;

}

set

{

if ((this.\_CanView != value))

{

this.OnCanViewChanging(value);

this.SendPropertyChanging();

this.\_CanView = value;

this.SendPropertyChanged("CanView");

this.OnCanViewChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_CanConfirm", DbType="Bit")]

public System.Nullable<bool> CanConfirm

{

get

{

return this.\_CanConfirm;

}

set

{

if ((this.\_CanConfirm != value))

{

this.OnCanConfirmChanging(value);

this.SendPropertyChanging();

this.\_CanConfirm = value;

this.SendPropertyChanged("CanConfirm");

this.OnCanConfirmChanged();

}

}

}

[global::System.Data.Linq.Mapping.AssociationAttribute(Name="RolesTable\_IndicatorsAndRolesMappingTable", Storage="\_RolesTable", ThisKey="FK\_RolesTable", OtherKey="RolesTableID", IsForeignKey=true)]

public RolesTable RolesTable

{

get

{

return this.\_RolesTable.Entity;

}

set

{

RolesTable previousValue = this.\_RolesTable.Entity;

if (((previousValue != value)

|| (this.\_RolesTable.HasLoadedOrAssignedValue == false)))

{

this.SendPropertyChanging();

if ((previousValue != null))

{

this.\_RolesTable.Entity = null;

previousValue.IndicatorsAndRolesMappingTable.Remove(this);

}

this.\_RolesTable.Entity = value;

if ((value != null))

{

value.IndicatorsAndRolesMappingTable.Add(this);

this.\_FK\_RolesTable = value.RolesTableID;

}

else

{

this.\_FK\_RolesTable = default(Nullable<int>);

}

this.SendPropertyChanged("RolesTable");

}

}

}

[global::System.Data.Linq.Mapping.AssociationAttribute(Name="IndicatorsTable\_IndicatorsAndRolesMappingTable", Storage="\_IndicatorsTable", ThisKey="FK\_Indicators", OtherKey="IndicatorsTableID", IsForeignKey=true)]

public IndicatorsTable IndicatorsTable

{

get

{

return this.\_IndicatorsTable.Entity;

}

set

{

IndicatorsTable previousValue = this.\_IndicatorsTable.Entity;

if (((previousValue != value)

|| (this.\_IndicatorsTable.HasLoadedOrAssignedValue == false)))

{

this.SendPropertyChanging();

if ((previousValue != null))

{

this.\_IndicatorsTable.Entity = null;

previousValue.IndicatorsAndRolesMappingTable.Remove(this);

}

this.\_IndicatorsTable.Entity = value;

if ((value != null))

{

value.IndicatorsAndRolesMappingTable.Add(this);

this.\_FK\_Indicators = value.IndicatorsTableID;

}

else

{

this.\_FK\_Indicators = default(Nullable<int>);

}

this.SendPropertyChanged("IndicatorsTable");

}

}

}

public event PropertyChangingEventHandler PropertyChanging;

public event PropertyChangedEventHandler PropertyChanged;

protected virtual void SendPropertyChanging()

{

if ((this.PropertyChanging != null))

{

this.PropertyChanging(this, emptyChangingEventArgs);

}

}

protected virtual void SendPropertyChanged(String propertyName)

{

if ((this.PropertyChanged != null))

{

this.PropertyChanged(this, new PropertyChangedEventArgs(propertyName));

}

}

}

[global::System.Data.Linq.Mapping.TableAttribute(Name="dbo.IndicatorsAndUsersMapping")]

public partial class IndicatorsAndUsersMapping : INotifyPropertyChanging, INotifyPropertyChanged

{

private static PropertyChangingEventArgs emptyChangingEventArgs = new PropertyChangingEventArgs(String.Empty);

private int \_IndicatorsAndUsersMappingID;

private bool \_Active;

private System.Nullable<bool> \_CanEdit;

private System.Nullable<bool> \_CanView;

private System.Nullable<bool> \_CanConfirm;

private System.Nullable<int> \_FK\_UsresTable;

private System.Nullable<int> \_FK\_IndicatorsTable;

private EntityRef<UsersTable> \_UsersTable;

private EntityRef<IndicatorsTable> \_IndicatorsTable;

#region Extensibility Method Definitions

partial void OnLoaded();

partial void OnValidate(System.Data.Linq.ChangeAction action);

partial void OnCreated();

partial void OnIndicatorsAndUsersMappingIDChanging(int value);

partial void OnIndicatorsAndUsersMappingIDChanged();

partial void OnActiveChanging(bool value);

partial void OnActiveChanged();

partial void OnCanEditChanging(System.Nullable<bool> value);

partial void OnCanEditChanged();

partial void OnCanViewChanging(System.Nullable<bool> value);

partial void OnCanViewChanged();

partial void OnCanConfirmChanging(System.Nullable<bool> value);

partial void OnCanConfirmChanged();

partial void OnFK\_UsresTableChanging(System.Nullable<int> value);

partial void OnFK\_UsresTableChanged();

partial void OnFK\_IndicatorsTableChanging(System.Nullable<int> value);

partial void OnFK\_IndicatorsTableChanged();

#endregion

public IndicatorsAndUsersMapping()

{

this.\_UsersTable = default(EntityRef<UsersTable>);

this.\_IndicatorsTable = default(EntityRef<IndicatorsTable>);

OnCreated();

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_IndicatorsAndUsersMappingID", AutoSync=AutoSync.OnInsert, DbType="Int NOT NULL IDENTITY", IsPrimaryKey=true, IsDbGenerated=true)]

public int IndicatorsAndUsersMappingID

{

get

{

return this.\_IndicatorsAndUsersMappingID;

}

set

{

if ((this.\_IndicatorsAndUsersMappingID != value))

{

this.OnIndicatorsAndUsersMappingIDChanging(value);

this.SendPropertyChanging();

this.\_IndicatorsAndUsersMappingID = value;

this.SendPropertyChanged("IndicatorsAndUsersMappingID");

this.OnIndicatorsAndUsersMappingIDChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_Active", DbType="Bit NOT NULL")]

public bool Active

{

get

{

return this.\_Active;

}

set

{

if ((this.\_Active != value))

{

this.OnActiveChanging(value);

this.SendPropertyChanging();

this.\_Active = value;

this.SendPropertyChanged("Active");

this.OnActiveChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_CanEdit", DbType="Bit")]

public System.Nullable<bool> CanEdit

{

get

{

return this.\_CanEdit;

}

set

{

if ((this.\_CanEdit != value))

{

this.OnCanEditChanging(value);

this.SendPropertyChanging();

this.\_CanEdit = value;

this.SendPropertyChanged("CanEdit");

this.OnCanEditChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_CanView", DbType="Bit")]

public System.Nullable<bool> CanView

{

get

{

return this.\_CanView;

}

set

{

if ((this.\_CanView != value))

{

this.OnCanViewChanging(value);

this.SendPropertyChanging();

this.\_CanView = value;

this.SendPropertyChanged("CanView");

this.OnCanViewChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_CanConfirm", DbType="Bit")]

public System.Nullable<bool> CanConfirm

{

get

{

return this.\_CanConfirm;

}

set

{

if ((this.\_CanConfirm != value))

{

this.OnCanConfirmChanging(value);

this.SendPropertyChanging();

this.\_CanConfirm = value;

this.SendPropertyChanged("CanConfirm");

this.OnCanConfirmChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_FK\_UsresTable", DbType="Int")]

public System.Nullable<int> FK\_UsresTable

{

get

{

return this.\_FK\_UsresTable;

}

set

{

if ((this.\_FK\_UsresTable != value))

{

if (this.\_UsersTable.HasLoadedOrAssignedValue)

{

throw new System.Data.Linq.ForeignKeyReferenceAlreadyHasValueException();

}

this.OnFK\_UsresTableChanging(value);

this.SendPropertyChanging();

this.\_FK\_UsresTable = value;

this.SendPropertyChanged("FK\_UsresTable");

this.OnFK\_UsresTableChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_FK\_IndicatorsTable", DbType="Int")]

public System.Nullable<int> FK\_IndicatorsTable

{

get

{

return this.\_FK\_IndicatorsTable;

}

set

{

if ((this.\_FK\_IndicatorsTable != value))

{

if (this.\_IndicatorsTable.HasLoadedOrAssignedValue)

{

throw new System.Data.Linq.ForeignKeyReferenceAlreadyHasValueException();

}

this.OnFK\_IndicatorsTableChanging(value);

this.SendPropertyChanging();

this.\_FK\_IndicatorsTable = value;

this.SendPropertyChanged("FK\_IndicatorsTable");

this.OnFK\_IndicatorsTableChanged();

}

}

}

[global::System.Data.Linq.Mapping.AssociationAttribute(Name="UsersTable\_IndicatorsAndUsersMapping", Storage="\_UsersTable", ThisKey="FK\_UsresTable", OtherKey="UsersTableID", IsForeignKey=true)]

public UsersTable UsersTable

{

get

{

return this.\_UsersTable.Entity;

}

set

{

UsersTable previousValue = this.\_UsersTable.Entity;

if (((previousValue != value)

|| (this.\_UsersTable.HasLoadedOrAssignedValue == false)))

{

this.SendPropertyChanging();

if ((previousValue != null))

{

this.\_UsersTable.Entity = null;

previousValue.IndicatorsAndUsersMapping.Remove(this);

}

this.\_UsersTable.Entity = value;

if ((value != null))

{

value.IndicatorsAndUsersMapping.Add(this);

this.\_FK\_UsresTable = value.UsersTableID;

}

else

{

this.\_FK\_UsresTable = default(Nullable<int>);

}

this.SendPropertyChanged("UsersTable");

}

}

}

[global::System.Data.Linq.Mapping.AssociationAttribute(Name="IndicatorsTable\_IndicatorsAndUsersMapping", Storage="\_IndicatorsTable", ThisKey="FK\_IndicatorsTable", OtherKey="IndicatorsTableID", IsForeignKey=true)]

public IndicatorsTable IndicatorsTable

{

get

{

return this.\_IndicatorsTable.Entity;

}

set

{

IndicatorsTable previousValue = this.\_IndicatorsTable.Entity;

if (((previousValue != value)

|| (this.\_IndicatorsTable.HasLoadedOrAssignedValue == false)))

{

this.SendPropertyChanging();

if ((previousValue != null))

{

this.\_IndicatorsTable.Entity = null;

previousValue.IndicatorsAndUsersMapping.Remove(this);

}

this.\_IndicatorsTable.Entity = value;

if ((value != null))

{

value.IndicatorsAndUsersMapping.Add(this);

this.\_FK\_IndicatorsTable = value.IndicatorsTableID;

}

else

{

this.\_FK\_IndicatorsTable = default(Nullable<int>);

}

this.SendPropertyChanged("IndicatorsTable");

}

}

}

public event PropertyChangingEventHandler PropertyChanging;

public event PropertyChangedEventHandler PropertyChanged;

protected virtual void SendPropertyChanging()

{

if ((this.PropertyChanging != null))

{

this.PropertyChanging(this, emptyChangingEventArgs);

}

}

protected virtual void SendPropertyChanged(String propertyName)

{

if ((this.PropertyChanged != null))

{

this.PropertyChanged(this, new PropertyChangedEventArgs(propertyName));

}

}

}

[global::System.Data.Linq.Mapping.TableAttribute(Name="dbo.ManualTable")]

public partial class ManualTable : INotifyPropertyChanging, INotifyPropertyChanged

{

private static PropertyChangingEventArgs emptyChangingEventArgs = new PropertyChangingEventArgs(String.Empty);

private int \_ManualID;

private System.Nullable<bool> \_Active;

private string \_ManualName;

private string \_ManualLink;

#region Extensibility Method Definitions

partial void OnLoaded();

partial void OnValidate(System.Data.Linq.ChangeAction action);

partial void OnCreated();

partial void OnManualIDChanging(int value);

partial void OnManualIDChanged();

partial void OnActiveChanging(System.Nullable<bool> value);

partial void OnActiveChanged();

partial void OnManualNameChanging(string value);

partial void OnManualNameChanged();

partial void OnManualLinkChanging(string value);

partial void OnManualLinkChanged();

#endregion

public ManualTable()

{

OnCreated();

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_ManualID", AutoSync=AutoSync.OnInsert, DbType="Int NOT NULL IDENTITY", IsPrimaryKey=true, IsDbGenerated=true)]

public int ManualID

{

get

{

return this.\_ManualID;

}

set

{

if ((this.\_ManualID != value))

{

this.OnManualIDChanging(value);

this.SendPropertyChanging();

this.\_ManualID = value;

this.SendPropertyChanged("ManualID");

this.OnManualIDChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_Active", DbType="Bit")]

public System.Nullable<bool> Active

{

get

{

return this.\_Active;

}

set

{

if ((this.\_Active != value))

{

this.OnActiveChanging(value);

this.SendPropertyChanging();

this.\_Active = value;

this.SendPropertyChanged("Active");

this.OnActiveChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_ManualName", DbType="VarChar(200)")]

public string ManualName

{

get

{

return this.\_ManualName;

}

set

{

if ((this.\_ManualName != value))

{

this.OnManualNameChanging(value);

this.SendPropertyChanging();

this.\_ManualName = value;

this.SendPropertyChanged("ManualName");

this.OnManualNameChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_ManualLink", DbType="VarChar(200)")]

public string ManualLink

{

get

{

return this.\_ManualLink;

}

set

{

if ((this.\_ManualLink != value))

{

this.OnManualLinkChanging(value);

this.SendPropertyChanging();

this.\_ManualLink = value;

this.SendPropertyChanged("ManualLink");

this.OnManualLinkChanged();

}

}

}

public event PropertyChangingEventHandler PropertyChanging;

public event PropertyChangedEventHandler PropertyChanged;

protected virtual void SendPropertyChanging()

{

if ((this.PropertyChanging != null))

{

this.PropertyChanging(this, emptyChangingEventArgs);

}

}

protected virtual void SendPropertyChanged(String propertyName)

{

if ((this.PropertyChanged != null))

{

this.PropertyChanged(this, new PropertyChangedEventArgs(propertyName));

}

}

}

[global::System.Data.Linq.Mapping.TableAttribute(Name="dbo.MultiUser")]

public partial class MultiUser : INotifyPropertyChanging, INotifyPropertyChanged

{

private static PropertyChangingEventArgs emptyChangingEventArgs = new PropertyChangingEventArgs(String.Empty);

private int \_MultiUserID;

private int \_FK\_UserCanAccess;

private int \_FK\_UserToAccess;

private bool \_Active;

private EntityRef<UsersTable> \_UsersTable;

private EntityRef<UsersTable> \_UsersTable1;

#region Extensibility Method Definitions

partial void OnLoaded();

partial void OnValidate(System.Data.Linq.ChangeAction action);

partial void OnCreated();

partial void OnMultiUserIDChanging(int value);

partial void OnMultiUserIDChanged();

partial void OnFK\_UserCanAccessChanging(int value);

partial void OnFK\_UserCanAccessChanged();

partial void OnFK\_UserToAccessChanging(int value);

partial void OnFK\_UserToAccessChanged();

partial void OnActiveChanging(bool value);

partial void OnActiveChanged();

#endregion

public MultiUser()

{

this.\_UsersTable = default(EntityRef<UsersTable>);

this.\_UsersTable1 = default(EntityRef<UsersTable>);

OnCreated();

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_MultiUserID", AutoSync=AutoSync.OnInsert, DbType="Int NOT NULL IDENTITY", IsPrimaryKey=true, IsDbGenerated=true)]

public int MultiUserID

{

get

{

return this.\_MultiUserID;

}

set

{

if ((this.\_MultiUserID != value))

{

this.OnMultiUserIDChanging(value);

this.SendPropertyChanging();

this.\_MultiUserID = value;

this.SendPropertyChanged("MultiUserID");

this.OnMultiUserIDChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_FK\_UserCanAccess", DbType="Int NOT NULL")]

public int FK\_UserCanAccess

{

get

{

return this.\_FK\_UserCanAccess;

}

set

{

if ((this.\_FK\_UserCanAccess != value))

{

if (this.\_UsersTable.HasLoadedOrAssignedValue)

{

throw new System.Data.Linq.ForeignKeyReferenceAlreadyHasValueException();

}

this.OnFK\_UserCanAccessChanging(value);

this.SendPropertyChanging();

this.\_FK\_UserCanAccess = value;

this.SendPropertyChanged("FK\_UserCanAccess");

this.OnFK\_UserCanAccessChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_FK\_UserToAccess", DbType="Int NOT NULL")]

public int FK\_UserToAccess

{

get

{

return this.\_FK\_UserToAccess;

}

set

{

if ((this.\_FK\_UserToAccess != value))

{

if (this.\_UsersTable1.HasLoadedOrAssignedValue)

{

throw new System.Data.Linq.ForeignKeyReferenceAlreadyHasValueException();

}

this.OnFK\_UserToAccessChanging(value);

this.SendPropertyChanging();

this.\_FK\_UserToAccess = value;

this.SendPropertyChanged("FK\_UserToAccess");

this.OnFK\_UserToAccessChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_Active", DbType="Bit NOT NULL")]

public bool Active

{

get

{

return this.\_Active;

}

set

{

if ((this.\_Active != value))

{

this.OnActiveChanging(value);

this.SendPropertyChanging();

this.\_Active = value;

this.SendPropertyChanged("Active");

this.OnActiveChanged();

}

}

}

[global::System.Data.Linq.Mapping.AssociationAttribute(Name="UsersTable\_MultiUser", Storage="\_UsersTable", ThisKey="FK\_UserCanAccess", OtherKey="UsersTableID", IsForeignKey=true)]

public UsersTable UsersTable

{

get

{

return this.\_UsersTable.Entity;

}

set

{

UsersTable previousValue = this.\_UsersTable.Entity;

if (((previousValue != value)

|| (this.\_UsersTable.HasLoadedOrAssignedValue == false)))

{

this.SendPropertyChanging();

if ((previousValue != null))

{

this.\_UsersTable.Entity = null;

previousValue.MultiUser.Remove(this);

}

this.\_UsersTable.Entity = value;

if ((value != null))

{

value.MultiUser.Add(this);

this.\_FK\_UserCanAccess = value.UsersTableID;

}

else

{

this.\_FK\_UserCanAccess = default(int);

}

this.SendPropertyChanged("UsersTable");

}

}

}

[global::System.Data.Linq.Mapping.AssociationAttribute(Name="UsersTable\_MultiUser1", Storage="\_UsersTable1", ThisKey="FK\_UserToAccess", OtherKey="UsersTableID", IsForeignKey=true)]

public UsersTable UsersTable1

{

get

{

return this.\_UsersTable1.Entity;

}

set

{

UsersTable previousValue = this.\_UsersTable1.Entity;

if (((previousValue != value)

|| (this.\_UsersTable1.HasLoadedOrAssignedValue == false)))

{

this.SendPropertyChanging();

if ((previousValue != null))

{

this.\_UsersTable1.Entity = null;

previousValue.MultiUser1.Remove(this);

}

this.\_UsersTable1.Entity = value;

if ((value != null))

{

value.MultiUser1.Add(this);

this.\_FK\_UserToAccess = value.UsersTableID;

}

else

{

this.\_FK\_UserToAccess = default(int);

}

this.SendPropertyChanged("UsersTable1");

}

}

}

public event PropertyChangingEventHandler PropertyChanging;

public event PropertyChangedEventHandler PropertyChanged;

protected virtual void SendPropertyChanging()

{

if ((this.PropertyChanging != null))

{

this.PropertyChanging(this, emptyChangingEventArgs);

}

}

protected virtual void SendPropertyChanged(String propertyName)

{

if ((this.PropertyChanged != null))

{

this.PropertyChanged(this, new PropertyChangedEventArgs(propertyName));

}

}

}

[global::System.Data.Linq.Mapping.TableAttribute(Name="dbo.PlannedIndicator")]

public partial class PlannedIndicator : INotifyPropertyChanging, INotifyPropertyChanged

{

private static PropertyChangingEventArgs emptyChangingEventArgs = new PropertyChangingEventArgs(String.Empty);

private int \_PlanedIndicatorID;

private bool \_Active;

private System.Nullable<double> \_Value;

private System.Nullable<System.DateTime> \_Date;

private int \_FK\_IndicatorsTable;

private EntityRef<IndicatorsTable> \_IndicatorsTable;

#region Extensibility Method Definitions

partial void OnLoaded();

partial void OnValidate(System.Data.Linq.ChangeAction action);

partial void OnCreated();

partial void OnPlanedIndicatorIDChanging(int value);

partial void OnPlanedIndicatorIDChanged();

partial void OnActiveChanging(bool value);

partial void OnActiveChanged();

partial void OnValueChanging(System.Nullable<double> value);

partial void OnValueChanged();

partial void OnDateChanging(System.Nullable<System.DateTime> value);

partial void OnDateChanged();

partial void OnFK\_IndicatorsTableChanging(int value);

partial void OnFK\_IndicatorsTableChanged();

#endregion

public PlannedIndicator()

{

this.\_IndicatorsTable = default(EntityRef<IndicatorsTable>);

OnCreated();

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_PlanedIndicatorID", AutoSync=AutoSync.OnInsert, DbType="Int NOT NULL IDENTITY", IsPrimaryKey=true, IsDbGenerated=true)]

public int PlanedIndicatorID

{

get

{

return this.\_PlanedIndicatorID;

}

set

{

if ((this.\_PlanedIndicatorID != value))

{

this.OnPlanedIndicatorIDChanging(value);

this.SendPropertyChanging();

this.\_PlanedIndicatorID = value;

this.SendPropertyChanged("PlanedIndicatorID");

this.OnPlanedIndicatorIDChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_Active", DbType="Bit NOT NULL")]

public bool Active

{

get

{

return this.\_Active;

}

set

{

if ((this.\_Active != value))

{

this.OnActiveChanging(value);

this.SendPropertyChanging();

this.\_Active = value;

this.SendPropertyChanged("Active");

this.OnActiveChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_Value", DbType="Float")]

public System.Nullable<double> Value

{

get

{

return this.\_Value;

}

set

{

if ((this.\_Value != value))

{

this.OnValueChanging(value);

this.SendPropertyChanging();

this.\_Value = value;

this.SendPropertyChanged("Value");

this.OnValueChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_Date", DbType="DateTime")]

public System.Nullable<System.DateTime> Date

{

get

{

return this.\_Date;

}

set

{

if ((this.\_Date != value))

{

this.OnDateChanging(value);

this.SendPropertyChanging();

this.\_Date = value;

this.SendPropertyChanged("Date");

this.OnDateChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_FK\_IndicatorsTable", DbType="Int NOT NULL")]

public int FK\_IndicatorsTable

{

get

{

return this.\_FK\_IndicatorsTable;

}

set

{

if ((this.\_FK\_IndicatorsTable != value))

{

if (this.\_IndicatorsTable.HasLoadedOrAssignedValue)

{

throw new System.Data.Linq.ForeignKeyReferenceAlreadyHasValueException();

}

this.OnFK\_IndicatorsTableChanging(value);

this.SendPropertyChanging();

this.\_FK\_IndicatorsTable = value;

this.SendPropertyChanged("FK\_IndicatorsTable");

this.OnFK\_IndicatorsTableChanged();

}

}

}

[global::System.Data.Linq.Mapping.AssociationAttribute(Name="IndicatorsTable\_PlannedIndicator", Storage="\_IndicatorsTable", ThisKey="FK\_IndicatorsTable", OtherKey="IndicatorsTableID", IsForeignKey=true)]

public IndicatorsTable IndicatorsTable

{

get

{

return this.\_IndicatorsTable.Entity;

}

set

{

IndicatorsTable previousValue = this.\_IndicatorsTable.Entity;

if (((previousValue != value)

|| (this.\_IndicatorsTable.HasLoadedOrAssignedValue == false)))

{

this.SendPropertyChanging();

if ((previousValue != null))

{

this.\_IndicatorsTable.Entity = null;

previousValue.PlannedIndicator.Remove(this);

}

this.\_IndicatorsTable.Entity = value;

if ((value != null))

{

value.PlannedIndicator.Add(this);

this.\_FK\_IndicatorsTable = value.IndicatorsTableID;

}

else

{

this.\_FK\_IndicatorsTable = default(int);

}

this.SendPropertyChanged("IndicatorsTable");

}

}

}

public event PropertyChangingEventHandler PropertyChanging;

public event PropertyChangedEventHandler PropertyChanged;

protected virtual void SendPropertyChanging()

{

if ((this.PropertyChanging != null))

{

this.PropertyChanging(this, emptyChangingEventArgs);

}

}

protected virtual void SendPropertyChanged(String propertyName)

{

if ((this.PropertyChanged != null))

{

this.PropertyChanged(this, new PropertyChangedEventArgs(propertyName));

}

}

}

[global::System.Data.Linq.Mapping.TableAttribute(Name="dbo.ReportArchiveAndBasicParametrsMappingTable")]

public partial class ReportArchiveAndBasicParametrsMappingTable : INotifyPropertyChanging, INotifyPropertyChanged

{

private static PropertyChangingEventArgs emptyChangingEventArgs = new PropertyChangingEventArgs(String.Empty);

private int \_ReportArchiveAndBasicParametrsMappingTableID;

private bool \_Active;

private int \_FK\_ReportArchiveTable;

private int \_FK\_BasicParametrsTable;

private EntityRef<BasicParametersTable> \_BasicParametersTable;

private EntityRef<ReportArchiveTable> \_ReportArchiveTable;

#region Extensibility Method Definitions

partial void OnLoaded();

partial void OnValidate(System.Data.Linq.ChangeAction action);

partial void OnCreated();

partial void OnReportArchiveAndBasicParametrsMappingTableIDChanging(int value);

partial void OnReportArchiveAndBasicParametrsMappingTableIDChanged();

partial void OnActiveChanging(bool value);

partial void OnActiveChanged();

partial void OnFK\_ReportArchiveTableChanging(int value);

partial void OnFK\_ReportArchiveTableChanged();

partial void OnFK\_BasicParametrsTableChanging(int value);

partial void OnFK\_BasicParametrsTableChanged();

#endregion

public ReportArchiveAndBasicParametrsMappingTable()

{

this.\_BasicParametersTable = default(EntityRef<BasicParametersTable>);

this.\_ReportArchiveTable = default(EntityRef<ReportArchiveTable>);

OnCreated();

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_ReportArchiveAndBasicParametrsMappingTableID", AutoSync=AutoSync.OnInsert, DbType="Int NOT NULL IDENTITY", IsPrimaryKey=true, IsDbGenerated=true)]

public int ReportArchiveAndBasicParametrsMappingTableID

{

get

{

return this.\_ReportArchiveAndBasicParametrsMappingTableID;

}

set

{

if ((this.\_ReportArchiveAndBasicParametrsMappingTableID != value))

{

this.OnReportArchiveAndBasicParametrsMappingTableIDChanging(value);

this.SendPropertyChanging();

this.\_ReportArchiveAndBasicParametrsMappingTableID = value;

this.SendPropertyChanged("ReportArchiveAndBasicParametrsMappingTableID");

this.OnReportArchiveAndBasicParametrsMappingTableIDChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_Active", DbType="Bit NOT NULL")]

public bool Active

{

get

{

return this.\_Active;

}

set

{

if ((this.\_Active != value))

{

this.OnActiveChanging(value);

this.SendPropertyChanging();

this.\_Active = value;

this.SendPropertyChanged("Active");

this.OnActiveChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_FK\_ReportArchiveTable", DbType="Int NOT NULL")]

public int FK\_ReportArchiveTable

{

get

{

return this.\_FK\_ReportArchiveTable;

}

set

{

if ((this.\_FK\_ReportArchiveTable != value))

{

if (this.\_ReportArchiveTable.HasLoadedOrAssignedValue)

{

throw new System.Data.Linq.ForeignKeyReferenceAlreadyHasValueException();

}

this.OnFK\_ReportArchiveTableChanging(value);

this.SendPropertyChanging();

this.\_FK\_ReportArchiveTable = value;

this.SendPropertyChanged("FK\_ReportArchiveTable");

this.OnFK\_ReportArchiveTableChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_FK\_BasicParametrsTable", DbType="Int NOT NULL")]

public int FK\_BasicParametrsTable

{

get

{

return this.\_FK\_BasicParametrsTable;

}

set

{

if ((this.\_FK\_BasicParametrsTable != value))

{

if (this.\_BasicParametersTable.HasLoadedOrAssignedValue)

{

throw new System.Data.Linq.ForeignKeyReferenceAlreadyHasValueException();

}

this.OnFK\_BasicParametrsTableChanging(value);

this.SendPropertyChanging();

this.\_FK\_BasicParametrsTable = value;

this.SendPropertyChanged("FK\_BasicParametrsTable");

this.OnFK\_BasicParametrsTableChanged();

}

}

}

[global::System.Data.Linq.Mapping.AssociationAttribute(Name="BasicParametersTable\_ReportArchiveAndBasicParametrsMappingTable", Storage="\_BasicParametersTable", ThisKey="FK\_BasicParametrsTable", OtherKey="BasicParametersTableID", IsForeignKey=true)]

public BasicParametersTable BasicParametersTable

{

get

{

return this.\_BasicParametersTable.Entity;

}

set

{

BasicParametersTable previousValue = this.\_BasicParametersTable.Entity;

if (((previousValue != value)

|| (this.\_BasicParametersTable.HasLoadedOrAssignedValue == false)))

{

this.SendPropertyChanging();

if ((previousValue != null))

{

this.\_BasicParametersTable.Entity = null;

previousValue.ReportArchiveAndBasicParametrsMappingTable.Remove(this);

}

this.\_BasicParametersTable.Entity = value;

if ((value != null))

{

value.ReportArchiveAndBasicParametrsMappingTable.Add(this);

this.\_FK\_BasicParametrsTable = value.BasicParametersTableID;

}

else

{

this.\_FK\_BasicParametrsTable = default(int);

}

this.SendPropertyChanged("BasicParametersTable");

}

}

}

[global::System.Data.Linq.Mapping.AssociationAttribute(Name="ReportArchiveTable\_ReportArchiveAndBasicParametrsMappingTable", Storage="\_ReportArchiveTable", ThisKey="FK\_ReportArchiveTable", OtherKey="ReportArchiveTableID", IsForeignKey=true)]

public ReportArchiveTable ReportArchiveTable

{

get

{

return this.\_ReportArchiveTable.Entity;

}

set

{

ReportArchiveTable previousValue = this.\_ReportArchiveTable.Entity;

if (((previousValue != value)

|| (this.\_ReportArchiveTable.HasLoadedOrAssignedValue == false)))

{

this.SendPropertyChanging();

if ((previousValue != null))

{

this.\_ReportArchiveTable.Entity = null;

previousValue.ReportArchiveAndBasicParametrsMappingTable.Remove(this);

}

this.\_ReportArchiveTable.Entity = value;

if ((value != null))

{

value.ReportArchiveAndBasicParametrsMappingTable.Add(this);

this.\_FK\_ReportArchiveTable = value.ReportArchiveTableID;

}

else

{

this.\_FK\_ReportArchiveTable = default(int);

}

this.SendPropertyChanged("ReportArchiveTable");

}

}

}

public event PropertyChangingEventHandler PropertyChanging;

public event PropertyChangedEventHandler PropertyChanged;

protected virtual void SendPropertyChanging()

{

if ((this.PropertyChanging != null))

{

this.PropertyChanging(this, emptyChangingEventArgs);

}

}

protected virtual void SendPropertyChanged(String propertyName)

{

if ((this.PropertyChanged != null))

{

this.PropertyChanged(this, new PropertyChangedEventArgs(propertyName));

}

}

}

[global::System.Data.Linq.Mapping.TableAttribute(Name="dbo.ReportArchiveAndCalculatedParametrsMappingTable")]

public partial class ReportArchiveAndCalculatedParametrsMappingTable : INotifyPropertyChanging, INotifyPropertyChanged

{

private static PropertyChangingEventArgs emptyChangingEventArgs = new PropertyChangingEventArgs(String.Empty);

private int \_ReportArchiveAndCalculatedParametrsMappingTableID;

private bool \_Active;

private int \_FK\_ReportArchiveTable;

private int \_FK\_CalculatedParametrsTable;

private EntityRef<CalculatedParametrs> \_CalculatedParametrs;

private EntityRef<ReportArchiveTable> \_ReportArchiveTable;

#region Extensibility Method Definitions

partial void OnLoaded();

partial void OnValidate(System.Data.Linq.ChangeAction action);

partial void OnCreated();

partial void OnReportArchiveAndCalculatedParametrsMappingTableIDChanging(int value);

partial void OnReportArchiveAndCalculatedParametrsMappingTableIDChanged();

partial void OnActiveChanging(bool value);

partial void OnActiveChanged();

partial void OnFK\_ReportArchiveTableChanging(int value);

partial void OnFK\_ReportArchiveTableChanged();

partial void OnFK\_CalculatedParametrsTableChanging(int value);

partial void OnFK\_CalculatedParametrsTableChanged();

#endregion

public ReportArchiveAndCalculatedParametrsMappingTable()

{

this.\_CalculatedParametrs = default(EntityRef<CalculatedParametrs>);

this.\_ReportArchiveTable = default(EntityRef<ReportArchiveTable>);

OnCreated();

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_ReportArchiveAndCalculatedParametrsMappingTableID", AutoSync=AutoSync.OnInsert, DbType="Int NOT NULL IDENTITY", IsPrimaryKey=true, IsDbGenerated=true)]

public int ReportArchiveAndCalculatedParametrsMappingTableID

{

get

{

return this.\_ReportArchiveAndCalculatedParametrsMappingTableID;

}

set

{

if ((this.\_ReportArchiveAndCalculatedParametrsMappingTableID != value))

{

this.OnReportArchiveAndCalculatedParametrsMappingTableIDChanging(value);

this.SendPropertyChanging();

this.\_ReportArchiveAndCalculatedParametrsMappingTableID = value;

this.SendPropertyChanged("ReportArchiveAndCalculatedParametrsMappingTableID");

this.OnReportArchiveAndCalculatedParametrsMappingTableIDChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_Active", DbType="Bit NOT NULL")]

public bool Active

{

get

{

return this.\_Active;

}

set

{

if ((this.\_Active != value))

{

this.OnActiveChanging(value);

this.SendPropertyChanging();

this.\_Active = value;

this.SendPropertyChanged("Active");

this.OnActiveChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_FK\_ReportArchiveTable", DbType="Int NOT NULL")]

public int FK\_ReportArchiveTable

{

get

{

return this.\_FK\_ReportArchiveTable;

}

set

{

if ((this.\_FK\_ReportArchiveTable != value))

{

if (this.\_ReportArchiveTable.HasLoadedOrAssignedValue)

{

throw new System.Data.Linq.ForeignKeyReferenceAlreadyHasValueException();

}

this.OnFK\_ReportArchiveTableChanging(value);

this.SendPropertyChanging();

this.\_FK\_ReportArchiveTable = value;

this.SendPropertyChanged("FK\_ReportArchiveTable");

this.OnFK\_ReportArchiveTableChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_FK\_CalculatedParametrsTable", DbType="Int NOT NULL")]

public int FK\_CalculatedParametrsTable

{

get

{

return this.\_FK\_CalculatedParametrsTable;

}

set

{

if ((this.\_FK\_CalculatedParametrsTable != value))

{

if (this.\_CalculatedParametrs.HasLoadedOrAssignedValue)

{

throw new System.Data.Linq.ForeignKeyReferenceAlreadyHasValueException();

}

this.OnFK\_CalculatedParametrsTableChanging(value);

this.SendPropertyChanging();

this.\_FK\_CalculatedParametrsTable = value;

this.SendPropertyChanged("FK\_CalculatedParametrsTable");

this.OnFK\_CalculatedParametrsTableChanged();

}

}

}

[global::System.Data.Linq.Mapping.AssociationAttribute(Name="CalculatedParametrs\_ReportArchiveAndCalculatedParametrsMappingTable", Storage="\_CalculatedParametrs", ThisKey="FK\_CalculatedParametrsTable", OtherKey="CalculatedParametrsID", IsForeignKey=true)]

public CalculatedParametrs CalculatedParametrs

{

get

{

return this.\_CalculatedParametrs.Entity;

}

set

{

CalculatedParametrs previousValue = this.\_CalculatedParametrs.Entity;

if (((previousValue != value)

|| (this.\_CalculatedParametrs.HasLoadedOrAssignedValue == false)))

{

this.SendPropertyChanging();

if ((previousValue != null))

{

this.\_CalculatedParametrs.Entity = null;

previousValue.ReportArchiveAndCalculatedParametrsMappingTable.Remove(this);

}

this.\_CalculatedParametrs.Entity = value;

if ((value != null))

{

value.ReportArchiveAndCalculatedParametrsMappingTable.Add(this);

this.\_FK\_CalculatedParametrsTable = value.CalculatedParametrsID;

}

else

{

this.\_FK\_CalculatedParametrsTable = default(int);

}

this.SendPropertyChanged("CalculatedParametrs");

}

}

}

[global::System.Data.Linq.Mapping.AssociationAttribute(Name="ReportArchiveTable\_ReportArchiveAndCalculatedParametrsMappingTable", Storage="\_ReportArchiveTable", ThisKey="FK\_ReportArchiveTable", OtherKey="ReportArchiveTableID", IsForeignKey=true)]

public ReportArchiveTable ReportArchiveTable

{

get

{

return this.\_ReportArchiveTable.Entity;

}

set

{

ReportArchiveTable previousValue = this.\_ReportArchiveTable.Entity;

if (((previousValue != value)

|| (this.\_ReportArchiveTable.HasLoadedOrAssignedValue == false)))

{

this.SendPropertyChanging();

if ((previousValue != null))

{

this.\_ReportArchiveTable.Entity = null;

previousValue.ReportArchiveAndCalculatedParametrsMappingTable.Remove(this);

}

this.\_ReportArchiveTable.Entity = value;

if ((value != null))

{

value.ReportArchiveAndCalculatedParametrsMappingTable.Add(this);

this.\_FK\_ReportArchiveTable = value.ReportArchiveTableID;

}

else

{

this.\_FK\_ReportArchiveTable = default(int);

}

this.SendPropertyChanged("ReportArchiveTable");

}

}

}

public event PropertyChangingEventHandler PropertyChanging;

public event PropertyChangedEventHandler PropertyChanged;

protected virtual void SendPropertyChanging()

{

if ((this.PropertyChanging != null))

{

this.PropertyChanging(this, emptyChangingEventArgs);

}

}

protected virtual void SendPropertyChanged(String propertyName)

{

if ((this.PropertyChanged != null))

{

this.PropertyChanged(this, new PropertyChangedEventArgs(propertyName));

}

}

}

[global::System.Data.Linq.Mapping.TableAttribute(Name="dbo.ReportArchiveAndIndicatorsMappingTable")]

public partial class ReportArchiveAndIndicatorsMappingTable : INotifyPropertyChanging, INotifyPropertyChanged

{

private static PropertyChangingEventArgs emptyChangingEventArgs = new PropertyChangingEventArgs(String.Empty);

private int \_ReportArchiveAndIndicatorsMappingTableID;

private bool \_Active;

private int \_FK\_ReportArchiveTable;

private int \_FK\_IndicatorsTable;

private EntityRef<ReportArchiveTable> \_ReportArchiveTable;

private EntityRef<IndicatorsTable> \_IndicatorsTable;

#region Extensibility Method Definitions

partial void OnLoaded();

partial void OnValidate(System.Data.Linq.ChangeAction action);

partial void OnCreated();

partial void OnReportArchiveAndIndicatorsMappingTableIDChanging(int value);

partial void OnReportArchiveAndIndicatorsMappingTableIDChanged();

partial void OnActiveChanging(bool value);

partial void OnActiveChanged();

partial void OnFK\_ReportArchiveTableChanging(int value);

partial void OnFK\_ReportArchiveTableChanged();

partial void OnFK\_IndicatorsTableChanging(int value);

partial void OnFK\_IndicatorsTableChanged();

#endregion

public ReportArchiveAndIndicatorsMappingTable()

{

this.\_ReportArchiveTable = default(EntityRef<ReportArchiveTable>);

this.\_IndicatorsTable = default(EntityRef<IndicatorsTable>);

OnCreated();

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_ReportArchiveAndIndicatorsMappingTableID", AutoSync=AutoSync.OnInsert, DbType="Int NOT NULL IDENTITY", IsPrimaryKey=true, IsDbGenerated=true)]

public int ReportArchiveAndIndicatorsMappingTableID

{

get

{

return this.\_ReportArchiveAndIndicatorsMappingTableID;

}

set

{

if ((this.\_ReportArchiveAndIndicatorsMappingTableID != value))

{

this.OnReportArchiveAndIndicatorsMappingTableIDChanging(value);

this.SendPropertyChanging();

this.\_ReportArchiveAndIndicatorsMappingTableID = value;

this.SendPropertyChanged("ReportArchiveAndIndicatorsMappingTableID");

this.OnReportArchiveAndIndicatorsMappingTableIDChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_Active", DbType="Bit NOT NULL")]

public bool Active

{

get

{

return this.\_Active;

}

set

{

if ((this.\_Active != value))

{

this.OnActiveChanging(value);

this.SendPropertyChanging();

this.\_Active = value;

this.SendPropertyChanged("Active");

this.OnActiveChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_FK\_ReportArchiveTable", DbType="Int NOT NULL")]

public int FK\_ReportArchiveTable

{

get

{

return this.\_FK\_ReportArchiveTable;

}

set

{

if ((this.\_FK\_ReportArchiveTable != value))

{

if (this.\_ReportArchiveTable.HasLoadedOrAssignedValue)

{

throw new System.Data.Linq.ForeignKeyReferenceAlreadyHasValueException();

}

this.OnFK\_ReportArchiveTableChanging(value);

this.SendPropertyChanging();

this.\_FK\_ReportArchiveTable = value;

this.SendPropertyChanged("FK\_ReportArchiveTable");

this.OnFK\_ReportArchiveTableChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_FK\_IndicatorsTable", DbType="Int NOT NULL")]

public int FK\_IndicatorsTable

{

get

{

return this.\_FK\_IndicatorsTable;

}

set

{

if ((this.\_FK\_IndicatorsTable != value))

{

if (this.\_IndicatorsTable.HasLoadedOrAssignedValue)

{

throw new System.Data.Linq.ForeignKeyReferenceAlreadyHasValueException();

}

this.OnFK\_IndicatorsTableChanging(value);

this.SendPropertyChanging();

this.\_FK\_IndicatorsTable = value;

this.SendPropertyChanged("FK\_IndicatorsTable");

this.OnFK\_IndicatorsTableChanged();

}

}

}

[global::System.Data.Linq.Mapping.AssociationAttribute(Name="ReportArchiveTable\_ReportArchiveAndIndicatorsMappingTable", Storage="\_ReportArchiveTable", ThisKey="FK\_ReportArchiveTable", OtherKey="ReportArchiveTableID", IsForeignKey=true)]

public ReportArchiveTable ReportArchiveTable

{

get

{

return this.\_ReportArchiveTable.Entity;

}

set

{

ReportArchiveTable previousValue = this.\_ReportArchiveTable.Entity;

if (((previousValue != value)

|| (this.\_ReportArchiveTable.HasLoadedOrAssignedValue == false)))

{

this.SendPropertyChanging();

if ((previousValue != null))

{

this.\_ReportArchiveTable.Entity = null;

previousValue.ReportArchiveAndIndicatorsMappingTable.Remove(this);

}

this.\_ReportArchiveTable.Entity = value;

if ((value != null))

{

value.ReportArchiveAndIndicatorsMappingTable.Add(this);

this.\_FK\_ReportArchiveTable = value.ReportArchiveTableID;

}

else

{

this.\_FK\_ReportArchiveTable = default(int);

}

this.SendPropertyChanged("ReportArchiveTable");

}

}

}

[global::System.Data.Linq.Mapping.AssociationAttribute(Name="IndicatorsTable\_ReportArchiveAndIndicatorsMappingTable", Storage="\_IndicatorsTable", ThisKey="FK\_IndicatorsTable", OtherKey="IndicatorsTableID", IsForeignKey=true)]

public IndicatorsTable IndicatorsTable

{

get

{

return this.\_IndicatorsTable.Entity;

}

set

{

IndicatorsTable previousValue = this.\_IndicatorsTable.Entity;

if (((previousValue != value)

|| (this.\_IndicatorsTable.HasLoadedOrAssignedValue == false)))

{

this.SendPropertyChanging();

if ((previousValue != null))

{

this.\_IndicatorsTable.Entity = null;

previousValue.ReportArchiveAndIndicatorsMappingTable.Remove(this);

}

this.\_IndicatorsTable.Entity = value;

if ((value != null))

{

value.ReportArchiveAndIndicatorsMappingTable.Add(this);

this.\_FK\_IndicatorsTable = value.IndicatorsTableID;

}

else

{

this.\_FK\_IndicatorsTable = default(int);

}

this.SendPropertyChanged("IndicatorsTable");

}

}

}

public event PropertyChangingEventHandler PropertyChanging;

public event PropertyChangedEventHandler PropertyChanged;

protected virtual void SendPropertyChanging()

{

if ((this.PropertyChanging != null))

{

this.PropertyChanging(this, emptyChangingEventArgs);

}

}

protected virtual void SendPropertyChanged(String propertyName)

{

if ((this.PropertyChanged != null))

{

this.PropertyChanged(this, new PropertyChangedEventArgs(propertyName));

}

}

}

[global::System.Data.Linq.Mapping.TableAttribute(Name="dbo.ReportArchiveAndLevelMappingTable")]

public partial class ReportArchiveAndLevelMappingTable : INotifyPropertyChanging, INotifyPropertyChanged

{

private static PropertyChangingEventArgs emptyChangingEventArgs = new PropertyChangingEventArgs(String.Empty);

private int \_ReportArchiveAndLevelMappingTableId;

private bool \_Active;

private int \_FK\_ReportArchiveTableId;

private int \_FK\_FirstLevelSubmisionTableId;

private EntityRef<FirstLevelSubdivisionTable> \_FirstLevelSubdivisionTable;

private EntityRef<ReportArchiveTable> \_ReportArchiveTable;

#region Extensibility Method Definitions

partial void OnLoaded();

partial void OnValidate(System.Data.Linq.ChangeAction action);

partial void OnCreated();

partial void OnReportArchiveAndLevelMappingTableIdChanging(int value);

partial void OnReportArchiveAndLevelMappingTableIdChanged();

partial void OnActiveChanging(bool value);

partial void OnActiveChanged();

partial void OnFK\_ReportArchiveTableIdChanging(int value);

partial void OnFK\_ReportArchiveTableIdChanged();

partial void OnFK\_FirstLevelSubmisionTableIdChanging(int value);

partial void OnFK\_FirstLevelSubmisionTableIdChanged();

#endregion

public ReportArchiveAndLevelMappingTable()

{

this.\_FirstLevelSubdivisionTable = default(EntityRef<FirstLevelSubdivisionTable>);

this.\_ReportArchiveTable = default(EntityRef<ReportArchiveTable>);

OnCreated();

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_ReportArchiveAndLevelMappingTableId", AutoSync=AutoSync.OnInsert, DbType="Int NOT NULL IDENTITY", IsPrimaryKey=true, IsDbGenerated=true)]

public int ReportArchiveAndLevelMappingTableId

{

get

{

return this.\_ReportArchiveAndLevelMappingTableId;

}

set

{

if ((this.\_ReportArchiveAndLevelMappingTableId != value))

{

this.OnReportArchiveAndLevelMappingTableIdChanging(value);

this.SendPropertyChanging();

this.\_ReportArchiveAndLevelMappingTableId = value;

this.SendPropertyChanged("ReportArchiveAndLevelMappingTableId");

this.OnReportArchiveAndLevelMappingTableIdChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_Active", DbType="Bit NOT NULL")]

public bool Active

{

get

{

return this.\_Active;

}

set

{

if ((this.\_Active != value))

{

this.OnActiveChanging(value);

this.SendPropertyChanging();

this.\_Active = value;

this.SendPropertyChanged("Active");

this.OnActiveChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_FK\_ReportArchiveTableId", DbType="Int NOT NULL")]

public int FK\_ReportArchiveTableId

{

get

{

return this.\_FK\_ReportArchiveTableId;

}

set

{

if ((this.\_FK\_ReportArchiveTableId != value))

{

if (this.\_ReportArchiveTable.HasLoadedOrAssignedValue)

{

throw new System.Data.Linq.ForeignKeyReferenceAlreadyHasValueException();

}

this.OnFK\_ReportArchiveTableIdChanging(value);

this.SendPropertyChanging();

this.\_FK\_ReportArchiveTableId = value;

this.SendPropertyChanged("FK\_ReportArchiveTableId");

this.OnFK\_ReportArchiveTableIdChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_FK\_FirstLevelSubmisionTableId", DbType="Int NOT NULL")]

public int FK\_FirstLevelSubmisionTableId

{

get

{

return this.\_FK\_FirstLevelSubmisionTableId;

}

set

{

if ((this.\_FK\_FirstLevelSubmisionTableId != value))

{

if (this.\_FirstLevelSubdivisionTable.HasLoadedOrAssignedValue)

{

throw new System.Data.Linq.ForeignKeyReferenceAlreadyHasValueException();

}

this.OnFK\_FirstLevelSubmisionTableIdChanging(value);

this.SendPropertyChanging();

this.\_FK\_FirstLevelSubmisionTableId = value;

this.SendPropertyChanged("FK\_FirstLevelSubmisionTableId");

this.OnFK\_FirstLevelSubmisionTableIdChanged();

}

}

}

[global::System.Data.Linq.Mapping.AssociationAttribute(Name="FirstLevelSubdivisionTable\_ReportArchiveAndLevelMappingTable", Storage="\_FirstLevelSubdivisionTable", ThisKey="FK\_FirstLevelSubmisionTableId", OtherKey="FirstLevelSubdivisionTableID", IsForeignKey=true)]

public FirstLevelSubdivisionTable FirstLevelSubdivisionTable

{

get

{

return this.\_FirstLevelSubdivisionTable.Entity;

}

set

{

FirstLevelSubdivisionTable previousValue = this.\_FirstLevelSubdivisionTable.Entity;

if (((previousValue != value)

|| (this.\_FirstLevelSubdivisionTable.HasLoadedOrAssignedValue == false)))

{

this.SendPropertyChanging();

if ((previousValue != null))

{

this.\_FirstLevelSubdivisionTable.Entity = null;

previousValue.ReportArchiveAndLevelMappingTable.Remove(this);

}

this.\_FirstLevelSubdivisionTable.Entity = value;

if ((value != null))

{

value.ReportArchiveAndLevelMappingTable.Add(this);

this.\_FK\_FirstLevelSubmisionTableId = value.FirstLevelSubdivisionTableID;

}

else

{

this.\_FK\_FirstLevelSubmisionTableId = default(int);

}

this.SendPropertyChanged("FirstLevelSubdivisionTable");

}

}

}

[global::System.Data.Linq.Mapping.AssociationAttribute(Name="ReportArchiveTable\_ReportArchiveAndLevelMappingTable", Storage="\_ReportArchiveTable", ThisKey="FK\_ReportArchiveTableId", OtherKey="ReportArchiveTableID", IsForeignKey=true)]

public ReportArchiveTable ReportArchiveTable

{

get

{

return this.\_ReportArchiveTable.Entity;

}

set

{

ReportArchiveTable previousValue = this.\_ReportArchiveTable.Entity;

if (((previousValue != value)

|| (this.\_ReportArchiveTable.HasLoadedOrAssignedValue == false)))

{

this.SendPropertyChanging();

if ((previousValue != null))

{

this.\_ReportArchiveTable.Entity = null;

previousValue.ReportArchiveAndLevelMappingTable.Remove(this);

}

this.\_ReportArchiveTable.Entity = value;

if ((value != null))

{

value.ReportArchiveAndLevelMappingTable.Add(this);

this.\_FK\_ReportArchiveTableId = value.ReportArchiveTableID;

}

else

{

this.\_FK\_ReportArchiveTableId = default(int);

}

this.SendPropertyChanged("ReportArchiveTable");

}

}

}

public event PropertyChangingEventHandler PropertyChanging;

public event PropertyChangedEventHandler PropertyChanged;

protected virtual void SendPropertyChanging()

{

if ((this.PropertyChanging != null))

{

this.PropertyChanging(this, emptyChangingEventArgs);

}

}

protected virtual void SendPropertyChanged(String propertyName)

{

if ((this.PropertyChanged != null))

{

this.PropertyChanged(this, new PropertyChangedEventArgs(propertyName));

}

}

}

[global::System.Data.Linq.Mapping.TableAttribute(Name="dbo.ReportArchiveTable")]

public partial class ReportArchiveTable : INotifyPropertyChanging, INotifyPropertyChanged

{

private static PropertyChangingEventArgs emptyChangingEventArgs = new PropertyChangingEventArgs(String.Empty);

private int \_ReportArchiveTableID;

private bool \_Active;

private bool \_Calculeted;

private bool \_Sent;

private System.Nullable<System.DateTime> \_SentDateTime;

private bool \_RecipientConfirmed;

private System.Nullable<System.DateTime> \_StartDateTime;

private System.Nullable<System.DateTime> \_EndDateTime;

private System.Nullable<System.DateTime> \_DateToSend;

private string \_Name;

private System.Nullable<System.DateTime> \_RecivedDateTime;

private System.Nullable<System.DateTime> \_ConfirmEndDay;

private System.Nullable<int> \_DaysBeforeToCalcForRector;

private EntitySet<CollectedBasicParametersTable> \_CollectedBasicParametersTable;

private EntitySet<CollectedCalculatedParametrs> \_CollectedCalculatedParametrs;

private EntitySet<CollectedIndicatorsForR> \_CollectedIndicatorsForR;

private EntitySet<CollectedIndocators> \_CollectedIndocators;

private EntitySet<ConfirmationHistory> \_ConfirmationHistory;

private EntitySet<EmailSendHistory> \_EmailSendHistory;

private EntitySet<ReportArchiveAndBasicParametrsMappingTable> \_ReportArchiveAndBasicParametrsMappingTable;

private EntitySet<ReportArchiveAndCalculatedParametrsMappingTable> \_ReportArchiveAndCalculatedParametrsMappingTable;

private EntitySet<ReportArchiveAndIndicatorsMappingTable> \_ReportArchiveAndIndicatorsMappingTable;

private EntitySet<ReportArchiveAndLevelMappingTable> \_ReportArchiveAndLevelMappingTable;

#region Extensibility Method Definitions

partial void OnLoaded();

partial void OnValidate(System.Data.Linq.ChangeAction action);

partial void OnCreated();

partial void OnReportArchiveTableIDChanging(int value);

partial void OnReportArchiveTableIDChanged();

partial void OnActiveChanging(bool value);

partial void OnActiveChanged();

partial void OnCalculetedChanging(bool value);

partial void OnCalculetedChanged();

partial void OnSentChanging(bool value);

partial void OnSentChanged();

partial void OnSentDateTimeChanging(System.Nullable<System.DateTime> value);

partial void OnSentDateTimeChanged();

partial void OnRecipientConfirmedChanging(bool value);

partial void OnRecipientConfirmedChanged();

partial void OnStartDateTimeChanging(System.Nullable<System.DateTime> value);

partial void OnStartDateTimeChanged();

partial void OnEndDateTimeChanging(System.Nullable<System.DateTime> value);

partial void OnEndDateTimeChanged();

partial void OnDateToSendChanging(System.Nullable<System.DateTime> value);

partial void OnDateToSendChanged();

partial void OnNameChanging(string value);

partial void OnNameChanged();

partial void OnRecivedDateTimeChanging(System.Nullable<System.DateTime> value);

partial void OnRecivedDateTimeChanged();

partial void OnConfirmEndDayChanging(System.Nullable<System.DateTime> value);

partial void OnConfirmEndDayChanged();

partial void OnDaysBeforeToCalcForRectorChanging(System.Nullable<int> value);

partial void OnDaysBeforeToCalcForRectorChanged();

#endregion

public ReportArchiveTable()

{

this.\_CollectedBasicParametersTable = new EntitySet<CollectedBasicParametersTable>(new Action<CollectedBasicParametersTable>(this.attach\_CollectedBasicParametersTable), new Action<CollectedBasicParametersTable>(this.detach\_CollectedBasicParametersTable));

this.\_CollectedCalculatedParametrs = new EntitySet<CollectedCalculatedParametrs>(new Action<CollectedCalculatedParametrs>(this.attach\_CollectedCalculatedParametrs), new Action<CollectedCalculatedParametrs>(this.detach\_CollectedCalculatedParametrs));

this.\_CollectedIndicatorsForR = new EntitySet<CollectedIndicatorsForR>(new Action<CollectedIndicatorsForR>(this.attach\_CollectedIndicatorsForR), new Action<CollectedIndicatorsForR>(this.detach\_CollectedIndicatorsForR));

this.\_CollectedIndocators = new EntitySet<CollectedIndocators>(new Action<CollectedIndocators>(this.attach\_CollectedIndocators), new Action<CollectedIndocators>(this.detach\_CollectedIndocators));

this.\_ConfirmationHistory = new EntitySet<ConfirmationHistory>(new Action<ConfirmationHistory>(this.attach\_ConfirmationHistory), new Action<ConfirmationHistory>(this.detach\_ConfirmationHistory));

this.\_EmailSendHistory = new EntitySet<EmailSendHistory>(new Action<EmailSendHistory>(this.attach\_EmailSendHistory), new Action<EmailSendHistory>(this.detach\_EmailSendHistory));

this.\_ReportArchiveAndBasicParametrsMappingTable = new EntitySet<ReportArchiveAndBasicParametrsMappingTable>(new Action<ReportArchiveAndBasicParametrsMappingTable>(this.attach\_ReportArchiveAndBasicParametrsMappingTable), new Action<ReportArchiveAndBasicParametrsMappingTable>(this.detach\_ReportArchiveAndBasicParametrsMappingTable));

this.\_ReportArchiveAndCalculatedParametrsMappingTable = new EntitySet<ReportArchiveAndCalculatedParametrsMappingTable>(new Action<ReportArchiveAndCalculatedParametrsMappingTable>(this.attach\_ReportArchiveAndCalculatedParametrsMappingTable), new Action<ReportArchiveAndCalculatedParametrsMappingTable>(this.detach\_ReportArchiveAndCalculatedParametrsMappingTable));

this.\_ReportArchiveAndIndicatorsMappingTable = new EntitySet<ReportArchiveAndIndicatorsMappingTable>(new Action<ReportArchiveAndIndicatorsMappingTable>(this.attach\_ReportArchiveAndIndicatorsMappingTable), new Action<ReportArchiveAndIndicatorsMappingTable>(this.detach\_ReportArchiveAndIndicatorsMappingTable));

this.\_ReportArchiveAndLevelMappingTable = new EntitySet<ReportArchiveAndLevelMappingTable>(new Action<ReportArchiveAndLevelMappingTable>(this.attach\_ReportArchiveAndLevelMappingTable), new Action<ReportArchiveAndLevelMappingTable>(this.detach\_ReportArchiveAndLevelMappingTable));

OnCreated();

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_ReportArchiveTableID", AutoSync=AutoSync.OnInsert, DbType="Int NOT NULL IDENTITY", IsPrimaryKey=true, IsDbGenerated=true)]

public int ReportArchiveTableID

{

get

{

return this.\_ReportArchiveTableID;

}

set

{

if ((this.\_ReportArchiveTableID != value))

{

this.OnReportArchiveTableIDChanging(value);

this.SendPropertyChanging();

this.\_ReportArchiveTableID = value;

this.SendPropertyChanged("ReportArchiveTableID");

this.OnReportArchiveTableIDChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_Active", DbType="Bit NOT NULL")]

public bool Active

{

get

{

return this.\_Active;

}

set

{

if ((this.\_Active != value))

{

this.OnActiveChanging(value);

this.SendPropertyChanging();

this.\_Active = value;

this.SendPropertyChanged("Active");

this.OnActiveChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_Calculeted", DbType="Bit NOT NULL")]

public bool Calculeted

{

get

{

return this.\_Calculeted;

}

set

{

if ((this.\_Calculeted != value))

{

this.OnCalculetedChanging(value);

this.SendPropertyChanging();

this.\_Calculeted = value;

this.SendPropertyChanged("Calculeted");

this.OnCalculetedChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_Sent", DbType="Bit NOT NULL")]

public bool Sent

{

get

{

return this.\_Sent;

}

set

{

if ((this.\_Sent != value))

{

this.OnSentChanging(value);

this.SendPropertyChanging();

this.\_Sent = value;

this.SendPropertyChanged("Sent");

this.OnSentChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_SentDateTime", DbType="DateTime")]

public System.Nullable<System.DateTime> SentDateTime

{

get

{

return this.\_SentDateTime;

}

set

{

if ((this.\_SentDateTime != value))

{

this.OnSentDateTimeChanging(value);

this.SendPropertyChanging();

this.\_SentDateTime = value;

this.SendPropertyChanged("SentDateTime");

this.OnSentDateTimeChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_RecipientConfirmed", DbType="Bit NOT NULL")]

public bool RecipientConfirmed

{

get

{

return this.\_RecipientConfirmed;

}

set

{

if ((this.\_RecipientConfirmed != value))

{

this.OnRecipientConfirmedChanging(value);

this.SendPropertyChanging();

this.\_RecipientConfirmed = value;

this.SendPropertyChanged("RecipientConfirmed");

this.OnRecipientConfirmedChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_StartDateTime", DbType="DateTime")]

public System.Nullable<System.DateTime> StartDateTime

{

get

{

return this.\_StartDateTime;

}

set

{

if ((this.\_StartDateTime != value))

{

this.OnStartDateTimeChanging(value);

this.SendPropertyChanging();

this.\_StartDateTime = value;

this.SendPropertyChanged("StartDateTime");

this.OnStartDateTimeChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_EndDateTime", DbType="DateTime")]

public System.Nullable<System.DateTime> EndDateTime

{

get

{

return this.\_EndDateTime;

}

set

{

if ((this.\_EndDateTime != value))

{

this.OnEndDateTimeChanging(value);

this.SendPropertyChanging();

this.\_EndDateTime = value;

this.SendPropertyChanged("EndDateTime");

this.OnEndDateTimeChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_DateToSend", DbType="DateTime")]

public System.Nullable<System.DateTime> DateToSend

{

get

{

return this.\_DateToSend;

}

set

{

if ((this.\_DateToSend != value))

{

this.OnDateToSendChanging(value);

this.SendPropertyChanging();

this.\_DateToSend = value;

this.SendPropertyChanged("DateToSend");

this.OnDateToSendChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_Name", DbType="VarChar(500) NOT NULL", CanBeNull=false)]

public string Name

{

get

{

return this.\_Name;

}

set

{

if ((this.\_Name != value))

{

this.OnNameChanging(value);

this.SendPropertyChanging();

this.\_Name = value;

this.SendPropertyChanged("Name");

this.OnNameChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_RecivedDateTime", DbType="DateTime")]

public System.Nullable<System.DateTime> RecivedDateTime

{

get

{

return this.\_RecivedDateTime;

}

set

{

if ((this.\_RecivedDateTime != value))

{

this.OnRecivedDateTimeChanging(value);

this.SendPropertyChanging();

this.\_RecivedDateTime = value;

this.SendPropertyChanged("RecivedDateTime");

this.OnRecivedDateTimeChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_ConfirmEndDay", DbType="DateTime")]

public System.Nullable<System.DateTime> ConfirmEndDay

{

get

{

return this.\_ConfirmEndDay;

}

set

{

if ((this.\_ConfirmEndDay != value))

{

this.OnConfirmEndDayChanging(value);

this.SendPropertyChanging();

this.\_ConfirmEndDay = value;

this.SendPropertyChanged("ConfirmEndDay");

this.OnConfirmEndDayChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_DaysBeforeToCalcForRector", DbType="Int")]

public System.Nullable<int> DaysBeforeToCalcForRector

{

get

{

return this.\_DaysBeforeToCalcForRector;

}

set

{

if ((this.\_DaysBeforeToCalcForRector != value))

{

this.OnDaysBeforeToCalcForRectorChanging(value);

this.SendPropertyChanging();

this.\_DaysBeforeToCalcForRector = value;

this.SendPropertyChanged("DaysBeforeToCalcForRector");

this.OnDaysBeforeToCalcForRectorChanged();

}

}

}

[global::System.Data.Linq.Mapping.AssociationAttribute(Name="ReportArchiveTable\_CollectedBasicParametersTable", Storage="\_CollectedBasicParametersTable", ThisKey="ReportArchiveTableID", OtherKey="FK\_ReportArchiveTable")]

public EntitySet<CollectedBasicParametersTable> CollectedBasicParametersTable

{

get

{

return this.\_CollectedBasicParametersTable;

}

set

{

this.\_CollectedBasicParametersTable.Assign(value);

}

}

[global::System.Data.Linq.Mapping.AssociationAttribute(Name="ReportArchiveTable\_CollectedCalculatedParametrs", Storage="\_CollectedCalculatedParametrs", ThisKey="ReportArchiveTableID", OtherKey="FK\_ReportArchiveTable")]

public EntitySet<CollectedCalculatedParametrs> CollectedCalculatedParametrs

{

get

{

return this.\_CollectedCalculatedParametrs;

}

set

{

this.\_CollectedCalculatedParametrs.Assign(value);

}

}

[global::System.Data.Linq.Mapping.AssociationAttribute(Name="ReportArchiveTable\_CollectedIndicatorsForR", Storage="\_CollectedIndicatorsForR", ThisKey="ReportArchiveTableID", OtherKey="FK\_ReportArchiveTable")]

public EntitySet<CollectedIndicatorsForR> CollectedIndicatorsForR

{

get

{

return this.\_CollectedIndicatorsForR;

}

set

{

this.\_CollectedIndicatorsForR.Assign(value);

}

}

[global::System.Data.Linq.Mapping.AssociationAttribute(Name="ReportArchiveTable\_CollectedIndocators", Storage="\_CollectedIndocators", ThisKey="ReportArchiveTableID", OtherKey="FK\_ReportArchiveTable")]

public EntitySet<CollectedIndocators> CollectedIndocators

{

get

{

return this.\_CollectedIndocators;

}

set

{

this.\_CollectedIndocators.Assign(value);

}

}

[global::System.Data.Linq.Mapping.AssociationAttribute(Name="ReportArchiveTable\_ConfirmationHistory", Storage="\_ConfirmationHistory", ThisKey="ReportArchiveTableID", OtherKey="FK\_ReportTable")]

public EntitySet<ConfirmationHistory> ConfirmationHistory

{

get

{

return this.\_ConfirmationHistory;

}

set

{

this.\_ConfirmationHistory.Assign(value);

}

}

[global::System.Data.Linq.Mapping.AssociationAttribute(Name="ReportArchiveTable\_EmailSendHistory", Storage="\_EmailSendHistory", ThisKey="ReportArchiveTableID", OtherKey="FK\_ReportsArchiveTable")]

public EntitySet<EmailSendHistory> EmailSendHistory

{

get

{

return this.\_EmailSendHistory;

}

set

{

this.\_EmailSendHistory.Assign(value);

}

}

[global::System.Data.Linq.Mapping.AssociationAttribute(Name="ReportArchiveTable\_ReportArchiveAndBasicParametrsMappingTable", Storage="\_ReportArchiveAndBasicParametrsMappingTable", ThisKey="ReportArchiveTableID", OtherKey="FK\_ReportArchiveTable")]

public EntitySet<ReportArchiveAndBasicParametrsMappingTable> ReportArchiveAndBasicParametrsMappingTable

{

get

{

return this.\_ReportArchiveAndBasicParametrsMappingTable;

}

set

{

this.\_ReportArchiveAndBasicParametrsMappingTable.Assign(value);

}

}

[global::System.Data.Linq.Mapping.AssociationAttribute(Name="ReportArchiveTable\_ReportArchiveAndCalculatedParametrsMappingTable", Storage="\_ReportArchiveAndCalculatedParametrsMappingTable", ThisKey="ReportArchiveTableID", OtherKey="FK\_ReportArchiveTable")]

public EntitySet<ReportArchiveAndCalculatedParametrsMappingTable> ReportArchiveAndCalculatedParametrsMappingTable

{

get

{

return this.\_ReportArchiveAndCalculatedParametrsMappingTable;

}

set

{

this.\_ReportArchiveAndCalculatedParametrsMappingTable.Assign(value);

}

}

[global::System.Data.Linq.Mapping.AssociationAttribute(Name="ReportArchiveTable\_ReportArchiveAndIndicatorsMappingTable", Storage="\_ReportArchiveAndIndicatorsMappingTable", ThisKey="ReportArchiveTableID", OtherKey="FK\_ReportArchiveTable")]

public EntitySet<ReportArchiveAndIndicatorsMappingTable> ReportArchiveAndIndicatorsMappingTable

{

get

{

return this.\_ReportArchiveAndIndicatorsMappingTable;

}

set

{

this.\_ReportArchiveAndIndicatorsMappingTable.Assign(value);

}

}

[global::System.Data.Linq.Mapping.AssociationAttribute(Name="ReportArchiveTable\_ReportArchiveAndLevelMappingTable", Storage="\_ReportArchiveAndLevelMappingTable", ThisKey="ReportArchiveTableID", OtherKey="FK\_ReportArchiveTableId")]

public EntitySet<ReportArchiveAndLevelMappingTable> ReportArchiveAndLevelMappingTable

{

get

{

return this.\_ReportArchiveAndLevelMappingTable;

}

set

{

this.\_ReportArchiveAndLevelMappingTable.Assign(value);

}

}

public event PropertyChangingEventHandler PropertyChanging;

public event PropertyChangedEventHandler PropertyChanged;

protected virtual void SendPropertyChanging()

{

if ((this.PropertyChanging != null))

{

this.PropertyChanging(this, emptyChangingEventArgs);

}

}

protected virtual void SendPropertyChanged(String propertyName)

{

if ((this.PropertyChanged != null))

{

this.PropertyChanged(this, new PropertyChangedEventArgs(propertyName));

}

}

private void attach\_CollectedBasicParametersTable(CollectedBasicParametersTable entity)

{

this.SendPropertyChanging();

entity.ReportArchiveTable = this;

}

private void detach\_CollectedBasicParametersTable(CollectedBasicParametersTable entity)

{

this.SendPropertyChanging();

entity.ReportArchiveTable = null;

}

private void attach\_CollectedCalculatedParametrs(CollectedCalculatedParametrs entity)

{

this.SendPropertyChanging();

entity.ReportArchiveTable = this;

}

private void detach\_CollectedCalculatedParametrs(CollectedCalculatedParametrs entity)

{

this.SendPropertyChanging();

entity.ReportArchiveTable = null;

}

private void attach\_CollectedIndicatorsForR(CollectedIndicatorsForR entity)

{

this.SendPropertyChanging();

entity.ReportArchiveTable = this;

}

private void detach\_CollectedIndicatorsForR(CollectedIndicatorsForR entity)

{

this.SendPropertyChanging();

entity.ReportArchiveTable = null;

}

private void attach\_CollectedIndocators(CollectedIndocators entity)

{

this.SendPropertyChanging();

entity.ReportArchiveTable = this;

}

private void detach\_CollectedIndocators(CollectedIndocators entity)

{

this.SendPropertyChanging();

entity.ReportArchiveTable = null;

}

private void attach\_ConfirmationHistory(ConfirmationHistory entity)

{

this.SendPropertyChanging();

entity.ReportArchiveTable = this;

}

private void detach\_ConfirmationHistory(ConfirmationHistory entity)

{

this.SendPropertyChanging();

entity.ReportArchiveTable = null;

}

private void attach\_EmailSendHistory(EmailSendHistory entity)

{

this.SendPropertyChanging();

entity.ReportArchiveTable = this;

}

private void detach\_EmailSendHistory(EmailSendHistory entity)

{

this.SendPropertyChanging();

entity.ReportArchiveTable = null;

}

private void attach\_ReportArchiveAndBasicParametrsMappingTable(ReportArchiveAndBasicParametrsMappingTable entity)

{

this.SendPropertyChanging();

entity.ReportArchiveTable = this;

}

private void detach\_ReportArchiveAndBasicParametrsMappingTable(ReportArchiveAndBasicParametrsMappingTable entity)

{

this.SendPropertyChanging();

entity.ReportArchiveTable = null;

}

private void attach\_ReportArchiveAndCalculatedParametrsMappingTable(ReportArchiveAndCalculatedParametrsMappingTable entity)

{

this.SendPropertyChanging();

entity.ReportArchiveTable = this;

}

private void detach\_ReportArchiveAndCalculatedParametrsMappingTable(ReportArchiveAndCalculatedParametrsMappingTable entity)

{

this.SendPropertyChanging();

entity.ReportArchiveTable = null;

}

private void attach\_ReportArchiveAndIndicatorsMappingTable(ReportArchiveAndIndicatorsMappingTable entity)

{

this.SendPropertyChanging();

entity.ReportArchiveTable = this;

}

private void detach\_ReportArchiveAndIndicatorsMappingTable(ReportArchiveAndIndicatorsMappingTable entity)

{

this.SendPropertyChanging();

entity.ReportArchiveTable = null;

}

private void attach\_ReportArchiveAndLevelMappingTable(ReportArchiveAndLevelMappingTable entity)

{

this.SendPropertyChanging();

entity.ReportArchiveTable = this;

}

private void detach\_ReportArchiveAndLevelMappingTable(ReportArchiveAndLevelMappingTable entity)

{

this.SendPropertyChanging();

entity.ReportArchiveTable = null;

}

}

[global::System.Data.Linq.Mapping.TableAttribute(Name="dbo.RolesTable")]

public partial class RolesTable : INotifyPropertyChanging, INotifyPropertyChanged

{

private static PropertyChangingEventArgs emptyChangingEventArgs = new PropertyChangingEventArgs(String.Empty);

private int \_RolesTableID;

private bool \_Active;

private string \_RoleName;

private System.Nullable<bool> \_IsHead;

private EntitySet<BasicParametersAndRolesMappingTable> \_BasicParametersAndRolesMappingTable;

private EntitySet<CalculatedParametrsAndRolesMappingTable> \_CalculatedParametrsAndRolesMappingTable;

private EntitySet<IndicatorsAndRolesMappingTable> \_IndicatorsAndRolesMappingTable;

#region Extensibility Method Definitions

partial void OnLoaded();

partial void OnValidate(System.Data.Linq.ChangeAction action);

partial void OnCreated();

partial void OnRolesTableIDChanging(int value);

partial void OnRolesTableIDChanged();

partial void OnActiveChanging(bool value);

partial void OnActiveChanged();

partial void OnRoleNameChanging(string value);

partial void OnRoleNameChanged();

partial void OnIsHeadChanging(System.Nullable<bool> value);

partial void OnIsHeadChanged();

#endregion

public RolesTable()

{

this.\_BasicParametersAndRolesMappingTable = new EntitySet<BasicParametersAndRolesMappingTable>(new Action<BasicParametersAndRolesMappingTable>(this.attach\_BasicParametersAndRolesMappingTable), new Action<BasicParametersAndRolesMappingTable>(this.detach\_BasicParametersAndRolesMappingTable));

this.\_CalculatedParametrsAndRolesMappingTable = new EntitySet<CalculatedParametrsAndRolesMappingTable>(new Action<CalculatedParametrsAndRolesMappingTable>(this.attach\_CalculatedParametrsAndRolesMappingTable), new Action<CalculatedParametrsAndRolesMappingTable>(this.detach\_CalculatedParametrsAndRolesMappingTable));

this.\_IndicatorsAndRolesMappingTable = new EntitySet<IndicatorsAndRolesMappingTable>(new Action<IndicatorsAndRolesMappingTable>(this.attach\_IndicatorsAndRolesMappingTable), new Action<IndicatorsAndRolesMappingTable>(this.detach\_IndicatorsAndRolesMappingTable));

OnCreated();

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_RolesTableID", AutoSync=AutoSync.OnInsert, DbType="Int NOT NULL IDENTITY", IsPrimaryKey=true, IsDbGenerated=true)]

public int RolesTableID

{

get

{

return this.\_RolesTableID;

}

set

{

if ((this.\_RolesTableID != value))

{

this.OnRolesTableIDChanging(value);

this.SendPropertyChanging();

this.\_RolesTableID = value;

this.SendPropertyChanged("RolesTableID");

this.OnRolesTableIDChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_Active", DbType="Bit NOT NULL")]

public bool Active

{

get

{

return this.\_Active;

}

set

{

if ((this.\_Active != value))

{

this.OnActiveChanging(value);

this.SendPropertyChanging();

this.\_Active = value;

this.SendPropertyChanged("Active");

this.OnActiveChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_RoleName", DbType="VarChar(100) NOT NULL", CanBeNull=false)]

public string RoleName

{

get

{

return this.\_RoleName;

}

set

{

if ((this.\_RoleName != value))

{

this.OnRoleNameChanging(value);

this.SendPropertyChanging();

this.\_RoleName = value;

this.SendPropertyChanged("RoleName");

this.OnRoleNameChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_IsHead", DbType="Bit")]

public System.Nullable<bool> IsHead

{

get

{

return this.\_IsHead;

}

set

{

if ((this.\_IsHead != value))

{

this.OnIsHeadChanging(value);

this.SendPropertyChanging();

this.\_IsHead = value;

this.SendPropertyChanged("IsHead");

this.OnIsHeadChanged();

}

}

}

[global::System.Data.Linq.Mapping.AssociationAttribute(Name="RolesTable\_BasicParametersAndRolesMappingTable", Storage="\_BasicParametersAndRolesMappingTable", ThisKey="RolesTableID", OtherKey="FK\_RolesTable")]

public EntitySet<BasicParametersAndRolesMappingTable> BasicParametersAndRolesMappingTable

{

get

{

return this.\_BasicParametersAndRolesMappingTable;

}

set

{

this.\_BasicParametersAndRolesMappingTable.Assign(value);

}

}

[global::System.Data.Linq.Mapping.AssociationAttribute(Name="RolesTable\_CalculatedParametrsAndRolesMappingTable", Storage="\_CalculatedParametrsAndRolesMappingTable", ThisKey="RolesTableID", OtherKey="FK\_RolesTable")]

public EntitySet<CalculatedParametrsAndRolesMappingTable> CalculatedParametrsAndRolesMappingTable

{

get

{

return this.\_CalculatedParametrsAndRolesMappingTable;

}

set

{

this.\_CalculatedParametrsAndRolesMappingTable.Assign(value);

}

}

[global::System.Data.Linq.Mapping.AssociationAttribute(Name="RolesTable\_IndicatorsAndRolesMappingTable", Storage="\_IndicatorsAndRolesMappingTable", ThisKey="RolesTableID", OtherKey="FK\_RolesTable")]

public EntitySet<IndicatorsAndRolesMappingTable> IndicatorsAndRolesMappingTable

{

get

{

return this.\_IndicatorsAndRolesMappingTable;

}

set

{

this.\_IndicatorsAndRolesMappingTable.Assign(value);

}

}

public event PropertyChangingEventHandler PropertyChanging;

public event PropertyChangedEventHandler PropertyChanged;

protected virtual void SendPropertyChanging()

{

if ((this.PropertyChanging != null))

{

this.PropertyChanging(this, emptyChangingEventArgs);

}

}

protected virtual void SendPropertyChanged(String propertyName)

{

if ((this.PropertyChanged != null))

{

this.PropertyChanged(this, new PropertyChangedEventArgs(propertyName));

}

}

private void attach\_BasicParametersAndRolesMappingTable(BasicParametersAndRolesMappingTable entity)

{

this.SendPropertyChanging();

entity.RolesTable = this;

}

private void detach\_BasicParametersAndRolesMappingTable(BasicParametersAndRolesMappingTable entity)

{

this.SendPropertyChanging();

entity.RolesTable = null;

}

private void attach\_CalculatedParametrsAndRolesMappingTable(CalculatedParametrsAndRolesMappingTable entity)

{

this.SendPropertyChanging();

entity.RolesTable = this;

}

private void detach\_CalculatedParametrsAndRolesMappingTable(CalculatedParametrsAndRolesMappingTable entity)

{

this.SendPropertyChanging();

entity.RolesTable = null;

}

private void attach\_IndicatorsAndRolesMappingTable(IndicatorsAndRolesMappingTable entity)

{

this.SendPropertyChanging();

entity.RolesTable = this;

}

private void detach\_IndicatorsAndRolesMappingTable(IndicatorsAndRolesMappingTable entity)

{

this.SendPropertyChanging();

entity.RolesTable = null;

}

}

[global::System.Data.Linq.Mapping.TableAttribute(Name="dbo.SecondLevelSubdivisionTable")]

public partial class SecondLevelSubdivisionTable : INotifyPropertyChanging, INotifyPropertyChanged

{

private static PropertyChangingEventArgs emptyChangingEventArgs = new PropertyChangingEventArgs(String.Empty);

private int \_SecondLevelSubdivisionTableID;

private System.Nullable<bool> \_Active;

private string \_Name;

private int \_FK\_FirstLevelSubdivisionTable;

private EntitySet<CollectedBasicParametersTable> \_CollectedBasicParametersTable;

private EntitySet<CollectedIndicatorsForR> \_CollectedIndicatorsForR;

private EntitySet<ThirdLevelSubdivisionTable> \_ThirdLevelSubdivisionTable;

private EntitySet<UsersTable> \_UsersTable;

private EntityRef<FirstLevelSubdivisionTable> \_FirstLevelSubdivisionTable;

#region Extensibility Method Definitions

partial void OnLoaded();

partial void OnValidate(System.Data.Linq.ChangeAction action);

partial void OnCreated();

partial void OnSecondLevelSubdivisionTableIDChanging(int value);

partial void OnSecondLevelSubdivisionTableIDChanged();

partial void OnActiveChanging(System.Nullable<bool> value);

partial void OnActiveChanged();

partial void OnNameChanging(string value);

partial void OnNameChanged();

partial void OnFK\_FirstLevelSubdivisionTableChanging(int value);

partial void OnFK\_FirstLevelSubdivisionTableChanged();

#endregion

public SecondLevelSubdivisionTable()

{

this.\_CollectedBasicParametersTable = new EntitySet<CollectedBasicParametersTable>(new Action<CollectedBasicParametersTable>(this.attach\_CollectedBasicParametersTable), new Action<CollectedBasicParametersTable>(this.detach\_CollectedBasicParametersTable));

this.\_CollectedIndicatorsForR = new EntitySet<CollectedIndicatorsForR>(new Action<CollectedIndicatorsForR>(this.attach\_CollectedIndicatorsForR), new Action<CollectedIndicatorsForR>(this.detach\_CollectedIndicatorsForR));

this.\_ThirdLevelSubdivisionTable = new EntitySet<ThirdLevelSubdivisionTable>(new Action<ThirdLevelSubdivisionTable>(this.attach\_ThirdLevelSubdivisionTable), new Action<ThirdLevelSubdivisionTable>(this.detach\_ThirdLevelSubdivisionTable));

this.\_UsersTable = new EntitySet<UsersTable>(new Action<UsersTable>(this.attach\_UsersTable), new Action<UsersTable>(this.detach\_UsersTable));

this.\_FirstLevelSubdivisionTable = default(EntityRef<FirstLevelSubdivisionTable>);

OnCreated();

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_SecondLevelSubdivisionTableID", AutoSync=AutoSync.OnInsert, DbType="Int NOT NULL IDENTITY", IsPrimaryKey=true, IsDbGenerated=true)]

public int SecondLevelSubdivisionTableID

{

get

{

return this.\_SecondLevelSubdivisionTableID;

}

set

{

if ((this.\_SecondLevelSubdivisionTableID != value))

{

this.OnSecondLevelSubdivisionTableIDChanging(value);

this.SendPropertyChanging();

this.\_SecondLevelSubdivisionTableID = value;

this.SendPropertyChanged("SecondLevelSubdivisionTableID");

this.OnSecondLevelSubdivisionTableIDChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_Active", DbType="Bit")]

public System.Nullable<bool> Active

{

get

{

return this.\_Active;

}

set

{

if ((this.\_Active != value))

{

this.OnActiveChanging(value);

this.SendPropertyChanging();

this.\_Active = value;

this.SendPropertyChanged("Active");

this.OnActiveChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_Name", DbType="VarChar(500) NOT NULL", CanBeNull=false)]

public string Name

{

get

{

return this.\_Name;

}

set

{

if ((this.\_Name != value))

{

this.OnNameChanging(value);

this.SendPropertyChanging();

this.\_Name = value;

this.SendPropertyChanged("Name");

this.OnNameChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_FK\_FirstLevelSubdivisionTable", DbType="Int NOT NULL")]

public int FK\_FirstLevelSubdivisionTable

{

get

{

return this.\_FK\_FirstLevelSubdivisionTable;

}

set

{

if ((this.\_FK\_FirstLevelSubdivisionTable != value))

{

if (this.\_FirstLevelSubdivisionTable.HasLoadedOrAssignedValue)

{

throw new System.Data.Linq.ForeignKeyReferenceAlreadyHasValueException();

}

this.OnFK\_FirstLevelSubdivisionTableChanging(value);

this.SendPropertyChanging();

this.\_FK\_FirstLevelSubdivisionTable = value;

this.SendPropertyChanged("FK\_FirstLevelSubdivisionTable");

this.OnFK\_FirstLevelSubdivisionTableChanged();

}

}

}

[global::System.Data.Linq.Mapping.AssociationAttribute(Name="SecondLevelSubdivisionTable\_CollectedBasicParametersTable", Storage="\_CollectedBasicParametersTable", ThisKey="SecondLevelSubdivisionTableID", OtherKey="FK\_SecondLevelSubdivisionTable")]

public EntitySet<CollectedBasicParametersTable> CollectedBasicParametersTable

{

get

{

return this.\_CollectedBasicParametersTable;

}

set

{

this.\_CollectedBasicParametersTable.Assign(value);

}

}

[global::System.Data.Linq.Mapping.AssociationAttribute(Name="SecondLevelSubdivisionTable\_CollectedIndicatorsForR", Storage="\_CollectedIndicatorsForR", ThisKey="SecondLevelSubdivisionTableID", OtherKey="FK\_SecondLevelSubdivisionTable")]

public EntitySet<CollectedIndicatorsForR> CollectedIndicatorsForR

{

get

{

return this.\_CollectedIndicatorsForR;

}

set

{

this.\_CollectedIndicatorsForR.Assign(value);

}

}

[global::System.Data.Linq.Mapping.AssociationAttribute(Name="SecondLevelSubdivisionTable\_ThirdLevelSubdivisionTable", Storage="\_ThirdLevelSubdivisionTable", ThisKey="SecondLevelSubdivisionTableID", OtherKey="FK\_SecondLevelSubdivisionTable")]

public EntitySet<ThirdLevelSubdivisionTable> ThirdLevelSubdivisionTable

{

get

{

return this.\_ThirdLevelSubdivisionTable;

}

set

{

this.\_ThirdLevelSubdivisionTable.Assign(value);

}

}

[global::System.Data.Linq.Mapping.AssociationAttribute(Name="SecondLevelSubdivisionTable\_UsersTable", Storage="\_UsersTable", ThisKey="SecondLevelSubdivisionTableID", OtherKey="FK\_SecondLevelSubdivisionTable")]

public EntitySet<UsersTable> UsersTable

{

get

{

return this.\_UsersTable;

}

set

{

this.\_UsersTable.Assign(value);

}

}

[global::System.Data.Linq.Mapping.AssociationAttribute(Name="FirstLevelSubdivisionTable\_SecondLevelSubdivisionTable", Storage="\_FirstLevelSubdivisionTable", ThisKey="FK\_FirstLevelSubdivisionTable", OtherKey="FirstLevelSubdivisionTableID", IsForeignKey=true)]

public FirstLevelSubdivisionTable FirstLevelSubdivisionTable

{

get

{

return this.\_FirstLevelSubdivisionTable.Entity;

}

set

{

FirstLevelSubdivisionTable previousValue = this.\_FirstLevelSubdivisionTable.Entity;

if (((previousValue != value)

|| (this.\_FirstLevelSubdivisionTable.HasLoadedOrAssignedValue == false)))

{

this.SendPropertyChanging();

if ((previousValue != null))

{

this.\_FirstLevelSubdivisionTable.Entity = null;

previousValue.SecondLevelSubdivisionTable.Remove(this);

}

this.\_FirstLevelSubdivisionTable.Entity = value;

if ((value != null))

{

value.SecondLevelSubdivisionTable.Add(this);

this.\_FK\_FirstLevelSubdivisionTable = value.FirstLevelSubdivisionTableID;

}

else

{

this.\_FK\_FirstLevelSubdivisionTable = default(int);

}

this.SendPropertyChanged("FirstLevelSubdivisionTable");

}

}

}

public event PropertyChangingEventHandler PropertyChanging;

public event PropertyChangedEventHandler PropertyChanged;

protected virtual void SendPropertyChanging()

{

if ((this.PropertyChanging != null))

{

this.PropertyChanging(this, emptyChangingEventArgs);

}

}

protected virtual void SendPropertyChanged(String propertyName)

{

if ((this.PropertyChanged != null))

{

this.PropertyChanged(this, new PropertyChangedEventArgs(propertyName));

}

}

private void attach\_CollectedBasicParametersTable(CollectedBasicParametersTable entity)

{

this.SendPropertyChanging();

entity.SecondLevelSubdivisionTable = this;

}

private void detach\_CollectedBasicParametersTable(CollectedBasicParametersTable entity)

{

this.SendPropertyChanging();

entity.SecondLevelSubdivisionTable = null;

}

private void attach\_CollectedIndicatorsForR(CollectedIndicatorsForR entity)

{

this.SendPropertyChanging();

entity.SecondLevelSubdivisionTable = this;

}

private void detach\_CollectedIndicatorsForR(CollectedIndicatorsForR entity)

{

this.SendPropertyChanging();

entity.SecondLevelSubdivisionTable = null;

}

private void attach\_ThirdLevelSubdivisionTable(ThirdLevelSubdivisionTable entity)

{

this.SendPropertyChanging();

entity.SecondLevelSubdivisionTable = this;

}

private void detach\_ThirdLevelSubdivisionTable(ThirdLevelSubdivisionTable entity)

{

this.SendPropertyChanging();

entity.SecondLevelSubdivisionTable = null;

}

private void attach\_UsersTable(UsersTable entity)

{

this.SendPropertyChanging();

entity.SecondLevelSubdivisionTable = this;

}

private void detach\_UsersTable(UsersTable entity)

{

this.SendPropertyChanging();

entity.SecondLevelSubdivisionTable = null;

}

}

[global::System.Data.Linq.Mapping.TableAttribute(Name="dbo.SpecializationTable")]

public partial class SpecializationTable : INotifyPropertyChanging, INotifyPropertyChanged

{

private static PropertyChangingEventArgs emptyChangingEventArgs = new PropertyChangingEventArgs(String.Empty);

private int \_SpecializationTableID;

private bool \_Active;

private string \_Name;

private string \_SpecializationNumber;

private System.Nullable<int> \_FK\_FieldOfExpertise;

private EntitySet<FourthLevelSubdivisionTable> \_FourthLevelSubdivisionTable;

private EntitySet<EducationCostTable> \_EducationCostTable;

private EntityRef<FieldOfExpertise> \_FieldOfExpertise;

#region Extensibility Method Definitions

partial void OnLoaded();

partial void OnValidate(System.Data.Linq.ChangeAction action);

partial void OnCreated();

partial void OnSpecializationTableIDChanging(int value);

partial void OnSpecializationTableIDChanged();

partial void OnActiveChanging(bool value);

partial void OnActiveChanged();

partial void OnNameChanging(string value);

partial void OnNameChanged();

partial void OnSpecializationNumberChanging(string value);

partial void OnSpecializationNumberChanged();

partial void OnFK\_FieldOfExpertiseChanging(System.Nullable<int> value);

partial void OnFK\_FieldOfExpertiseChanged();

#endregion

public SpecializationTable()

{

this.\_FourthLevelSubdivisionTable = new EntitySet<FourthLevelSubdivisionTable>(new Action<FourthLevelSubdivisionTable>(this.attach\_FourthLevelSubdivisionTable), new Action<FourthLevelSubdivisionTable>(this.detach\_FourthLevelSubdivisionTable));

this.\_EducationCostTable = new EntitySet<EducationCostTable>(new Action<EducationCostTable>(this.attach\_EducationCostTable), new Action<EducationCostTable>(this.detach\_EducationCostTable));

this.\_FieldOfExpertise = default(EntityRef<FieldOfExpertise>);

OnCreated();

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_SpecializationTableID", AutoSync=AutoSync.OnInsert, DbType="Int NOT NULL IDENTITY", IsPrimaryKey=true, IsDbGenerated=true)]

public int SpecializationTableID

{

get

{

return this.\_SpecializationTableID;

}

set

{

if ((this.\_SpecializationTableID != value))

{

this.OnSpecializationTableIDChanging(value);

this.SendPropertyChanging();

this.\_SpecializationTableID = value;

this.SendPropertyChanged("SpecializationTableID");

this.OnSpecializationTableIDChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_Active", DbType="Bit NOT NULL")]

public bool Active

{

get

{

return this.\_Active;

}

set

{

if ((this.\_Active != value))

{

this.OnActiveChanging(value);

this.SendPropertyChanging();

this.\_Active = value;

this.SendPropertyChanged("Active");

this.OnActiveChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_Name", DbType="VarChar(1000) NOT NULL", CanBeNull=false)]

public string Name

{

get

{

return this.\_Name;

}

set

{

if ((this.\_Name != value))

{

this.OnNameChanging(value);

this.SendPropertyChanging();

this.\_Name = value;

this.SendPropertyChanged("Name");

this.OnNameChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_SpecializationNumber", DbType="VarChar(50)")]

public string SpecializationNumber

{

get

{

return this.\_SpecializationNumber;

}

set

{

if ((this.\_SpecializationNumber != value))

{

this.OnSpecializationNumberChanging(value);

this.SendPropertyChanging();

this.\_SpecializationNumber = value;

this.SendPropertyChanged("SpecializationNumber");

this.OnSpecializationNumberChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_FK\_FieldOfExpertise", DbType="Int")]

public System.Nullable<int> FK\_FieldOfExpertise

{

get

{

return this.\_FK\_FieldOfExpertise;

}

set

{

if ((this.\_FK\_FieldOfExpertise != value))

{

if (this.\_FieldOfExpertise.HasLoadedOrAssignedValue)

{

throw new System.Data.Linq.ForeignKeyReferenceAlreadyHasValueException();

}

this.OnFK\_FieldOfExpertiseChanging(value);

this.SendPropertyChanging();

this.\_FK\_FieldOfExpertise = value;

this.SendPropertyChanged("FK\_FieldOfExpertise");

this.OnFK\_FieldOfExpertiseChanged();

}

}

}

[global::System.Data.Linq.Mapping.AssociationAttribute(Name="SpecializationTable\_FourthLevelSubdivisionTable", Storage="\_FourthLevelSubdivisionTable", ThisKey="SpecializationTableID", OtherKey="FK\_Specialization")]

public EntitySet<FourthLevelSubdivisionTable> FourthLevelSubdivisionTable

{

get

{

return this.\_FourthLevelSubdivisionTable;

}

set

{

this.\_FourthLevelSubdivisionTable.Assign(value);

}

}

[global::System.Data.Linq.Mapping.AssociationAttribute(Name="SpecializationTable\_EducationCostTable", Storage="\_EducationCostTable", ThisKey="SpecializationTableID", OtherKey="FK\_Specialization")]

public EntitySet<EducationCostTable> EducationCostTable

{

get

{

return this.\_EducationCostTable;

}

set

{

this.\_EducationCostTable.Assign(value);

}

}

[global::System.Data.Linq.Mapping.AssociationAttribute(Name="FieldOfExpertise\_SpecializationTable", Storage="\_FieldOfExpertise", ThisKey="FK\_FieldOfExpertise", OtherKey="FieldOfExpertiseID", IsForeignKey=true)]

public FieldOfExpertise FieldOfExpertise

{

get

{

return this.\_FieldOfExpertise.Entity;

}

set

{

FieldOfExpertise previousValue = this.\_FieldOfExpertise.Entity;

if (((previousValue != value)

|| (this.\_FieldOfExpertise.HasLoadedOrAssignedValue == false)))

{

this.SendPropertyChanging();

if ((previousValue != null))

{

this.\_FieldOfExpertise.Entity = null;

previousValue.SpecializationTable.Remove(this);

}

this.\_FieldOfExpertise.Entity = value;

if ((value != null))

{

value.SpecializationTable.Add(this);

this.\_FK\_FieldOfExpertise = value.FieldOfExpertiseID;

}

else

{

this.\_FK\_FieldOfExpertise = default(Nullable<int>);

}

this.SendPropertyChanged("FieldOfExpertise");

}

}

}

public event PropertyChangingEventHandler PropertyChanging;

public event PropertyChangedEventHandler PropertyChanged;

protected virtual void SendPropertyChanging()

{

if ((this.PropertyChanging != null))

{

this.PropertyChanging(this, emptyChangingEventArgs);

}

}

protected virtual void SendPropertyChanged(String propertyName)

{

if ((this.PropertyChanged != null))

{

this.PropertyChanged(this, new PropertyChangedEventArgs(propertyName));

}

}

private void attach\_FourthLevelSubdivisionTable(FourthLevelSubdivisionTable entity)

{

this.SendPropertyChanging();

entity.SpecializationTable = this;

}

private void detach\_FourthLevelSubdivisionTable(FourthLevelSubdivisionTable entity)

{

this.SendPropertyChanging();

entity.SpecializationTable = null;

}

private void attach\_EducationCostTable(EducationCostTable entity)

{

this.SendPropertyChanging();

entity.SpecializationTable = this;

}

private void detach\_EducationCostTable(EducationCostTable entity)

{

this.SendPropertyChanging();

entity.SpecializationTable = null;

}

}

[global::System.Data.Linq.Mapping.TableAttribute(Name="dbo.ThirdLevelParametrs")]

public partial class ThirdLevelParametrs : INotifyPropertyChanging, INotifyPropertyChanged

{

private static PropertyChangingEventArgs emptyChangingEventArgs = new PropertyChangingEventArgs(String.Empty);

private int \_ThirdLevelParametrsID;

private bool \_Active;

private bool \_CanGraduate;

private System.Nullable<bool> \_IsBasic;

private EntityRef<ThirdLevelSubdivisionTable> \_ThirdLevelSubdivisionTable;

#region Extensibility Method Definitions

partial void OnLoaded();

partial void OnValidate(System.Data.Linq.ChangeAction action);

partial void OnCreated();

partial void OnThirdLevelParametrsIDChanging(int value);

partial void OnThirdLevelParametrsIDChanged();

partial void OnActiveChanging(bool value);

partial void OnActiveChanged();

partial void OnCanGraduateChanging(bool value);

partial void OnCanGraduateChanged();

partial void OnIsBasicChanging(System.Nullable<bool> value);

partial void OnIsBasicChanged();

#endregion

public ThirdLevelParametrs()

{

this.\_ThirdLevelSubdivisionTable = default(EntityRef<ThirdLevelSubdivisionTable>);

OnCreated();

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_ThirdLevelParametrsID", DbType="Int NOT NULL", IsPrimaryKey=true)]

public int ThirdLevelParametrsID

{

get

{

return this.\_ThirdLevelParametrsID;

}

set

{

if ((this.\_ThirdLevelParametrsID != value))

{

if (this.\_ThirdLevelSubdivisionTable.HasLoadedOrAssignedValue)

{

throw new System.Data.Linq.ForeignKeyReferenceAlreadyHasValueException();

}

this.OnThirdLevelParametrsIDChanging(value);

this.SendPropertyChanging();

this.\_ThirdLevelParametrsID = value;

this.SendPropertyChanged("ThirdLevelParametrsID");

this.OnThirdLevelParametrsIDChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_Active", DbType="Bit NOT NULL")]

public bool Active

{

get

{

return this.\_Active;

}

set

{

if ((this.\_Active != value))

{

this.OnActiveChanging(value);

this.SendPropertyChanging();

this.\_Active = value;

this.SendPropertyChanged("Active");

this.OnActiveChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_CanGraduate", DbType="Bit NOT NULL")]

public bool CanGraduate

{

get

{

return this.\_CanGraduate;

}

set

{

if ((this.\_CanGraduate != value))

{

this.OnCanGraduateChanging(value);

this.SendPropertyChanging();

this.\_CanGraduate = value;

this.SendPropertyChanged("CanGraduate");

this.OnCanGraduateChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_IsBasic", DbType="Bit")]

public System.Nullable<bool> IsBasic

{

get

{

return this.\_IsBasic;

}

set

{

if ((this.\_IsBasic != value))

{

this.OnIsBasicChanging(value);

this.SendPropertyChanging();

this.\_IsBasic = value;

this.SendPropertyChanged("IsBasic");

this.OnIsBasicChanged();

}

}

}

[global::System.Data.Linq.Mapping.AssociationAttribute(Name="ThirdLevelSubdivisionTable\_ThirdLevelParametrs", Storage="\_ThirdLevelSubdivisionTable", ThisKey="ThirdLevelParametrsID", OtherKey="ThirdLevelSubdivisionTableID", IsForeignKey=true)]

public ThirdLevelSubdivisionTable ThirdLevelSubdivisionTable

{

get

{

return this.\_ThirdLevelSubdivisionTable.Entity;

}

set

{

ThirdLevelSubdivisionTable previousValue = this.\_ThirdLevelSubdivisionTable.Entity;

if (((previousValue != value)

|| (this.\_ThirdLevelSubdivisionTable.HasLoadedOrAssignedValue == false)))

{

this.SendPropertyChanging();

if ((previousValue != null))

{

this.\_ThirdLevelSubdivisionTable.Entity = null;

previousValue.ThirdLevelParametrs = null;

}

this.\_ThirdLevelSubdivisionTable.Entity = value;

if ((value != null))

{

value.ThirdLevelParametrs = this;

this.\_ThirdLevelParametrsID = value.ThirdLevelSubdivisionTableID;

}

else

{

this.\_ThirdLevelParametrsID = default(int);

}

this.SendPropertyChanged("ThirdLevelSubdivisionTable");

}

}

}

public event PropertyChangingEventHandler PropertyChanging;

public event PropertyChangedEventHandler PropertyChanged;

protected virtual void SendPropertyChanging()

{

if ((this.PropertyChanging != null))

{

this.PropertyChanging(this, emptyChangingEventArgs);

}

}

protected virtual void SendPropertyChanged(String propertyName)

{

if ((this.PropertyChanged != null))

{

this.PropertyChanged(this, new PropertyChangedEventArgs(propertyName));

}

}

}

[global::System.Data.Linq.Mapping.TableAttribute(Name="dbo.ThirdLevelSubdivisionTable")]

public partial class ThirdLevelSubdivisionTable : INotifyPropertyChanging, INotifyPropertyChanged

{

private static PropertyChangingEventArgs emptyChangingEventArgs = new PropertyChangingEventArgs(String.Empty);

private int \_ThirdLevelSubdivisionTableID;

private bool \_Active;

private string \_Name;

private int \_FK\_SecondLevelSubdivisionTable;

private EntitySet<CollectedBasicParametersTable> \_CollectedBasicParametersTable;

private EntitySet<CollectedIndicatorsForR> \_CollectedIndicatorsForR;

private EntitySet<FourthLevelSubdivisionTable> \_FourthLevelSubdivisionTable;

private EntityRef<ThirdLevelParametrs> \_ThirdLevelParametrs;

private EntitySet<UsersTable> \_UsersTable;

private EntityRef<SecondLevelSubdivisionTable> \_SecondLevelSubdivisionTable;

#region Extensibility Method Definitions

partial void OnLoaded();

partial void OnValidate(System.Data.Linq.ChangeAction action);

partial void OnCreated();

partial void OnThirdLevelSubdivisionTableIDChanging(int value);

partial void OnThirdLevelSubdivisionTableIDChanged();

partial void OnActiveChanging(bool value);

partial void OnActiveChanged();

partial void OnNameChanging(string value);

partial void OnNameChanged();

partial void OnFK\_SecondLevelSubdivisionTableChanging(int value);

partial void OnFK\_SecondLevelSubdivisionTableChanged();

#endregion

public ThirdLevelSubdivisionTable()

{

this.\_CollectedBasicParametersTable = new EntitySet<CollectedBasicParametersTable>(new Action<CollectedBasicParametersTable>(this.attach\_CollectedBasicParametersTable), new Action<CollectedBasicParametersTable>(this.detach\_CollectedBasicParametersTable));

this.\_CollectedIndicatorsForR = new EntitySet<CollectedIndicatorsForR>(new Action<CollectedIndicatorsForR>(this.attach\_CollectedIndicatorsForR), new Action<CollectedIndicatorsForR>(this.detach\_CollectedIndicatorsForR));

this.\_FourthLevelSubdivisionTable = new EntitySet<FourthLevelSubdivisionTable>(new Action<FourthLevelSubdivisionTable>(this.attach\_FourthLevelSubdivisionTable), new Action<FourthLevelSubdivisionTable>(this.detach\_FourthLevelSubdivisionTable));

this.\_ThirdLevelParametrs = default(EntityRef<ThirdLevelParametrs>);

this.\_UsersTable = new EntitySet<UsersTable>(new Action<UsersTable>(this.attach\_UsersTable), new Action<UsersTable>(this.detach\_UsersTable));

this.\_SecondLevelSubdivisionTable = default(EntityRef<SecondLevelSubdivisionTable>);

OnCreated();

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_ThirdLevelSubdivisionTableID", AutoSync=AutoSync.OnInsert, DbType="Int NOT NULL IDENTITY", IsPrimaryKey=true, IsDbGenerated=true)]

public int ThirdLevelSubdivisionTableID

{

get

{

return this.\_ThirdLevelSubdivisionTableID;

}

set

{

if ((this.\_ThirdLevelSubdivisionTableID != value))

{

this.OnThirdLevelSubdivisionTableIDChanging(value);

this.SendPropertyChanging();

this.\_ThirdLevelSubdivisionTableID = value;

this.SendPropertyChanged("ThirdLevelSubdivisionTableID");

this.OnThirdLevelSubdivisionTableIDChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_Active", DbType="Bit NOT NULL")]

public bool Active

{

get

{

return this.\_Active;

}

set

{

if ((this.\_Active != value))

{

this.OnActiveChanging(value);

this.SendPropertyChanging();

this.\_Active = value;

this.SendPropertyChanged("Active");

this.OnActiveChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_Name", DbType="VarChar(500) NOT NULL", CanBeNull=false)]

public string Name

{

get

{

return this.\_Name;

}

set

{

if ((this.\_Name != value))

{

this.OnNameChanging(value);

this.SendPropertyChanging();

this.\_Name = value;

this.SendPropertyChanged("Name");

this.OnNameChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_FK\_SecondLevelSubdivisionTable", DbType="Int NOT NULL")]

public int FK\_SecondLevelSubdivisionTable

{

get

{

return this.\_FK\_SecondLevelSubdivisionTable;

}

set

{

if ((this.\_FK\_SecondLevelSubdivisionTable != value))

{

if (this.\_SecondLevelSubdivisionTable.HasLoadedOrAssignedValue)

{

throw new System.Data.Linq.ForeignKeyReferenceAlreadyHasValueException();

}

this.OnFK\_SecondLevelSubdivisionTableChanging(value);

this.SendPropertyChanging();

this.\_FK\_SecondLevelSubdivisionTable = value;

this.SendPropertyChanged("FK\_SecondLevelSubdivisionTable");

this.OnFK\_SecondLevelSubdivisionTableChanged();

}

}

}

[global::System.Data.Linq.Mapping.AssociationAttribute(Name="ThirdLevelSubdivisionTable\_CollectedBasicParametersTable", Storage="\_CollectedBasicParametersTable", ThisKey="ThirdLevelSubdivisionTableID", OtherKey="FK\_ThirdLevelSubdivisionTable")]

public EntitySet<CollectedBasicParametersTable> CollectedBasicParametersTable

{

get

{

return this.\_CollectedBasicParametersTable;

}

set

{

this.\_CollectedBasicParametersTable.Assign(value);

}

}

[global::System.Data.Linq.Mapping.AssociationAttribute(Name="ThirdLevelSubdivisionTable\_CollectedIndicatorsForR", Storage="\_CollectedIndicatorsForR", ThisKey="ThirdLevelSubdivisionTableID", OtherKey="FK\_ThirdLevelSubdivisionTable")]

public EntitySet<CollectedIndicatorsForR> CollectedIndicatorsForR

{

get

{

return this.\_CollectedIndicatorsForR;

}

set

{

this.\_CollectedIndicatorsForR.Assign(value);

}

}

[global::System.Data.Linq.Mapping.AssociationAttribute(Name="ThirdLevelSubdivisionTable\_FourthLevelSubdivisionTable", Storage="\_FourthLevelSubdivisionTable", ThisKey="ThirdLevelSubdivisionTableID", OtherKey="FK\_ThirdLevelSubdivisionTable")]

public EntitySet<FourthLevelSubdivisionTable> FourthLevelSubdivisionTable

{

get

{

return this.\_FourthLevelSubdivisionTable;

}

set

{

this.\_FourthLevelSubdivisionTable.Assign(value);

}

}

[global::System.Data.Linq.Mapping.AssociationAttribute(Name="ThirdLevelSubdivisionTable\_ThirdLevelParametrs", Storage="\_ThirdLevelParametrs", ThisKey="ThirdLevelSubdivisionTableID", OtherKey="ThirdLevelParametrsID", IsUnique=true, IsForeignKey=false)]

public ThirdLevelParametrs ThirdLevelParametrs

{

get

{

return this.\_ThirdLevelParametrs.Entity;

}

set

{

ThirdLevelParametrs previousValue = this.\_ThirdLevelParametrs.Entity;

if (((previousValue != value)

|| (this.\_ThirdLevelParametrs.HasLoadedOrAssignedValue == false)))

{

this.SendPropertyChanging();

if ((previousValue != null))

{

this.\_ThirdLevelParametrs.Entity = null;

previousValue.ThirdLevelSubdivisionTable = null;

}

this.\_ThirdLevelParametrs.Entity = value;

if ((value != null))

{

value.ThirdLevelSubdivisionTable = this;

}

this.SendPropertyChanged("ThirdLevelParametrs");

}

}

}

[global::System.Data.Linq.Mapping.AssociationAttribute(Name="ThirdLevelSubdivisionTable\_UsersTable", Storage="\_UsersTable", ThisKey="ThirdLevelSubdivisionTableID", OtherKey="FK\_ThirdLevelSubdivisionTable")]

public EntitySet<UsersTable> UsersTable

{

get

{

return this.\_UsersTable;

}

set

{

this.\_UsersTable.Assign(value);

}

}

[global::System.Data.Linq.Mapping.AssociationAttribute(Name="SecondLevelSubdivisionTable\_ThirdLevelSubdivisionTable", Storage="\_SecondLevelSubdivisionTable", ThisKey="FK\_SecondLevelSubdivisionTable", OtherKey="SecondLevelSubdivisionTableID", IsForeignKey=true)]

public SecondLevelSubdivisionTable SecondLevelSubdivisionTable

{

get

{

return this.\_SecondLevelSubdivisionTable.Entity;

}

set

{

SecondLevelSubdivisionTable previousValue = this.\_SecondLevelSubdivisionTable.Entity;

if (((previousValue != value)

|| (this.\_SecondLevelSubdivisionTable.HasLoadedOrAssignedValue == false)))

{

this.SendPropertyChanging();

if ((previousValue != null))

{

this.\_SecondLevelSubdivisionTable.Entity = null;

previousValue.ThirdLevelSubdivisionTable.Remove(this);

}

this.\_SecondLevelSubdivisionTable.Entity = value;

if ((value != null))

{

value.ThirdLevelSubdivisionTable.Add(this);

this.\_FK\_SecondLevelSubdivisionTable = value.SecondLevelSubdivisionTableID;

}

else

{

this.\_FK\_SecondLevelSubdivisionTable = default(int);

}

this.SendPropertyChanged("SecondLevelSubdivisionTable");

}

}

}

public event PropertyChangingEventHandler PropertyChanging;

public event PropertyChangedEventHandler PropertyChanged;

protected virtual void SendPropertyChanging()

{

if ((this.PropertyChanging != null))

{

this.PropertyChanging(this, emptyChangingEventArgs);

}

}

protected virtual void SendPropertyChanged(String propertyName)

{

if ((this.PropertyChanged != null))

{

this.PropertyChanged(this, new PropertyChangedEventArgs(propertyName));

}

}

private void attach\_CollectedBasicParametersTable(CollectedBasicParametersTable entity)

{

this.SendPropertyChanging();

entity.ThirdLevelSubdivisionTable = this;

}

private void detach\_CollectedBasicParametersTable(CollectedBasicParametersTable entity)

{

this.SendPropertyChanging();

entity.ThirdLevelSubdivisionTable = null;

}

private void attach\_CollectedIndicatorsForR(CollectedIndicatorsForR entity)

{

this.SendPropertyChanging();

entity.ThirdLevelSubdivisionTable = this;

}

private void detach\_CollectedIndicatorsForR(CollectedIndicatorsForR entity)

{

this.SendPropertyChanging();

entity.ThirdLevelSubdivisionTable = null;

}

private void attach\_FourthLevelSubdivisionTable(FourthLevelSubdivisionTable entity)

{

this.SendPropertyChanging();

entity.ThirdLevelSubdivisionTable = this;

}

private void detach\_FourthLevelSubdivisionTable(FourthLevelSubdivisionTable entity)

{

this.SendPropertyChanging();

entity.ThirdLevelSubdivisionTable = null;

}

private void attach\_UsersTable(UsersTable entity)

{

this.SendPropertyChanging();

entity.ThirdLevelSubdivisionTable = this;

}

private void detach\_UsersTable(UsersTable entity)

{

this.SendPropertyChanging();

entity.ThirdLevelSubdivisionTable = null;

}

}

[global::System.Data.Linq.Mapping.TableAttribute(Name="dbo.UsersTable")]

public partial class UsersTable : INotifyPropertyChanging, INotifyPropertyChanged

{

private static PropertyChangingEventArgs emptyChangingEventArgs = new PropertyChangingEventArgs(String.Empty);

private int \_UsersTableID;

private bool \_Active;

private string \_Login;

private string \_Password;

private string \_Email;

private System.Nullable<int> \_FK\_FirstLevelSubdivisionTable;

private System.Nullable<int> \_FK\_SecondLevelSubdivisionTable;

private System.Nullable<int> \_FK\_ThirdLevelSubdivisionTable;

private System.Nullable<int> \_FK\_FourthLevelSubdivisionTable;

private System.Nullable<int> \_FK\_FifthLevelSubdivisionTable;

private System.Nullable<int> \_AccessLevel;

private System.Nullable<int> \_FK\_ZeroLevelSubdivisionTable;

private string \_PassCode;

private System.Nullable<bool> \_Confirmed;

private string \_Position;

private EntitySet<BasicParametrsAndUsersMapping> \_BasicParametrsAndUsersMapping;

private EntitySet<CalculatedParametrsAndUsersMapping> \_CalculatedParametrsAndUsersMapping;

private EntitySet<CollectedBasicParametersTable> \_CollectedBasicParametersTable;

private EntitySet<CollectedCalculatedParametrs> \_CollectedCalculatedParametrs;

private EntitySet<CollectedIndocators> \_CollectedIndocators;

private EntitySet<ConfirmationHistory> \_ConfirmationHistory;

private EntitySet<IndicatorsAndUsersMapping> \_IndicatorsAndUsersMapping;

private EntitySet<MultiUser> \_MultiUser;

private EntitySet<MultiUser> \_MultiUser1;

private EntityRef<FifthLevelSubdivisionTable> \_FifthLevelSubdivisionTable;

private EntityRef<FirstLevelSubdivisionTable> \_FirstLevelSubdivisionTable;

private EntityRef<FourthLevelSubdivisionTable> \_FourthLevelSubdivisionTable;

private EntityRef<SecondLevelSubdivisionTable> \_SecondLevelSubdivisionTable;

private EntityRef<ThirdLevelSubdivisionTable> \_ThirdLevelSubdivisionTable;

private EntityRef<ZeroLevelSubdivisionTable> \_ZeroLevelSubdivisionTable;

#region Extensibility Method Definitions

partial void OnLoaded();

partial void OnValidate(System.Data.Linq.ChangeAction action);

partial void OnCreated();

partial void OnUsersTableIDChanging(int value);

partial void OnUsersTableIDChanged();

partial void OnActiveChanging(bool value);

partial void OnActiveChanged();

partial void OnLoginChanging(string value);

partial void OnLoginChanged();

partial void OnPasswordChanging(string value);

partial void OnPasswordChanged();

partial void OnEmailChanging(string value);

partial void OnEmailChanged();

partial void OnFK\_FirstLevelSubdivisionTableChanging(System.Nullable<int> value);

partial void OnFK\_FirstLevelSubdivisionTableChanged();

partial void OnFK\_SecondLevelSubdivisionTableChanging(System.Nullable<int> value);

partial void OnFK\_SecondLevelSubdivisionTableChanged();

partial void OnFK\_ThirdLevelSubdivisionTableChanging(System.Nullable<int> value);

partial void OnFK\_ThirdLevelSubdivisionTableChanged();

partial void OnFK\_FourthLevelSubdivisionTableChanging(System.Nullable<int> value);

partial void OnFK\_FourthLevelSubdivisionTableChanged();

partial void OnFK\_FifthLevelSubdivisionTableChanging(System.Nullable<int> value);

partial void OnFK\_FifthLevelSubdivisionTableChanged();

partial void OnAccessLevelChanging(System.Nullable<int> value);

partial void OnAccessLevelChanged();

partial void OnFK\_ZeroLevelSubdivisionTableChanging(System.Nullable<int> value);

partial void OnFK\_ZeroLevelSubdivisionTableChanged();

partial void OnPassCodeChanging(string value);

partial void OnPassCodeChanged();

partial void OnConfirmedChanging(System.Nullable<bool> value);

partial void OnConfirmedChanged();

partial void OnPositionChanging(string value);

partial void OnPositionChanged();

#endregion

public UsersTable()

{

this.\_BasicParametrsAndUsersMapping = new EntitySet<BasicParametrsAndUsersMapping>(new Action<BasicParametrsAndUsersMapping>(this.attach\_BasicParametrsAndUsersMapping), new Action<BasicParametrsAndUsersMapping>(this.detach\_BasicParametrsAndUsersMapping));

this.\_CalculatedParametrsAndUsersMapping = new EntitySet<CalculatedParametrsAndUsersMapping>(new Action<CalculatedParametrsAndUsersMapping>(this.attach\_CalculatedParametrsAndUsersMapping), new Action<CalculatedParametrsAndUsersMapping>(this.detach\_CalculatedParametrsAndUsersMapping));

this.\_CollectedBasicParametersTable = new EntitySet<CollectedBasicParametersTable>(new Action<CollectedBasicParametersTable>(this.attach\_CollectedBasicParametersTable), new Action<CollectedBasicParametersTable>(this.detach\_CollectedBasicParametersTable));

this.\_CollectedCalculatedParametrs = new EntitySet<CollectedCalculatedParametrs>(new Action<CollectedCalculatedParametrs>(this.attach\_CollectedCalculatedParametrs), new Action<CollectedCalculatedParametrs>(this.detach\_CollectedCalculatedParametrs));

this.\_CollectedIndocators = new EntitySet<CollectedIndocators>(new Action<CollectedIndocators>(this.attach\_CollectedIndocators), new Action<CollectedIndocators>(this.detach\_CollectedIndocators));

this.\_ConfirmationHistory = new EntitySet<ConfirmationHistory>(new Action<ConfirmationHistory>(this.attach\_ConfirmationHistory), new Action<ConfirmationHistory>(this.detach\_ConfirmationHistory));

this.\_IndicatorsAndUsersMapping = new EntitySet<IndicatorsAndUsersMapping>(new Action<IndicatorsAndUsersMapping>(this.attach\_IndicatorsAndUsersMapping), new Action<IndicatorsAndUsersMapping>(this.detach\_IndicatorsAndUsersMapping));

this.\_MultiUser = new EntitySet<MultiUser>(new Action<MultiUser>(this.attach\_MultiUser), new Action<MultiUser>(this.detach\_MultiUser));

this.\_MultiUser1 = new EntitySet<MultiUser>(new Action<MultiUser>(this.attach\_MultiUser1), new Action<MultiUser>(this.detach\_MultiUser1));

this.\_FifthLevelSubdivisionTable = default(EntityRef<FifthLevelSubdivisionTable>);

this.\_FirstLevelSubdivisionTable = default(EntityRef<FirstLevelSubdivisionTable>);

this.\_FourthLevelSubdivisionTable = default(EntityRef<FourthLevelSubdivisionTable>);

this.\_SecondLevelSubdivisionTable = default(EntityRef<SecondLevelSubdivisionTable>);

this.\_ThirdLevelSubdivisionTable = default(EntityRef<ThirdLevelSubdivisionTable>);

this.\_ZeroLevelSubdivisionTable = default(EntityRef<ZeroLevelSubdivisionTable>);

OnCreated();

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_UsersTableID", AutoSync=AutoSync.OnInsert, DbType="Int NOT NULL IDENTITY", IsPrimaryKey=true, IsDbGenerated=true)]

public int UsersTableID

{

get

{

return this.\_UsersTableID;

}

set

{

if ((this.\_UsersTableID != value))

{

this.OnUsersTableIDChanging(value);

this.SendPropertyChanging();

this.\_UsersTableID = value;

this.SendPropertyChanged("UsersTableID");

this.OnUsersTableIDChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_Active", DbType="Bit NOT NULL")]

public bool Active

{

get

{

return this.\_Active;

}

set

{

if ((this.\_Active != value))

{

this.OnActiveChanging(value);

this.SendPropertyChanging();

this.\_Active = value;

this.SendPropertyChanged("Active");

this.OnActiveChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_Login", DbType="VarChar(50) NOT NULL", CanBeNull=false)]

public string Login

{

get

{

return this.\_Login;

}

set

{

if ((this.\_Login != value))

{

this.OnLoginChanging(value);

this.SendPropertyChanging();

this.\_Login = value;

this.SendPropertyChanged("Login");

this.OnLoginChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_Password", DbType="VarChar(50) NOT NULL", CanBeNull=false)]

public string Password

{

get

{

return this.\_Password;

}

set

{

if ((this.\_Password != value))

{

this.OnPasswordChanging(value);

this.SendPropertyChanging();

this.\_Password = value;

this.SendPropertyChanged("Password");

this.OnPasswordChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_Email", DbType="VarChar(100) NOT NULL", CanBeNull=false)]

public string Email

{

get

{

return this.\_Email;

}

set

{

if ((this.\_Email != value))

{

this.OnEmailChanging(value);

this.SendPropertyChanging();

this.\_Email = value;

this.SendPropertyChanged("Email");

this.OnEmailChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_FK\_FirstLevelSubdivisionTable", DbType="Int")]

public System.Nullable<int> FK\_FirstLevelSubdivisionTable

{

get

{

return this.\_FK\_FirstLevelSubdivisionTable;

}

set

{

if ((this.\_FK\_FirstLevelSubdivisionTable != value))

{

if (this.\_FirstLevelSubdivisionTable.HasLoadedOrAssignedValue)

{

throw new System.Data.Linq.ForeignKeyReferenceAlreadyHasValueException();

}

this.OnFK\_FirstLevelSubdivisionTableChanging(value);

this.SendPropertyChanging();

this.\_FK\_FirstLevelSubdivisionTable = value;

this.SendPropertyChanged("FK\_FirstLevelSubdivisionTable");

this.OnFK\_FirstLevelSubdivisionTableChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_FK\_SecondLevelSubdivisionTable", DbType="Int")]

public System.Nullable<int> FK\_SecondLevelSubdivisionTable

{

get

{

return this.\_FK\_SecondLevelSubdivisionTable;

}

set

{

if ((this.\_FK\_SecondLevelSubdivisionTable != value))

{

if (this.\_SecondLevelSubdivisionTable.HasLoadedOrAssignedValue)

{

throw new System.Data.Linq.ForeignKeyReferenceAlreadyHasValueException();

}

this.OnFK\_SecondLevelSubdivisionTableChanging(value);

this.SendPropertyChanging();

this.\_FK\_SecondLevelSubdivisionTable = value;

this.SendPropertyChanged("FK\_SecondLevelSubdivisionTable");

this.OnFK\_SecondLevelSubdivisionTableChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_FK\_ThirdLevelSubdivisionTable", DbType="Int")]

public System.Nullable<int> FK\_ThirdLevelSubdivisionTable

{

get

{

return this.\_FK\_ThirdLevelSubdivisionTable;

}

set

{

if ((this.\_FK\_ThirdLevelSubdivisionTable != value))

{

if (this.\_ThirdLevelSubdivisionTable.HasLoadedOrAssignedValue)

{

throw new System.Data.Linq.ForeignKeyReferenceAlreadyHasValueException();

}

this.OnFK\_ThirdLevelSubdivisionTableChanging(value);

this.SendPropertyChanging();

this.\_FK\_ThirdLevelSubdivisionTable = value;

this.SendPropertyChanged("FK\_ThirdLevelSubdivisionTable");

this.OnFK\_ThirdLevelSubdivisionTableChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_FK\_FourthLevelSubdivisionTable", DbType="Int")]

public System.Nullable<int> FK\_FourthLevelSubdivisionTable

{

get

{

return this.\_FK\_FourthLevelSubdivisionTable;

}

set

{

if ((this.\_FK\_FourthLevelSubdivisionTable != value))

{

if (this.\_FourthLevelSubdivisionTable.HasLoadedOrAssignedValue)

{

throw new System.Data.Linq.ForeignKeyReferenceAlreadyHasValueException();

}

this.OnFK\_FourthLevelSubdivisionTableChanging(value);

this.SendPropertyChanging();

this.\_FK\_FourthLevelSubdivisionTable = value;

this.SendPropertyChanged("FK\_FourthLevelSubdivisionTable");

this.OnFK\_FourthLevelSubdivisionTableChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_FK\_FifthLevelSubdivisionTable", DbType="Int")]

public System.Nullable<int> FK\_FifthLevelSubdivisionTable

{

get

{

return this.\_FK\_FifthLevelSubdivisionTable;

}

set

{

if ((this.\_FK\_FifthLevelSubdivisionTable != value))

{

if (this.\_FifthLevelSubdivisionTable.HasLoadedOrAssignedValue)

{

throw new System.Data.Linq.ForeignKeyReferenceAlreadyHasValueException();

}

this.OnFK\_FifthLevelSubdivisionTableChanging(value);

this.SendPropertyChanging();

this.\_FK\_FifthLevelSubdivisionTable = value;

this.SendPropertyChanged("FK\_FifthLevelSubdivisionTable");

this.OnFK\_FifthLevelSubdivisionTableChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_AccessLevel", DbType="Int")]

public System.Nullable<int> AccessLevel

{

get

{

return this.\_AccessLevel;

}

set

{

if ((this.\_AccessLevel != value))

{

this.OnAccessLevelChanging(value);

this.SendPropertyChanging();

this.\_AccessLevel = value;

this.SendPropertyChanged("AccessLevel");

this.OnAccessLevelChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_FK\_ZeroLevelSubdivisionTable", DbType="Int")]

public System.Nullable<int> FK\_ZeroLevelSubdivisionTable

{

get

{

return this.\_FK\_ZeroLevelSubdivisionTable;

}

set

{

if ((this.\_FK\_ZeroLevelSubdivisionTable != value))

{

if (this.\_ZeroLevelSubdivisionTable.HasLoadedOrAssignedValue)

{

throw new System.Data.Linq.ForeignKeyReferenceAlreadyHasValueException();

}

this.OnFK\_ZeroLevelSubdivisionTableChanging(value);

this.SendPropertyChanging();

this.\_FK\_ZeroLevelSubdivisionTable = value;

this.SendPropertyChanged("FK\_ZeroLevelSubdivisionTable");

this.OnFK\_ZeroLevelSubdivisionTableChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_PassCode", DbType="VarChar(50)")]

public string PassCode

{

get

{

return this.\_PassCode;

}

set

{

if ((this.\_PassCode != value))

{

this.OnPassCodeChanging(value);

this.SendPropertyChanging();

this.\_PassCode = value;

this.SendPropertyChanged("PassCode");

this.OnPassCodeChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_Confirmed", DbType="Bit")]

public System.Nullable<bool> Confirmed

{

get

{

return this.\_Confirmed;

}

set

{

if ((this.\_Confirmed != value))

{

this.OnConfirmedChanging(value);

this.SendPropertyChanging();

this.\_Confirmed = value;

this.SendPropertyChanged("Confirmed");

this.OnConfirmedChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_Position", DbType="VarChar(500)")]

public string Position

{

get

{

return this.\_Position;

}

set

{

if ((this.\_Position != value))

{

this.OnPositionChanging(value);

this.SendPropertyChanging();

this.\_Position = value;

this.SendPropertyChanged("Position");

this.OnPositionChanged();

}

}

}

[global::System.Data.Linq.Mapping.AssociationAttribute(Name="UsersTable\_BasicParametrsAndUsersMapping", Storage="\_BasicParametrsAndUsersMapping", ThisKey="UsersTableID", OtherKey="FK\_UsersTable")]

public EntitySet<BasicParametrsAndUsersMapping> BasicParametrsAndUsersMapping

{

get

{

return this.\_BasicParametrsAndUsersMapping;

}

set

{

this.\_BasicParametrsAndUsersMapping.Assign(value);

}

}

[global::System.Data.Linq.Mapping.AssociationAttribute(Name="UsersTable\_CalculatedParametrsAndUsersMapping", Storage="\_CalculatedParametrsAndUsersMapping", ThisKey="UsersTableID", OtherKey="FK\_UsersTable")]

public EntitySet<CalculatedParametrsAndUsersMapping> CalculatedParametrsAndUsersMapping

{

get

{

return this.\_CalculatedParametrsAndUsersMapping;

}

set

{

this.\_CalculatedParametrsAndUsersMapping.Assign(value);

}

}

[global::System.Data.Linq.Mapping.AssociationAttribute(Name="UsersTable\_CollectedBasicParametersTable", Storage="\_CollectedBasicParametersTable", ThisKey="UsersTableID", OtherKey="FK\_UsersTable")]

public EntitySet<CollectedBasicParametersTable> CollectedBasicParametersTable

{

get

{

return this.\_CollectedBasicParametersTable;

}

set

{

this.\_CollectedBasicParametersTable.Assign(value);

}

}

[global::System.Data.Linq.Mapping.AssociationAttribute(Name="UsersTable\_CollectedCalculatedParametrs", Storage="\_CollectedCalculatedParametrs", ThisKey="UsersTableID", OtherKey="FK\_UsersTable")]

public EntitySet<CollectedCalculatedParametrs> CollectedCalculatedParametrs

{

get

{

return this.\_CollectedCalculatedParametrs;

}

set

{

this.\_CollectedCalculatedParametrs.Assign(value);

}

}

[global::System.Data.Linq.Mapping.AssociationAttribute(Name="UsersTable\_CollectedIndocators", Storage="\_CollectedIndocators", ThisKey="UsersTableID", OtherKey="FK\_UsersTable")]

public EntitySet<CollectedIndocators> CollectedIndocators

{

get

{

return this.\_CollectedIndocators;

}

set

{

this.\_CollectedIndocators.Assign(value);

}

}

[global::System.Data.Linq.Mapping.AssociationAttribute(Name="UsersTable\_ConfirmationHistory", Storage="\_ConfirmationHistory", ThisKey="UsersTableID", OtherKey="FK\_UsersTable")]

public EntitySet<ConfirmationHistory> ConfirmationHistory

{

get

{

return this.\_ConfirmationHistory;

}

set

{

this.\_ConfirmationHistory.Assign(value);

}

}

[global::System.Data.Linq.Mapping.AssociationAttribute(Name="UsersTable\_IndicatorsAndUsersMapping", Storage="\_IndicatorsAndUsersMapping", ThisKey="UsersTableID", OtherKey="FK\_UsresTable")]

public EntitySet<IndicatorsAndUsersMapping> IndicatorsAndUsersMapping

{

get

{

return this.\_IndicatorsAndUsersMapping;

}

set

{

this.\_IndicatorsAndUsersMapping.Assign(value);

}

}

[global::System.Data.Linq.Mapping.AssociationAttribute(Name="UsersTable\_MultiUser", Storage="\_MultiUser", ThisKey="UsersTableID", OtherKey="FK\_UserCanAccess")]

public EntitySet<MultiUser> MultiUser

{

get

{

return this.\_MultiUser;

}

set

{

this.\_MultiUser.Assign(value);

}

}

[global::System.Data.Linq.Mapping.AssociationAttribute(Name="UsersTable\_MultiUser1", Storage="\_MultiUser1", ThisKey="UsersTableID", OtherKey="FK\_UserToAccess")]

public EntitySet<MultiUser> MultiUser1

{

get

{

return this.\_MultiUser1;

}

set

{

this.\_MultiUser1.Assign(value);

}

}

[global::System.Data.Linq.Mapping.AssociationAttribute(Name="FifthLevelSubdivisionTable\_UsersTable", Storage="\_FifthLevelSubdivisionTable", ThisKey="FK\_FifthLevelSubdivisionTable", OtherKey="FifthLevelSubdivisionTableID", IsForeignKey=true)]

public FifthLevelSubdivisionTable FifthLevelSubdivisionTable

{

get

{

return this.\_FifthLevelSubdivisionTable.Entity;

}

set

{

FifthLevelSubdivisionTable previousValue = this.\_FifthLevelSubdivisionTable.Entity;

if (((previousValue != value)

|| (this.\_FifthLevelSubdivisionTable.HasLoadedOrAssignedValue == false)))

{

this.SendPropertyChanging();

if ((previousValue != null))

{

this.\_FifthLevelSubdivisionTable.Entity = null;

previousValue.UsersTable.Remove(this);

}

this.\_FifthLevelSubdivisionTable.Entity = value;

if ((value != null))

{

value.UsersTable.Add(this);

this.\_FK\_FifthLevelSubdivisionTable = value.FifthLevelSubdivisionTableID;

}

else

{

this.\_FK\_FifthLevelSubdivisionTable = default(Nullable<int>);

}

this.SendPropertyChanged("FifthLevelSubdivisionTable");

}

}

}

[global::System.Data.Linq.Mapping.AssociationAttribute(Name="FirstLevelSubdivisionTable\_UsersTable", Storage="\_FirstLevelSubdivisionTable", ThisKey="FK\_FirstLevelSubdivisionTable", OtherKey="FirstLevelSubdivisionTableID", IsForeignKey=true)]

public FirstLevelSubdivisionTable FirstLevelSubdivisionTable

{

get

{

return this.\_FirstLevelSubdivisionTable.Entity;

}

set

{

FirstLevelSubdivisionTable previousValue = this.\_FirstLevelSubdivisionTable.Entity;

if (((previousValue != value)

|| (this.\_FirstLevelSubdivisionTable.HasLoadedOrAssignedValue == false)))

{

this.SendPropertyChanging();

if ((previousValue != null))

{

this.\_FirstLevelSubdivisionTable.Entity = null;

previousValue.UsersTable.Remove(this);

}

this.\_FirstLevelSubdivisionTable.Entity = value;

if ((value != null))

{

value.UsersTable.Add(this);

this.\_FK\_FirstLevelSubdivisionTable = value.FirstLevelSubdivisionTableID;

}

else

{

this.\_FK\_FirstLevelSubdivisionTable = default(Nullable<int>);

}

this.SendPropertyChanged("FirstLevelSubdivisionTable");

}

}

}

[global::System.Data.Linq.Mapping.AssociationAttribute(Name="FourthLevelSubdivisionTable\_UsersTable", Storage="\_FourthLevelSubdivisionTable", ThisKey="FK\_FourthLevelSubdivisionTable", OtherKey="FourthLevelSubdivisionTableID", IsForeignKey=true)]

public FourthLevelSubdivisionTable FourthLevelSubdivisionTable

{

get

{

return this.\_FourthLevelSubdivisionTable.Entity;

}

set

{

FourthLevelSubdivisionTable previousValue = this.\_FourthLevelSubdivisionTable.Entity;

if (((previousValue != value)

|| (this.\_FourthLevelSubdivisionTable.HasLoadedOrAssignedValue == false)))

{

this.SendPropertyChanging();

if ((previousValue != null))

{

this.\_FourthLevelSubdivisionTable.Entity = null;

previousValue.UsersTable.Remove(this);

}

this.\_FourthLevelSubdivisionTable.Entity = value;

if ((value != null))

{

value.UsersTable.Add(this);

this.\_FK\_FourthLevelSubdivisionTable = value.FourthLevelSubdivisionTableID;

}

else

{

this.\_FK\_FourthLevelSubdivisionTable = default(Nullable<int>);

}

this.SendPropertyChanged("FourthLevelSubdivisionTable");

}

}

}

[global::System.Data.Linq.Mapping.AssociationAttribute(Name="SecondLevelSubdivisionTable\_UsersTable", Storage="\_SecondLevelSubdivisionTable", ThisKey="FK\_SecondLevelSubdivisionTable", OtherKey="SecondLevelSubdivisionTableID", IsForeignKey=true)]

public SecondLevelSubdivisionTable SecondLevelSubdivisionTable

{

get

{

return this.\_SecondLevelSubdivisionTable.Entity;

}

set

{

SecondLevelSubdivisionTable previousValue = this.\_SecondLevelSubdivisionTable.Entity;

if (((previousValue != value)

|| (this.\_SecondLevelSubdivisionTable.HasLoadedOrAssignedValue == false)))

{

this.SendPropertyChanging();

if ((previousValue != null))

{

this.\_SecondLevelSubdivisionTable.Entity = null;

previousValue.UsersTable.Remove(this);

}

this.\_SecondLevelSubdivisionTable.Entity = value;

if ((value != null))

{

value.UsersTable.Add(this);

this.\_FK\_SecondLevelSubdivisionTable = value.SecondLevelSubdivisionTableID;

}

else

{

this.\_FK\_SecondLevelSubdivisionTable = default(Nullable<int>);

}

this.SendPropertyChanged("SecondLevelSubdivisionTable");

}

}

}

[global::System.Data.Linq.Mapping.AssociationAttribute(Name="ThirdLevelSubdivisionTable\_UsersTable", Storage="\_ThirdLevelSubdivisionTable", ThisKey="FK\_ThirdLevelSubdivisionTable", OtherKey="ThirdLevelSubdivisionTableID", IsForeignKey=true)]

public ThirdLevelSubdivisionTable ThirdLevelSubdivisionTable

{

get

{

return this.\_ThirdLevelSubdivisionTable.Entity;

}

set

{

ThirdLevelSubdivisionTable previousValue = this.\_ThirdLevelSubdivisionTable.Entity;

if (((previousValue != value)

|| (this.\_ThirdLevelSubdivisionTable.HasLoadedOrAssignedValue == false)))

{

this.SendPropertyChanging();

if ((previousValue != null))

{

this.\_ThirdLevelSubdivisionTable.Entity = null;

previousValue.UsersTable.Remove(this);

}

this.\_ThirdLevelSubdivisionTable.Entity = value;

if ((value != null))

{

value.UsersTable.Add(this);

this.\_FK\_ThirdLevelSubdivisionTable = value.ThirdLevelSubdivisionTableID;

}

else

{

this.\_FK\_ThirdLevelSubdivisionTable = default(Nullable<int>);

}

this.SendPropertyChanged("ThirdLevelSubdivisionTable");

}

}

}

[global::System.Data.Linq.Mapping.AssociationAttribute(Name="ZeroLevelSubdivisionTable\_UsersTable", Storage="\_ZeroLevelSubdivisionTable", ThisKey="FK\_ZeroLevelSubdivisionTable", OtherKey="ZeroLevelSubdivisionTableID", IsForeignKey=true)]

public ZeroLevelSubdivisionTable ZeroLevelSubdivisionTable

{

get

{

return this.\_ZeroLevelSubdivisionTable.Entity;

}

set

{

ZeroLevelSubdivisionTable previousValue = this.\_ZeroLevelSubdivisionTable.Entity;

if (((previousValue != value)

|| (this.\_ZeroLevelSubdivisionTable.HasLoadedOrAssignedValue == false)))

{

this.SendPropertyChanging();

if ((previousValue != null))

{

this.\_ZeroLevelSubdivisionTable.Entity = null;

previousValue.UsersTable.Remove(this);

}

this.\_ZeroLevelSubdivisionTable.Entity = value;

if ((value != null))

{

value.UsersTable.Add(this);

this.\_FK\_ZeroLevelSubdivisionTable = value.ZeroLevelSubdivisionTableID;

}

else

{

this.\_FK\_ZeroLevelSubdivisionTable = default(Nullable<int>);

}

this.SendPropertyChanged("ZeroLevelSubdivisionTable");

}

}

}

public event PropertyChangingEventHandler PropertyChanging;

public event PropertyChangedEventHandler PropertyChanged;

protected virtual void SendPropertyChanging()

{

if ((this.PropertyChanging != null))

{

this.PropertyChanging(this, emptyChangingEventArgs);

}

}

protected virtual void SendPropertyChanged(String propertyName)

{

if ((this.PropertyChanged != null))

{

this.PropertyChanged(this, new PropertyChangedEventArgs(propertyName));

}

}

private void attach\_BasicParametrsAndUsersMapping(BasicParametrsAndUsersMapping entity)

{

this.SendPropertyChanging();

entity.UsersTable = this;

}

private void detach\_BasicParametrsAndUsersMapping(BasicParametrsAndUsersMapping entity)

{

this.SendPropertyChanging();

entity.UsersTable = null;

}

private void attach\_CalculatedParametrsAndUsersMapping(CalculatedParametrsAndUsersMapping entity)

{

this.SendPropertyChanging();

entity.UsersTable = this;

}

private void detach\_CalculatedParametrsAndUsersMapping(CalculatedParametrsAndUsersMapping entity)

{

this.SendPropertyChanging();

entity.UsersTable = null;

}

private void attach\_CollectedBasicParametersTable(CollectedBasicParametersTable entity)

{

this.SendPropertyChanging();

entity.UsersTable = this;

}

private void detach\_CollectedBasicParametersTable(CollectedBasicParametersTable entity)

{

this.SendPropertyChanging();

entity.UsersTable = null;

}

private void attach\_CollectedCalculatedParametrs(CollectedCalculatedParametrs entity)

{

this.SendPropertyChanging();

entity.UsersTable = this;

}

private void detach\_CollectedCalculatedParametrs(CollectedCalculatedParametrs entity)

{

this.SendPropertyChanging();

entity.UsersTable = null;

}

private void attach\_CollectedIndocators(CollectedIndocators entity)

{

this.SendPropertyChanging();

entity.UsersTable = this;

}

private void detach\_CollectedIndocators(CollectedIndocators entity)

{

this.SendPropertyChanging();

entity.UsersTable = null;

}

private void attach\_ConfirmationHistory(ConfirmationHistory entity)

{

this.SendPropertyChanging();

entity.UsersTable = this;

}

private void detach\_ConfirmationHistory(ConfirmationHistory entity)

{

this.SendPropertyChanging();

entity.UsersTable = null;

}

private void attach\_IndicatorsAndUsersMapping(IndicatorsAndUsersMapping entity)

{

this.SendPropertyChanging();

entity.UsersTable = this;

}

private void detach\_IndicatorsAndUsersMapping(IndicatorsAndUsersMapping entity)

{

this.SendPropertyChanging();

entity.UsersTable = null;

}

private void attach\_MultiUser(MultiUser entity)

{

this.SendPropertyChanging();

entity.UsersTable = this;

}

private void detach\_MultiUser(MultiUser entity)

{

this.SendPropertyChanging();

entity.UsersTable = null;

}

private void attach\_MultiUser1(MultiUser entity)

{

this.SendPropertyChanging();

entity.UsersTable1 = this;

}

private void detach\_MultiUser1(MultiUser entity)

{

this.SendPropertyChanging();

entity.UsersTable1 = null;

}

}

[global::System.Data.Linq.Mapping.TableAttribute(Name="dbo.EducationCostTable")]

public partial class EducationCostTable : INotifyPropertyChanging, INotifyPropertyChanged

{

private static PropertyChangingEventArgs emptyChangingEventArgs = new PropertyChangingEventArgs(String.Empty);

private int \_EducationCostTableID;

private System.Nullable<bool> \_Active;

private System.Nullable<System.DateTime> \_Date;

private System.Nullable<int> \_FK\_Specialization;

private System.Nullable<double> \_CostOfCommercOch;

private System.Nullable<double> \_CostOfCommercOchIn;

private System.Nullable<double> \_CostOfCommercZaoch;

private System.Nullable<double> \_CostOfCommercEvening;

private EntityRef<SpecializationTable> \_SpecializationTable;

#region Extensibility Method Definitions

partial void OnLoaded();

partial void OnValidate(System.Data.Linq.ChangeAction action);

partial void OnCreated();

partial void OnEducationCostTableIDChanging(int value);

partial void OnEducationCostTableIDChanged();

partial void OnActiveChanging(System.Nullable<bool> value);

partial void OnActiveChanged();

partial void OnDateChanging(System.Nullable<System.DateTime> value);

partial void OnDateChanged();

partial void OnFK\_SpecializationChanging(System.Nullable<int> value);

partial void OnFK\_SpecializationChanged();

partial void OnCostOfCommercOchChanging(System.Nullable<double> value);

partial void OnCostOfCommercOchChanged();

partial void OnCostOfCommercOchInChanging(System.Nullable<double> value);

partial void OnCostOfCommercOchInChanged();

partial void OnCostOfCommercZaochChanging(System.Nullable<double> value);

partial void OnCostOfCommercZaochChanged();

partial void OnCostOfCommercEveningChanging(System.Nullable<double> value);

partial void OnCostOfCommercEveningChanged();

#endregion

public EducationCostTable()

{

this.\_SpecializationTable = default(EntityRef<SpecializationTable>);

OnCreated();

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_EducationCostTableID", AutoSync=AutoSync.OnInsert, DbType="Int NOT NULL IDENTITY", IsPrimaryKey=true, IsDbGenerated=true)]

public int EducationCostTableID

{

get

{

return this.\_EducationCostTableID;

}

set

{

if ((this.\_EducationCostTableID != value))

{

this.OnEducationCostTableIDChanging(value);

this.SendPropertyChanging();

this.\_EducationCostTableID = value;

this.SendPropertyChanged("EducationCostTableID");

this.OnEducationCostTableIDChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_Active", DbType="Bit")]

public System.Nullable<bool> Active

{

get

{

return this.\_Active;

}

set

{

if ((this.\_Active != value))

{

this.OnActiveChanging(value);

this.SendPropertyChanging();

this.\_Active = value;

this.SendPropertyChanged("Active");

this.OnActiveChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_Date", DbType="DateTime")]

public System.Nullable<System.DateTime> Date

{

get

{

return this.\_Date;

}

set

{

if ((this.\_Date != value))

{

this.OnDateChanging(value);

this.SendPropertyChanging();

this.\_Date = value;

this.SendPropertyChanged("Date");

this.OnDateChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_FK\_Specialization", DbType="Int")]

public System.Nullable<int> FK\_Specialization

{

get

{

return this.\_FK\_Specialization;

}

set

{

if ((this.\_FK\_Specialization != value))

{

if (this.\_SpecializationTable.HasLoadedOrAssignedValue)

{

throw new System.Data.Linq.ForeignKeyReferenceAlreadyHasValueException();

}

this.OnFK\_SpecializationChanging(value);

this.SendPropertyChanging();

this.\_FK\_Specialization = value;

this.SendPropertyChanged("FK\_Specialization");

this.OnFK\_SpecializationChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_CostOfCommercOch", DbType="Float")]

public System.Nullable<double> CostOfCommercOch

{

get

{

return this.\_CostOfCommercOch;

}

set

{

if ((this.\_CostOfCommercOch != value))

{

this.OnCostOfCommercOchChanging(value);

this.SendPropertyChanging();

this.\_CostOfCommercOch = value;

this.SendPropertyChanged("CostOfCommercOch");

this.OnCostOfCommercOchChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_CostOfCommercOchIn", DbType="Float")]

public System.Nullable<double> CostOfCommercOchIn

{

get

{

return this.\_CostOfCommercOchIn;

}

set

{

if ((this.\_CostOfCommercOchIn != value))

{

this.OnCostOfCommercOchInChanging(value);

this.SendPropertyChanging();

this.\_CostOfCommercOchIn = value;

this.SendPropertyChanged("CostOfCommercOchIn");

this.OnCostOfCommercOchInChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_CostOfCommercZaoch", DbType="Float")]

public System.Nullable<double> CostOfCommercZaoch

{

get

{

return this.\_CostOfCommercZaoch;

}

set

{

if ((this.\_CostOfCommercZaoch != value))

{

this.OnCostOfCommercZaochChanging(value);

this.SendPropertyChanging();

this.\_CostOfCommercZaoch = value;

this.SendPropertyChanged("CostOfCommercZaoch");

this.OnCostOfCommercZaochChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_CostOfCommercEvening", DbType="Float")]

public System.Nullable<double> CostOfCommercEvening

{

get

{

return this.\_CostOfCommercEvening;

}

set

{

if ((this.\_CostOfCommercEvening != value))

{

this.OnCostOfCommercEveningChanging(value);

this.SendPropertyChanging();

this.\_CostOfCommercEvening = value;

this.SendPropertyChanged("CostOfCommercEvening");

this.OnCostOfCommercEveningChanged();

}

}

}

[global::System.Data.Linq.Mapping.AssociationAttribute(Name="SpecializationTable\_EducationCostTable", Storage="\_SpecializationTable", ThisKey="FK\_Specialization", OtherKey="SpecializationTableID", IsForeignKey=true)]

public SpecializationTable SpecializationTable

{

get

{

return this.\_SpecializationTable.Entity;

}

set

{

SpecializationTable previousValue = this.\_SpecializationTable.Entity;

if (((previousValue != value)

|| (this.\_SpecializationTable.HasLoadedOrAssignedValue == false)))

{

this.SendPropertyChanging();

if ((previousValue != null))

{

this.\_SpecializationTable.Entity = null;

previousValue.EducationCostTable.Remove(this);

}

this.\_SpecializationTable.Entity = value;

if ((value != null))

{

value.EducationCostTable.Add(this);

this.\_FK\_Specialization = value.SpecializationTableID;

}

else

{

this.\_FK\_Specialization = default(Nullable<int>);

}

this.SendPropertyChanged("SpecializationTable");

}

}

}

public event PropertyChangingEventHandler PropertyChanging;

public event PropertyChangedEventHandler PropertyChanged;

protected virtual void SendPropertyChanging()

{

if ((this.PropertyChanging != null))

{

this.PropertyChanging(this, emptyChangingEventArgs);

}

}

protected virtual void SendPropertyChanged(String propertyName)

{

if ((this.PropertyChanged != null))

{

this.PropertyChanged(this, new PropertyChangedEventArgs(propertyName));

}

}

}

[global::System.Data.Linq.Mapping.TableAttribute(Name="dbo.IndicatorsTable")]

public partial class IndicatorsTable : INotifyPropertyChanging, INotifyPropertyChanged

{

private static PropertyChangingEventArgs emptyChangingEventArgs = new PropertyChangingEventArgs(String.Empty);

private int \_IndicatorsTableID;

private System.Nullable<bool> \_Active;

private string \_Name;

private string \_Measure;

private string \_Formula;

private System.Nullable<int> \_FK\_IndicatorClass;

private System.Nullable<int> \_DataType;

private System.Nullable<int> \_SortID;

private EntitySet<CollectedIndicatorsForR> \_CollectedIndicatorsForR;

private EntitySet<CollectedIndocators> \_CollectedIndocators;

private EntitySet<ConfirmationHistory> \_ConfirmationHistory;

private EntitySet<IndicatorsAndRolesMappingTable> \_IndicatorsAndRolesMappingTable;

private EntitySet<IndicatorsAndUsersMapping> \_IndicatorsAndUsersMapping;

private EntitySet<PlannedIndicator> \_PlannedIndicator;

private EntitySet<ReportArchiveAndIndicatorsMappingTable> \_ReportArchiveAndIndicatorsMappingTable;

private EntityRef<IndicatorClass> \_IndicatorClass;

#region Extensibility Method Definitions

partial void OnLoaded();

partial void OnValidate(System.Data.Linq.ChangeAction action);

partial void OnCreated();

partial void OnIndicatorsTableIDChanging(int value);

partial void OnIndicatorsTableIDChanged();

partial void OnActiveChanging(System.Nullable<bool> value);

partial void OnActiveChanged();

partial void OnNameChanging(string value);

partial void OnNameChanged();

partial void OnMeasureChanging(string value);

partial void OnMeasureChanged();

partial void OnFormulaChanging(string value);

partial void OnFormulaChanged();

partial void OnFK\_IndicatorClassChanging(System.Nullable<int> value);

partial void OnFK\_IndicatorClassChanged();

partial void OnDataTypeChanging(System.Nullable<int> value);

partial void OnDataTypeChanged();

partial void OnSortIDChanging(System.Nullable<int> value);

partial void OnSortIDChanged();

#endregion

public IndicatorsTable()

{

this.\_CollectedIndicatorsForR = new EntitySet<CollectedIndicatorsForR>(new Action<CollectedIndicatorsForR>(this.attach\_CollectedIndicatorsForR), new Action<CollectedIndicatorsForR>(this.detach\_CollectedIndicatorsForR));

this.\_CollectedIndocators = new EntitySet<CollectedIndocators>(new Action<CollectedIndocators>(this.attach\_CollectedIndocators), new Action<CollectedIndocators>(this.detach\_CollectedIndocators));

this.\_ConfirmationHistory = new EntitySet<ConfirmationHistory>(new Action<ConfirmationHistory>(this.attach\_ConfirmationHistory), new Action<ConfirmationHistory>(this.detach\_ConfirmationHistory));

this.\_IndicatorsAndRolesMappingTable = new EntitySet<IndicatorsAndRolesMappingTable>(new Action<IndicatorsAndRolesMappingTable>(this.attach\_IndicatorsAndRolesMappingTable), new Action<IndicatorsAndRolesMappingTable>(this.detach\_IndicatorsAndRolesMappingTable));

this.\_IndicatorsAndUsersMapping = new EntitySet<IndicatorsAndUsersMapping>(new Action<IndicatorsAndUsersMapping>(this.attach\_IndicatorsAndUsersMapping), new Action<IndicatorsAndUsersMapping>(this.detach\_IndicatorsAndUsersMapping));

this.\_PlannedIndicator = new EntitySet<PlannedIndicator>(new Action<PlannedIndicator>(this.attach\_PlannedIndicator), new Action<PlannedIndicator>(this.detach\_PlannedIndicator));

this.\_ReportArchiveAndIndicatorsMappingTable = new EntitySet<ReportArchiveAndIndicatorsMappingTable>(new Action<ReportArchiveAndIndicatorsMappingTable>(this.attach\_ReportArchiveAndIndicatorsMappingTable), new Action<ReportArchiveAndIndicatorsMappingTable>(this.detach\_ReportArchiveAndIndicatorsMappingTable));

this.\_IndicatorClass = default(EntityRef<IndicatorClass>);

OnCreated();

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_IndicatorsTableID", AutoSync=AutoSync.OnInsert, DbType="Int NOT NULL IDENTITY", IsPrimaryKey=true, IsDbGenerated=true)]

public int IndicatorsTableID

{

get

{

return this.\_IndicatorsTableID;

}

set

{

if ((this.\_IndicatorsTableID != value))

{

this.OnIndicatorsTableIDChanging(value);

this.SendPropertyChanging();

this.\_IndicatorsTableID = value;

this.SendPropertyChanged("IndicatorsTableID");

this.OnIndicatorsTableIDChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_Active", DbType="Bit")]

public System.Nullable<bool> Active

{

get

{

return this.\_Active;

}

set

{

if ((this.\_Active != value))

{

this.OnActiveChanging(value);

this.SendPropertyChanging();

this.\_Active = value;

this.SendPropertyChanged("Active");

this.OnActiveChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_Name", DbType="VarChar(1000) NOT NULL", CanBeNull=false)]

public string Name

{

get

{

return this.\_Name;

}

set

{

if ((this.\_Name != value))

{

this.OnNameChanging(value);

this.SendPropertyChanging();

this.\_Name = value;

this.SendPropertyChanged("Name");

this.OnNameChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_Measure", DbType="VarChar(10) NOT NULL", CanBeNull=false)]

public string Measure

{

get

{

return this.\_Measure;

}

set

{

if ((this.\_Measure != value))

{

this.OnMeasureChanging(value);

this.SendPropertyChanging();

this.\_Measure = value;

this.SendPropertyChanged("Measure");

this.OnMeasureChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_Formula", DbType="VarChar(5000)")]

public string Formula

{

get

{

return this.\_Formula;

}

set

{

if ((this.\_Formula != value))

{

this.OnFormulaChanging(value);

this.SendPropertyChanging();

this.\_Formula = value;

this.SendPropertyChanged("Formula");

this.OnFormulaChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_FK\_IndicatorClass", DbType="Int")]

public System.Nullable<int> FK\_IndicatorClass

{

get

{

return this.\_FK\_IndicatorClass;

}

set

{

if ((this.\_FK\_IndicatorClass != value))

{

if (this.\_IndicatorClass.HasLoadedOrAssignedValue)

{

throw new System.Data.Linq.ForeignKeyReferenceAlreadyHasValueException();

}

this.OnFK\_IndicatorClassChanging(value);

this.SendPropertyChanging();

this.\_FK\_IndicatorClass = value;

this.SendPropertyChanged("FK\_IndicatorClass");

this.OnFK\_IndicatorClassChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_DataType", DbType="Int")]

public System.Nullable<int> DataType

{

get

{

return this.\_DataType;

}

set

{

if ((this.\_DataType != value))

{

this.OnDataTypeChanging(value);

this.SendPropertyChanging();

this.\_DataType = value;

this.SendPropertyChanged("DataType");

this.OnDataTypeChanged();

}

}

}

[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="\_SortID", DbType="Int")]

public System.Nullable<int> SortID

{

get

{

return this.\_SortID;

}

set

{

if ((this.\_SortID != value))

{

this.OnSortIDChanging(value);

this.SendPropertyChanging();

this.\_SortID = value;

this.SendPropertyChanged("SortID");

this.OnSortIDChanged();

}

}

}

[global::System.Data.Linq.Mapping.AssociationAttribute(Name="IndicatorsTable\_CollectedIndicatorsForR", Storage="\_CollectedIndicatorsForR", ThisKey="IndicatorsTableID", OtherKey="FK\_IndicatorsTable")]

public EntitySet<CollectedIndicatorsForR> CollectedIndicatorsForR

{

get

{

return this.\_CollectedIndicatorsForR;

}

set

{

this.\_CollectedIndicatorsForR.Assign(value);

}

}

[global::System.Data.Linq.Mapping.AssociationAttribute(Name="IndicatorsTable\_CollectedIndocators", Storage="\_CollectedIndocators", ThisKey="IndicatorsTableID", OtherKey="FK\_Indicators")]

public EntitySet<CollectedIndocators> CollectedIndocators

{

get

{

return this.\_CollectedIndocators;

}

set

{

this.\_CollectedIndocators.Assign(value);

}

}

[global::System.Data.Linq.Mapping.AssociationAttribute(Name="IndicatorsTable\_ConfirmationHistory", Storage="\_ConfirmationHistory", ThisKey="IndicatorsTableID", OtherKey="FK\_IndicatorsTable")]

public EntitySet<ConfirmationHistory> ConfirmationHistory

{

get

{

return this.\_ConfirmationHistory;

}

set

{

this.\_ConfirmationHistory.Assign(value);

}

}

[global::System.Data.Linq.Mapping.AssociationAttribute(Name="IndicatorsTable\_IndicatorsAndRolesMappingTable", Storage="\_IndicatorsAndRolesMappingTable", ThisKey="IndicatorsTableID", OtherKey="FK\_Indicators")]

public EntitySet<IndicatorsAndRolesMappingTable> IndicatorsAndRolesMappingTable

{

get

{

return this.\_IndicatorsAndRolesMappingTable;

}

set

{

this.\_IndicatorsAndRolesMappingTable.Assign(value);

}

}

[global::System.Data.Linq.Mapping.AssociationAttribute(Name="IndicatorsTable\_IndicatorsAndUsersMapping", Storage="\_IndicatorsAndUsersMapping", ThisKey="IndicatorsTableID", OtherKey="FK\_IndicatorsTable")]

public EntitySet<IndicatorsAndUsersMapping> IndicatorsAndUsersMapping

{

get

{

return this.\_IndicatorsAndUsersMapping;

}

set

{

this.\_IndicatorsAndUsersMapping.Assign(value);

}

}

[global::System.Data.Linq.Mapping.AssociationAttribute(Name="IndicatorsTable\_PlannedIndicator", Storage="\_PlannedIndicator", ThisKey="IndicatorsTableID", OtherKey="FK\_IndicatorsTable")]

public EntitySet<PlannedIndicator> PlannedIndicator

{

get

{

return this.\_PlannedIndicator;

}

set

{

this.\_PlannedIndicator.Assign(value);

}

}

[global::System.Data.Linq.Mapping.AssociationAttribute(Name="IndicatorsTable\_ReportArchiveAndIndicatorsMappingTable", Storage="\_ReportArchiveAndIndicatorsMappingTable", ThisKey="IndicatorsTableID", OtherKey="FK\_IndicatorsTable")]

public EntitySet<ReportArchiveAndIndicatorsMappingTable> ReportArchiveAndIndicatorsMappingTable

{

get

{

return this.\_ReportArchiveAndIndicatorsMappingTable;

}

set

{

this.\_ReportArchiveAndIndicatorsMappingTable.Assign(value);

}

}

[global::System.Data.Linq.Mapping.AssociationAttribute(Name="IndicatorClass\_IndicatorsTable", Storage="\_IndicatorClass", ThisKey="FK\_IndicatorClass", OtherKey="IndicatorClassID", IsForeignKey=true)]

public IndicatorClass IndicatorClass

{

get

{

return this.\_IndicatorClass.Entity;

}

set

{

IndicatorClass previousValue = this.\_IndicatorClass.Entity;

if (((previousValue != value)

|| (this.\_IndicatorClass.HasLoadedOrAssignedValue == false)))

{

this.SendPropertyChanging();

if ((previousValue != null))

{

this.\_IndicatorClass.Entity = null;

previousValue.IndicatorsTable.Remove(this);

}

this.\_IndicatorClass.Entity = value;

if ((value != null))

{

value.IndicatorsTable.Add(this);

this.\_FK\_IndicatorClass = value.IndicatorClassID;

}

else

{

this.\_FK\_IndicatorClass = default(Nullable<int>);

}

this.SendPropertyChanged("IndicatorClass");

}

}

}

public event PropertyChangingEventHandler PropertyChanging;

public event PropertyChangedEventHandler PropertyChanged;

protected virtual void SendPropertyChanging()

{

if ((this.PropertyChanging != null))

{

this.PropertyChanging(this, emptyChangingEventArgs);

}

}

protected virtual void SendPropertyChanged(String propertyName)

{

if ((this.PropertyChanged != null))

{

this.PropertyChanged(this, new PropertyChangedEventArgs(propertyName));

}

}

private void attach\_CollectedIndicatorsForR(CollectedIndicatorsForR entity)

{

this.SendPropertyChanging();

entity.IndicatorsTable = this;

}

private void detach\_CollectedIndicatorsForR(CollectedIndicatorsForR entity)

{

this.SendPropertyChanging();

entity.IndicatorsTable = null;

}

private void attach\_CollectedIndocators(CollectedIndocators entity)

{

this.SendPropertyChanging();

entity.IndicatorsTable = this;

}

private void detach\_CollectedIndocators(CollectedIndocators entity)

{

this.SendPropertyChanging();

entity.IndicatorsTable = null;

}

private void attach\_ConfirmationHistory(ConfirmationHistory entity)

{

this.SendPropertyChanging();

entity.IndicatorsTable = this;

}

private void detach\_ConfirmationHistory(ConfirmationHistory entity)

{

this.SendPropertyChanging();

entity.IndicatorsTable = null;

}

private void attach\_IndicatorsAndRolesMappingTable(IndicatorsAndRolesMappingTable entity)

{

this.SendPropertyChanging();

entity.IndicatorsTable = this;

}

private void detach\_IndicatorsAndRolesMappingTable(IndicatorsAndRolesMappingTable entity)

{

this.SendPropertyChanging();

entity.IndicatorsTable = null;

}

private void attach\_IndicatorsAndUsersMapping(IndicatorsAndUsersMapping entity)

{

this.SendPropertyChanging();

entity.IndicatorsTable = this;

}

private void detach\_IndicatorsAndUsersMapping(IndicatorsAndUsersMapping entity)

{

this.SendPropertyChanging();

entity.IndicatorsTable = null;

}

private void attach\_PlannedIndicator(PlannedIndicator entity)

{

this.SendPropertyChanging();

entity.IndicatorsTable = this;

}

private void detach\_PlannedIndicator(PlannedIndicator entity)

{

this.SendPropertyChanging();

entity.IndicatorsTable = null;

}

private void attach\_ReportArchiveAndIndicatorsMappingTable(ReportArchiveAndIndicatorsMappingTable entity)

{

this.SendPropertyChanging();

entity.IndicatorsTable = this;

}

private void detach\_ReportArchiveAndIndicatorsMappingTable(ReportArchiveAndIndicatorsMappingTable entity)

{

this.SendPropertyChanging();

entity.IndicatorsTable = null;

}

}

}

#pragma warning restore 1591

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Filename: MessageBox.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

namespace KPIWeb

{

public class MessageBox

{

public static void Show(System.Web.UI.UserControl parent, string message)

{

Show(parent, "MessageBoxShow", message);

}

public static void Show(System.Web.UI.UserControl parent, string functionName, string message)

{

if (parent != null && parent.Page != null)

{

Show(parent.Page, functionName, message);

}

}

public static void Show(System.Web.UI.Page parent, string message)

{

Show(parent, "MessageBoxShow", message);

}

public static void Show(System.Web.UI.Page parent, string functionName, string message)

{

if (parent != null)

{

parent.RegisterStartupScript(functionName, "<script language=\"JavaScript\"> function " + functionName + " (){ alert('" + message.Replace("\\", "\\\\").Replace("\r\n", "\\n").Replace("'", "\\'") + "');} </script>");

}

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Filename: MultiUser.aspx.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

using System.Data;

using System.Web.Security;

namespace KPIWeb

{

public partial class MultiUser1 : System.Web.UI.Page

{

protected void Page\_Load(object sender, EventArgs e)

{

Serialization UserSer = (Serialization)Session["UserID"];

if (UserSer == null)

{

Response.Redirect("~/Account/UserLogin.aspx");

}

KPIWebDataContext KPIWebDataContext = new KPIWebDataContext();

UsersTable user = (from usersTables in KPIWebDataContext.UsersTable

where usersTables.UsersTableID == UserSer.Id

select usersTables).FirstOrDefault();

if (user != null)

{

List<MultiUser> MultiuserList = (from a in KPIWebDataContext.MultiUser

where a.Active == true

&& a.FK\_UserCanAccess == user.UsersTableID

select a).ToList();

if (MultiuserList.Count>0)

{

List<UsersTable> UsersList =new List<UsersTable>();

UsersList.Add(user);

foreach (MultiUser MultUs in MultiuserList)

{

UsersList.Add ((from a in KPIWebDataContext.UsersTable where a.UsersTableID == MultUs.FK\_UserToAccess && a.Active == true select a).FirstOrDefault());

}

DataTable dataTable = new DataTable();

dataTable.Columns.Add(new DataColumn("Name", typeof(string)));

dataTable.Columns.Add(new DataColumn("UserID", typeof(string)));

foreach (UsersTable User in UsersList)

{

DataRow dataRow = dataTable.NewRow();

if (User.Position!=null)

{

dataRow["Name"] = User.Position;

}

else

{

dataRow["Name"] = User.Email;

}

dataRow["UserID"] = User.UsersTableID;

dataTable.Rows.Add(dataRow);

}

GridView1.DataSource = dataTable;

GridView1.DataBind();

}

else

{

Response.Redirect("~/Default.aspx");

}

}

else

{

Response.Redirect("~/Default.aspx");

}

}

public void Directions(UsersTable user)

{

if (user.Position == null)

{

FormsAuthentication.SetAuthCookie(user.Email, true);

}

else if (user.Position.Length > 2)

{

FormsAuthentication.SetAuthCookie(user.Position, true);

}

else

{

FormsAuthentication.SetAuthCookie(user.Email, true);

}

int accessLevel = (int)user.AccessLevel;

if (accessLevel == 10)

{

Response.Redirect("~/AutomationDepartment/Main.aspx");

}

else if (accessLevel == 9)

{

Response.Redirect("~/StatisticsDepartment/MonitoringMain.aspx");

}

else if (accessLevel == 5)

{

Response.Redirect("~/Rector/RectorMain.aspx");

}

else if (accessLevel == 3)

{

Response.Redirect("~/FinKadr/OtdelChooseReport.aspx");

}

else if (accessLevel == 7)

{

Response.Redirect("~/Rector/RMain.aspx");

}

else if (accessLevel == 0)

{

Response.Redirect("~/Reports\_/ChooseReport.aspx");

}

else //если входим сюда то что то не так) скорей всего пользователю не присвоен уровень в UsersTable

{

FormsAuthentication.SignOut();

Session.Abandon();

Response.Redirect("~/Account/UserLogin.aspx");

}

}

protected void GoToMainClick(object sender, EventArgs e)

{

Button button = (Button)sender;

{

KPIWebDataContext KPIWebDataContext = new KPIWebDataContext();

UsersTable user = (from usersTables in KPIWebDataContext.UsersTable

where usersTables.UsersTableID == Convert.ToUInt32(button.CommandArgument)

select usersTables).FirstOrDefault();

Serialization UserSerId = new Serialization(user.UsersTableID);

Session["UserID"] = UserSerId;

Directions(user);

}

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Filename: MultiUser.aspx.designer.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//------------------------------------------------------------------------------

// <auto-generated>

// This code was generated by a tool.

//

// Changes to this file may cause incorrect behavior and will be lost if

// the code is regenerated.

// </auto-generated>

//------------------------------------------------------------------------------

namespace KPIWeb {

public partial class MultiUser1 {

/// <summary>

/// GridView1 control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.GridView GridView1;

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Filename: OtdelSession.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using System;

using System.Collections.Generic;

using System.Linq;

using System.Security.Cryptography.X509Certificates;

using System.Web;

using KPIWeb.FinKadr;

using KPIWeb.Rector;

namespace KPIWeb

{

[Serializable]

public class OtdelSession

{

public OtdelResult.Struct sesStruct { get; set; }

public int sesViewType { get; set; } // тип просмотра // 0 - просмотр для структурных подразделений // 1 - просмотр

// public int sesLevelID { get; set; } // ID структуры

public int sesParamID { get; set; } // ID параметра

public int sesParamType { get; set; }// Тип параметра // 0 - целевой показатель / 1 - расчетный показатель / 2 - базовый показатель

public int sesReportID { get; set; }

public string sesName { get; set; }

public int sesSpecID { get; set; }

public OtdelSession(OtdelResult.Struct \_sesStruct, int \_sesViewType, int \_sesParamID, int \_sesParamType, int \_sesReportID, int \_sesSpecID)

{

this.sesStruct = \_sesStruct;

this.sesViewType = \_sesViewType;

this.sesParamID = \_sesParamID;

this.sesParamType = \_sesParamType;

this.sesReportID = \_sesReportID;

this.sesSpecID = \_sesSpecID;

}

public OtdelSession(OtdelResult.Struct \_sesStruct, int \_sesViewType, int \_sesParamID, int \_sesParamType, int \_sesReportID, int \_sesSpecID, string \_sesName)

{

this.sesStruct = \_sesStruct;

this.sesViewType = \_sesViewType;

this.sesParamID = \_sesParamID;

this.sesParamType = \_sesParamType;

this.sesReportID = \_sesReportID;

this.sesName = \_sesName;

this.sesSpecID = \_sesSpecID;

}

}

[Serializable]

public class OtdelHistorySession

{

public OtdelSession[] OtdelSession = new OtdelSession[10];//{ get; set; }

public int SessionCount { get; set; }

public int CurrentSession { get; set; }

public bool Visible { get; set; }

}

[Serializable]

public class OtdelParametrType

{

public int paramType { set; get; }

public OtdelParametrType(int \_paramType)

{

this.paramType = \_paramType;

}

}

[Serializable]

public class OtdelShowUnConfirmed

{

public bool DoShowUnConfirmed { set; get; }

public OtdelShowUnConfirmed(bool DoShowUnConfirmed\_)

{

this.DoShowUnConfirmed = DoShowUnConfirmed\_;

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Filename: OwnersProcessInfo.aspx.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

using System.Data;

namespace KPIWeb

{

public partial class OwnersProcessInfo : System.Web.UI.Page

{

public class MyObject

{

public int Id;

public int ParentId;

public string Name;

}

private void BindTree(IEnumerable<MyObject> list, TreeNode parentNode)

{

var nodes = list.Where(x => parentNode == null ? x.ParentId == 0 : x.ParentId == int.Parse(parentNode.Value));

foreach (var node in nodes)

{

TreeNode newNode = new TreeNode(node.Name, node.Id.ToString());

newNode.SelectAction = TreeNodeSelectAction.None;

if (parentNode == null)

{

TreeView1.Nodes.Add(newNode);

}

else

{

parentNode.ChildNodes.Add(newNode);

}

BindTree(list, newNode);

}

}

protected void Page\_Load(object sender, EventArgs e)

{

// Label5.Text = getColoredName(0, 0, 0, 0, 0, "");

Label1.Text = getColoredName(1, 1, 0, 0, 0, "Утверждено");

Label2.Text = getColoredName(1, 0, 1, 0, 0, "Ожидает утверждения");

Label3.Text = getColoredName(1, 0, 0, 1, 0, "В процессе заполнения");

Label4.Text = getColoredName(1, 0, 0, 0, 1, "Отчет не открывался");

Label5.Text = getColoredName(1, 0, 0, 0, 0, "В процессе заполнения подразделениями");

}

public string getColoredName(int all,int conf,int toconf,int started, int notstarted,string name)

{

if (all == 0)

{

//return name;

return null;

}

else if (all == conf)

{

//return "<font style=\"color:#27a327;font-weight: bold;\">" + name + "</font> ";

return null;

}

else if(all == notstarted)

{

return "<font style=\"color:#92000a;font-weight: bold;\">" + name + "</font> ";

}

else if (all == started)

{

return "<font style=\"color:#f4a900;font-weight: bold;\">" + name + "</font> ";

}

else if (all == toconf)

{

return "<font style=\"color:#27a327;font-weight: bold;\">" + name + "</font> ";

}

else

{

return "<font style=\"color:#044857;font-weight: bold;\">" + name + "</font> ";

}

}

protected void Button1\_Click(object sender, EventArgs e)

{

if (TextBox1.Text == "123\_")

{

Label1.Visible = true;

Label2.Visible = true;

Label3.Visible = true;

Label4.Visible = true;

Label5.Visible = true;

KPIWebDataContext kPiDataContext = new KPIWebDataContext();

//////////////////////////////////////////////////////////////////////////

DataTable dataTable = new DataTable();

dataTable.Columns.Add(new DataColumn("First", typeof(string)));

dataTable.Columns.Add(new DataColumn("Second", typeof(string)));

dataTable.Columns.Add(new DataColumn("Third", typeof(string)));

dataTable.Columns.Add(new DataColumn("Email", typeof(string)));

List<ThirdLevelSubdivisionTable> ThirdLevelWithEditList = (from a in kPiDataContext.ThirdLevelSubdivisionTable

join b in kPiDataContext.UsersTable

on a.ThirdLevelSubdivisionTableID equals b.FK\_ThirdLevelSubdivisionTable

join c in kPiDataContext.BasicParametrsAndUsersMapping

on b.UsersTableID equals c.FK\_UsersTable

where a.Active == true

&& b.Active == true

&& c.Active == true

&& c.CanEdit == true

select a).Distinct().ToList();

List<ThirdLevelSubdivisionTable> ThirdLevelNotNullValue = (from a in kPiDataContext.ThirdLevelSubdivisionTable

join b in kPiDataContext.CollectedBasicParametersTable

on a.ThirdLevelSubdivisionTableID equals b.FK\_ThirdLevelSubdivisionTable

where b.CollectedValue!=null

select a).Distinct().ToList();

foreach (ThirdLevelSubdivisionTable cur in ThirdLevelNotNullValue)

{

ThirdLevelWithEditList.Remove(cur);

}

foreach (ThirdLevelSubdivisionTable CurrentThirdLevel in ThirdLevelWithEditList)

{

DataRow dataRow2 = dataTable.NewRow();

dataRow2["First"] = (from a in kPiDataContext.SecondLevelSubdivisionTable

join b in kPiDataContext.FirstLevelSubdivisionTable

on a.FK\_FirstLevelSubdivisionTable equals b.FirstLevelSubdivisionTableID

where a.SecondLevelSubdivisionTableID == CurrentThirdLevel.FK\_SecondLevelSubdivisionTable

select b.Name).FirstOrDefault();

dataRow2["Second"] = (from a in kPiDataContext.SecondLevelSubdivisionTable

where a.SecondLevelSubdivisionTableID == CurrentThirdLevel.FK\_SecondLevelSubdivisionTable

select a.Name).FirstOrDefault();

dataRow2["Third"] = CurrentThirdLevel.Name;

dataRow2["Email"] = (from a in kPiDataContext.UsersTable

join b in kPiDataContext.BasicParametrsAndUsersMapping

on a.UsersTableID equals b.FK\_UsersTable

where a.FK\_ThirdLevelSubdivisionTable == CurrentThirdLevel.ThirdLevelSubdivisionTableID

select a.Email).FirstOrDefault();

dataTable.Rows.Add(dataRow2);

}

GridView1.DataSource = dataTable;

GridView1.DataBind();

///////////////////////////////////////////////////////////////////////////////

TreeView1.DataSource = null;

TreeView1.DataBind();

//Label1.Visible = true;

List<MyObject> list = new List<MyObject>();

int allZeroLevel = 0;

int insertZeroLevel = 0;

int confirmZeroLevel = 0;

int i = 1;

int UsrCnt = 0;

List<UsersTable> Users;

string tmp;

string tmp2;

#region get zero leve list

List<ZeroLevelSubdivisionTable> zeroLevelList = (from a in kPiDataContext.ZeroLevelSubdivisionTable

where a.Active == true

select a).ToList();

#endregion

// колво утвердивших (% ) // % ожидающих утверждения // % начавших заполнение

foreach (ZeroLevelSubdivisionTable zeroLevelItem in zeroLevelList)//по каждому университету

{

#region get first level list

List<FirstLevelSubdivisionTable> firstLevelList = (from b in kPiDataContext.FirstLevelSubdivisionTable

where b.FK\_ZeroLevelSubvisionTable == zeroLevelItem.ZeroLevelSubdivisionTableID

&& b.Active == true

select b).ToList();

#endregion

#region

int all0 = (from a in kPiDataContext.ThirdLevelParametrs where a.Active == true select a).Count();

int toconf0 = (from a in kPiDataContext.ThirdLevelParametrs

join b in kPiDataContext.CollectedBasicParametersTable

on a.ThirdLevelParametrsID equals b.FK\_ThirdLevelSubdivisionTable

where b.Status == 3

select a).Distinct().Count();

int conf0 = (from a in kPiDataContext.ThirdLevelParametrs

join b in kPiDataContext.CollectedBasicParametersTable

on a.ThirdLevelParametrsID equals b.FK\_ThirdLevelSubdivisionTable

where b.Status == 4

select a).Distinct().Count();

int started0 = (from a in kPiDataContext.ThirdLevelParametrs

join b in kPiDataContext.CollectedBasicParametersTable

on a.ThirdLevelParametrsID equals b.FK\_ThirdLevelSubdivisionTable

where b.CollectedValue != null

select a).Distinct().Count() - toconf0 - conf0;

int notstarted0 =all0 - toconf0 - started0 - conf0;

#endregion

i++;

int par0 = i;

tmp2 = getColoredName(all0,conf0,toconf0,started0,notstarted0, zeroLevelItem.Name);

if (tmp2!=null)

list.Add(new MyObject() { Id = i, ParentId = 0, Name = tmp2 + " : " + all0.ToString() + "/" + conf0.ToString() + "/" + toconf0.ToString() + "/" + started0.ToString() + "/" + notstarted0.ToString() });

foreach (FirstLevelSubdivisionTable firstLevelItem in firstLevelList)//по каждой академии

{

#region get second level list

List<SecondLevelSubdivisionTable> secondLevelList =

(from d in kPiDataContext.SecondLevelSubdivisionTable

where d.FK\_FirstLevelSubdivisionTable == firstLevelItem.FirstLevelSubdivisionTableID

&& d.Active == true

select d).ToList();

#endregion

#region

all0 = (from a in kPiDataContext.ThirdLevelSubdivisionTable

join b in kPiDataContext.SecondLevelSubdivisionTable

on a.FK\_SecondLevelSubdivisionTable equals b.SecondLevelSubdivisionTableID

where a.Active == true

&& b.FK\_FirstLevelSubdivisionTable == firstLevelItem.FirstLevelSubdivisionTableID

select a).Count();

toconf0 = (from a in kPiDataContext.ThirdLevelParametrs

join b in kPiDataContext.CollectedBasicParametersTable

on a.ThirdLevelParametrsID equals b.FK\_ThirdLevelSubdivisionTable

where b.Status == 3

&& b.FK\_FirstLevelSubdivisionTable == firstLevelItem.FirstLevelSubdivisionTableID

select a).Distinct().Count();

conf0 = (from a in kPiDataContext.ThirdLevelParametrs

join b in kPiDataContext.CollectedBasicParametersTable

on a.ThirdLevelParametrsID equals b.FK\_ThirdLevelSubdivisionTable

where b.Status == 4

&& b.FK\_FirstLevelSubdivisionTable == firstLevelItem.FirstLevelSubdivisionTableID

select a).Distinct().Count();

started0 = (from a in kPiDataContext.ThirdLevelParametrs

join b in kPiDataContext.CollectedBasicParametersTable

on a.ThirdLevelParametrsID equals b.FK\_ThirdLevelSubdivisionTable

where

b.FK\_FirstLevelSubdivisionTable == firstLevelItem.FirstLevelSubdivisionTableID

&& b.CollectedValue != null

select a).Distinct().Count() - toconf0 - conf0;

notstarted0 = all0 - toconf0 - started0 - conf0;

#endregion

i++;

int par1 = i;

//tmp2 = firstLevelItem.Name;

tmp2 = getColoredName(all0, conf0, toconf0, started0, notstarted0, firstLevelItem.Name);

if (tmp2 != null)

list.Add(new MyObject() { Id = i, ParentId = par0, Name = tmp2 + " : " + all0.ToString() + "/" + conf0.ToString() + "/" + toconf0.ToString() + "/" + started0.ToString() + "/" + notstarted0.ToString() });

foreach (SecondLevelSubdivisionTable secondLevelItem in secondLevelList)//по каждому факультету

{

#region get third level list

List<ThirdLevelSubdivisionTable> thirdLevelList =

(from f in kPiDataContext.ThirdLevelSubdivisionTable

where f.FK\_SecondLevelSubdivisionTable == secondLevelItem.SecondLevelSubdivisionTableID

&& f.Active == true

select f).ToList();

#endregion

#region

all0 = (from a in kPiDataContext.ThirdLevelSubdivisionTable

where a.Active == true

&& a.FK\_SecondLevelSubdivisionTable == secondLevelItem.SecondLevelSubdivisionTableID

select a).Count();

conf0 = (from a in kPiDataContext.ThirdLevelParametrs

join b in kPiDataContext.CollectedBasicParametersTable

on a.ThirdLevelParametrsID equals b.FK\_ThirdLevelSubdivisionTable

where b.Status == 4

&& b.FK\_SecondLevelSubdivisionTable == secondLevelItem.SecondLevelSubdivisionTableID

select a).Distinct().Count();

toconf0 = (from a in kPiDataContext.ThirdLevelParametrs

join b in kPiDataContext.CollectedBasicParametersTable

on a.ThirdLevelParametrsID equals b.FK\_ThirdLevelSubdivisionTable

where b.Status == 3

&& b.FK\_SecondLevelSubdivisionTable == secondLevelItem.SecondLevelSubdivisionTableID

select a).Distinct().Count();

started0 = (from a in kPiDataContext.ThirdLevelParametrs

join b in kPiDataContext.CollectedBasicParametersTable

on a.ThirdLevelParametrsID equals b.FK\_ThirdLevelSubdivisionTable

where

b.FK\_SecondLevelSubdivisionTable == secondLevelItem.SecondLevelSubdivisionTableID

&& b.CollectedValue != null

select a).Distinct().Count() - conf0 - toconf0;

notstarted0 = all0 - toconf0 - started0 - conf0;

#endregion

i++;

int par2 = i;

//tmp2 = secondLevelItem.Name;

tmp2 = getColoredName(all0, conf0, toconf0, started0, notstarted0, secondLevelItem.Name);

if (tmp2 != null)

list.Add(new MyObject() { Id = i, ParentId = par1, Name = tmp2 + " : " + all0.ToString() + "/" + conf0.ToString() + "/" + toconf0.ToString() + "/" + started0.ToString() + "/" + notstarted0.ToString() });

foreach (ThirdLevelSubdivisionTable thirdLevelItem in thirdLevelList)//по кафедре

{

#region

all0 = (from a in kPiDataContext.ThirdLevelSubdivisionTable

where a.Active == true

&& a.ThirdLevelSubdivisionTableID == thirdLevelItem.ThirdLevelSubdivisionTableID

select a).Count();

conf0 = (from a in kPiDataContext.ThirdLevelParametrs

join b in kPiDataContext.CollectedBasicParametersTable

on a.ThirdLevelParametrsID equals b.FK\_ThirdLevelSubdivisionTable

where b.Status == 4

&& b.FK\_ThirdLevelSubdivisionTable == thirdLevelItem.ThirdLevelSubdivisionTableID

select a).Distinct().Count();

toconf0 = (from a in kPiDataContext.ThirdLevelParametrs

join b in kPiDataContext.CollectedBasicParametersTable

on a.ThirdLevelParametrsID equals b.FK\_ThirdLevelSubdivisionTable

where b.Status == 3

&& b.FK\_ThirdLevelSubdivisionTable == thirdLevelItem.ThirdLevelSubdivisionTableID

select a).Distinct().Count();

started0 = (from a in kPiDataContext.ThirdLevelParametrs

join b in kPiDataContext.CollectedBasicParametersTable

on a.ThirdLevelParametrsID equals b.FK\_ThirdLevelSubdivisionTable

where

b.FK\_ThirdLevelSubdivisionTable == thirdLevelItem.ThirdLevelSubdivisionTableID

&& b.CollectedValue != null

select a).Distinct().Count() - toconf0 - conf0;

notstarted0 = all0 - toconf0 - started0 - conf0;

#endregion

i++;

int par3 = i;

// tmp2 = thirdLevelItem.Name;

tmp2 = getColoredName(all0, conf0, toconf0, started0, notstarted0, thirdLevelItem.Name);

if (tmp2 != null)

list.Add(new MyObject() { Id = i, ParentId = par2, Name = tmp2 + " : " + all0.ToString() + "/" + conf0.ToString() + "/" + toconf0.ToString() + "/" + started0.ToString() + "/" + notstarted0.ToString() });

}

}

}

}

BindTree(list, null);

TreeView1.CollapseAll();

}

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Filename: OwnersProcessInfo.aspx.designer.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//------------------------------------------------------------------------------

// <auto-generated>

// This code was generated by a tool.

//

// Changes to this file may cause incorrect behavior and will be lost if

// the code is regenerated.

// </auto-generated>

//------------------------------------------------------------------------------

namespace KPIWeb {

public partial class OwnersProcessInfo {

/// <summary>

/// TextBox1 control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.TextBox TextBox1;

/// <summary>

/// Button1 control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.Button Button1;

/// <summary>

/// GridView1 control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.GridView GridView1;

/// <summary>

/// Label1 control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.Label Label1;

/// <summary>

/// Label2 control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.Label Label2;

/// <summary>

/// Label3 control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.Label Label3;

/// <summary>

/// Label4 control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.Label Label4;

/// <summary>

/// Label5 control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.Label Label5;

/// <summary>

/// TreeView1 control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.TreeView TreeView1;

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Filename: Polish.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

namespace KPIWeb

{

public class Polish

{

static private bool IsDelimeter(char c)

{

if ((" =".IndexOf(c) != -1))

return true;

return false;

}

static private bool IsOperator(char с)

{

if (("+-/\*^()".IndexOf(с) != -1))

return true;

return false;

}

static private byte GetPriority(char s)

{

switch (s)

{

case '(': return 0;

case ')': return 1;

case '+': return 2;

case '-': return 3;

case '\*': return 4;

case '/': return 4;

case '^': return 5;

default: return 6;

}

}

static public double Calculate(string input)

{

string input2 = input;

string output = GetExpression(input2); //Преобразовываем выражение в постфиксную запись

double result = Counting(output); //Решаем полученное выражение

return result; //Возвращаем результат

}

static private string GetExpression(string input)

{

string output = string.Empty; //Строка для хранения выражения

Stack<char> operStack = new Stack<char>(); //Стек для хранения операторов

for (int i = 0; i < input.Length; i++) //Для каждого символа в входной строке

{

//Разделители пропускаем

if (IsDelimeter(input[i]))

continue; //Переходим к следующему символу

//Если символ - цифра, то считываем все число

if (Char.IsDigit(input[i])) //Если цифра

{

//Читаем до разделителя или оператора, что бы получить число

while (!IsDelimeter(input[i]) && !IsOperator(input[i]))

{

output += input[i]; //Добавляем каждую цифру числа к нашей строке

i++; //Переходим к следующему символу

if (i == input.Length) break; //Если символ - последний, то выходим из цикла

}

output += " "; //Дописываем после числа пробел в строку с выражением

i--; //Возвращаемся на один символ назад, к символу перед разделителем

}

//Если символ - оператор

if (IsOperator(input[i])) //Если оператор

{

if (input[i] == '(') //Если символ - открывающая скобка

operStack.Push(input[i]); //Записываем её в стек

else if (input[i] == ')') //Если символ - закрывающая скобка

{

//Выписываем все операторы до открывающей скобки в строку

char s = operStack.Pop();

while (s != '(')

{

output += s.ToString() + ' ';

s = operStack.Pop();

}

}

else //Если любой другой оператор

{

if (operStack.Count > 0) //Если в стеке есть элементы

if (GetPriority(input[i]) <= GetPriority(operStack.Peek())) //И если приоритет нашего оператора меньше или равен приоритету оператора на вершине стека

output += operStack.Pop().ToString() + " "; //То добавляем последний оператор из стека в строку с выражением

operStack.Push(char.Parse(input[i].ToString())); //Если стек пуст, или же приоритет оператора выше - добавляем операторов на вершину стека

}

}

}

//Когда прошли по всем символам, выкидываем из стека все оставшиеся там операторы в строку

while (operStack.Count > 0)

output += operStack.Pop() + " ";

return output; //Возвращаем выражение в постфиксной записи

}

static private double Counting(string input)

{

double result = 0; //Результат

Stack<double> temp = new Stack<double>(); //Dhtvtyysq стек для решения

for (int i = 0; i < input.Length; i++) //Для каждого символа в строке

{

//Если символ - цифра, то читаем все число и записываем на вершину стека

if (Char.IsDigit(input[i]))

{

string a = string.Empty;

while (!IsDelimeter(input[i]) && !IsOperator(input[i])) //Пока не разделитель

{

a += input[i]; //Добавляем

i++;

if (i == input.Length) break;

}

temp.Push(double.Parse(a)); //Записываем в стек

i--;

}

else if (IsOperator(input[i])) //Если символ - оператор

{

//Берем два последних значения из стека

double a = temp.Pop();

double b = temp.Pop();

switch (input[i]) //И производим над ними действие, согласно оператору

{

case '+': result = b + a; break;

case '-': result = b - a; break;

case '\*': result = b \* a; break;

case '/': result = b / a; break;

case '^': result = double.Parse(Math.Pow(double.Parse(b.ToString()), double.Parse(a.ToString())).ToString()); break;

}

temp.Push(result); //Результат вычисления записываем обратно в стек

}

}

return temp.Peek(); //Забираем результат всех вычислений из стека и возвращаем его

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Filename: RectorSession.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using System;

using System.Collections.Generic;

using System.Linq;

using System.Security.Cryptography.X509Certificates;

using System.Web;

using KPIWeb.Rector;

namespace KPIWeb

{

[Serializable]

public class RectorSession

{

public ForRCalc.Struct sesStruct { get; set; }

public int sesViewType { get; set; } // тип просмотра // 0 - просмотр для структурных подразделений // 1 - просмотр

// public int sesLevelID { get; set; } // ID структуры

public int sesParamID { get; set; } // ID параметра

public int sesParamType { get; set; }// Тип параметра // 0 - целевой показатель / 1 - расчетный показатель / 2 - базовый показатель

public int sesReportID { get; set; }

public string sesName { get; set; }

public int sesSpecID { get; set; }

public RectorSession(ForRCalc.Struct \_sesStruct, int \_sesViewType, int \_sesParamID, int \_sesParamType, int \_sesReportID, int \_sesSpecID)

{

this.sesStruct = \_sesStruct;

this.sesViewType = \_sesViewType;

this.sesParamID = \_sesParamID;

this.sesParamType = \_sesParamType;

this.sesReportID = \_sesReportID;

this.sesSpecID = \_sesSpecID;

}

public RectorSession(ForRCalc.Struct \_sesStruct, int \_sesViewType, int \_sesParamID, int \_sesParamType, int \_sesReportID, int \_sesSpecID, string \_sesName)

{

this.sesStruct = \_sesStruct;

this.sesViewType = \_sesViewType;

this.sesParamID = \_sesParamID;

this.sesParamType = \_sesParamType;

this.sesReportID = \_sesReportID;

this.sesName = \_sesName;

this.sesSpecID = \_sesSpecID;

}

}

[Serializable]

public class RectorHistorySession

{

public RectorSession[] RectorSession = new RectorSession[10];//{ get; set; }

public int SessionCount { get; set; }

public int CurrentSession { get; set; }

public bool Visible { get; set; }

}

[Serializable]

public class ParametrType

{

public int paramType { set; get; }

public ParametrType(int \_paramType)

{

this.paramType = \_paramType;

}

}

[Serializable]

public class ShowUnConfirmed

{

public bool DoShowUnConfirmed { set; get; }

public ShowUnConfirmed(bool DoShowUnConfirmed\_)

{

this.DoShowUnConfirmed = DoShowUnConfirmed\_;

}

}

[Serializable]

public class RectorChartSession

{

string name;

string info;

public List<int> IndicatorsList;

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Filename: RegisterProcessInfo.aspx.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

namespace KPIWeb

{

public partial class RegisterProcessInfo : System.Web.UI.Page

{

public class MyObject

{

public int Id;

public int ParentId;

public string Name;

//public string UrlAddr;

//public int Active;

}

private void BindTree(IEnumerable<MyObject> list, TreeNode parentNode)

{

var nodes = list.Where(x => parentNode == null ? x.ParentId == 0 : x.ParentId == int.Parse(parentNode.Value));

foreach (var node in nodes)

{

TreeNode newNode = new TreeNode(node.Name, node.Id.ToString());

/\* if (node.Active == 1)

{

//newNode.NavigateUrl = node.UrlAddr;

}

else\*/

newNode.SelectAction = TreeNodeSelectAction.None;

if (parentNode == null)

{

TreeView1.Nodes.Add(newNode);

}

else

{

parentNode.ChildNodes.Add(newNode);

}

BindTree(list, newNode);

}

}

protected void Page\_Load(object sender, EventArgs e)

{

}

protected void Button1\_Click(object sender, EventArgs e)

{

if (TextBox1.Text == "123\_")

{

TreeView1.DataSource=null;

TreeView1.DataBind();

Label1.Visible = true;

KPIWebDataContext kPiDataContext = new KPIWebDataContext();

List<MyObject> list = new List<MyObject>();

int allZeroLevel = 0;

int insertZeroLevel = 0;

int confirmZeroLevel = 0;

int i = 1;

int UsrCnt = 0;

List<UsersTable> Users;

string tmp;

string tmp2;

#region get zero leve list

List<ZeroLevelSubdivisionTable> zeroLevelList = (from a in kPiDataContext.ZeroLevelSubdivisionTable

where a.Active == true

select a).ToList();

#endregion

foreach (ZeroLevelSubdivisionTable zeroLevelItem in zeroLevelList)//по каждому университету

{

#region get first level list

List<FirstLevelSubdivisionTable> firstLevelList = (from b in kPiDataContext.FirstLevelSubdivisionTable

/\* join c in kPiDataContext.ReportArchiveAndLevelMappingTable

on b.FirstLevelSubdivisionTableID equals c.FK\_FirstLevelSubmisionTableId\*/

where b.FK\_ZeroLevelSubvisionTable == zeroLevelItem.ZeroLevelSubdivisionTableID

&& b.Active == true

// && c.Active == true

select b).ToList();

#endregion

//TextBox1.Text+="\_\_" + zeroLevelItem.Name +"\n";

i++;

int par0 = i;

UsrCnt = 0;

Users = (from a in kPiDataContext.UsersTable

where a.Active == true

&& a.AccessLevel!=9

&& a.AccessLevel!=10

&& a.FK\_ZeroLevelSubdivisionTable == zeroLevelItem.ZeroLevelSubdivisionTableID

&& a.FK\_FirstLevelSubdivisionTable == null

select a).ToList();

tmp = "";

foreach (UsersTable curuser in Users)

{

if (curuser.Confirmed == true)

{

tmp += "<font style=\"color:#00a34f;font-weight: bold;\">" + curuser.Email + "</font> ";

}

else

{

tmp += "<font style=\"color:#990000;font-weight: bold;\">" + curuser.Email + "</font> ";

}

}

UsrCnt = (from a in kPiDataContext.UsersTable

where a.Active == true

&& a.FK\_ZeroLevelSubdivisionTable == zeroLevelItem.ZeroLevelSubdivisionTableID

&& a.AccessLevel != 9

&& a.AccessLevel != 10

select a).Count();

tmp2 = "";

if (UsrCnt > 0)

{

tmp2 = "<font style=\"color:#0000FF;font-weight: bold;\">" + zeroLevelItem.Name + "</font> ";

}

else

{

tmp2 = zeroLevelItem.Name;

}

list.Add(new MyObject() { Id = i, ParentId = 0, Name = tmp2 + " ( " + UsrCnt.ToString() + ") " + tmp });

foreach (FirstLevelSubdivisionTable firstLevelItem in firstLevelList)//по каждой академии

{

#region get second level list

List<SecondLevelSubdivisionTable> secondLevelList =

(from d in kPiDataContext.SecondLevelSubdivisionTable

where d.FK\_FirstLevelSubdivisionTable == firstLevelItem.FirstLevelSubdivisionTableID

&& d.Active == true

select d).ToList();

#endregion

// TextBox1.Text +="\_\_\_\_" + firstLevelItem.Name+"\n";

i++;

int par1 = i;

UsrCnt = 0;

Users = (from a in kPiDataContext.UsersTable

where a.Active == true

&& a.FK\_FirstLevelSubdivisionTable == firstLevelItem.FirstLevelSubdivisionTableID

&& a.FK\_SecondLevelSubdivisionTable == null

select a).ToList();

tmp = "";

foreach (UsersTable curuser in Users)

{

if (curuser.Confirmed == true)

{

tmp += "<font style=\"color:#00a34f;font-weight: bold;\">" + curuser.Email + "</font> ";

}

else

{

tmp += "<font style=\"color:#990000;font-weight: bold;\">" + curuser.Email + "</font> ";

}

}

UsrCnt = (from a in kPiDataContext.UsersTable

where a.Active == true

&& a.FK\_FirstLevelSubdivisionTable == firstLevelItem.FirstLevelSubdivisionTableID

select a).Count();

tmp2 = "";

if (UsrCnt > 0)

{

tmp2 = "<font style=\"color:#0000FF;font-weight: bold;\">" + firstLevelItem.Name + "</font> ";

}

else

{

tmp2 = firstLevelItem.Name;

}

list.Add(new MyObject() { Id = i, ParentId = par0, Name = tmp2 + " (" + UsrCnt.ToString() + ") " + tmp });

foreach (SecondLevelSubdivisionTable secondLevelItem in secondLevelList)//по каждому факультету

{

// TextBox1.Text += "\_\_\_\_\_\_" + secondLevelItem.Name + "\n";

#region get third level list

List<ThirdLevelSubdivisionTable> thirdLevelList =

(from f in kPiDataContext.ThirdLevelSubdivisionTable

where f.FK\_SecondLevelSubdivisionTable == secondLevelItem.SecondLevelSubdivisionTableID

&& f.Active == true

select f).ToList();

#endregion

i++;

int par2 = i;

UsrCnt = 0;

Users = (from a in kPiDataContext.UsersTable

where a.Active == true

&& a.FK\_SecondLevelSubdivisionTable == secondLevelItem.SecondLevelSubdivisionTableID

&& a.FK\_ThirdLevelSubdivisionTable == null

select a).ToList();

tmp = "";

foreach (UsersTable curuser in Users)

{

if (curuser.Confirmed == true)

{

tmp += "<font style=\"color:#00a34f;font-weight: bold;\">" + curuser.Email + "</font> ";

}

else

{

tmp += "<font style=\"color:#990000;font-weight: bold;\">" + curuser.Email + "</font> ";

}

}

UsrCnt = (from a in kPiDataContext.UsersTable

where a.Active == true

&& a.FK\_SecondLevelSubdivisionTable == secondLevelItem.SecondLevelSubdivisionTableID

select a).Count();

tmp2 = "";

if (UsrCnt > 0)

{

tmp2 = "<font style=\"color:#0000FF;font-weight: bold;\">" + secondLevelItem.Name + "</font> ";

}

else

{

tmp2 = secondLevelItem.Name;

}

list.Add(new MyObject() { Id = i, ParentId = par1, Name = tmp2 + " (" + UsrCnt.ToString() + ") " + tmp });

foreach (ThirdLevelSubdivisionTable thirdLevelItem in thirdLevelList)//по кафедре

{

//TextBox1.Text += "\_\_\_\_\_\_\_\_" + thirdLevelItem.Name + "\n";

#region get fourth level list

/\*

List<FourthLevelSubdivisionTable> fourthLevelList = (from g in kPiDataContext.FourthLevelSubdivisionTable

where

g.FK\_ThirdLevelSubdivisionTable ==

thirdLevelItem.ThirdLevelSubdivisionTableID

&& g.Active == true

select g).ToList();

\*/

#endregion

i++;

int par3 = i;

UsrCnt = 0;

Users = (from a in kPiDataContext.UsersTable

where a.Active == true

&& a.FK\_ThirdLevelSubdivisionTable == thirdLevelItem.ThirdLevelSubdivisionTableID

&& a.FK\_FourthLevelSubdivisionTable == null

select a).ToList();

tmp = "";

foreach (UsersTable curuser in Users)

{

if (curuser.Confirmed == true)

{

tmp += "<font style=\"color:#00a34f;font-weight: bold;\">" + curuser.Email + "</font> ";

}

else

{

tmp += "<font style=\"color:#990000;font-weight: bold;\">" + curuser.Email + "</font> ";

}

}

UsrCnt = (from a in kPiDataContext.UsersTable

where a.Active == true

&& a.FK\_ThirdLevelSubdivisionTable == thirdLevelItem.ThirdLevelSubdivisionTableID

select a).Count();

tmp2 = "";

if (UsrCnt > 0)

{

tmp2 = "<font style=\"color:#0000FF;font-weight: bold;\">" + thirdLevelItem.Name + "</font> ";

}

else

{

tmp2 = thirdLevelItem.Name;

}

list.Add(new MyObject() { Id = i, ParentId = par2, Name = tmp2 + " (" + UsrCnt.ToString() + ") " + tmp });

/\*

foreach (FourthLevelSubdivisionTable fourthLevelItem in fourthLevelList)//по специальности

{

TextBox1.Text += "\_\_\_\_\_\_\_\_\_\_" + fourthLevelItem.Name + "\n";

}

\*/

}

}

}

}

BindTree(list, null);

TreeView1.CollapseAll();

//TreeView1.ExpandAll();

}

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Filename: RegisterProcessInfo.aspx.designer.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//------------------------------------------------------------------------------

// <auto-generated>

// This code was generated by a tool.

//

// Changes to this file may cause incorrect behavior and will be lost if

// the code is regenerated.

// </auto-generated>

//------------------------------------------------------------------------------

namespace KPIWeb {

public partial class RegisterProcessInfo {

/// <summary>

/// TextBox1 control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.TextBox TextBox1;

/// <summary>

/// Button1 control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.Button Button1;

/// <summary>

/// Label1 control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.Label Label1;

/// <summary>

/// TreeView1 control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.TreeView TreeView1;

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Filename: Result.aspx.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using System;

using System.Collections.Generic;

using System.Data;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

namespace KPIWeb.Rector

{

public partial class Result : System.Web.UI.Page

{

[Serializable]

public class Struct // класс описываюший структурные подразделения

{

public int Lv\_0 { get; set; }

public int Lv\_1 { get; set; }

public int Lv\_2 { get; set; }

public int Lv\_3 { get; set; }

public int Lv\_4 { get; set; }

public int Lv\_5 { get; set; }

public string Name { get; set; }

public Struct(int lv0, int lv1, int lv2, int lv3, int lv4, int lv5, string name)

{

Lv\_0 = lv0;

Lv\_1 = lv1;

Lv\_2 = lv2;

Lv\_3 = lv3;

Lv\_4 = lv4;

Lv\_5 = lv5;

Name = name;

}

public Struct(int lv0, int lv1, int lv2, int lv3, int lv4, string name)

{

Lv\_0 = lv0;

Lv\_1 = lv1;

Lv\_2 = lv2;

Lv\_3 = lv3;

Lv\_4 = lv4;

Lv\_5 = 0;

Name = name;

}

public Struct(int lv0, int lv1, int lv2, int lv3, string name)

{

Lv\_0 = lv0;

Lv\_1 = lv1;

Lv\_2 = lv2;

Lv\_3 = lv3;

Lv\_4 = 0;

Lv\_5 = 0;

Name = name;

}

public Struct(int lv0, int lv1, int lv2, string name)

{

Lv\_0 = lv0;

Lv\_1 = lv1;

Lv\_2 = lv2;

Lv\_3 = 0;

Lv\_4 = 0;

Lv\_5 = 0;

Name = name;

}

public Struct(int lv0, int lv1, string name)

{

Lv\_0 = lv0;

Lv\_1 = lv1;

Lv\_2 = 0;

Lv\_3 = 0;

Lv\_4 = 0;

Lv\_5 = 0;

Name = name;

}

public Struct(int lv0, string name)

{

Lv\_0 = lv0;

Lv\_1 = 0;

Lv\_2 = 0;

Lv\_3 = 0;

Lv\_4 = 0;

Lv\_5 = 0;

Name = name;

}

}

public List<Struct> GetChildStructList (Struct ParentStruct)

{

List<Struct> tmpStrucList = new List<Struct>();

int Level = 5;

Level = ParentStruct.Lv\_5 == 0 ? 4 : Level;

Level = ParentStruct.Lv\_4 == 0 ? 3 : Level;

Level = ParentStruct.Lv\_3 == 0 ? 2 : Level;

Level = ParentStruct.Lv\_2 == 0 ? 1 : Level;

Level = ParentStruct.Lv\_1 == 0 ? 0 : Level;

Level = ParentStruct.Lv\_0 == 0 ? -1 : Level;

KPIWebDataContext kpiWebDataContext = new KPIWebDataContext();

switch (Level)

{

case -1: // возвращаем все нулевого уровня, хотя там должен быть только КФУ

{

tmpStrucList = (from a in kpiWebDataContext.ZeroLevelSubdivisionTable

where a.Active == true

select new Struct(1,"") {

Lv\_0 = (int)a.ZeroLevelSubdivisionTableID,

Lv\_1 = 0,

Lv\_2 = 0,

Lv\_3 = 0,

Lv\_4 = 0,

Lv\_5 = 0,

Name = a.Name

}).ToList();

break;

}

case 0: // возвращаем все универы

{

tmpStrucList = (from a in kpiWebDataContext.FirstLevelSubdivisionTable

where a.Active == true

&& a.FK\_ZeroLevelSubvisionTable == ParentStruct.Lv\_0

select new Struct(1,"")

{

Lv\_0 = (int)a.FK\_ZeroLevelSubvisionTable,

Lv\_1 = (int)a.FirstLevelSubdivisionTableID,

Lv\_2 = 0,

Lv\_3 = 0,

Lv\_4 = 0,

Lv\_5 = 0,

Name = a.Name

}).ToList();

break;

}

case 1: // возвращаем все факультеты

{

tmpStrucList = (from a in kpiWebDataContext.FirstLevelSubdivisionTable

join b in kpiWebDataContext.SecondLevelSubdivisionTable

on a.FirstLevelSubdivisionTableID equals b.FK\_FirstLevelSubdivisionTable

where a.Active == true

&& b.Active == true

&& a.FK\_ZeroLevelSubvisionTable == ParentStruct.Lv\_0

&& b.FK\_FirstLevelSubdivisionTable == ParentStruct.Lv\_1

select new Struct(1,"")

{

Lv\_0 = (int)a.FK\_ZeroLevelSubvisionTable,

Lv\_1 = (int)a.FirstLevelSubdivisionTableID,

Lv\_2 = (int)b.SecondLevelSubdivisionTableID,

Lv\_3 = 0,

Lv\_4 = 0,

Lv\_5 = 0,

Name = b.Name

}).ToList();

break;

}

case 2: // возвращаем все кафедры

{

tmpStrucList = (from a in kpiWebDataContext.FirstLevelSubdivisionTable

join b in kpiWebDataContext.SecondLevelSubdivisionTable

on a.FirstLevelSubdivisionTableID equals b.FK\_FirstLevelSubdivisionTable

join c in kpiWebDataContext.ThirdLevelSubdivisionTable

on b.SecondLevelSubdivisionTableID equals c.FK\_SecondLevelSubdivisionTable

where a.Active == true

&& b.Active == true

&& c.Active == true

&& a.FK\_ZeroLevelSubvisionTable == ParentStruct.Lv\_0

&& b.FK\_FirstLevelSubdivisionTable == ParentStruct.Lv\_1

&& c.FK\_SecondLevelSubdivisionTable == ParentStruct.Lv\_2

select new Struct(1,"")

{

Lv\_0 = (int)a.FK\_ZeroLevelSubvisionTable,

Lv\_1 = (int)a.FirstLevelSubdivisionTableID,

Lv\_2 = (int)b.SecondLevelSubdivisionTableID,

Lv\_3 = (int)c.ThirdLevelSubdivisionTableID,

Lv\_4 = 0,

Lv\_5 = 0,

Name = c.Name

}).ToList();

break;

}

default:

{

//error не будем раскладывать до специальностей

break;

}

}

return tmpStrucList;

}

public float GetCalculatedWithParams(Struct StructToCalcFor, int ParamType, int ParamID,int ReportID) // читает показатель

{

KPIWebDataContext kpiWebDataContext = new KPIWebDataContext();

float result = 0;

if (ParamType == 0) // считаем индикатор

{

IndicatorsTable Indicator = (from a in kpiWebDataContext.IndicatorsTable

where a.IndicatorsTableID == ParamID

select a).FirstOrDefault();

return (float)CalculateAbb.CalculateForLevel(1, Abbreviature.CalculatedAbbToFormula(Indicator.Formula)

, ReportID, StructToCalcFor.Lv\_0

, StructToCalcFor.Lv\_1, StructToCalcFor.Lv\_2, StructToCalcFor.Lv\_3, StructToCalcFor.Lv\_4, StructToCalcFor.Lv\_5, 0);

}

else if (ParamType == 1) // считаем рассчетный

{

CalculatedParametrs Calculated = (from a in kpiWebDataContext.CalculatedParametrs

where a.CalculatedParametrsID == ParamID

select a).FirstOrDefault();

return (float)CalculateAbb.CalculateForLevel(1, Calculated.Formula, ReportID, StructToCalcFor.Lv\_0

, StructToCalcFor.Lv\_1, StructToCalcFor.Lv\_2, StructToCalcFor.Lv\_3, StructToCalcFor.Lv\_4, StructToCalcFor.Lv\_5, 0);

}

else if (ParamType == 2) // суммируем базовый

{

return (float)CalculateAbb.SumForLevel(ParamID,ReportID,StructToCalcFor.Lv\_0

, StructToCalcFor.Lv\_1, StructToCalcFor.Lv\_2, StructToCalcFor.Lv\_3, StructToCalcFor.Lv\_4, StructToCalcFor.Lv\_5);

}

else

{

//error

}

return result;

}

public int StructDeepness(Struct CurrentStruct)

{

int tmp=0;

if (CurrentStruct.Lv\_0 != 0) tmp++;

if (CurrentStruct.Lv\_1 != 0) tmp++;

if (CurrentStruct.Lv\_2 != 0) tmp++;

if (CurrentStruct.Lv\_3 != 0) tmp++;

if (CurrentStruct.Lv\_4 != 0) tmp++;

if (CurrentStruct.Lv\_5 != 0) tmp++;

return tmp;

}

public Struct StructDeeper(Struct parentStruct, int nextID)

{

Struct tmp = parentStruct;

if (tmp.Lv\_0 == 0)

{

tmp.Lv\_0 = nextID;

return tmp;

}

if (tmp.Lv\_1 == 0)

{

tmp.Lv\_1 = nextID;

return tmp;

}

if (tmp.Lv\_2 == 0)

{

tmp.Lv\_2 = nextID;

return tmp;

}

if (tmp.Lv\_3 == 0)

{

tmp.Lv\_3 = nextID;

return tmp;

}

if (tmp.Lv\_4 == 0)

{

tmp.Lv\_4 = nextID;

return tmp;

}

if (tmp.Lv\_5 == 0)

{

tmp.Lv\_5 = nextID;

return tmp;

}

return tmp;

} //добавляет ID к первому в структуре нулю

public int GetLastID(Struct currentStruct)

{

if (currentStruct.Lv\_0 == 0)

{

return 0;

}

if (currentStruct.Lv\_1 == 0)

{

return currentStruct.Lv\_0;

}

if (currentStruct.Lv\_2 == 0)

{

return currentStruct.Lv\_1;

}

if (currentStruct.Lv\_3 == 0)

{

return currentStruct.Lv\_2;

}

if (currentStruct.Lv\_4 == 0)

{

return currentStruct.Lv\_3;

}

return 0;

} //определяет последнее не нулевое значение в структуре

protected void Page\_Load(object sender, EventArgs e)

{

if (!IsPostBack)

{

KPIWebDataContext kpiWebDataContext = new KPIWebDataContext();

//Принимаемые через сессию параметры

RectorSession rectorResultSession = (RectorSession) Session["rectorResultSession"];

if (rectorResultSession == null)

{

Response.Redirect("~/Default.aspx");

}

Struct mainStruct = rectorResultSession.sesStruct;

int ViewType = rectorResultSession.sesViewType;

int ParamID = rectorResultSession.sesParamID;

int ParamType = rectorResultSession.sesParamType;

int ReportID = rectorResultSession.sesReportID;

///////////////////////

DataTable dataTable = new DataTable();

dataTable.Columns.Add(new DataColumn("ID", typeof (string)));

dataTable.Columns.Add(new DataColumn("Number", typeof (string)));

dataTable.Columns.Add(new DataColumn("Name", typeof (string)));

dataTable.Columns.Add(new DataColumn("StartDate", typeof (string)));

dataTable.Columns.Add(new DataColumn("EndDate", typeof (string)));

dataTable.Columns.Add(new DataColumn("Value", typeof (string)));

dataTable.Columns.Add(new DataColumn("Title", typeof(string)));

ReportTitle.Text = (from a in kpiWebDataContext.ReportArchiveTable

where a.ReportArchiveTableID == ReportID

select a.Name).FirstOrDefault().ToString();

if (ViewType == 0) // просмотр для структурных подразделений

{

#region преднастройка страницы

string title="";

if (ParamType==0)

{

PageName.Text = "Значения для индикатора; \"";

PageName.Text += (from a in kpiWebDataContext.IndicatorsTable

where a.IndicatorsTableID == ParamID

select a.Name).FirstOrDefault();

PageName.Text += "\";";

}

else if (ParamType == 1)

{

PageName.Text = "Значения для расчетного показателя: \"";

PageName.Text += (from a in kpiWebDataContext.CalculatedParametrs

where a.CalculatedParametrsID == ParamID

select a.Name).FirstOrDefault();

PageName.Text += "\";";

}

else if (ParamType == 2)

{

PageName.Text = "Значения для базового показателя: \"";

PageName.Text += (from a in kpiWebDataContext.BasicParametersTable

where a.BasicParametersTableID == ParamID

select a.Name).FirstOrDefault();

PageName.Text += "\";";

}

#region useless switch

int tmpLevel = StructDeepness(mainStruct);

switch (tmpLevel)

{

case 0:

{

title = "Подразделения";

break;

}

case 1:

{

title = "Подразделения";

break;

}

case 2:

{

title = "Подразделения";

break;

}

case 3:

{

title = "Подразделения";

break;

}

case 4:

{

title = "Подразделения";

break;

}

case 5:

{

title = "Подразделения";

break;

}

default:

{

title = "Подразделения";

break;

}

}

#endregion

#endregion

#region fill grid

List<Struct> currentStructList = GetChildStructList(mainStruct);

foreach (Struct currentStruct in currentStructList)

{

DataRow dataRow = dataTable.NewRow();

dataRow["ID"] = GetLastID(currentStruct).ToString();

dataRow["Number"] = "num";

dataRow["Name"] = currentStruct.Name;

dataRow["StartDate"] = "nun";

dataRow["EndDate"] = "nun";

dataRow["Value"] = GetCalculatedWithParams(currentStruct, ParamType, ParamID, ReportID).ToString();

dataTable.Rows.Add(dataRow);

}

#endregion

Grid.DataSource = dataTable;

Grid.Columns[2].HeaderText = title;

Grid.DataBind();

#region постнастройка страницы

Grid.Columns[8].Visible = false;

Grid.Columns[7].Visible = false;

Grid.Columns[4].Visible = false;

Grid.Columns[3].Visible = false;

Grid.Columns[1].Visible = false;

Grid.Columns[0].HeaderText = title;

if (StructDeepness(currentStructList[0]) > 3) // дальше углубляться нельзя

{

Grid.Columns[6].Visible = false;

}

#endregion

}

else if (ViewType == 1) // просмотр для показателей (верхние 3 шт)

{

#region преднастройка страницы

string title="";

if (ParamType==0)

{

PageName.Text = "Значения индикторов для КФУ";

title = "Индикаторы";

}

else if (ParamType == 1)

{

PageName.Text = "Значения расчетных показателей используемых для расчета индикатора: \"";

PageName.Text += (from a in kpiWebDataContext.IndicatorsTable

where a.IndicatorsTableID == ParamID

select a.Name).FirstOrDefault();

PageName.Text += "\" для КФУ";

title = "Расчетные показатели";

}

else if (ParamType == 2)

{

PageName.Text = "Значения базовых показателей используемых для расчета расчетного показателя\"";

PageName.Text += (from a in kpiWebDataContext.CalculatedParametrs

where a.CalculatedParametrsID == ParamID

select a.Name).FirstOrDefault();

PageName.Text += "\" для КФУ";

title = "Базовые показатели";

}

#endregion

#region main

if (ParamType == 0)//считаем индикатор

{

List<IndicatorsTable> Indicators = (from a in kpiWebDataContext.IndicatorsTable

where a.Active == true

select a).ToList();

foreach (IndicatorsTable CurrentIndicator in Indicators)

{

DataRow dataRow = dataTable.NewRow();

dataRow["ID"] = CurrentIndicator.IndicatorsTableID;//GetLastID(currentStruct).ToString();

dataRow["Number"] = "num";

dataRow["Name"] = CurrentIndicator.Name;

dataRow["StartDate"] = "nun";

dataRow["EndDate"] = "nun";

dataRow["Value"] = GetCalculatedWithParams(mainStruct, ParamType, CurrentIndicator.IndicatorsTableID, ReportID).ToString();

dataTable.Rows.Add(dataRow);

}

}

if (ParamType == 1) //показываем рассчетный входящий в ID Индикатора

{

//ID - это айди Индиктора

IndicatorsTable Indicator = (from a in kpiWebDataContext.IndicatorsTable

where a.IndicatorsTableID == ParamID

select a).FirstOrDefault();

List<CalculatedParametrs> CalculatedList = Abbreviature.GetCalculatedList(Indicator.Formula);

foreach (CalculatedParametrs CurrentCalculated in CalculatedList)

{

DataRow dataRow = dataTable.NewRow();

dataRow["ID"] = CurrentCalculated.CalculatedParametrsID;//GetLastID(currentStruct).ToString();

dataRow["Number"] = "num";

dataRow["Name"] = CurrentCalculated.Name;

dataRow["StartDate"] = "nun";

dataRow["EndDate"] = "nun";

dataRow["Value"] = GetCalculatedWithParams(mainStruct, ParamType, CurrentCalculated.CalculatedParametrsID, ReportID).ToString();

dataTable.Rows.Add(dataRow);

}

}

if (ParamType == 2)//

{

//ID - Рассчетного айдишник

CalculatedParametrs Calculated = (from a in kpiWebDataContext.CalculatedParametrs

where a.CalculatedParametrsID == ParamID

select a).FirstOrDefault();

List <BasicParametersTable> BasicList = Abbreviature.GetBasicList(Calculated.Formula);

foreach (BasicParametersTable CurrebtBasic in BasicList)

{

DataRow dataRow = dataTable.NewRow();

dataRow["ID"] = CurrebtBasic.BasicParametersTableID;//GetLastID(currentStruct).ToString();

dataRow["Number"] = "num";

dataRow["Name"] = CurrebtBasic.Name;

dataRow["StartDate"] = "nun";

dataRow["EndDate"] = "nun";

dataRow["Value"] = GetCalculatedWithParams(mainStruct, ParamType, CurrebtBasic.BasicParametersTableID, ReportID).ToString();

dataTable.Rows.Add(dataRow);

}

}

#endregion

Grid.DataSource = dataTable;

Grid.Columns[2].HeaderText = title;

Grid.DataBind();

#region постнастройки страницы

Grid.Columns[8].Visible = false;

Grid.Columns[4].Visible = false;

Grid.Columns[3].Visible = false;

Grid.Columns[1].Visible = false;

if (ParamType == 2) // дальше углубляться нельзя

{

Grid.Columns[7].Visible = false;

}

#endregion

}

else

{

//error // wrong ViewType

}

}

}

protected void Button1Click(object sender, EventArgs e) //по структуре

{

Button button = (Button)sender;

{

RectorSession rectorResultSession = (RectorSession) Session["rectorResultSession"];

if (rectorResultSession == null)

{

Response.Redirect("~/Default.aspx");

}

//rectorResultSession.sesParamID = Convert.ToInt32(button.CommandArgument);

//rectorResultSession.sesViewType = 0;

if (rectorResultSession.sesViewType == 1)

{//впервые перешли на разложение по структуре

rectorResultSession.sesParamID = Convert.ToInt32(button.CommandArgument);

rectorResultSession.sesViewType = 0;

rectorResultSession.sesStruct.Lv\_0 = 1;

rectorResultSession.sesStruct.Lv\_1 = 0;

rectorResultSession.sesStruct.Lv\_2 = 0;

rectorResultSession.sesStruct.Lv\_3 = 0;

rectorResultSession.sesStruct.Lv\_4 = 0;

rectorResultSession.sesStruct.Lv\_5 = 0;

}

else

{

rectorResultSession.sesStruct = StructDeeper(rectorResultSession.sesStruct, Convert.ToInt32(button.CommandArgument));

}

Session["rectorResultSession"] = rectorResultSession;

Response.Redirect("~/Rector/Result.aspx");

}

}

protected void Button2Click(object sender, EventArgs e)

{

Button button = (Button)sender;

{

RectorSession rectorResultSession = (RectorSession)Session["rectorResultSession"];

if (rectorResultSession == null)

{

Response.Redirect("~/Default.aspx");

}

//rectorResultSession.sesStruct = StructDeeper(rectorResultSession.sesStruct, Convert.ToInt32(button.CommandArgument));

rectorResultSession.sesParamID = Convert.ToInt32(button.CommandArgument);

rectorResultSession.sesParamType++;

Session["rectorResultSession"] = rectorResultSession;

Response.Redirect("~/Rector/Result.aspx");

}

}

protected void Button3Click(object sender, EventArgs e)

{

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Filename: Result.aspx.designer.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//------------------------------------------------------------------------------

// <автоматически создаваемое>

// Этот код создан программой.

//

// Изменения в этом файле могут привести к неправильной работе и будут потеряны в случае

// повторной генерации кода.

// </автоматически создаваемое>

//------------------------------------------------------------------------------

namespace KPIWeb.Rector {

public partial class Result {

/// <summary>

/// ReportTitle элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Label ReportTitle;

/// <summary>

/// PageName элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Label PageName;

/// <summary>

/// Grid элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.GridView Grid;

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Filename: Serialization.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

namespace KPIWeb

{

[Serializable]

public class Serialization

{

public int Id { get; set; }

public int ReportArchiveID { get; set; }

public string ReportStr { get; set; }

public int mode { get; set; }

public int l0 { get; set; }

public int l1 { get; set; }

public int l2 { get; set; }

public int l3 { get; set; }

public int l4 { get; set; }

public int l5 { get; set; }

public Serialization(int IdTmp)

{

this.Id = IdTmp;

}

public Serialization(string StrTmp)

{

this.ReportStr = StrTmp;

}

public Serialization(int ArchiveIdTmp,object obj)

{

this.ReportArchiveID = ArchiveIdTmp;

}

public Serialization(int mode, object obj, object obj2)

{

this.mode = mode;

}

public Serialization(int l0\_,int l1\_, int l2\_, int l3\_, int l4\_, int l5\_)

{

this.l0 = l0\_;

this.l1 = l1\_;

this.l2 = l2\_;

this.l3 = l3\_;

this.l4 = l4\_;

this.l5 = l5\_;

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Filename: Site.Master.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using System;

using System.Collections.Generic;

using System.Security.Claims;

using System.Security.Principal;

using System.Web;

using System.Web.Security;

using System.Web.UI;

using System.Web.UI.WebControls;

namespace KPIWeb

{

public partial class SiteMaster : MasterPage

{

private const string AntiXsrfTokenKey = "\_\_AntiXsrfToken";

private const string AntiXsrfUserNameKey = "\_\_AntiXsrfUserName";

private string \_antiXsrfTokenValue;

protected void Page\_Init(object sender, EventArgs e)

{

/\*

// The code below helps to protect against XSRF attacks

var requestCookie = Request.Cookies[AntiXsrfTokenKey];

Guid requestCookieGuidValue;

if (requestCookie != null && Guid.TryParse(requestCookie.Value, out requestCookieGuidValue))

{

// Use the Anti-XSRF token from the cookie

\_antiXsrfTokenValue = requestCookie.Value;

Page.ViewStateUserKey = \_antiXsrfTokenValue;

}

else

{

// Generate a new Anti-XSRF token and save to the cookie

\_antiXsrfTokenValue = Guid.NewGuid().ToString("N");

Page.ViewStateUserKey = \_antiXsrfTokenValue;

var responseCookie = new HttpCookie(AntiXsrfTokenKey)

{

HttpOnly = true,

Value = \_antiXsrfTokenValue

};

if (FormsAuthentication.RequireSSL && Request.IsSecureConnection)

{

responseCookie.Secure = true;

}

Response.Cookies.Set(responseCookie);

}

Page.PreLoad += master\_Page\_PreLoad;

\*/

}

protected void master\_Page\_PreLoad(object sender, EventArgs e)

{

if (!IsPostBack)

{

// Set Anti-XSRF token

ViewState[AntiXsrfTokenKey] = Page.ViewStateUserKey;

ViewState[AntiXsrfUserNameKey] = Context.User.Identity.Name ?? String.Empty;

}

else

{

// Validate the Anti-XSRF token

if ((string)ViewState[AntiXsrfTokenKey] != \_antiXsrfTokenValue

|| (string)ViewState[AntiXsrfUserNameKey] != (Context.User.Identity.Name ?? String.Empty))

{

throw new InvalidOperationException("Validation of Anti-XSRF token failed.");

}

}

}

protected void Page\_Load(object sender, EventArgs e)

{

}

protected void Unnamed\_LoggingOut(object sender, LoginCancelEventArgs e)

{

FormsAuthentication.SignOut();

Session.Abandon();

Response.Redirect("~/Account/UserLogin.aspx");

//Context.GetOwinContext().Authentication.SignOut();

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Filename: Site.Master.designer.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//------------------------------------------------------------------------------

// <auto-generated>

// This code was generated by a tool.

//

// Changes to this file may cause incorrect behavior and will be lost if

// the code is regenerated.

// </auto-generated>

//------------------------------------------------------------------------------

namespace KPIWeb {

public partial class SiteMaster {

/// <summary>

/// MainContent control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.ContentPlaceHolder MainContent;

/// <summary>

/// loading control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.Panel loading;

/// <summary>

/// gears control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.Panel gears;

/// <summary>

/// im1 control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.Panel im1;

/// <summary>

/// im2 control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.Panel im2;

/// <summary>

/// loading\_2 control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.Panel loading\_2;

/// <summary>

/// bgpanel\_of\_circlebar control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.Panel bgpanel\_of\_circlebar;

/// <summary>

/// im3\_circle control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.Panel im3\_circle;

/// <summary>

/// im3\_bg\_circle control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.Panel im3\_bg\_circle;

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Filename: Site.Mobile.Master.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

namespace KPIWeb

{

public partial class Site\_Mobile : System.Web.UI.MasterPage

{

protected void Page\_Load(object sender, EventArgs e)

{

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Filename: Site.Mobile.Master.designer.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//------------------------------------------------------------------------------

// <auto-generated>

// This code was generated by a tool.

//

// Changes to this file may cause incorrect behavior and will be lost if

// the code is regenerated.

// </auto-generated>

//------------------------------------------------------------------------------

namespace KPIWeb {

public partial class Site\_Mobile {

/// <summary>

/// HeadContent control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.ContentPlaceHolder HeadContent;

/// <summary>

/// form1 control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.HtmlControls.HtmlForm form1;

/// <summary>

/// FeaturedContent control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.ContentPlaceHolder FeaturedContent;

/// <summary>

/// MainContent control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.ContentPlaceHolder MainContent;

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Filename: Startup.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using Microsoft.Owin;

using Owin;

[assembly: OwinStartupAttribute(typeof(KPIWeb.Startup))]

namespace KPIWeb

{

public partial class Startup {

public void Configuration(IAppBuilder app) {

ConfigureAuth(app);

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Filename: ViewSwitcher.ascx.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.Routing;

using System.Web.UI;

using System.Web.UI.WebControls;

using Microsoft.AspNet.FriendlyUrls.Resolvers;

namespace KPIWeb

{

public partial class ViewSwitcher : System.Web.UI.UserControl

{

protected string CurrentView { get; private set; }

protected string AlternateView { get; private set; }

protected string SwitchUrl { get; private set; }

protected void Page\_Load(object sender, EventArgs e)

{

// Determine current view

var isMobile = WebFormsFriendlyUrlResolver.IsMobileView(new HttpContextWrapper(Context));

CurrentView = isMobile ? "Mobile" : "Desktop";

// Determine alternate view

AlternateView = isMobile ? "Desktop" : "Mobile";

// Create switch URL from the route, e.g. ~/\_\_FriendlyUrls\_SwitchView/Mobile?ReturnUrl=/Page

var switchViewRouteName = "AspNet.FriendlyUrls.SwitchView";

var switchViewRoute = RouteTable.Routes[switchViewRouteName];

if (switchViewRoute == null)

{

// Friendly URLs is not enabled or the name of the switch view route is out of sync

this.Visible = false;

return;

}

var url = GetRouteUrl(switchViewRouteName, new { view = AlternateView, \_\_FriendlyUrls\_SwitchViews = true });

url += "?ReturnUrl=" + HttpUtility.UrlEncode(Request.RawUrl);

SwitchUrl = url;

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Filename: ViewSwitcher.ascx.designer.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//------------------------------------------------------------------------------

// <auto-generated>

// This code was generated by a tool.

//

// Changes to this file may cause incorrect behavior and will be lost if

// the code is regenerated.

// </auto-generated>

//------------------------------------------------------------------------------

namespace KPIWeb {

public partial class ViewSwitcher {

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Filename: DataBaseCommunicator.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using System;

using System.Collections.Generic;

using System.Configuration;

using System.Data;

using System.Data.SqlClient;

using System.Linq;

using System.Reflection;

using System.Web;

namespace WebApplication3

{

public class DataBaseCommunicator

{

//public static List<Users> GetUsersTable()

//{

// List<Users> returnValue = new List<Users>();

// using (SqlConnection dbConnection = new SqlConnection())

// using (SqlCommand dbCommand = dbConnection.CreateCommand())

// {

// string SqlCommandText = "SELECT ALL [id\_users],[active],[login],[password],[fk\_first\_stage],[fk\_second\_stage],[fk\_third\_stage],[fk\_fourth\_stage],[fk\_fifth\_stage] FROM [crimeanfeder\_3].[dbo].[Users]";

// dbCommand.CommandType = CommandType.Text;

// dbCommand.CommandText = SqlCommandText;

// returnValue = ExecuteSQL<Users>(dbConnection, dbCommand);

// // if (returnValue == null) throw new Exception("Can't get data from Users table.");

// }

// return returnValue;

//}

protected static List<T> ExecuteSQL<T>(SqlConnection connection, SqlCommand cmd)

{

var result = new List<T>();

cmd.Connection = connection;

Type type = typeof(T);

ConstructorInfo constructor = type.GetConstructor(Type.EmptyTypes);

try

{

if (connection.State != ConnectionState.Open)

{

connection.Open();

}

if (constructor != null)

{

using (SqlDataReader reader = cmd.ExecuteReader())

{

while (reader.Read())

{

object obj = constructor.Invoke(null);

int cnt = reader.FieldCount;

for (int i = 0; i < cnt; i++)

{

string name = reader.GetName(i);

object val = reader[i];

if (val is DBNull)

{

val = null;

}

var property = type.GetProperty(name,

BindingFlags.IgnoreCase | BindingFlags.Instance |

BindingFlags.Public);

if (property != null)

{

property.SetValue(obj, val, null);

}

}

result.Add((T)obj);

}

reader.Close();

}

}

}

catch (SqlException sqlEx)

{

//LogHandler.LogWriter.WriteError(sqlEx);

}

finally

{

connection.Close();

}

return result;

}

//public static Users Get\_User(List<Users> list\_users, string login, string password)

//{

// Users returnvalue = null;

// foreach (Users users in list\_users)

// {

// if (users.login.Equals(login) && users.password.Equals(password))

// returnvalue = users;

// }

// return returnvalue;

//}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Filename: OpenAuthProviders.ascx.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using Microsoft.Owin.Security;

using Microsoft.AspNet.Identity;

using System;

using System.Collections.Generic;

using System.Globalization;

using System.Linq;

using System.Web;

namespace KPIWeb.Account

{

public partial class OpenAuthProviders : System.Web.UI.UserControl

{

protected void Page\_Load(object sender, EventArgs e)

{

if (IsPostBack)

{

var provider = Request.Form["provider"];

if (provider == null)

{

return;

}

// Request a redirect to the external login provider

string redirectUrl = ResolveUrl(String.Format(CultureInfo.InvariantCulture, "~/Account/RegisterExternalLogin?{0}={1}&returnUrl={2}", IdentityHelper.ProviderNameKey, provider, ReturnUrl));

var properties = new AuthenticationProperties() { RedirectUri = redirectUrl };

// Add xsrf verification when linking accounts

if (Context.User.Identity.IsAuthenticated)

{

properties.Dictionary[IdentityHelper.XsrfKey] = Context.User.Identity.GetUserId();

}

Context.GetOwinContext().Authentication.Challenge(properties, provider);

Response.StatusCode = 401;

Response.End();

}

}

public string ReturnUrl { get; set; }

public IEnumerable<string> GetProviderNames()

{

return Context.GetOwinContext().Authentication.GetExternalAuthenticationTypes().Select(t => t.AuthenticationType);

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Filename: OpenAuthProviders.ascx.designer.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//------------------------------------------------------------------------------

// <auto-generated>

// This code was generated by a tool.

//

// Changes to this file may cause incorrect behavior and will be lost if

// the code is regenerated.

// </auto-generated>

//------------------------------------------------------------------------------

namespace KPIWeb.Account {

public partial class OpenAuthProviders {

/// <summary>

/// providerDetails control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.ListView providerDetails;

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Filename: PersonalInfo.aspx.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

namespace KPIWeb

{

public partial class PersonalInfo : System.Web.UI.Page

{

protected void Page\_Load(object sender, EventArgs e)

{

//TextBox1.Text = "";

Serialization UserSer = (Serialization) Session["UserID"];

if (UserSer == null)

{

Response.Redirect("~/Default.aspx");

}

int userID = UserSer.Id;

ViewState["ID"] = userID;

KPIWebDataContext kPiDataContext = new KPIWebDataContext();

UsersTable userTable =

(from a in kPiDataContext.UsersTable where a.UsersTableID == userID select a).FirstOrDefault();

////////////////////////////////////////////////////////////////////////////

if (!IsPostBack)

{

Label1.Text += userTable.Login;

Label2.Text += userTable.Email;

Label3.Text = (from zero in kPiDataContext.ZeroLevelSubdivisionTable

where zero.ZeroLevelSubdivisionTableID == userTable.FK\_ZeroLevelSubdivisionTable

select zero.Name).FirstOrDefault();

Label4.Text = (from b in kPiDataContext.FirstLevelSubdivisionTable

where b.FirstLevelSubdivisionTableID == userTable.FK\_FirstLevelSubdivisionTable

select b.Name).FirstOrDefault();

Label5.Text = (from c in kPiDataContext.SecondLevelSubdivisionTable

where c.SecondLevelSubdivisionTableID == userTable.FK\_SecondLevelSubdivisionTable

select c.Name).FirstOrDefault();

Label6.Text = (from d in kPiDataContext.ThirdLevelSubdivisionTable

where d.ThirdLevelSubdivisionTableID == userTable.FK\_ThirdLevelSubdivisionTable

select d.Name).FirstOrDefault();

Label7.Text = (from f in kPiDataContext.FourthLevelSubdivisionTable

where f.FourthLevelSubdivisionTableID == userTable.FK\_FourthLevelSubdivisionTable

select f.Name).FirstOrDefault();

Label8.Text = (from g in kPiDataContext.FifthLevelSubdivisionTable

where g.FifthLevelSubdivisionTableID == userTable.FK\_FifthLevelSubdivisionTable

select g.Name).FirstOrDefault();

/\*TextBox1.Text += "Ваш email " + userTable.Email + Environment.NewLine;

TextBox1.Text += (from zero in kPiDataContext.ZeroLevelSubdivisionTable where zero.ZeroLevelSubdivisionTableID == userTable.FK\_ZeroLevelSubdivisionTable select zero.Name).FirstOrDefault() + Environment.NewLine;

TextBox1.Text += (from b in kPiDataContext.FirstLevelSubdivisionTable where b.FirstLevelSubdivisionTableID == userTable.FK\_FirstLevelSubdivisionTable select b.Name).FirstOrDefault() + Environment.NewLine;

TextBox1.Text += (from c in kPiDataContext.SecondLevelSubdivisionTable where c.SecondLevelSubdivisionTableID == userTable.FK\_SecondLevelSubdivisionTable select c.Name).FirstOrDefault() + Environment.NewLine;

TextBox1.Text += (from d in kPiDataContext.ThirdLevelSubdivisionTable where d.ThirdLevelSubdivisionTableID == userTable.FK\_ThirdLevelSubdivisionTable select d.Name).FirstOrDefault() + Environment.NewLine;

TextBox1.Text += (from f in kPiDataContext.FourthLevelSubdivisionTable where f.FourthLevelSubdivisionTableID == userTable.FK\_FourthLevelSubdivisionTable select f.Name).FirstOrDefault() + Environment.NewLine;

TextBox1.Text += (from g in kPiDataContext.FifthLevelSubdivisionTable where g.FifthLevelSubdivisionTableID == userTable.FK\_FifthLevelSubdivisionTable select g.Name).FirstOrDefault() + Environment.NewLine; \*/

}

}

protected void Button1\_Click(object sender, EventArgs e)

{

TextBox1.Text = "";

Label9.Visible = true;

Label10.Visible = true;

Label11.Visible = true;

TextBox1.Visible = true;

TextBox2.Visible = true;

TextBox3.Visible = true;

Button2.Visible = true;

SetFocus(Button2);

}

protected void Button2\_Click(object sender, EventArgs e)

{

KPIWebDataContext kpiWebDataContext = new KPIWebDataContext();

UsersTable user = (from a in kpiWebDataContext.UsersTable where a.UsersTableID == (int)ViewState["ID"] select a).FirstOrDefault();

if ( (user!= null) && (TextBox1.Text.Equals(user.Password)) && (TextBox2.Text.Any()) && (TextBox2.Text.Equals(TextBox3.Text)) )

{

user.Password = TextBox2.Text;

kpiWebDataContext.SubmitChanges();

Page.ClientScript.RegisterClientScriptBlock(typeof(Page), "Script",

"alert('Пароль успешно изменен!');", true);

Response.Redirect("~/Default.aspx");

}

else

{

Page.ClientScript.RegisterClientScriptBlock(typeof(Page), "Script",

"alert('Произошла ошибка, проверьте правильность данных!');", true);

}

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Filename: PersonalInfo.aspx.designer.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//------------------------------------------------------------------------------

// <автоматически создаваемое>

// Этот код создан программой.

//

// Изменения в этом файле могут привести к неправильной работе и будут потеряны в случае

// повторной генерации кода.

// </автоматически создаваемое>

//------------------------------------------------------------------------------

namespace KPIWeb {

public partial class PersonalInfo {

/// <summary>

/// Label1 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Label Label1;

/// <summary>

/// Label2 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Label Label2;

/// <summary>

/// Label3 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Label Label3;

/// <summary>

/// Label4 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Label Label4;

/// <summary>

/// Label5 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Label Label5;

/// <summary>

/// Label6 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Label Label6;

/// <summary>

/// Label7 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Label Label7;

/// <summary>

/// Label8 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Label Label8;

/// <summary>

/// Button1 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Button Button1;

/// <summary>

/// Label9 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Label Label9;

/// <summary>

/// TextBox1 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.TextBox TextBox1;

/// <summary>

/// Label10 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Label Label10;

/// <summary>

/// TextBox2 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.TextBox TextBox2;

/// <summary>

/// Label11 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Label Label11;

/// <summary>

/// TextBox3 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.TextBox TextBox3;

/// <summary>

/// Button2 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Button Button2;

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Filename: RestorePassword.aspx.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using System;

using System.Collections.Generic;

using System.Configuration;

using System.Linq;

using System.Text;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

namespace KPIWeb.Account

{

public partial class RestorePassword : System.Web.UI.Page

{

protected void Page\_Load(object sender, EventArgs e)

{

}

protected void Button1\_Click(object sender, EventArgs e)

{

KPIWebDataContext kPiDataContext = new KPIWebDataContext();

UsersTable user = (from a in kPiDataContext.UsersTable where a.Email == TextBox1.Text && a.Active select a).FirstOrDefault();

if (user != null)

{

if (user.PassCode != null && user.PassCode.Any())

{

DisplayAlert("На " + user.Email +

" уже отправлена инструкция по восстановлению пароля." + Environment.NewLine + "Если в течении 20 минут письмо не пришло, обратитесь к администрации." +Environment.NewLine+ "Чтобы узнать как связаться с нами перейдите в раздел \"Контакты\" в верхней части экрана.");

}

else

{

string passCode = RandomString(25);

user.PassCode = passCode;

user.Confirmed = false;

kPiDataContext.SubmitChanges();

EmailTemplate EmailParams = (from a in kPiDataContext.EmailTemplate

where a.Name == "PasswordRecover"

&& a.Active == true

select a).FirstOrDefault();

if (EmailParams != null)

Action.MassMailing(user.Email, EmailParams.EmailTitle,

EmailParams.EmailContent.Replace("#SiteName#", ConfigurationManager.AppSettings.Get("SiteName")).Replace("#LINK#", ConfigurationManager.AppSettings.Get("SiteName") + "/Account/UserRegister?&id=" + passCode)

, null);

/\*

Action.MassMailing(user.Email, "Восстановление пароля в системе ИАС 'КФУ-Программа развития'",

"Здравствуйте!" + Environment.NewLine +

"На ваш почтовый адрес был сформирован запрос на восстановление пароля в системе ИАС 'КФУ-Программа развития!'" +

Environment.NewLine +

"Для того, чтобы указать новый пароль перейдите по ссылке:" + Environment.NewLine +

ConfigurationManager.AppSettings.Get("SiteName") + "/Account/UserRegister?&id=" + passCode +

Environment.NewLine +

"Спасибо!"

, null);

\*/

DisplayAlert("На email " + user.Email +

" было выслано письмо с дальнейшими указаниями по восстановлению пароля");

}

}

else

{

DisplayAlert("Пользователь с таким email адресом в системе не зарегистрирован!");

}

}

private string RandomString(int size)

{

StringBuilder builder = new StringBuilder();

Random random = new Random();

char ch;

for (int i = 0; i < size; i++)

{

ch = Convert.ToChar(Convert.ToInt32(Math.Floor(26 \* random.NextDouble() + 65)));

builder.Append(ch);

}

return builder.ToString();

}

private void DisplayAlert(string message)

{

ClientScript.RegisterStartupScript(

this.GetType(),

Guid.NewGuid().ToString(),

string.Format("alert('{0}');window.location.href = 'UserLogin.aspx'",

message.Replace("'", @"\'").Replace("\n", "\\n").Replace("\r", "\\r")),

true);

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Filename: RestorePassword.aspx.designer.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//------------------------------------------------------------------------------

// <auto-generated>

// This code was generated by a tool.

//

// Changes to this file may cause incorrect behavior and will be lost if

// the code is regenerated.

// </auto-generated>

//------------------------------------------------------------------------------

namespace KPIWeb.Account {

public partial class RestorePassword {

/// <summary>

/// Label3 control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.Label Label3;

/// <summary>

/// Label4 control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.Label Label4;

/// <summary>

/// TextBox1 control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.TextBox TextBox1;

/// <summary>

/// Button1 control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.Button Button1;

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Filename: TEST.aspx.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

namespace KPIWeb.Account

{

public partial class TEST : System.Web.UI.Page

{

protected void Page\_Load(object sender, EventArgs e)

{

}

protected void abbreviatureStringToPolishString(string abbreviaturetring)

{

}

protected void Button1\_Click(object sender, EventArgs e)

{

Label1.Text = Polish.Calculate(TextBox1.Text).ToString();

}

protected void Button2\_Click(object sender, EventArgs e)

{

Label1.Text = CalculateAbb.Calculate(TextBox1.Text).ToString();

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Filename: TEST.aspx.designer.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//------------------------------------------------------------------------------

// <автоматически создаваемое>

// Этот код создан программой.

//

// Изменения в этом файле могут привести к неправильной работе и будут потеряны в случае

// повторной генерации кода.

// </автоматически создаваемое>

//------------------------------------------------------------------------------

namespace KPIWeb.Account {

public partial class TEST {

/// <summary>

/// form1 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.HtmlControls.HtmlForm form1;

/// <summary>

/// TextBox1 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.TextBox TextBox1;

/// <summary>

/// Button1 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Button Button1;

/// <summary>

/// Button2 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Button Button2;

/// <summary>

/// Label1 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Label Label1;

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Filename: UserLogin.aspx.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

using System.IO;

using System.Net;

using System.Text;

namespace KPIWeb.Account

{

public partial class UserLogin : System.Web.UI.Page

{

protected void Page\_Load(object sender, EventArgs e)

{

Serialization UserSer = (Serialization)Session["UserID"];

if (UserSer != null)

{

Response.Redirect("~/Default.aspx");

}

}

protected void Button1\_Click(object sender, EventArgs e)

{

try

{

if (IsValid)

{

KPIWebDataContext KPIWebDataContext = new KPIWebDataContext();

UsersTable user = (from usersTables in KPIWebDataContext.UsersTable

where

( (usersTables.Login == UserName.Text) || (usersTables.Email == UserName.Text) )

&&

usersTables.Password == Password.Text &&

usersTables.Active == true

select usersTables).FirstOrDefault();

if (user != null )

{

LogHandler.LogWriter.WriteLog(LogCategory.INFO,"0LN0: User " + user.Email + " login in from ip: " + Dns.GetHostEntry(Dns.GetHostName()).AddressList.Where(ip => ip.AddressFamily == System.Net.Sockets.AddressFamily.InterNetwork).Select(ip => ip.ToString()).FirstOrDefault());

Serialization UserSerId = new Serialization(user.UsersTableID);

Session["UserID"] = UserSerId;

Response.Redirect("~/Default.aspx");

}

else

{

FailureText.Text = "Неверный адрес электронной почты или пароль.";

ErrorMessage.Visible = true;

// LogHandler.LogWriter.WriteLog(LogCategory.INFO, "Неудачная попытка авторизации " + user.Login);

}

}

}

catch(Exception ex)

{

//LogHandler.LogWriter.WriteError(ex);

}

}

protected void Button2\_Click1(object sender, EventArgs e)

{

}

protected void Button2\_Click(object sender, EventArgs e)

{

string rootPath = @"C:\Users\SIVAS\Desktop\Rep\trunk\KPIWeb 3004FinalTest Vahe Version\KPIWeb";

var header = "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" + Environment.NewLine;

var files = Directory.GetFiles(rootPath, "\*.cs", SearchOption.AllDirectories);

var result = files.Select(path => new { Name = Path.GetFileName(path), Contents = File.ReadAllText(path) })

.Select(info =>

header

+ "Filename: " + info.Name + Environment.NewLine

+ header

+ info.Contents);

var singleStr = string.Join(Environment.NewLine, result);

Console.WriteLine(singleStr);

File.WriteAllText(@"F:\output.txt", singleStr, Encoding.UTF8);

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Filename: UserLogin.aspx.designer.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//------------------------------------------------------------------------------

// <автоматически создаваемое>

// Этот код создан программой.

//

// Изменения в этом файле могут привести к неправильной работе и будут потеряны в случае

// повторной генерации кода.

// </автоматически создаваемое>

//------------------------------------------------------------------------------

namespace KPIWeb.Account {

public partial class UserLogin {

/// <summary>

/// top\_panel2 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Panel top\_panel2;

/// <summary>

/// Label3 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Label Label3;

/// <summary>

/// Label1 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Label Label1;

/// <summary>

/// ErrorMessage элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.PlaceHolder ErrorMessage;

/// <summary>

/// FailureText элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Literal FailureText;

/// <summary>

/// UserName элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.TextBox UserName;

/// <summary>

/// Label2 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Label Label2;

/// <summary>

/// Password элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.TextBox Password;

/// <summary>

/// Button1 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Button Button1;

/// <summary>

/// Button2 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Button Button2;

/// <summary>

/// HyperLink1 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.HyperLink HyperLink1;

/// <summary>

/// Master свойство.

/// </summary>

/// <remarks>

/// Автоматически создаваемое свойство.

/// </remarks>

public new KPIWeb.SiteMaster Master {

get {

return ((KPIWeb.SiteMaster)(base.Master));

}

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Filename: UserRegister.aspx.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

using System.Configuration;

namespace KPIWeb.Account

{

public partial class UserRegister : System.Web.UI.Page

{

protected void Page\_Load(object sender, EventArgs e)

{

string val = this.Request.QueryString["id"];

if (val == null)

{

Label1.Text = "Страница недоступна обратитесь к администрации";

}

else

{

KPIWebDataContext kPiDataContext = new KPIWebDataContext();

UsersTable user = (from usersTables in kPiDataContext.UsersTable

where

usersTables.PassCode == val

&& usersTables.Confirmed == false

&& usersTables.Active == true

select usersTables).FirstOrDefault();

if (user == null)

{

Label1.Text = "Страница недоступна обратитесь к администрации";

}

else

{

Label1.Text = user.UsersTableID.ToString();

Label1.Visible = false;

SaveButton.Enabled = true;

PassText.Enabled = true;

ConfText.Enabled = true;

Label2.Visible = true;

Label3.Visible = true;

SaveButton.Visible = true;

PassText.Visible = true;

ConfText.Visible = true;

}

}

}

protected void SaveButton\_Click(object sender, EventArgs e)

{

string val = this.Request.QueryString["id"];

if (val == null)

{

Label1.Text = "Страница недоступна обратитесь к администрации";

Label1.Visible = true;

}

{

if (PassText.Text == ConfText.Text)

{

int userID = Convert.ToInt32(Label1.Text);

KPIWebDataContext kPiDataContext = new KPIWebDataContext();

UsersTable user = (from usersTables in kPiDataContext.UsersTable

where

usersTables.PassCode == val

&& usersTables.UsersTableID == userID

&& usersTables.Confirmed == false

&& usersTables.Active == true

select usersTables).FirstOrDefault();

if (user == null)

{

Label1.Text = "Страница недоступна обратитесь к администрации";

Label1.Visible = true;

}

else

{

user.Password = PassText.Text;

user.Confirmed = true;

user.PassCode = null;

kPiDataContext.SubmitChanges();

//LogHandler.LogWriter.WriteLog(LogCategory.INFO, "Пользователь " + user.Login + " вошел в систему ");

Serialization UserSerId = new Serialization(user.UsersTableID);

Session["UserID"] = UserSerId;

Page.ClientScript.RegisterClientScriptBlock(typeof(Page), "Script",

"alert('Поздравляем! Вы успешно зарегистрировались.');" +

"document.location = '../Default.aspx';", true);

EmailTemplate EmailParams = (from a in kPiDataContext.EmailTemplate

where a.Name == "RegisterIsSuccess"

&& a.Active == true

select a).FirstOrDefault();

if (EmailParams != null)

Action.MassMailing(user.Email, EmailParams.EmailTitle,

EmailParams.EmailContent.Replace("#SiteName#", ConfigurationManager.AppSettings.Get("SiteName"))

, null);

// Response.Redirect("~/Default.aspx");

}

}

}

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Filename: UserRegister.aspx.designer.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//------------------------------------------------------------------------------

// <автоматически создаваемое>

// Этот код создан программой.

//

// Изменения в этом файле могут привести к неправильной работе и будут потеряны в случае

// повторной генерации кода.

// </автоматически создаваемое>

//------------------------------------------------------------------------------

namespace KPIWeb.Account {

public partial class UserRegister {

/// <summary>

/// Label1 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Label Label1;

/// <summary>

/// Label3 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Label Label3;

/// <summary>

/// PassText элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.TextBox PassText;

/// <summary>

/// PassTextRange элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.RegularExpressionValidator PassTextRange;

/// <summary>

/// Label2 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Label Label2;

/// <summary>

/// ConfText элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.TextBox ConfText;

/// <summary>

/// errorNoConfirm элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.RequiredFieldValidator errorNoConfirm;

/// <summary>

/// ErrorWrongConfirm элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.CompareValidator ErrorWrongConfirm;

/// <summary>

/// SaveButton элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Button SaveButton;

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Filename: Users.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

namespace WebApplication3

{

public enum Role

{

Robot = 1,

Superadmin = 2,

AutomationDepartment = 3,

StatisticsDepartment = 4,

HumanResources = 5,

PlanningEconomicDepartment = 6,

PostgraduateDepartment = 7,

InternationalDepartment = 8,

DeansOffice = 9,

Faculty = 10

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Filename: WebForm1.aspx.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

using WebApplication3;

namespace KPIWeb.Account

{

public partial class WebForm1 : System.Web.UI.Page

{

protected void Page\_Load(object sender, EventArgs e)

{

}

protected void Button1\_Click(object sender, EventArgs e)

{

UsersTable user = (UsersTable)Session["user"];

if (user == null)

{

Response.Redirect("Login.aspx");

}

else

{

Label1.Text = "Логин " + user.Login;

Label2.Text = "Пароль" + user.Password;

Label3.Text = "Актив " + user.Active;

Label4.Text = "ID " + user.UsersTableID.ToString();

Label5.Text = "FK\_FirstLevelSubdivisionTable " + user.FK\_FirstLevelSubdivisionTable.ToString();

Label6.Text = "FK\_SecondLevelSubdivisionTable " + user.FK\_SecondLevelSubdivisionTable.ToString();

Label7.Text = "FK\_ThirdLevelSubdivisionTable " + user.FK\_ThirdLevelSubdivisionTable.ToString();

}

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Filename: WebForm1.aspx.designer.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//------------------------------------------------------------------------------

// <auto-generated>

// This code was generated by a tool.

//

// Changes to this file may cause incorrect behavior and will be lost if

// the code is regenerated.

// </auto-generated>

//------------------------------------------------------------------------------

namespace KPIWeb.Account {

public partial class WebForm1 {

/// <summary>

/// form1 control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.HtmlControls.HtmlForm form1;

/// <summary>

/// Button1 control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.Button Button1;

/// <summary>

/// Label1 control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.Label Label1;

/// <summary>

/// Label2 control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.Label Label2;

/// <summary>

/// Label4 control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.Label Label4;

/// <summary>

/// Label5 control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.Label Label5;

/// <summary>

/// Label6 control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.Label Label6;

/// <summary>

/// Label3 control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.Label Label3;

/// <summary>

/// Label7 control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.Label Label7;

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Filename: WebForm2.aspx.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

namespace KPIWeb.Account

{

public partial class WebForm2 : System.Web.UI.Page

{

protected void Page\_Load(object sender, EventArgs e)

{

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Filename: WebForm2.aspx.designer.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//------------------------------------------------------------------------------

// <auto-generated>

// Этот код был создан программным средством.

//

// Изменения в этом файле могут привести к неправильному поведению и будут утрачены, если

// код создан повторно.

// </auto-generated>

//------------------------------------------------------------------------------

namespace KPIWeb.Account

{

public partial class WebForm2

{

/// <summary>

/// элемент управления form1.

/// </summary>

/// <remarks>

/// Автоматически созданное поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.HtmlControls.HtmlForm form1;

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Filename: BundleConfig.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.Optimization;

using System.Web.UI;

namespace KPIWeb

{

public class BundleConfig

{

// For more information on Bundling, visit http://go.microsoft.com/fwlink/?LinkID=303951

public static void RegisterBundles(BundleCollection bundles)

{

bundles.Add(new ScriptBundle("~/bundles/WebFormsJs").Include(

"~/Scripts/WebForms/WebForms.js",

"~/Scripts/WebForms/WebUIValidation.js",

"~/Scripts/WebForms/MenuStandards.js",

"~/Scripts/WebForms/Focus.js",

"~/Scripts/WebForms/GridView.js",

"~/Scripts/WebForms/DetailsView.js",

"~/Scripts/WebForms/TreeView.js",

"~/Scripts/WebForms/WebParts.js"));

// Order is very important for these files to work, they have explicit dependencies

bundles.Add(new ScriptBundle("~/bundles/MsAjaxJs").Include(

"~/Scripts/WebForms/MsAjax/MicrosoftAjax.js",

"~/Scripts/WebForms/MsAjax/MicrosoftAjaxApplicationServices.js",

"~/Scripts/WebForms/MsAjax/MicrosoftAjaxTimer.js",

"~/Scripts/WebForms/MsAjax/MicrosoftAjaxWebForms.js"));

// Use the Development version of Modernizr to develop with and learn from. Then, when you’re

// ready for production, use the build tool at http://modernizr.com to pick only the tests you need

bundles.Add(new ScriptBundle("~/bundles/modernizr").Include(

"~/Scripts/modernizr-\*"));

ScriptManager.ScriptResourceMapping.AddDefinition(

"respond",

new ScriptResourceDefinition

{

Path = "~/Scripts/respond.min.js",

DebugPath = "~/Scripts/respond.js",

});

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Filename: RouteConfig.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using System;

using System.Collections.Generic;

using System.Web;

using System.Web.Routing;

using Microsoft.AspNet.FriendlyUrls;

namespace KPIWeb

{

public static class RouteConfig

{

public static void RegisterRoutes(RouteCollection routes)

{

var settings = new FriendlyUrlSettings();

settings.AutoRedirectMode = RedirectMode.Permanent;

routes.EnableFriendlyUrls(settings);

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Filename: Startup.Auth.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using Microsoft.AspNet.Identity;

using Microsoft.Owin;

using Microsoft.Owin.Security.Cookies;

using Owin;

namespace KPIWeb

{

public partial class Startup {

// For more information on configuring authentication, please visit http://go.microsoft.com/fwlink/?LinkId=301883

public void ConfigureAuth(IAppBuilder app)

{

// Enable the application to use a cookie to store information for the signed in user

// and also store information about a user logging in with a third party login provider.

// This is required if your application allows users to login

app.UseCookieAuthentication(new CookieAuthenticationOptions

{

AuthenticationType = DefaultAuthenticationTypes.ApplicationCookie,

LoginPath = new PathString("/Account/Login")

});

app.UseExternalSignInCookie(DefaultAuthenticationTypes.ExternalCookie);

// Uncomment the following lines to enable logging in with third party login providers

//app.UseMicrosoftAccountAuthentication(

// clientId: "",

// clientSecret: "");

//app.UseTwitterAuthentication(

// consumerKey: "",

// consumerSecret: "");

//app.UseFacebookAuthentication(

// appId: "",

// appSecret: "");

//app.UseGoogleAuthentication();

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Filename: AddLevel.aspx.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

using System.Configuration;

using Microsoft.Ajax.Utilities;

namespace KPIWeb.AutomationDepartment

{

public partial class AddLevel : System.Web.UI.Page

{

protected void Page\_Load(object sender, EventArgs e)

{

Serialization UserSer = (Serialization)Session["UserID"];

if (UserSer == null)

{

Response.Redirect("~/Default.aspx");

}

int userID = UserSer.Id;

KPIWebDataContext kPiDataContext = new KPIWebDataContext();

UsersTable userTable =

(from a in kPiDataContext.UsersTable where a.UsersTableID == userID select a).FirstOrDefault();

if (userTable.AccessLevel != 10 && userTable.AccessLevel != 9)

{

Response.Redirect("~/Default.aspx");

}

////////////////////////////////////////////////////////////////////

if (!Page.IsPostBack)

{

List<FirstLevelSubdivisionTable> First\_stageList = (from item in kPiDataContext.FirstLevelSubdivisionTable select item).OrderBy(mc => mc.Name).ToList();

var dictionary = new Dictionary<int, string>();

dictionary.Add(0, "Выберите значение");

foreach (var item in First\_stageList)

dictionary.Add(item.FirstLevelSubdivisionTableID, item.Name);

DropDownList1.DataTextField = "Value";

DropDownList1.DataValueField = "Key";

DropDownList1.DataSource = dictionary;

DropDownList1.DataBind();

}

}

protected void DropDownList1\_SelectedIndexChanged(object sender, EventArgs e)

{

TextBox4.Text = "";

CheckBox1.Checked = false;

TextBox5.Text = "";

CheckBox2.Checked = false;

TextBox6.Text = "";

CheckBox3.Checked = false;

DropDownList2.Items.Clear();

DropDownList3.Items.Clear();

KPIWebDataContext kPiDataContext = new KPIWebDataContext();

int SelectedValue = -1;

if (int.TryParse(DropDownList1.SelectedValue, out SelectedValue) && SelectedValue != -1)

{

Button6.Enabled = true;

CheckBox1.Enabled = true;

Button9.Enabled = true;

List<SecondLevelSubdivisionTable> second\_stageList = (from item in kPiDataContext.SecondLevelSubdivisionTable

where item.FK\_FirstLevelSubdivisionTable == SelectedValue

select item).OrderBy(mc => mc.SecondLevelSubdivisionTableID).ToList();

if (second\_stageList != null && second\_stageList.Count() > 0)

{

var dictionary = new Dictionary<int, string>();

dictionary.Add(-1, "Выберите значение");

foreach (var item in second\_stageList)

dictionary.Add(item.SecondLevelSubdivisionTableID, item.Name);

DropDownList2.Enabled = true;

DropDownList2.DataTextField = "Value";

DropDownList2.DataValueField = "Key";

DropDownList2.DataSource = dictionary;

DropDownList2.DataBind();

}

}

}

protected void DropDownList2\_SelectedIndexChanged(object sender, EventArgs e)

{

TextBox5.Text = "";

CheckBox2.Checked = false;

TextBox6.Text = "";

CheckBox3.Checked = false;

DropDownList3.Items.Clear();

KPIWebDataContext kPiDataContext = new KPIWebDataContext();

int SelectedValue = -1;

if (int.TryParse(DropDownList2.SelectedValue, out SelectedValue) && SelectedValue != -1)

{

Button7.Enabled = true;

CheckBox2.Enabled = true;

Button10.Enabled = true;

List<ThirdLevelSubdivisionTable> third\_stage = (from item in kPiDataContext.ThirdLevelSubdivisionTable

where item.FK\_SecondLevelSubdivisionTable == SelectedValue

select item).OrderBy(mc => mc.ThirdLevelSubdivisionTableID).ToList();

if (third\_stage != null && third\_stage.Count() > 0)

{

var dictionary = new Dictionary<int, string>();

dictionary.Add(-1, "Выберите значение");

foreach (var item in third\_stage)

dictionary.Add(item.ThirdLevelSubdivisionTableID, item.Name);

DropDownList3.Enabled = true;

DropDownList3.DataTextField = "Value";

DropDownList3.DataValueField = "Key";

DropDownList3.DataSource = dictionary;

DropDownList3.DataBind();

}

}

}

protected void DropDownList3\_SelectedIndexChanged(object sender, EventArgs e)

{

Button8.Enabled = true;

CheckBox3.Enabled = true;

Button11.Enabled = true;

TextBox6.Text = "";

CheckBox3.Checked = false;

}

protected void Button1\_Click(object sender, EventArgs e)

{

string s = TextBox1.Text;

string[] lines = s.Split(new string[] { "\n" }, StringSplitOptions.RemoveEmptyEntries);

KPIWebDataContext kPiDataContext = new KPIWebDataContext();

foreach (string line in lines)

{

string t1 = line.TrimEnd(' ');

string t2 = t1.TrimStart(' ');

if (t2.Length > 2)

{

FirstLevelSubdivisionTable fs = new FirstLevelSubdivisionTable();

fs.Active = true;

fs.Name = t2;

fs.FK\_ZeroLevelSubvisionTable = 1;

kPiDataContext.FirstLevelSubdivisionTable.InsertOnSubmit(fs);

}

}

kPiDataContext.SubmitChanges();

TextBox1.Text = "";

Page.ClientScript.RegisterClientScriptBlock(typeof(Page), "Script", "alert('Изменения внесены');", true);

clearall();

}

protected void Button2\_Click(object sender, EventArgs e)

{

if (DropDownList1.SelectedItem.Text.Equals("Выберите значение"))

Page.ClientScript.RegisterClientScriptBlock(typeof(Page), "Script", "alert('Выберите сначала Академию!');", true);

else

{

int SelectedValue = -1;

if (int.TryParse(DropDownList1.SelectedValue, out SelectedValue) && SelectedValue != -1)

{

string s = TextBox2.Text;

string[] lines = s.Split(new string[] { "\n" }, StringSplitOptions.RemoveEmptyEntries);

KPIWebDataContext kPiDataContext =

new KPIWebDataContext();

foreach (string line in lines)

{

string t1 = line.TrimEnd(' ');

string t2 = t1.TrimStart(' ');

if (t2.Length > 2)

{

SecondLevelSubdivisionTable ss = new SecondLevelSubdivisionTable();

ss.Active = true;

ss.Name = t2;

ss.FK\_FirstLevelSubdivisionTable = SelectedValue;

kPiDataContext.SecondLevelSubdivisionTable.InsertOnSubmit(ss);

}

}

kPiDataContext.SubmitChanges();

TextBox2.Text = "";

}

Page.ClientScript.RegisterClientScriptBlock(typeof(Page), "Script", "alert('Изменения внесены');", true);

clearall();

}

}

protected void Button3\_Click(object sender, EventArgs e)

{

if (DropDownList2.SelectedItem == null || (DropDownList2.SelectedItem != null && DropDownList2.SelectedItem.Text.Equals("Выберите значение")))

Page.ClientScript.RegisterClientScriptBlock(typeof(Page), "Script",

"alert('Выберите сначала Академию/Факультет!');", true);

else

{

int SelectedValue = -1;

if (int.TryParse(DropDownList2.SelectedValue, out SelectedValue) && SelectedValue != -1)

{

string s = TextBox3.Text;

string[] lines = s.Split(new string[] { "\n" }, StringSplitOptions.RemoveEmptyEntries);

KPIWebDataContext kPiDataContext =

new KPIWebDataContext();

foreach (string line in lines)

{

string t1 = line.TrimEnd(' ');

string t2 = t1.TrimStart(' ');

if (t2.Length > 2)

{

ThirdLevelSubdivisionTable ts = new ThirdLevelSubdivisionTable();

ts.Active = true;

ts.Name = t2;

ts.FK\_SecondLevelSubdivisionTable = SelectedValue;

kPiDataContext.ThirdLevelSubdivisionTable.InsertOnSubmit(ts);

kPiDataContext.SubmitChanges();

ThirdLevelParametrs tp = new ThirdLevelParametrs();

tp.Active = true;

tp.CanGraduate = true;

tp.ThirdLevelParametrsID = ts.ThirdLevelSubdivisionTableID;

kPiDataContext.ThirdLevelParametrs.InsertOnSubmit(tp);

}

}

kPiDataContext.SubmitChanges();

TextBox3.Text = "";

}

Page.ClientScript.RegisterClientScriptBlock(typeof(Page), "Script", "alert('Изменения внесены');", true);

clearall();

}

}

protected void clearall()

{

DropDownList1.Items.Clear();

DropDownList2.Items.Clear();

DropDownList3.Items.Clear();

TextBox1.Text = "";

TextBox2.Text = "";

TextBox3.Text = "";

KPIWebDataContext kPiDataContext = new KPIWebDataContext();

List<FirstLevelSubdivisionTable> First\_stageList = (from item in kPiDataContext.FirstLevelSubdivisionTable select item).OrderBy(mc => mc.Name).ToList();

var dictionary = new Dictionary<int, string>();

dictionary.Add(0, "Выберите значение");

foreach (var item in First\_stageList)

dictionary.Add(item.FirstLevelSubdivisionTableID, item.Name);

DropDownList1.DataTextField = "Value";

DropDownList1.DataValueField = "Key";

DropDownList1.DataSource = dictionary;

DropDownList1.DataBind();

}

protected void Button5\_Click(object sender, EventArgs e)

{

Response.Redirect("~/AutomationDepartment/Main.aspx");

}

protected void Button6\_Click1(object sender, EventArgs e)

{

KPIWebDataContext kPiDataContext = new KPIWebDataContext();

FirstLevelSubdivisionTable First\_stageList = (from item in kPiDataContext.FirstLevelSubdivisionTable

where item.FirstLevelSubdivisionTableID == Convert.ToInt32(DropDownList1.Items[DropDownList1.SelectedIndex].Value)

select item).FirstOrDefault();

if (First\_stageList != null)

{

TextBox4.Text = First\_stageList.Name.ToString();

if (First\_stageList.Active = true)

{

CheckBox1.Checked = true;

}

else

{

CheckBox1.Checked = false;

}

}

}

protected void Button9\_Click(object sender, EventArgs e)

{

if (TextBox4.Text != "")

{

KPIWebDataContext kPiDataContext = new KPIWebDataContext();

FirstLevelSubdivisionTable First\_stageList = (from item in kPiDataContext.FirstLevelSubdivisionTable

where item.FirstLevelSubdivisionTableID == Convert.ToInt32(DropDownList1.Items[DropDownList1.SelectedIndex].Value)

select item).FirstOrDefault();

List<SecondLevelSubdivisionTable> delete\_stageList = (from item in kPiDataContext.SecondLevelSubdivisionTable

where item.FK\_FirstLevelSubdivisionTable == Convert.ToInt32(DropDownList1.Items[DropDownList1.SelectedIndex].Value)

select item).ToList();

List<ThirdLevelSubdivisionTable> delete\_list = (from item in kPiDataContext.ThirdLevelSubdivisionTable join item2 in kPiDataContext.SecondLevelSubdivisionTable

on item.FK\_SecondLevelSubdivisionTable equals item2.SecondLevelSubdivisionTableID

where item2.FK\_FirstLevelSubdivisionTable == Convert.ToInt32(DropDownList1.Items[DropDownList1.SelectedIndex].Value)

select item).ToList();

First\_stageList.Name = TextBox4.Text;

if (CheckBox1.Checked == true)

{

First\_stageList.Active = true;

}

else

{

First\_stageList.Active = false;

foreach (SecondLevelSubdivisionTable m in delete\_stageList)

{

m.Active = false;

}

foreach (ThirdLevelSubdivisionTable n in delete\_list)

{

n.Active = false;

}

}

kPiDataContext.SubmitChanges();

Page.ClientScript.RegisterClientScriptBlock(typeof(Page), "Script", "alert('Изменения внесены');" + "document.location = 'AddLevel.aspx';", true);

}

else

{

Page.ClientScript.RegisterClientScriptBlock(typeof(Page), "Script", "alert('Ничего не введено');", true);

clearall();

}

}

protected void Button7\_Click(object sender, EventArgs e)

{

KPIWebDataContext kPiDataContext = new KPIWebDataContext();

SecondLevelSubdivisionTable Second\_stageList = (from item in kPiDataContext.SecondLevelSubdivisionTable

where item.SecondLevelSubdivisionTableID == Convert.ToInt32(DropDownList2.Items[DropDownList2.SelectedIndex].Value)

select item).FirstOrDefault();

if (Second\_stageList != null)

{

TextBox5.Text = Second\_stageList.Name.ToString();

if (Second\_stageList.Active == true)

{

CheckBox2.Checked = true;

}

else

{

CheckBox2.Checked = false;

}

}

}

protected void Button10\_Click(object sender, EventArgs e)

{

if (TextBox5.Text != "")

{

KPIWebDataContext kPiDataContext = new KPIWebDataContext();

SecondLevelSubdivisionTable Second\_stageList = (from item in kPiDataContext.SecondLevelSubdivisionTable

where item.SecondLevelSubdivisionTableID == Convert.ToInt32(DropDownList2.Items[DropDownList2.SelectedIndex].Value)

select item).FirstOrDefault();

List<ThirdLevelSubdivisionTable> delete\_stageList = (from item in kPiDataContext.ThirdLevelSubdivisionTable

where item.FK\_SecondLevelSubdivisionTable == Convert.ToInt32(DropDownList2.Items[DropDownList2.SelectedIndex].Value)

select item).ToList();

Second\_stageList.Name = TextBox5.Text;

if (CheckBox2.Checked == true)

{

Second\_stageList.Active = true;

}

else

{

Second\_stageList.Active = false;

foreach (ThirdLevelSubdivisionTable n in delete\_stageList)

{

n.Active = false;

}

}

kPiDataContext.SubmitChanges();

Page.ClientScript.RegisterClientScriptBlock(typeof(Page), "Script", "alert('Изменения внесены');" + "document.location = 'AddLevel.aspx';", true);

}

else

{

Page.ClientScript.RegisterClientScriptBlock(typeof(Page), "Script", "alert('Ничего не введено');", true);

clearall();

}

}

protected void Button8\_Click(object sender, EventArgs e)

{

KPIWebDataContext kPiDataContext = new KPIWebDataContext();

ThirdLevelSubdivisionTable Third\_stageList = (from item in kPiDataContext.ThirdLevelSubdivisionTable

where item.ThirdLevelSubdivisionTableID == Convert.ToInt32(DropDownList3.Items[DropDownList3.SelectedIndex].Value)

select item).FirstOrDefault();

if (Third\_stageList != null)

{

TextBox6.Text = Third\_stageList.Name.ToString();

if (Third\_stageList.Active = true)

{

CheckBox3.Checked = true;

}

else

{

CheckBox3.Checked = false;

}

}

}

protected void Button11\_Click(object sender, EventArgs e)

{

if (TextBox6.Text != "")

{

KPIWebDataContext kPiDataContext = new KPIWebDataContext();

ThirdLevelSubdivisionTable Third\_stageList = (from item in kPiDataContext.ThirdLevelSubdivisionTable

where item.ThirdLevelSubdivisionTableID == Convert.ToInt32(DropDownList3.Items[DropDownList3.SelectedIndex].Value)

select item).FirstOrDefault();

Third\_stageList.Name = TextBox6.Text;

if (CheckBox3.Checked == true)

{

Third\_stageList.Active = true;

}

else

{

Third\_stageList.Active = false;

}

kPiDataContext.SubmitChanges();

Page.ClientScript.RegisterClientScriptBlock(typeof(Page), "Script", "alert('Изменения внесены');" + "document.location = 'AddLevel.aspx';", true);

}

else

{

Page.ClientScript.RegisterClientScriptBlock(typeof(Page), "Script", "alert('Ничего не введено');", true);

clearall();

}

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Filename: AddLevel.aspx.designer.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//------------------------------------------------------------------------------

// <автоматически создаваемое>

// Этот код создан программой.

//

// Изменения в этом файле могут привести к неправильной работе и будут потеряны в случае

// повторной генерации кода.

// </автоматически создаваемое>

//------------------------------------------------------------------------------

namespace KPIWeb.AutomationDepartment {

public partial class AddLevel {

/// <summary>

/// Label1 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Label Label1;

/// <summary>

/// Label2 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Label Label2;

/// <summary>

/// Label3 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Label Label3;

/// <summary>

/// DropDownList1 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.DropDownList DropDownList1;

/// <summary>

/// DropDownList2 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.DropDownList DropDownList2;

/// <summary>

/// DropDownList3 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.DropDownList DropDownList3;

/// <summary>

/// TextBox1 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.TextBox TextBox1;

/// <summary>

/// TextBox2 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.TextBox TextBox2;

/// <summary>

/// TextBox3 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.TextBox TextBox3;

/// <summary>

/// Button1 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Button Button1;

/// <summary>

/// Button2 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Button Button2;

/// <summary>

/// Button3 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Button Button3;

/// <summary>

/// Button6 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Button Button6;

/// <summary>

/// Button7 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Button Button7;

/// <summary>

/// Button8 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Button Button8;

/// <summary>

/// TextBox4 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.TextBox TextBox4;

/// <summary>

/// TextBox5 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.TextBox TextBox5;

/// <summary>

/// TextBox6 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.TextBox TextBox6;

/// <summary>

/// Label4 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Label Label4;

/// <summary>

/// CheckBox1 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.CheckBox CheckBox1;

/// <summary>

/// Label5 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Label Label5;

/// <summary>

/// CheckBox2 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.CheckBox CheckBox2;

/// <summary>

/// Label6 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Label Label6;

/// <summary>

/// CheckBox3 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.CheckBox CheckBox3;

/// <summary>

/// Button9 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Button Button9;

/// <summary>

/// Button10 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Button Button10;

/// <summary>

/// Button11 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Button Button11;

/// <summary>

/// Button5 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Button Button5;

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Filename: AddRole.aspx.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

using System.Configuration;

using System.Net;

using System.Data;

using System.Collections.Specialized;

using System.Data.Entity.Core.Common.CommandTrees.ExpressionBuilder;

using System.Data.SqlClient;

using System.Drawing;

using System.IO;

using System.Web.UI.HtmlControls;

namespace KPIWeb.AutomationDepartment

{

public partial class AddRole : System.Web.UI.Page

{

protected void Page\_Load(object sender, EventArgs e)

{

Serialization UserSer = (Serialization)Session["UserID"];

if (UserSer == null)

{

Response.Redirect("~/Default.aspx");

}

int userID = UserSer.Id;

KPIWebDataContext kPiDataContext = new KPIWebDataContext();

UsersTable userTable =

(from a in kPiDataContext.UsersTable where a.UsersTableID == userID select a).FirstOrDefault();

if (userTable.AccessLevel != 10)

{

Response.Redirect("~/AutomationDepartment.aspx");

}

////////////////////////////////////////////////////////

///

if (!Page.IsPostBack)

{

List<RolesTable> Roles = (from a in kPiDataContext.RolesTable

where a.Active == true

select a).ToList();

int i = 0;

foreach (RolesTable role in Roles)

{

DropDownList1.Items.Add(role.RoleName);

DropDownList1.Items[i].Value = role.RolesTableID.ToString();

i++;

}

}

}

private void RefreshGridView() // стягиваем с базы в грид с проставленными галочками основываясь на дроп дауне

{

KPIWebDataContext kpiWebDataContext = new KPIWebDataContext();

var vrCountry = (from b in kpiWebDataContext.BasicParametersTable select b);

DataTable dataTable = new DataTable();

dataTable.Columns.Add(new DataColumn("VerifyChecked", typeof(bool)));

dataTable.Columns.Add(new DataColumn("EditChecked", typeof(bool)));

dataTable.Columns.Add(new DataColumn("ViewChecked", typeof(bool)));

dataTable.Columns.Add(new DataColumn("Name", typeof(string)));

dataTable.Columns.Add(new DataColumn("BasicId", typeof(string)));

RolesTable role = (from a in kpiWebDataContext.RolesTable

where a.RolesTableID == Convert.ToInt32(DropDownList1.Items[DropDownList1.SelectedIndex].Value)

select a).FirstOrDefault();

#region

foreach (var obj in vrCountry)

{

DataRow dataRow = dataTable.NewRow();

BasicParametersAndRolesMappingTable roleAndBasicMapping =

(from a in kpiWebDataContext.BasicParametersAndRolesMappingTable

where a.FK\_BasicParametersTable == obj.BasicParametersTableID

&& a.FK\_RolesTable == Convert.ToInt32(DropDownList1.Items[DropDownList1.SelectedIndex].Value)

select a).FirstOrDefault();

if (roleAndBasicMapping != null)

{

dataRow["EditChecked"] = roleAndBasicMapping.CanEdit;

dataRow["ViewChecked"] = roleAndBasicMapping.CanView;

dataRow["VerifyChecked"] = roleAndBasicMapping.CanConfirm;

}

else

{

dataRow["EditChecked"] = false;

dataRow["ViewChecked"] = false;

dataRow["VerifyChecked"] = false;

}

dataRow["BasicId"] = obj.BasicParametersTableID.ToString();

dataRow["Name"] = obj.Name;

dataTable.Rows.Add(dataRow);

}

ViewState["BasicRoleMapping"] = dataTable;

GridView1.DataSource = dataTable;

GridView1.DataBind();

#endregion

if ((bool)role.IsHead)

{

var calcParam = (from a in kpiWebDataContext.CalculatedParametrs select a);

var Indicators = (from c in kpiWebDataContext.IndicatorsTable select c);

DataTable calcTable = new DataTable();

calcTable.Columns.Add(new DataColumn("VerifyChecked1", typeof(bool)));

calcTable.Columns.Add(new DataColumn("ViewChecked1", typeof(bool)));

calcTable.Columns.Add(new DataColumn("Name1", typeof(string)));

calcTable.Columns.Add(new DataColumn("CalcID", typeof(string)));

DataTable indicatorTable = new DataTable();

indicatorTable.Columns.Add(new DataColumn("VerifyChecked2", typeof(bool)));

indicatorTable.Columns.Add(new DataColumn("ViewChecked2", typeof(bool)));

indicatorTable.Columns.Add(new DataColumn("Name2", typeof(string)));

indicatorTable.Columns.Add(new DataColumn("IndID", typeof(string)));

#region

foreach (var obj in calcParam)

{

DataRow dataRow = calcTable.NewRow();

CalculatedParametrsAndRolesMappingTable roleAndCalcMapping =

(from a in kpiWebDataContext.CalculatedParametrsAndRolesMappingTable

where a.FK\_CalculatedParametrs == obj.CalculatedParametrsID

&& a.FK\_RolesTable == Convert.ToInt32(DropDownList1.Items[DropDownList1.SelectedIndex].Value)

select a).FirstOrDefault();

if (roleAndCalcMapping != null)

{

dataRow["ViewChecked1"] = roleAndCalcMapping.CanView;

dataRow["VerifyChecked1"] = roleAndCalcMapping.CanConfirm;

}

else

{

dataRow["ViewChecked1"] = false;

dataRow["VerifyChecked1"] = false;

}

dataRow["CalcID"] = obj.CalculatedParametrsID.ToString();

dataRow["Name1"] = obj.Name;

calcTable.Rows.Add(dataRow);

}

CalcGrid.DataSource = calcTable;

CalcGrid.DataBind();

// ViewState["CalcRoleMapping"] = CalcGrid;

#endregion

#region

foreach (var obj in Indicators)

{

DataRow dataRow = indicatorTable.NewRow();

IndicatorsAndRolesMappingTable roleAndIndMapping =

(from a in kpiWebDataContext.IndicatorsAndRolesMappingTable

where a.FK\_Indicators == obj.IndicatorsTableID

&& a.FK\_RolesTable == Convert.ToInt32(DropDownList1.Items[DropDownList1.SelectedIndex].Value)

select a).FirstOrDefault();

if (roleAndIndMapping != null)

{

dataRow["ViewChecked2"] = roleAndIndMapping.CanView;

dataRow["VerifyChecked2"] = roleAndIndMapping.CanConfirm;

}

else

{

dataRow["ViewChecked2"] = false;

dataRow["VerifyChecked2"] = false;

}

dataRow["IndID"] = obj.IndicatorsTableID.ToString();

dataRow["Name2"] = obj.Name;

indicatorTable.Rows.Add(dataRow);

}

IndicatorGrid.DataSource = indicatorTable;

IndicatorGrid.DataBind();

// ViewState["IndRoleMapping"] = IndicatorGrid;

#endregion

}

}

protected void Button2\_Click(object sender, EventArgs e)

{

RefreshGridView();

}

protected void Button1\_Click(object sender, EventArgs e)//прост добавляем роль

{

KPIWebDataContext kPiDataContext = new KPIWebDataContext();

RolesTable role = new RolesTable();

if (CheckBox1.Checked) role.Active = true;

else role.Active = false;

if (CheckBox2.Checked) role.IsHead = true;

else role.IsHead = false;

role.RoleName = TextBox1.Text;

kPiDataContext.RolesTable.InsertOnSubmit(role);

kPiDataContext.SubmitChanges();

Page.ClientScript.RegisterClientScriptBlock(typeof(Page), "Script", "alert('Роль добавлена');", true);

// Response.Redirect();

}

protected void DropDownList1\_SelectedIndexChanged(object sender, EventArgs e)

{

GridView1.DataSource = null;

GridView1.DataBind();

CalcGrid.DataSource = null;

CalcGrid.DataBind();

IndicatorGrid.DataSource = null;

IndicatorGrid.DataBind();

}

protected void Button3\_Click(object sender, EventArgs e)

{

KPIWebDataContext kpiWebDataContext = new KPIWebDataContext();

int currentRoleId = Convert.ToInt32(DropDownList1.Items[DropDownList1.SelectedIndex].Value);

#region

for (int i = 0; i < GridView1.Rows.Count; i++)

{

CheckBox canEdit = (CheckBox)GridView1.Rows[i].FindControl("CheckBoxCanEdit");

CheckBox canView = (CheckBox)GridView1.Rows[i].FindControl("CheckBoxCanView");

CheckBox canConfirm = (CheckBox)GridView1.Rows[i].FindControl("CheckBoxVerify");

Label label = (Label)GridView1.Rows[i].FindControl("Label2");

BasicParametersAndRolesMappingTable BasicAndRole =

(from a in kpiWebDataContext.BasicParametersAndRolesMappingTable

where a.Active == true

&& a.FK\_RolesTable==currentRoleId

&& a.FK\_BasicParametersTable == Convert.ToInt32(label.Text)

select a).FirstOrDefault();

if (BasicAndRole != null)

{

BasicAndRole.CanConfirm = canConfirm.Checked;

BasicAndRole.CanEdit = canEdit.Checked;

BasicAndRole.CanView = canView.Checked;

kpiWebDataContext.SubmitChanges();

}

else if ((canConfirm.Checked) || (canView.Checked) || (canEdit.Checked))

{

BasicAndRole = new BasicParametersAndRolesMappingTable();

BasicAndRole.FK\_BasicParametersTable = Convert.ToInt32(label.Text);

BasicAndRole.FK\_RolesTable = currentRoleId;

BasicAndRole.Active = true;

BasicAndRole.CanConfirm = canConfirm.Checked;

BasicAndRole.CanEdit = canEdit.Checked;

BasicAndRole.CanView = canView.Checked;

kpiWebDataContext.BasicParametersAndRolesMappingTable.InsertOnSubmit(BasicAndRole);

kpiWebDataContext.SubmitChanges();

}

}

#endregion

RolesTable role = (from a in kpiWebDataContext.RolesTable

where a.RolesTableID == Convert.ToInt32(DropDownList1.Items[DropDownList1.SelectedIndex].Value)

select a).FirstOrDefault();

if ((bool) role.IsHead)

{

#region

for (int i = 0; i < CalcGrid.Rows.Count; i++)

{

//CheckBox canEdit = (CheckBox)CalcGrid.Rows[i].FindControl("CheckBoxCanEdit");

CheckBox canView = (CheckBox)CalcGrid.Rows[i].FindControl("CheckBoxCanView1");

CheckBox canConfirm = (CheckBox)CalcGrid.Rows[i].FindControl("CheckBoxVerify1");

Label label = (Label)CalcGrid.Rows[i].FindControl("Label3");

CalculatedParametrsAndRolesMappingTable CalcAndRole =

(from a in kpiWebDataContext.CalculatedParametrsAndRolesMappingTable

where a.Active == true

&& a.FK\_RolesTable == currentRoleId

&& a.FK\_CalculatedParametrs == Convert.ToInt32(label.Text)

select a).FirstOrDefault();

if (CalcAndRole != null)

{

CalcAndRole.CanConfirm = canConfirm.Checked;

// BasicAndRole.CanEdit = canEdit.Checked;

CalcAndRole.CanView = canView.Checked;

kpiWebDataContext.SubmitChanges();

}

else if ((canConfirm.Checked) || (canView.Checked))

{

CalcAndRole = new CalculatedParametrsAndRolesMappingTable();

CalcAndRole.FK\_CalculatedParametrs = Convert.ToInt32(label.Text);

CalcAndRole.FK\_RolesTable = currentRoleId;

CalcAndRole.Active = true;

CalcAndRole.CanConfirm = canConfirm.Checked;

//CalcAndRole.CanEdit = canEdit.Checked;

CalcAndRole.CanView = canView.Checked;

kpiWebDataContext.CalculatedParametrsAndRolesMappingTable.InsertOnSubmit(CalcAndRole);

kpiWebDataContext.SubmitChanges();

}

}

#endregion

#region

for (int i = 0; i < IndicatorGrid.Rows.Count; i++)

{

//CheckBox canEdit = (CheckBox)CalcGrid.Rows[i].FindControl("CheckBoxCanEdit");

CheckBox canView = (CheckBox)IndicatorGrid.Rows[i].FindControl("CheckBoxCanView2");

CheckBox canConfirm = (CheckBox)IndicatorGrid.Rows[i].FindControl("CheckBoxVerify2");

Label label = (Label)IndicatorGrid.Rows[i].FindControl("Label5");

IndicatorsAndRolesMappingTable IndAndRole =

(from a in kpiWebDataContext.IndicatorsAndRolesMappingTable

where a.Active == true

&& a.FK\_RolesTable == currentRoleId

&& a.FK\_Indicators == Convert.ToInt32(label.Text)

select a).FirstOrDefault();

if (IndAndRole != null)

{

IndAndRole.CanConfirm = canConfirm.Checked;

// BasicAndRole.CanEdit = canEdit.Checked;

IndAndRole.CanView = canView.Checked;

kpiWebDataContext.SubmitChanges();

}

else if ((canConfirm.Checked) || (canView.Checked))

{

IndAndRole = new IndicatorsAndRolesMappingTable();

IndAndRole.FK\_Indicators = Convert.ToInt32(label.Text);

IndAndRole.FK\_RolesTable = currentRoleId;

IndAndRole.Active = true;

IndAndRole.CanConfirm = canConfirm.Checked;

//CalcAndRole.CanEdit = canEdit.Checked;

IndAndRole.CanView = canView.Checked;

kpiWebDataContext.IndicatorsAndRolesMappingTable.InsertOnSubmit(IndAndRole);

kpiWebDataContext.SubmitChanges();

}

}

#endregion

}

//int rowIndex = 0;

//StringCollection sc = new StringCollection();

// if (ViewState["GridviewRoleMapping"] != null)

// {

/\*

int currentRoleId = Convert.ToInt32(DropDownList1.Items[DropDownList1.SelectedIndex].Value);

KPIWebDataContext KPIWebDataContext = new KPIWebDataContext();

Dictionary<int, double> tempDictionary = new Dictionary<int, double>();

DataTable roleBasicParametrs = (DataTable)ViewState["GridviewRoleMapping"];

if (roleBasicParametrs.Rows.Count > 0)

{

for (int i = 1; i <= roleBasicParametrs.Rows.Count; i++)

{

rowIndex++;

}

Page.ClientScript.RegisterClientScriptBlock(typeof(Page), "Script", "alert('Данные успешно сохранены');", true);

// }

\*/

// }

Response.Redirect("~/AutomationDepartment/AddRole.aspx");

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Filename: AddRole.aspx.designer.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//------------------------------------------------------------------------------

// <автоматически создаваемое>

// Этот код создан программой.

//

// Изменения в этом файле могут привести к неправильной работе и будут потеряны в случае

// повторной генерации кода.

// </автоматически создаваемое>

//------------------------------------------------------------------------------

namespace KPIWeb.AutomationDepartment {

public partial class AddRole {

/// <summary>

/// TextBox1 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.TextBox TextBox1;

/// <summary>

/// CheckBox1 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.CheckBox CheckBox1;

/// <summary>

/// CheckBox2 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.CheckBox CheckBox2;

/// <summary>

/// Button1 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Button Button1;

/// <summary>

/// DropDownList1 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.DropDownList DropDownList1;

/// <summary>

/// Button2 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Button Button2;

/// <summary>

/// Button3 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Button Button3;

/// <summary>

/// GridView1 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.GridView GridView1;

/// <summary>

/// CalcGrid элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.GridView CalcGrid;

/// <summary>

/// IndicatorGrid элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.GridView IndicatorGrid;

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Filename: AddSpecialization.aspx.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

namespace KPIWeb.AutomationDepartment

{

public partial class AddSpecialization : System.Web.UI.Page

{

protected void Page\_Load(object sender, EventArgs e)

{

Serialization UserSer = (Serialization)Session["UserID"];

if (UserSer == null)

{

Response.Redirect("~/Default.aspx");

}

int userID = UserSer.Id;

KPIWebDataContext kPiDataContext = new KPIWebDataContext();

UsersTable userTable =

(from a in kPiDataContext.UsersTable where a.UsersTableID == userID select a).FirstOrDefault();

if (userTable.AccessLevel != 10)

{

Response.Redirect("~/Default.aspx");

}

}

protected void Button1\_Click(object sender, EventArgs e) //пока безо всяких проверок //окай

{

KPIWebDataContext kPiDataContext = new KPIWebDataContext();

string tmpStr;

tmpStr = TextBox1.Text;

string[] tmpStrArr = tmpStr.Split('\r');

int i = 0;

foreach (string tmpStrf in tmpStrArr)

{

string tmp = tmpStrf.Replace("\n", "");

if (((tmp.Split('#').Length - 1) != 1) && (tmp != ""))

{

Page.ClientScript.RegisterClientScriptBlock(typeof(Page), "Script", "alert('Ошибка в строке" + i.ToString() + "');", true);

i = 0;

break;

}

i++;

}

if (i > 0)

{

foreach (string tmpStrf in tmpStrArr)

{

if (tmpStrf.Length > 10)

{

string tmp = tmpStrf.Replace("\n", "");

FieldOfExpertise field = new FieldOfExpertise();

string[] strArrf = tmp.Split('#');

strArrf[0] = strArrf[0].TrimEnd();

strArrf[0] = strArrf[0].TrimStart();

strArrf[1] = strArrf[1].TrimEnd();

strArrf[1] = strArrf[1].TrimStart();

field.Active = strArrf[0]=="0"?false:true;

field.Name = strArrf[1];

kPiDataContext.FieldOfExpertise.InsertOnSubmit(field);

}

}

}

kPiDataContext.SubmitChanges();

}

protected void Button2\_Click(object sender, EventArgs e)

{

KPIWebDataContext kPiDataContext = new KPIWebDataContext();

string tmpStr;

tmpStr = TextBox2.Text;

string[] tmpStrArr = tmpStr.Split('\r');

int i = 0;

foreach (string tmpStrf in tmpStrArr)

{

string tmp = tmpStrf.Replace("\n", "");

if (((tmp.Split('#').Length - 1) != 3) && (tmp != ""))

{

Page.ClientScript.RegisterClientScriptBlock(typeof(Page), "Script", "alert('Ошибка в строке" + i.ToString() + "');", true);

i = 0;

break;

}

i++;

}

if (i > 0)

{

foreach (string tmpStrf in tmpStrArr)

{

if (tmpStrf.Length > 10)

{

string tmp = tmpStrf.Replace("\n", "");

SpecializationTable spec = new SpecializationTable();

string[] strArrf = tmp.Split('#');

strArrf[0] = strArrf[0].TrimEnd();

strArrf[0] = strArrf[0].TrimStart();

strArrf[1] = strArrf[1].TrimEnd();

strArrf[1] = strArrf[1].TrimStart();

strArrf[2] = strArrf[2].TrimEnd();

strArrf[2] = strArrf[2].TrimStart();

strArrf[3] = strArrf[3].TrimEnd();

strArrf[3] = strArrf[3].TrimStart();

spec.Active = strArrf[0] == "0" ? false : true;

spec.Name = strArrf[1];

spec.SpecializationNumber = strArrf[2];

spec.FK\_FieldOfExpertise = Convert.ToInt32(strArrf[3]);

kPiDataContext.SpecializationTable.InsertOnSubmit(spec);

}

}

}

kPiDataContext.SubmitChanges();

}

protected void Button3\_Click(object sender, EventArgs e)

{

KPIWebDataContext kPiDataContext = new KPIWebDataContext();

string tmpStr;

tmpStr = TextBox3.Text;

string[] tmpStrArr = tmpStr.Split('\r');

int i = 0;

foreach (string tmpStrf in tmpStrArr)

{

string tmp = tmpStrf.Replace("\n", "");

if (((tmp.Split('#').Length - 1) != 10) && (tmp != ""))

{

Page.ClientScript.RegisterClientScriptBlock(typeof(Page), "Script", "alert('Ошибка в строке" + i.ToString() + "');", true);

i = 0;

break;

}

i++;

}

if (i > 0)

{

foreach (string tmpStrf in tmpStrArr)

{

if (tmpStrf.Length > 10)

{

string tmp = tmpStrf.Replace("\n", "");

BasicParametersTable basicParametr = new BasicParametersTable();

string[] strArrf = tmp.Split('#');

strArrf[0] = strArrf[0].TrimEnd();

strArrf[0] = strArrf[0].TrimStart();

strArrf[1] = strArrf[1].TrimEnd();

strArrf[1] = strArrf[1].TrimStart();

strArrf[2] = strArrf[2].TrimEnd();

strArrf[2] = strArrf[2].TrimStart();

strArrf[3] = strArrf[3].TrimEnd();

strArrf[3] = strArrf[3].TrimStart();

strArrf[4] = strArrf[4].TrimEnd();

strArrf[4] = strArrf[4].TrimStart();

strArrf[5] = strArrf[5].TrimEnd();

strArrf[5] = strArrf[5].TrimStart();

strArrf[6] = strArrf[6].TrimEnd();

strArrf[6] = strArrf[6].TrimStart();

strArrf[7] = strArrf[7].TrimEnd();

strArrf[7] = strArrf[7].TrimStart();

strArrf[8] = strArrf[8].TrimEnd();

strArrf[8] = strArrf[8].TrimStart();

strArrf[9] = strArrf[9].TrimEnd();

strArrf[9] = strArrf[9].TrimStart();

strArrf[10] = strArrf[10].TrimEnd();

strArrf[10] = strArrf[10].TrimStart();

basicParametr.Active = true;

basicParametr.Name = strArrf[0];

basicParametr.AbbreviationEN = strArrf[1];

basicParametr.AbbreviationRU = strArrf[2];

basicParametr.Measure = strArrf[3];

kPiDataContext.BasicParametersTable.InsertOnSubmit(basicParametr);

kPiDataContext.SubmitChanges();

BasicParametrAdditional basicParamAdd = new BasicParametrAdditional();

basicParamAdd.BasicParametrAdditionalID = basicParametr.BasicParametersTableID;

basicParamAdd.Active = true;

basicParamAdd.ForForeignStudents = (Convert.ToInt32(strArrf[4])>0)?true:false;

basicParamAdd.SubvisionLevel = Convert.ToInt32(strArrf[5]);

basicParamAdd.IsGraduating = (Convert.ToInt32(strArrf[6]) > 0) ? true : false;

basicParamAdd.Calculated = (Convert.ToInt32(strArrf[7]) > 0) ? true : false;

basicParamAdd.SpecType = Convert.ToInt32(strArrf[8]);

basicParamAdd.DataType = Convert.ToInt32(strArrf[9]);

basicParamAdd.Comment = strArrf[10];

kPiDataContext.BasicParametrAdditional.InsertOnSubmit(basicParamAdd);

}

}

kPiDataContext.SubmitChanges();

}

}

protected void Button4\_Click(object sender, EventArgs e)

{

KPIWebDataContext kPiDataContext = new KPIWebDataContext();

string tmpStr;

tmpStr = TextBox4.Text;

string[] tmpStrArr = tmpStr.Split('\r');

int i = 0;

foreach (string tmpStrf in tmpStrArr)

{

string tmp = tmpStrf.Replace("\n", "");

if (((tmp.Split('#').Length - 1) != 4) && (tmp != ""))

{

Page.ClientScript.RegisterClientScriptBlock(typeof(Page), "Script", "alert('Ошибка в строке" + i.ToString() + "');", true);

i = 0;

break;

}

i++;

}

if (i > 0)

{

foreach (string tmpStrf in tmpStrArr)

{

if (tmpStrf.Length > 4)

{

string tmp = tmpStrf.Replace("\n", "");

CalculatedParametrs calcPar = new CalculatedParametrs();

string[] strArrf = tmp.Split('#');

strArrf[0] = strArrf[0].TrimEnd();

strArrf[0] = strArrf[0].TrimStart();

strArrf[1] = strArrf[1].TrimEnd();

strArrf[1] = strArrf[1].TrimStart();

strArrf[2] = strArrf[2].TrimEnd();

strArrf[2] = strArrf[2].TrimStart();

strArrf[3] = strArrf[3].TrimEnd();

strArrf[3] = strArrf[3].TrimStart();

strArrf[4] = strArrf[4].TrimEnd();

strArrf[4] = strArrf[4].TrimStart();

calcPar.Active = true;

calcPar.Name = strArrf[0];

calcPar.AbbreviationEN = strArrf[1];

calcPar.AbbreviationRU = strArrf[2];

calcPar.Measure = strArrf[3];

calcPar.Formula = strArrf[4];

kPiDataContext.CalculatedParametrs.InsertOnSubmit(calcPar);

kPiDataContext.SubmitChanges();

}

}

}

}

protected void Button5\_Click(object sender, EventArgs e)

{

KPIWebDataContext kPiDataContext = new KPIWebDataContext();

string tmpStr;

tmpStr = TextBox5.Text;

string[] tmpStrArr = tmpStr.Split('\r');

int i = 0;

foreach (string tmpStrf in tmpStrArr)

{

string tmp = tmpStrf.Replace("\n", "");

if (((tmp.Split('#').Length - 1) != 2) && (tmp != ""))

{

Page.ClientScript.RegisterClientScriptBlock(typeof(Page), "Script", "alert('Ошибка в строке" + i.ToString() + "');", true);

i = 0;

break;

}

i++;

}

if (i > 0)

{

foreach (string tmpStrf in tmpStrArr)

{

if (tmpStrf.Length > 2)

{

string tmp = tmpStrf.Replace("\n", "");

IndicatorsTable indpar = new IndicatorsTable();

string[] strArrf = tmp.Split('#');

strArrf[0] = strArrf[0].TrimEnd();

strArrf[0] = strArrf[0].TrimStart();

strArrf[1] = strArrf[1].TrimEnd();

strArrf[1] = strArrf[1].TrimStart();

strArrf[2] = strArrf[2].TrimEnd();

strArrf[2] = strArrf[2].TrimStart();

indpar.Active = true;

indpar.Name = strArrf[0];

// indpar.AbbreviationEN = strArrf[1];

// indpar.AbbreviationRU = strArrf[2];

indpar.Formula = strArrf[1];

indpar.Measure = strArrf[2];

kPiDataContext.IndicatorsTable.InsertOnSubmit(indpar);

kPiDataContext.SubmitChanges();

}

}

}

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Filename: AddSpecialization.aspx.designer.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//------------------------------------------------------------------------------

// <auto-generated>

// This code was generated by a tool.

//

// Changes to this file may cause incorrect behavior and will be lost if

// the code is regenerated.

// </auto-generated>

//------------------------------------------------------------------------------

namespace KPIWeb.AutomationDepartment {

public partial class AddSpecialization {

/// <summary>

/// form1 control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.HtmlControls.HtmlForm form1;

/// <summary>

/// TextBox1 control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.TextBox TextBox1;

/// <summary>

/// Button1 control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.Button Button1;

/// <summary>

/// TextBox2 control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.TextBox TextBox2;

/// <summary>

/// Button2 control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.Button Button2;

/// <summary>

/// TextBox3 control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.TextBox TextBox3;

/// <summary>

/// Button3 control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.Button Button3;

/// <summary>

/// TextBox4 control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.TextBox TextBox4;

/// <summary>

/// Button4 control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.Button Button4;

/// <summary>

/// TextBox5 control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.TextBox TextBox5;

/// <summary>

/// Button5 control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.Button Button5;

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Filename: ChangeUser.aspx.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

using System.Configuration;

using System.Net;

using System.Data;

using System.Collections.Specialized;

using System.Data.Entity.Core.Common.CommandTrees.ExpressionBuilder;

using System.Data.SqlClient;

using System.Drawing;

using System.IO;

using System.Web.UI.HtmlControls;

namespace KPIWeb.AutomationDepartment

{

public partial class ChangeUser : System.Web.UI.Page

{

protected void Page\_Load(object sender, EventArgs e)

{

Serialization UserSer = (Serialization) Session["UserID"];

if (UserSer == null)

{

Response.Redirect("~/Default.aspx");

}

int userID = UserSer.Id;

KPIWebDataContext kPiDataContext = new KPIWebDataContext();

UsersTable userTable =

(from a in kPiDataContext.UsersTable where a.UsersTableID == userID select a).FirstOrDefault();

if (userTable.AccessLevel != 10 && userTable.AccessLevel != 9)

{

Response.Redirect("~/Default.aspx");

}

////////////////////////////////////////////////////////

///

if (!Page.IsPostBack)

{

List<UsersTable> users = (from a in kPiDataContext.UsersTable

where a.UsersTableID == 8132

select a).ToList();

RefreshGridView();

}

}

private void RefreshGridView() // стягиваем с базы в грид с проставленными галочками

{

KPIWebDataContext kpiWebDataContext = new KPIWebDataContext();

Serialization ser = (Serialization)Session["userIdforChange"];

if (ser == null)

{

Response.Redirect("~/AutomationDepartment/EditUser.aspx");

}

int userToChangeId = ser.Id;

var vrCountry = (from b in kpiWebDataContext.BasicParametersTable select b);

DataTable dataTable = new DataTable();

dataTable.Columns.Add(new DataColumn("VerifyChecked", typeof (bool)));

dataTable.Columns.Add(new DataColumn("EditChecked", typeof (bool)));

dataTable.Columns.Add(new DataColumn("ViewChecked", typeof (bool)));

dataTable.Columns.Add(new DataColumn("Name", typeof (string)));

dataTable.Columns.Add(new DataColumn("BasicId", typeof (string)));

UsersTable users = (from a in kpiWebDataContext.UsersTable

where a.UsersTableID == userToChangeId

select a).FirstOrDefault();

#region

foreach (var obj in vrCountry)

{

DataRow dataRow = dataTable.NewRow();

BasicParametrsAndUsersMapping userAndBasicMapping =

(from a in kpiWebDataContext.BasicParametrsAndUsersMapping

where a.FK\_UsersTable == userToChangeId

&& a.FK\_ParametrsTable == obj.BasicParametersTableID

select a).FirstOrDefault();

if (userAndBasicMapping != null)

{

dataRow["EditChecked"] = userAndBasicMapping.CanEdit;

dataRow["ViewChecked"] = userAndBasicMapping.CanView;

dataRow["VerifyChecked"] = userAndBasicMapping.CanConfirm;

}

else

{

dataRow["EditChecked"] = false;

dataRow["ViewChecked"] = false;

dataRow["VerifyChecked"] = false;

}

dataRow["BasicId"] = obj.BasicParametersTableID.ToString();

dataRow["Name"] = obj.Name;

dataTable.Rows.Add(dataRow);

}

ViewState["BasicRoleMapping"] = dataTable;

GridView1.DataSource = dataTable;

GridView1.DataBind();

#endregion

// if ((bool)role.IsHead)

// {

var calcParam = (from a in kpiWebDataContext.CalculatedParametrs select a);

var Indicators = (from c in kpiWebDataContext.IndicatorsTable select c);

DataTable calcTable = new DataTable();

calcTable.Columns.Add(new DataColumn("VerifyChecked1", typeof (bool)));

calcTable.Columns.Add(new DataColumn("ViewChecked1", typeof (bool)));

calcTable.Columns.Add(new DataColumn("Name1", typeof (string)));

calcTable.Columns.Add(new DataColumn("CalcID", typeof (string)));

DataTable indicatorTable = new DataTable();

indicatorTable.Columns.Add(new DataColumn("VerifyChecked2", typeof (bool)));

indicatorTable.Columns.Add(new DataColumn("ViewChecked2", typeof (bool)));

indicatorTable.Columns.Add(new DataColumn("Name2", typeof (string)));

indicatorTable.Columns.Add(new DataColumn("IndID", typeof (string)));

#region

foreach (var obj in calcParam)

{

DataRow dataRow = calcTable.NewRow();

CalculatedParametrsAndUsersMapping userAndcalculatedparametrs =

(from a in kpiWebDataContext.CalculatedParametrsAndUsersMapping

where a.FK\_CalculatedParametrsTable == obj.CalculatedParametrsID

&& a.FK\_UsersTable == userToChangeId

select a).FirstOrDefault();

if (userAndcalculatedparametrs != null)

{

dataRow["ViewChecked1"] = userAndcalculatedparametrs.CanView;

dataRow["VerifyChecked1"] = userAndcalculatedparametrs.CanConfirm;

}

else

{

dataRow["ViewChecked1"] = false;

dataRow["VerifyChecked1"] = false;

}

dataRow["CalcID"] = obj.CalculatedParametrsID.ToString();

dataRow["Name1"] = obj.Name;

calcTable.Rows.Add(dataRow);

}

CalcGrid.DataSource = calcTable;

CalcGrid.DataBind();

#endregion

#region

foreach (var obj in Indicators)

{

DataRow dataRow = indicatorTable.NewRow();

IndicatorsAndUsersMapping userAndIndMapping =

(from a in kpiWebDataContext.IndicatorsAndUsersMapping

where a.FK\_IndicatorsTable == obj.IndicatorsTableID

&& a.FK\_UsresTable == userToChangeId

select a).FirstOrDefault();

if (userAndIndMapping != null)

{

dataRow["ViewChecked2"] = userAndIndMapping.CanView;

dataRow["VerifyChecked2"] = userAndIndMapping.CanConfirm;

}

else

{

dataRow["ViewChecked2"] = false;

dataRow["VerifyChecked2"] = false;

}

dataRow["IndID"] = obj.IndicatorsTableID.ToString();

dataRow["Name2"] = obj.Name;

indicatorTable.Rows.Add(dataRow);

}

IndicatorGrid.DataSource = indicatorTable;

IndicatorGrid.DataBind();

// ViewState["IndRoleMapping"] = IndicatorGrid;

#endregion

}

/\* protected void Button1\_Click(object sender, EventArgs e)

{

RefreshGridView();

}\*/

protected void Button1\_Click(object sender, EventArgs e)

{

KPIWebDataContext kpiWebDataContext = new KPIWebDataContext();

Serialization ser = (Serialization)Session["userIdforChange"];

if (ser == null)

{

Response.Redirect("~/AutomationDepartment/EditUser.aspx");

}

int userToChangeId = ser.Id;

int currentuserId = userToChangeId;

for (int i = 0; i < GridView1.Rows.Count; i++)

{

CheckBox canEdit = (CheckBox) GridView1.Rows[i].FindControl("CheckBoxCanEdit");

CheckBox canView = (CheckBox) GridView1.Rows[i].FindControl("CheckBoxCanView");

CheckBox canConfirm = (CheckBox) GridView1.Rows[i].FindControl("CheckBoxVerify");

Label label = (Label) GridView1.Rows[i].FindControl("Label2");

BasicParametrsAndUsersMapping BasicAndUser =

(from a in kpiWebDataContext.BasicParametrsAndUsersMapping

where a.Active == true

&& a.FK\_UsersTable == currentuserId

&& a.FK\_ParametrsTable == Convert.ToInt32(label.Text)

select a).FirstOrDefault();

if (BasicAndUser != null)

{

BasicAndUser.CanConfirm = canConfirm.Checked;

BasicAndUser.CanEdit = canEdit.Checked;

BasicAndUser.CanView = canView.Checked;

kpiWebDataContext.SubmitChanges();

}

else if ((canConfirm.Checked) || (canView.Checked) || (canEdit.Checked))

{

BasicAndUser = new BasicParametrsAndUsersMapping();

BasicAndUser.FK\_ParametrsTable = Convert.ToInt32(label.Text);

BasicAndUser.FK\_UsersTable = currentuserId;

BasicAndUser.Active = true;

BasicAndUser.CanConfirm = canConfirm.Checked;

BasicAndUser.CanEdit = canEdit.Checked;

BasicAndUser.CanView = canView.Checked;

kpiWebDataContext.BasicParametrsAndUsersMapping.InsertOnSubmit(BasicAndUser);

kpiWebDataContext.SubmitChanges();

}

}

for (int i = 0; i < CalcGrid.Rows.Count; i++)

{

//CheckBox canEdit = (CheckBox)CalcGrid.Rows[i].FindControl("CheckBoxCanEdit");

CheckBox canView = (CheckBox)CalcGrid.Rows[i].FindControl("CheckBoxCanView1");

CheckBox canConfirm = (CheckBox)CalcGrid.Rows[i].FindControl("CheckBoxVerify1");

Label label = (Label)CalcGrid.Rows[i].FindControl("Label3");

CalculatedParametrsAndUsersMapping CalcAndUser =

(from a in kpiWebDataContext.CalculatedParametrsAndUsersMapping

where a.Active == true

&& a.FK\_UsersTable == userToChangeId

&& a.FK\_CalculatedParametrsTable == Convert.ToInt32(label.Text)

select a).FirstOrDefault();

if (CalcAndUser != null)

{

CalcAndUser.CanConfirm = canConfirm.Checked;

// BasicAndRole.CanEdit = canEdit.Checked;

CalcAndUser.CanView = canView.Checked;

kpiWebDataContext.SubmitChanges();

}

else if ((canConfirm.Checked) || (canView.Checked))

{

CalcAndUser = new CalculatedParametrsAndUsersMapping();

CalcAndUser.FK\_CalculatedParametrsTable = Convert.ToInt32(label.Text);

CalcAndUser.FK\_UsersTable = userToChangeId;

CalcAndUser.Active = true;

CalcAndUser.CanConfirm = canConfirm.Checked;

//CalcAndRole.CanEdit = canEdit.Checked;

CalcAndUser.CanView = canView.Checked;

kpiWebDataContext.CalculatedParametrsAndUsersMapping.InsertOnSubmit(CalcAndUser);

kpiWebDataContext.SubmitChanges();

}

}

for (int i = 0; i < IndicatorGrid.Rows.Count; i++)

{

//CheckBox canEdit = (CheckBox)CalcGrid.Rows[i].FindControl("CheckBoxCanEdit");

CheckBox canView = (CheckBox)IndicatorGrid.Rows[i].FindControl("CheckBoxCanView2");

CheckBox canConfirm = (CheckBox)IndicatorGrid.Rows[i].FindControl("CheckBoxVerify2");

Label label = (Label)IndicatorGrid.Rows[i].FindControl("Label5");

IndicatorsAndUsersMapping IndAnduser =

(from a in kpiWebDataContext.IndicatorsAndUsersMapping

where a.Active == true

&& a.FK\_UsresTable == userToChangeId

&& a.FK\_IndicatorsTable == Convert.ToInt32(label.Text)

select a).FirstOrDefault();

if (IndAnduser != null)

{

IndAnduser.CanConfirm = canConfirm.Checked;

// BasicAndRole.CanEdit = canEdit.Checked;

IndAnduser.CanView = canView.Checked;

kpiWebDataContext.SubmitChanges();

}

else if ((canConfirm.Checked) || (canView.Checked))

{

IndAnduser = new IndicatorsAndUsersMapping();

IndAnduser.FK\_IndicatorsTable = Convert.ToInt32(label.Text);

IndAnduser.FK\_UsresTable = userToChangeId;

IndAnduser.Active = true;

IndAnduser.CanConfirm = canConfirm.Checked;

//CalcAndRole.CanEdit = canEdit.Checked;

IndAnduser.CanView = canView.Checked;

kpiWebDataContext.IndicatorsAndUsersMapping.InsertOnSubmit(IndAnduser);

kpiWebDataContext.SubmitChanges();

}

}

Response.Redirect("~/AutomationDepartment/EditUser.aspx");

}

/\* protected void Button2\_Click(object sender, EventArgs e)

{

RefreshGridView();

}

protected void Button1\_Click(object sender, EventArgs e)//прост добавляем роль

{/\*

KPIWebDataContext kPiDataContext = new KPIWebDataContext();

RolesTable role = new RolesTable();

if (CheckBox1.Checked) role.Active = true;

else role.Active = false;

if (CheckBox2.Checked) role.IsHead = true;

else role.IsHead = false;

role.RoleName = TextBox1.Text;

kPiDataContext.RolesTable.InsertOnSubmit(role);

kPiDataContext.SubmitChanges();

Page.ClientScript.RegisterClientScriptBlock(typeof(Page), "Script", "alert('Роль добавлена');", true);

// Response.Redirect();\*/

}

/\* protected void DropDownList1\_SelectedIndexChanged(object sender, EventArgs e)

{

GridView1.DataSource = null;

GridView1.DataBind();

CalcGrid.DataSource = null;

CalcGrid.DataBind();

IndicatorGrid.DataSource = null;

IndicatorGrid.DataBind();

}

protected void Button1\_Click(object sender, EventArgs e)

{

KPIWebDataContext kpiWebDataContext = new KPIWebDataContext();

int currentRoleId = 8132

#region

for ( int i = 0; i < GridView1.Rows.Count; i++)

{

CheckBox canEdit = (CheckBox)GridView1.Rows[i].FindControl("CheckBoxCanEdit");

CheckBox canView = (CheckBox)GridView1.Rows[i].FindControl("CheckBoxCanView");

CheckBox canConfirm = (CheckBox)GridView1.Rows[i].FindControl("CheckBoxVerify");

Label label = (Label)GridView1.Rows[i].FindControl("Label2");

BasicParametersAndRolesMappingTable BasicAndRole =

(from a in kpiWebDataContext.BasicParametersAndRolesMappingTable

where a.Active == true

&& a.FK\_RolesTable==currentRoleId

&& a.FK\_BasicParametersTable == Convert.ToInt32(label.Text)

select a).FirstOrDefault();

if (BasicAndRole != null)

{

BasicAndRole.CanConfirm = canConfirm.Checked;

BasicAndRole.CanEdit = canEdit.Checked;

BasicAndRole.CanView = canView.Checked;

kpiWebDataContext.SubmitChanges();

}

else if ((canConfirm.Checked) || (canView.Checked) || (canEdit.Checked))

{

BasicAndRole = new BasicParametersAndRolesMappingTable();

BasicAndRole.FK\_BasicParametersTable = Convert.ToInt32(label.Text);

BasicAndRole.FK\_RolesTable = currentRoleId;

BasicAndRole.Active = true;

BasicAndRole.CanConfirm = canConfirm.Checked;

BasicAndRole.CanEdit = canEdit.Checked;

BasicAndRole.CanView = canView.Checked;

kpiWebDataContext.BasicParametersAndRolesMappingTable.InsertOnSubmit(BasicAndRole);

kpiWebDataContext.SubmitChanges();

}

}

#endregion

RolesTable role = (from a in kpiWebDataContext.RolesTable

where a.RolesTableID == Convert.ToInt32(DropDownList1.Items[DropDownList1.SelectedIndex].Value)

select a).FirstOrDefault();

if ((bool) role.IsHead)

{

#region

for (int i = 0; i < CalcGrid.Rows.Count; i++)

{

//CheckBox canEdit = (CheckBox)CalcGrid.Rows[i].FindControl("CheckBoxCanEdit");

CheckBox canView = (CheckBox)CalcGrid.Rows[i].FindControl("CheckBoxCanView1");

CheckBox canConfirm = (CheckBox)CalcGrid.Rows[i].FindControl("CheckBoxVerify1");

Label label = (Label)CalcGrid.Rows[i].FindControl("Label3");

CalculatedParametrsAndRolesMappingTable CalcAndRole =

(from a in kpiWebDataContext.CalculatedParametrsAndRolesMappingTable

where a.Active == true

&& a.FK\_RolesTable == currentRoleId

&& a.FK\_CalculatedParametrs == Convert.ToInt32(label.Text)

select a).FirstOrDefault();

if (CalcAndRole != null)

{

CalcAndRole.CanConfirm = canConfirm.Checked;

// BasicAndRole.CanEdit = canEdit.Checked;

CalcAndRole.CanView = canView.Checked;

kpiWebDataContext.SubmitChanges();

}

else if ((canConfirm.Checked) || (canView.Checked))

{

CalcAndRole = new CalculatedParametrsAndRolesMappingTable();

CalcAndRole.FK\_CalculatedParametrs = Convert.ToInt32(label.Text);

CalcAndRole.FK\_RolesTable = currentRoleId;

CalcAndRole.Active = true;

CalcAndRole.CanConfirm = canConfirm.Checked;

//CalcAndRole.CanEdit = canEdit.Checked;

CalcAndRole.CanView = canView.Checked;

kpiWebDataContext.CalculatedParametrsAndRolesMappingTable.InsertOnSubmit(CalcAndRole);

kpiWebDataContext.SubmitChanges();

}

}

#endregion

#region

for (int i = 0; i < IndicatorGrid.Rows.Count; i++)

{

//CheckBox canEdit = (CheckBox)CalcGrid.Rows[i].FindControl("CheckBoxCanEdit");

CheckBox canView = (CheckBox)IndicatorGrid.Rows[i].FindControl("CheckBoxCanView2");

CheckBox canConfirm = (CheckBox)IndicatorGrid.Rows[i].FindControl("CheckBoxVerify2");

Label label = (Label)IndicatorGrid.Rows[i].FindControl("Label5");

IndicatorsAndRolesMappingTable IndAndRole =

(from a in kpiWebDataContext.IndicatorsAndRolesMappingTable

where a.Active == true

&& a.FK\_RolesTable == currentRoleId

&& a.FK\_Indicators == Convert.ToInt32(label.Text)

select a).FirstOrDefault();

if (IndAndRole != null)

{

IndAndRole.CanConfirm = canConfirm.Checked;

// BasicAndRole.CanEdit = canEdit.Checked;

IndAndRole.CanView = canView.Checked;

kpiWebDataContext.SubmitChanges();

}

else if ((canConfirm.Checked) || (canView.Checked))

{

IndAndRole = new IndicatorsAndRolesMappingTable();

IndAndRole.FK\_Indicators = Convert.ToInt32(label.Text);

IndAndRole.FK\_RolesTable = currentRoleId;

IndAndRole.Active = true;

IndAndRole.CanConfirm = canConfirm.Checked;

//CalcAndRole.CanEdit = canEdit.Checked;

IndAndRole.CanView = canView.Checked;

kpiWebDataContext.IndicatorsAndRolesMappingTable.InsertOnSubmit(IndAndRole);

kpiWebDataContext.SubmitChanges();

}

}

#endregion

}\*/

//int rowIndex = 0;

//StringCollection sc = new StringCollection();

// if (ViewState["GridviewRoleMapping"] != null)

// {

/\*

int currentRoleId = Convert.ToInt32(DropDownList1.Items[DropDownList1.SelectedIndex].Value);

KPIWebDataContext KPIWebDataContext = new KPIWebDataContext();

Dictionary<int, double> tempDictionary = new Dictionary<int, double>();

DataTable roleBasicParametrs = (DataTable)ViewState["GridviewRoleMapping"];

if (roleBasicParametrs.Rows.Count > 0)

{

for (int i = 1; i <= roleBasicParametrs.Rows.Count; i++)

{

rowIndex++;

}

Page.ClientScript.RegisterClientScriptBlock(typeof(Page), "Script", "alert('Данные успешно сохранены');", true);

// }

\*/

// }

// Response.Redirect("~/AutomationDepartment/AddRole.aspx");

}

//}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Filename: ChangeUser.aspx.designer.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//------------------------------------------------------------------------------

// <автоматически создаваемое>

// Этот код создан программой.

//

// Изменения в этом файле могут привести к неправильной работе и будут потеряны в случае

// повторной генерации кода.

// </автоматически создаваемое>

//------------------------------------------------------------------------------

namespace KPIWeb.AutomationDepartment {

public partial class ChangeUser {

/// <summary>

/// form1 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.HtmlControls.HtmlForm form1;

/// <summary>

/// Button1 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Button Button1;

/// <summary>

/// GridView1 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.GridView GridView1;

/// <summary>

/// CalcGrid элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.GridView CalcGrid;

/// <summary>

/// IndicatorGrid элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.GridView IndicatorGrid;

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Filename: EditUser.aspx.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using System;

using System.Collections.Generic;

using System.Configuration;

using System.Data;

using System.Linq;

using System.Net;

using System.Text.RegularExpressions;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

namespace KPIWeb.AutomationDepartment

{

public partial class EditUser : System.Web.UI.Page

{

protected void Page\_Load(object sender, EventArgs e)

{

Serialization UserSer = (Serialization)Session["UserID"];

if (UserSer == null)

{

Response.Redirect("~/Default.aspx");

}

int userID = UserSer.Id;

KPIWebDataContext kPiDataContext = new KPIWebDataContext();

UsersTable userTable =

(from a in kPiDataContext.UsersTable where a.UsersTableID == userID select a).FirstOrDefault();

ViewState["User"] = userTable.Email;

if ((userTable.AccessLevel != 10)&&(userTable.AccessLevel != 9))

{

Response.Redirect("~/Default.aspx");

}

////////////////////////////////////////////////////////

ViewState["Password"] = "112233";

if (!IsPostBack)

{

RefreshGrid();

}

}

private void RefreshGrid()

{

DataTable dataTable = new DataTable();

dataTable.Columns.Add(new DataColumn("FourthlvlId", typeof(string)));

dataTable.Columns.Add(new DataColumn("Login", typeof(string)));

dataTable.Columns.Add(new DataColumn("Password", typeof(string)));

dataTable.Columns.Add(new DataColumn("Email", typeof(string)));

dataTable.Columns.Add(new DataColumn("Firstlvl", typeof(string)));

dataTable.Columns.Add(new DataColumn("Secondlvl", typeof(string)));

dataTable.Columns.Add(new DataColumn("Thirdlvl", typeof(string)));

dataTable.Columns.Add(new DataColumn("Fourthlvl", typeof(string)));

dataTable.Columns.Add(new DataColumn("Fifthlvl", typeof(string)));

dataTable.Columns.Add(new DataColumn("Acceslvl", typeof(string)));

dataTable.Columns.Add(new DataColumn("Zerolvl", typeof(string)));

dataTable.Columns.Add(new DataColumn("Confirmed", typeof(string)));

dataTable.Columns.Add(new DataColumn("Position", typeof(string)));

dataTable.Columns.Add(new DataColumn("DeleteUserButton", typeof(string)));

dataTable.Columns.Add(new DataColumn("SaveUserButton", typeof(string)));

dataTable.Columns.Add(new DataColumn("ChangeUserButton", typeof(string)));

using (KPIWebDataContext kpiWebDataContext = new KPIWebDataContext())

{

List<UsersTable> users;

if (TextBox2.Text.Any())

{

users = (from a in kpiWebDataContext.UsersTable where a.Active == true && (a.Login.Contains(TextBox2.Text) || a.Email.Contains(TextBox2.Text) || a.Position.Contains(TextBox2.Text)) select a).ToList();

}

else

{

users = (from a in kpiWebDataContext.UsersTable where a.Active == true select a).ToList();

}

List<ZeroLevelSubdivisionTable> zeroLevelSubdivisionTable = (from a in kpiWebDataContext.ZeroLevelSubdivisionTable select a).ToList();

List<FirstLevelSubdivisionTable> firstLevelSubdivisionTable = (from a in kpiWebDataContext.FirstLevelSubdivisionTable select a).ToList();

List<SecondLevelSubdivisionTable> secondLevelSubdivisionTable = (from a in kpiWebDataContext.SecondLevelSubdivisionTable select a).ToList();

List<ThirdLevelSubdivisionTable> thirdLevelSubdivisionTable = (from a in kpiWebDataContext.ThirdLevelSubdivisionTable select a).ToList();

List<FourthLevelSubdivisionTable> fourthLevelSubdivisionTable = (from a in kpiWebDataContext.FourthLevelSubdivisionTable select a).ToList();

List<FifthLevelSubdivisionTable> fifthLevelSubdivisionTable = (from a in kpiWebDataContext.FifthLevelSubdivisionTable select a).ToList();

foreach (var user in users)

{

DataRow dataRow = dataTable.NewRow();

dataRow["FourthlvlId"] = user.UsersTableID;

dataRow["Login"] = user.Login;

// if (CheckBox1.Checked && TextBox1.Text == ViewState["Password"].ToString())

// {

dataRow["Password"] = user.Password;

dataRow["Email"] = user.Email;

// }

//else

// {

// dataRow["Password"] = "\*\*\*\*\*\*\*\*";

// dataRow["Email"] = "\*\*\*\*\*\*\*\*";

// }

dataRow["ChangeUserButton"] = user.UsersTableID;

dataRow["Firstlvl"] = (from a in firstLevelSubdivisionTable

where a.FirstLevelSubdivisionTableID == user.FK\_FirstLevelSubdivisionTable

select a.Name).FirstOrDefault();

dataRow["Secondlvl"] = (from a in secondLevelSubdivisionTable

where a.SecondLevelSubdivisionTableID == user.FK\_SecondLevelSubdivisionTable

select a.Name).FirstOrDefault();

dataRow["Thirdlvl"] = (from a in thirdLevelSubdivisionTable

where a.ThirdLevelSubdivisionTableID == user.FK\_ThirdLevelSubdivisionTable

select a.Name).FirstOrDefault();

dataRow["Fourthlvl"] = (from a in fourthLevelSubdivisionTable

where a.FourthLevelSubdivisionTableID == user.FK\_FourthLevelSubdivisionTable

select a.Name).FirstOrDefault();

dataRow["Fifthlvl"] = (from a in fifthLevelSubdivisionTable

where a.FifthLevelSubdivisionTableID == user.FK\_FirstLevelSubdivisionTable

select a.Name).FirstOrDefault();

dataRow["Acceslvl"] = user.AccessLevel;

dataRow["Zerolvl"] = (from a in zeroLevelSubdivisionTable

where a.ZeroLevelSubdivisionTableID == user.FK\_ZeroLevelSubdivisionTable

select a.Name).FirstOrDefault();

dataRow["Confirmed"] = user.Confirmed;

dataRow["Position"] = user.Position;

dataTable.Rows.Add(dataRow);

}

ViewState["GridviewUsers"] = dataTable;

GridView1.DataSource = dataTable;

GridView1.DataBind();

}

}

protected void CheckBox1\_CheckedChanged(object sender, EventArgs e)

{

RefreshGrid();

}

protected void DeleteUserButtonClick(object sender, EventArgs e)

{

// if (TextBox1.Text == ViewState["Password"].ToString())

// {

if (!CheckBox2.Checked)

{

Button button = (Button) sender;

{

using (KPIWebDataContext kPiDataContext = new KPIWebDataContext())

{

UsersTable user =

(from a in kPiDataContext.UsersTable

where a.UsersTableID == Convert.ToInt32(button.CommandArgument)

select a).FirstOrDefault();

user.Active = false;

kPiDataContext.SubmitChanges();

LogHandler.LogWriter.WriteLog(LogCategory.INFO, "0EU2: AdminUser " + ViewState["User"] + "DELETE user: "+ user.Email + " from ip: "+Dns.GetHostEntry(Dns.GetHostName()).AddressList.Where(ip => ip.AddressFamily == System.Net.Sockets.AddressFamily.InterNetwork).Select(ip => ip.ToString()).FirstOrDefault());

}

RefreshGrid();

}

}

else

{

DisplayAlert("Снимите предохранитель");

}

// }

// else

// Page.ClientScript.RegisterClientScriptBlock(typeof(Page), "Script",

// "alert('Введите пароль');", true);

}

protected void SaveUserButtonClick(object sender, EventArgs e)

{

//if (TextBox1.Text == ViewState["Password"].ToString())

//{

int rowIndex = 0;

Button button = (Button)sender;

{

if (ViewState["GridviewUsers"] != null)

{

DataTable dataTable = (DataTable)ViewState["GridviewUsers"];

if (dataTable.Rows.Count > 0)

{

for (int i = 1; i <= dataTable.Rows.Count; i++)

{

Label TextBoxFourthlvlId = (Label)GridView1.Rows[rowIndex].FindControl("FourthlvlId");

if (TextBoxFourthlvlId.Text.Equals(button.CommandArgument.ToString()))

{

TextBox TextBoxLogin = (TextBox)GridView1.Rows[rowIndex].FindControl("Login");

TextBox TextBoxPassword = (TextBox)GridView1.Rows[rowIndex].FindControl("Password");

TextBox TextBoxEmail = (TextBox)GridView1.Rows[rowIndex].FindControl("Email");

TextBox TextBoxFirstlvl = (TextBox)GridView1.Rows[rowIndex].FindControl("Firstlvl");

TextBox TextBoxSecondlvl =

(TextBox)GridView1.Rows[rowIndex].FindControl("Secondlvl");

TextBox TextBoxThirdlvl = (TextBox)GridView1.Rows[rowIndex].FindControl("Thirdlvl");

TextBox TextBoxFourthlvl =

(TextBox)GridView1.Rows[rowIndex].FindControl("Fourthlvl");

TextBox TextBoxFifthlvl = (TextBox)GridView1.Rows[rowIndex].FindControl("Fifthlvl");

TextBox TextBoxAcceslvl = (TextBox)GridView1.Rows[rowIndex].FindControl("Acceslvl");

TextBox TextBoxZerolvl = (TextBox)GridView1.Rows[rowIndex].FindControl("Zerolvl");

TextBox TextBoxConfirmed = (TextBox)GridView1.Rows[rowIndex].FindControl("Confirmed");

TextBox TextBoxPosition = (TextBox)GridView1.Rows[rowIndex].FindControl("Position");

using (KPIWebDataContext kPiDataContext = new KPIWebDataContext())

{

UsersTable user =

(from a in kPiDataContext.UsersTable

where a.UsersTableID == Convert.ToInt32(button.CommandArgument)

select a).FirstOrDefault();

user.Login = TextBoxLogin.Text;

if (!TextBoxPassword.Text.Equals("\*\*\*\*\*\*\*\*") || (!Regex.IsMatch(TextBoxPassword.Text, @"/\*")) || (!TextBoxEmail.Text.Equals("\*\*\*\*\*\*\*\*") || (!Regex.IsMatch(TextBoxEmail.Text, @"/\*"))))

{

user.Password = TextBoxPassword.Text;

user.Email = TextBoxEmail.Text;

}

if (TextBoxFirstlvl.Text.Any())

user.FK\_FirstLevelSubdivisionTable = (from a in kPiDataContext.UsersTable

where a.UsersTableID == Convert.ToInt32(TextBoxFourthlvlId.Text)

select a.FK\_FirstLevelSubdivisionTable).FirstOrDefault();

if (TextBoxSecondlvl.Text.Any())

user.FK\_SecondLevelSubdivisionTable = (from a in kPiDataContext.UsersTable

where a.UsersTableID == Convert.ToInt32(TextBoxFourthlvlId.Text)

select a.FK\_SecondLevelSubdivisionTable).FirstOrDefault();

if (TextBoxThirdlvl.Text.Any())

user.FK\_ThirdLevelSubdivisionTable = (from a in kPiDataContext.UsersTable

where a.UsersTableID == Convert.ToInt32(TextBoxFourthlvlId.Text)

select a.FK\_ThirdLevelSubdivisionTable).FirstOrDefault();

if (TextBoxFourthlvl.Text.Any())

user.FK\_FourthLevelSubdivisionTable = (from a in kPiDataContext.UsersTable

where a.UsersTableID == Convert.ToInt32(TextBoxFourthlvlId.Text)

select a.FK\_FourthLevelSubdivisionTable).FirstOrDefault();

if (TextBoxFifthlvl.Text.Any())

user.FK\_FifthLevelSubdivisionTable = (from a in kPiDataContext.UsersTable

where a.UsersTableID == Convert.ToInt32(TextBoxFourthlvlId.Text)

select a.FK\_FifthLevelSubdivisionTable).FirstOrDefault();

if (TextBoxAcceslvl.Text.Any())

user.AccessLevel = Convert.ToInt32(TextBoxAcceslvl.Text);

if (TextBoxZerolvl.Text.Any())

user.FK\_ZeroLevelSubdivisionTable = (from a in kPiDataContext.UsersTable

where a.UsersTableID == Convert.ToInt32(TextBoxFourthlvlId.Text)

select a.FK\_ZeroLevelSubdivisionTable).FirstOrDefault();

if (TextBoxPosition.Text.Any())

user.Position = Convert.ToString(TextBoxPosition.Text);

kPiDataContext.SubmitChanges();

LogHandler.LogWriter.WriteLog(LogCategory.INFO, "0EU3: AdminUser " + ViewState["User"] + " SAVE data of user: " + user.Email + " from ip: " + Dns.GetHostEntry(Dns.GetHostName()).AddressList.Where(ip => ip.AddressFamily == System.Net.Sockets.AddressFamily.InterNetwork).Select(ip => ip.ToString()).FirstOrDefault());

}

}

rowIndex++;

}

}

}

}

// }

// else Page.ClientScript.RegisterClientScriptBlock(typeof(Page), "Script",

// "alert('Введите пароль');", true);

}

protected void Button1\_Click(object sender, EventArgs e)

{

RefreshGrid();

}

protected void CheckBox1\_CheckedChanged1(object sender, EventArgs e)

{

if (!TextBox1.Text.Any()) CheckBox1.Checked = false;

RefreshGrid();

}

protected void Button2\_Click(object sender, EventArgs e)

{

LogHandler.LogWriter.WriteLog(LogCategory.INFO, "0EU1: User " + ViewState["User"] + " moved to page /AutomationDepartment/Regisration.aspx");

Response.Redirect("~/AutomationDepartment/Regisration.aspx");

}

protected void ChangeUserButtonClick (object sender, EventArgs e)

{

Button button = (Button)sender;

{

Serialization ser = new Serialization(Convert.ToInt32(button.CommandArgument));

Session["userIdforChange"] = ser;

LogHandler.LogWriter.WriteLog(LogCategory.INFO, "0EU4: AdminUser " + ViewState["User"] + "moved to page /AutomationDepartment/ChangeUser.aspx" + "from ip: "+ Dns.GetHostEntry(Dns.GetHostName()).AddressList.Where(ip => ip.AddressFamily == System.Net.Sockets.AddressFamily.InterNetwork).Select(ip => ip.ToString()).FirstOrDefault());

Response.Redirect("~/AutomationDepartment/ChangeUser.aspx");

}

}

protected void GridView1\_SelectedIndexChanged(object sender, EventArgs e)

{

}

protected void Button3\_Click(object sender, EventArgs e)

{

if (!CheckBox2.Checked)

{/\*

LogHandler.LogWriter.WriteLog(LogCategory.INFO, "0EU0: MassMailing was started by: " + ViewState["User"]);

KPIWebDataContext kPiDataContext = new KPIWebDataContext();

var users =

(from a in kPiDataContext.UsersTable where a.Active && a.Confirmed == false select a).ToList();

EmailTemplate EmailParams = (from a in kPiDataContext.EmailTemplate

where a.Name == "InviteToRegister"

&& a.Active == true

select a).FirstOrDefault();

foreach (var user in users)

{

Action.MassMailing(user.Email, EmailParams.EmailTitle,

EmailParams.EmailContent.Replace("#LINK#",

ConfigurationManager.AppSettings.Get("SiteName") + "/Account/UserRegister?&id=" +

user.PassCode), null);

}\*/

Response.Redirect("~/AutomationDepartment/SendInvite.aspx");

}

else

{

DisplayAlert("Снимите предохранитель");

}

}

private void DisplayAlert(string message)

{

ClientScript.RegisterStartupScript(

this.GetType(),

Guid.NewGuid().ToString(),

string.Format("alert('{0}');",

message.Replace("'", @"\'").Replace("\n", "\\n").Replace("\r", "\\r")),

true);

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Filename: EditUser.aspx.designer.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//------------------------------------------------------------------------------

// <auto-generated>

// This code was generated by a tool.

//

// Changes to this file may cause incorrect behavior and will be lost if

// the code is regenerated.

// </auto-generated>

//------------------------------------------------------------------------------

namespace KPIWeb.AutomationDepartment {

public partial class EditUser {

/// <summary>

/// Button2 control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.Button Button2;

/// <summary>

/// Label1 control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.Label Label1;

/// <summary>

/// TextBox1 control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.TextBox TextBox1;

/// <summary>

/// Label2 control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.Label Label2;

/// <summary>

/// TextBox2 control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.TextBox TextBox2;

/// <summary>

/// Button1 control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.Button Button1;

/// <summary>

/// CheckBox1 control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.CheckBox CheckBox1;

/// <summary>

/// GridView1 control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.GridView GridView1;

/// <summary>

/// Button3 control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.Button Button3;

/// <summary>

/// CheckBox2 control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.CheckBox CheckBox2;

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Filename: IsAvailibleMappingRole.aspx.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using System;

using System.Collections.Generic;

using System.Configuration;

using System.Data;

using System.Linq;

using System.Reflection;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

namespace KPIWeb

{

public partial class WebForm2 : System.Web.UI.Page

{

protected void Page\_Load(object sender, EventArgs e)

{

Serialization UserSer = (Serialization)Session["UserID"];

if (UserSer == null)

{

Response.Redirect("~/Default.aspx");

}

int userID = UserSer.Id;

KPIWebDataContext kPiDataContext = new KPIWebDataContext();

UsersTable userTable =

(from a in kPiDataContext.UsersTable where a.UsersTableID == userID select a).FirstOrDefault();

if (userTable.AccessLevel != 10)

{

Response.Redirect("~/Default.aspx");

}

///////////////////////////////////////////////////////////////////////////////////////////////////

KPIWebDataContext kpiWebDataContext = new KPIWebDataContext();

var vrCountry = (from a in kpiWebDataContext.BasicParametersTable

select a).Except(from b in kpiWebDataContext.BasicParametersTable

join c in kpiWebDataContext.BasicParametersAndRolesMappingTable on b.BasicParametersTableID

equals c.FK\_BasicParametersTable select b).ToList();

Dictionary<string, int> tempDictionary = new Dictionary<string, int>();

foreach (var obj in vrCountry)

{

tempDictionary.Add(obj.Name,obj.BasicParametersTableID);

}

if (tempDictionary.Count == 0)

{

Page.ClientScript.RegisterClientScriptBlock(typeof (Page), "Script","alert('Все базовые параметры распределены');", true);

}

else

{

DataTable dt = LINQToDataTable(tempDictionary);

GridView1.DataSource = dt;

GridView1.DataBind();

}

}

public DataTable LINQToDataTable<T>(IEnumerable<T> varlist)

{

DataTable dtReturn = new DataTable();

PropertyInfo[] oProps = null;

if (varlist == null) return dtReturn;

foreach (T rec in varlist)

{

if (oProps == null)

{

oProps = ((Type)rec.GetType()).GetProperties();

foreach (PropertyInfo pi in oProps)

{

Type colType = pi.PropertyType;

if ((colType.IsGenericType) && (colType.GetGenericTypeDefinition()

==typeof(Nullable<>)))

{

colType = colType.GetGenericArguments()[0];

}

dtReturn.Columns.Add(new DataColumn(pi.Name, colType));

}

}

DataRow dr = dtReturn.NewRow();

foreach (PropertyInfo pi in oProps)

{

dr[pi.Name] = pi.GetValue(rec, null) == null ?DBNull.Value :pi.GetValue

(rec,null);

}

dtReturn.Rows.Add(dr);

}

return dtReturn;

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Filename: IsAvailibleMappingRole.aspx.designer.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//------------------------------------------------------------------------------

// <автоматически создаваемое>

// Этот код создан программой.

//

// Изменения в этом файле могут привести к неправильной работе и будут потеряны в случае

// повторной генерации кода.

// </автоматически создаваемое>

//------------------------------------------------------------------------------

namespace KPIWeb {

public partial class WebForm2 {

/// <summary>

/// GridView1 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.GridView GridView1;

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Filename: Main.aspx.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using System;

using System.Collections.Generic;

using System.Configuration;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

using KPIWeb.Rector;

namespace KPIWeb.AutomationDepartment

{

public partial class Main : System.Web.UI.Page

{

protected void Page\_Load(object sender, EventArgs e)

{

Serialization UserSer = (Serialization)Session["UserID"];

if (UserSer == null)

{

Response.Redirect("~/Default.aspx");

}

int userID = UserSer.Id;

KPIWebDataContext kPiDataContext = new KPIWebDataContext();

UsersTable userTable =

(from a in kPiDataContext.UsersTable where a.UsersTableID == userID select a).FirstOrDefault();

if (userTable.AccessLevel != 10)

{

Response.Redirect("~/Default.aspx");

}

}

protected void Button1\_Click(object sender, EventArgs e)

{

Response.Redirect("~/AutomationDepartment/AddLevel.aspx");

}

protected void Button2\_Click(object sender, EventArgs e)

{

Response.Redirect("~/AutomationDepartment/Regisration.aspx");

}

protected void Button3\_Click(object sender, EventArgs e)

{

Response.Redirect("~/AutomationDepartment/RoleMapping.aspx");

}

protected void Button4\_Click(object sender, EventArgs e)

{

Response.Redirect("~/AutomationDepartment/IsAvailibleMappingRole.aspx");

}

protected void Button5\_Click(object sender, EventArgs e)

{

Response.Redirect("~/StatisticsDepartment/Indicators.aspx");

}

protected void Button6\_Click(object sender, EventArgs e)

{

Response.Redirect("~/StatisticsDepartment/BasicParametrs.aspx");

}

protected void Button7\_Click(object sender, EventArgs e)

{

Response.Redirect("~/StatisticsDepartment/ReportViewer.aspx");

}

protected void Button8\_Click(object sender, EventArgs e)

{

Response.Redirect("~/AutomationDepartment/AddRole.aspx");

}

protected void Button9\_Click(object sender, EventArgs e)

{

Response.Redirect("~/AutomationDepartment/AddSpecialization.aspx");

}

protected void Button10\_Click(object sender, EventArgs e)

{

Response.Redirect("~/AutomationDepartment/EditUser.aspx");

}

protected void Button11\_Click(object sender, EventArgs e)

{

Response.Redirect("~/Account/Register\_.aspx");

}

protected void Button11\_Click1(object sender, EventArgs e)

{

ForRCalc.Struct mainStruct = new ForRCalc.Struct(1, "");

RectorSession rectorResultSession = new RectorSession (mainStruct, 1, 0, 0, 4009,0);

Session["rectorResultSession"] = rectorResultSession;

RectorHistorySession RectorHistory = new RectorHistorySession();

RectorHistory.SessionCount = 1;

RectorHistory.CurrentSession = 0;

RectorHistory.RectorSession[RectorHistory.CurrentSession] = rectorResultSession;

Session["rectorHistory"] = RectorHistory;

Response.Redirect("~/Rector/Result.aspx");

}

protected void Button12\_Click(object sender, EventArgs e)

{

Response.Redirect("~/PlannedIndicator.aspx");

}

protected void Button13\_Click(object sender, EventArgs e)

{

Response.Redirect("~/AutomationDepartment/WatchingParamtrsState.aspx");

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Filename: Main.aspx.designer.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//------------------------------------------------------------------------------

// <автоматически создаваемое>

// Этот код создан программой.

//

// Изменения в этом файле могут привести к неправильной работе и будут потеряны в случае

// повторной генерации кода.

// </автоматически создаваемое>

//------------------------------------------------------------------------------

namespace KPIWeb.AutomationDepartment {

public partial class Main {

/// <summary>

/// Button1 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Button Button1;

/// <summary>

/// Button2 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Button Button2;

/// <summary>

/// Button3 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Button Button3;

/// <summary>

/// Button4 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Button Button4;

/// <summary>

/// Button5 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Button Button5;

/// <summary>

/// Button6 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Button Button6;

/// <summary>

/// Button7 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Button Button7;

/// <summary>

/// Button8 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Button Button8;

/// <summary>

/// Button9 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Button Button9;

/// <summary>

/// Button10 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Button Button10;

/// <summary>

/// Button12 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Button Button12;

/// <summary>

/// Button11 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Button Button11;

/// <summary>

/// Button13 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Button Button13;

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Filename: Regisration.aspx.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using System;

using System.Collections.Generic;

using System.Configuration;

using System.Data;

using System.Linq;

using System.Net;

using System.Net.Mail;

using System.Text;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

namespace KPIWeb.AutomationDepartment

{

public partial class Regisration : System.Web.UI.Page

{

private string RandomString(int size)

{

StringBuilder builder = new StringBuilder();

Random random = new Random();

char ch;

for (int i = 0; i < size; i++)

{

ch = Convert.ToChar(Convert.ToInt32(Math.Floor(26 \* random.NextDouble() + 65)));

builder.Append(ch);

}

return builder.ToString();

}

protected void FillGridVIews(int reportID\_)

{

KPIWebDataContext kPiDataContext = new KPIWebDataContext();

///////////////////////////////////////////////////////////////////////////////////////////////////////

List<IndicatorsTable> indicatorTable =

(from item in kPiDataContext.IndicatorsTable where item.Active == true select item).OrderBy(c => c.SortID).ToList();

DataTable dataTableIndicator = new DataTable();

dataTableIndicator.Columns.Add(new DataColumn("IndicatorID", typeof(string)));

dataTableIndicator.Columns.Add(new DataColumn("IndicatorName", typeof(string)));

dataTableIndicator.Columns.Add(new DataColumn("IndicatorEditCheckBox", typeof(bool)));

dataTableIndicator.Columns.Add(new DataColumn("IndicatorViewCheckBox", typeof(bool)));

dataTableIndicator.Columns.Add(new DataColumn("IndicatorConfirmCheckBox", typeof(bool)));

foreach (IndicatorsTable indicator in indicatorTable)

{

DataRow dataRow = dataTableIndicator.NewRow();

dataRow["IndicatorID"] = indicator.IndicatorsTableID.ToString();

dataRow["IndicatorName"] = indicator.Name;

dataRow["IndicatorEditCheckBox"] = false;

dataRow["IndicatorViewCheckBox"] = false;

dataRow["IndicatorConfirmCheckBox"] = false;

/\*dataRow["IndicatorCheckBox"] = ((from a in kPiDataContext.ReportArchiveAndIndicatorsMappingTable

where a.Active == true

&& a.FK\_IndicatorsTable == indicator.IndicatorsTableID

&& a.FK\_ReportArchiveTable == reportID

select a).Count() > 0) ? true : false;\*/

dataTableIndicator.Rows.Add(dataRow);

}

Gridview1.DataSource = dataTableIndicator;

Gridview1.DataBind();

////////////////////////////////////////////////////////////////////////////////////////////////////////

List<CalculatedParametrs> calcParams =

(from item in kPiDataContext.CalculatedParametrs where item.Active == true select item).ToList();

DataTable dataTableCalc = new DataTable();

dataTableCalc.Columns.Add(new DataColumn("CalculatedParametrsID", typeof(string)));

dataTableCalc.Columns.Add(new DataColumn("CalculatedParametrsName", typeof(string)));

dataTableCalc.Columns.Add(new DataColumn("CalculatedParametrsEditCheckBox", typeof(bool)));

dataTableCalc.Columns.Add(new DataColumn("CalculatedParametrsViewCheckBox", typeof(bool)));

dataTableCalc.Columns.Add(new DataColumn("CalculatedParametrsConfirmCheckBox", typeof(bool)));

foreach (CalculatedParametrs calcParam in calcParams)

{

DataRow dataRow = dataTableCalc.NewRow();

dataRow["CalculatedParametrsID"] = calcParam.CalculatedParametrsID.ToString();

dataRow["CalculatedParametrsName"] = calcParam.Name;

dataRow["CalculatedParametrsEditCheckBox"] = false;

dataRow["CalculatedParametrsViewCheckBox"] = false;

dataRow["CalculatedParametrsConfirmCheckBox"] = false;

/\*dataRow["CalculatedParametrsCheckBox"] = ((from a in kPiDataContext.ReportArchiveAndCalculatedParametrsMappingTable

where a.Active == true

&& a.FK\_CalculatedParametrsTable == calcParam.CalculatedParametrsID

&& a.FK\_ReportArchiveTable == reportID

select a).Count() > 0) ? true : false;\*/

dataTableCalc.Rows.Add(dataRow);

}

Gridview2.DataSource = dataTableCalc;

Gridview2.DataBind();

////////////////////////////////////////////////////////////////////////////////////////////////////////

List<BasicParametersTable> basicParams =

(from item in kPiDataContext.BasicParametersTable where item.Active == true select item).ToList();

DataTable dataTableBasic = new DataTable();

dataTableBasic.Columns.Add(new DataColumn("BasicParametrsID", typeof(string)));

dataTableBasic.Columns.Add(new DataColumn("BasicParametrsName", typeof(string)));

dataTableBasic.Columns.Add(new DataColumn("BasicParametrsEditCheckBox", typeof(bool)));

dataTableBasic.Columns.Add(new DataColumn("BasicParametrsViewCheckBox", typeof(bool)));

dataTableBasic.Columns.Add(new DataColumn("BasicParametrsConfirmCheckBox", typeof(bool)));

foreach (BasicParametersTable basic in basicParams)

{

DataRow dataRow = dataTableBasic.NewRow();

dataRow["BasicParametrsID"] = basic.BasicParametersTableID.ToString();

dataRow["BasicParametrsName"] = basic.Name;

dataRow["BasicParametrsEditCheckBox"] = false;

dataRow["BasicParametrsViewCheckBox"] = false;

dataRow["BasicParametrsConfirmCheckBox"] = false;

/\*dataRow["BasicParametrsCheckBox"] = ((from a in kPiDataContext.ReportArchiveAndBasicParametrsMappingTable

where a.Active == true

&& a.FK\_BasicParametrsTable == basic.BasicParametersTableID

&& a.FK\_ReportArchiveTable == reportID

select a).Count() > 0) ? true : false;\*/

dataTableBasic.Rows.Add(dataRow);

}

Gridview3.DataSource = dataTableBasic;

Gridview3.DataBind();

///////////////////////////////////////////////////////////////////////////////////////////////////

ViewState["BasicDataTable"] = dataTableBasic;

ViewState["CalculateDataTable"] = dataTableCalc;

ViewState["IndicatorDataTable"] = dataTableIndicator;

}

protected void DropDownList1\_SelectedIndexChanged(object sender, EventArgs e)

{

DropDownList2.Items.Clear();

DropDownList3.Items.Clear();

KPIWebDataContext kPiDataContext = new KPIWebDataContext();

int SelectedValue = -1;

if (int.TryParse(DropDownList1.SelectedValue, out SelectedValue) && SelectedValue != -1)

{

List<SecondLevelSubdivisionTable> second\_stageList =

(from item in kPiDataContext.SecondLevelSubdivisionTable

where item.FK\_FirstLevelSubdivisionTable == SelectedValue

select item).OrderBy(mc => mc.SecondLevelSubdivisionTableID).ToList();

if (second\_stageList != null && second\_stageList.Count() > 0)

{

var dictionary = new Dictionary<int, string>();

dictionary.Add(-1, "Выберите значение");

foreach (var item in second\_stageList)

dictionary.Add(item.SecondLevelSubdivisionTableID, item.Name);

DropDownList2.DataTextField = "Value";

DropDownList2.DataValueField = "Key";

DropDownList2.DataSource = dictionary;

DropDownList2.DataBind();

}

else

{

//Page.ClientScript.RegisterClientScriptBlock(typeof(Page), "Script", "alert('Произошла ошибка.');", true);

}

}

else

{

Page.ClientScript.RegisterClientScriptBlock(typeof(Page), "Script", "alert('Произошла ошибка.');", true);

}

}

protected void Page\_Load(object sender, EventArgs e)

{

Serialization UserSer = (Serialization)Session["UserID"];

if (UserSer == null)

{

Response.Redirect("~/Account/Login.aspx");

}

int userID = UserSer.Id;

KPIWebDataContext kPiDataContext = new KPIWebDataContext();

UsersTable userTable =

(from a in kPiDataContext.UsersTable where a.UsersTableID == userID select a).FirstOrDefault();

if ((userTable.AccessLevel != 10) && (userTable.AccessLevel != 9))

{

Response.Redirect("~/Account/Login.aspx");

}

///////////////////////////////////////////////////проверили на админа

if (!Page.IsPostBack)

{

ViewState["Login"] = userTable.Email;

Gridview1.Visible = false;

Label25.Visible = false;

Gridview2.Visible = false;

Label26.Visible = false;

Gridview3.Visible = false;

Label27.Visible = false;

DropDownList4.Visible = false;

Label24.Visible = false;

List<FirstLevelSubdivisionTable> First\_stageList =

(from item in kPiDataContext.FirstLevelSubdivisionTable

select item).OrderBy(mc => mc.Name).ToList();

var dictionary = new Dictionary<int, string>();

dictionary.Add(0, "Выберите значение");

foreach (var item in First\_stageList)

dictionary.Add(item.FirstLevelSubdivisionTableID, item.Name);

FillGridVIews(0);

DropDownList1.DataTextField = "Value";

DropDownList1.DataValueField = "Key";

DropDownList1.DataSource = dictionary;

DropDownList1.DataBind();

/// записали академии в дроп даун

///

/// в зависимости от того кто вошел изменяем интерфейс

if (userTable.AccessLevel == 9)

{

UserNameText.Enabled = false;

UserNameLabel.Enabled = false;

errorNoName.Enabled = false;

UserNameText.Visible = false;

UserNameLabel.Visible = false;

errorNoName.Visible = false;

PasswordText.Enabled = false;

PassLabel.Enabled = false;

errorNoPass.Enabled = false;

PasswordText.Visible = false;

PassLabel.Visible = false;

errorNoPass.Visible = false;

ConfirmPasswordText.Enabled = false;

ConfPassLabel.Enabled = false;

errorNoConfirm.Enabled = false;

ErrorWrongConfirm.Enabled = false;

ConfirmPasswordText.Visible = false;

ConfPassLabel.Visible = false;

errorNoConfirm.Visible = false;

ErrorWrongConfirm.Visible = false;

}

}

}

protected void DropDownList4\_SelectedIndexChanged(object sender, EventArgs e)

{

}

protected void CheckBox1\_CheckedChanged(object sender, EventArgs e)

{

// if

}

protected void DropDownList5\_SelectedIndexChanged(object sender, EventArgs e)

{

Serialization UserSer = (Serialization)Session["UserID"];

if (UserSer == null)

{

Response.Redirect("~/Account/Login.aspx");

}

int userID = UserSer.Id;

KPIWebDataContext kPiDataContext = new KPIWebDataContext();

UsersTable userTable =

(from a in kPiDataContext.UsersTable where a.UsersTableID == userID select a).FirstOrDefault();

Gridview1.Visible = false;

Label25.Visible = false;

Gridview2.Visible = false;

Label26.Visible = false;

Gridview3.Visible = false;

Label27.Visible = false;

DropDownList4.Visible = false;

Label24.Visible = false;

if (DropDownList5.SelectedIndex == 0 || DropDownList5.SelectedIndex == 3)

{

}

else if (DropDownList5.SelectedIndex == 5)

{

/\* Gridview1.Visible = true;

Label25.Visible = true;

Gridview2.Visible = true;

Label26.Visible = true;

Gridview3.Visible = true;

Label27.Visible = true;

DropDownList4.Visible = true;

Label24.Visible = true;\*/

}

else if (DropDownList5.SelectedIndex == 1)

{

List<RolesTable> Roles = (from a in kPiDataContext.RolesTable

where a.Active == true

&& a.IsHead == false

select a).ToList();

int i = 1;

DropDownList4.Items.Clear();

DropDownList4.Items.Add("Выберите шаблон");

foreach (RolesTable role in Roles)

{

DropDownList4.Items.Add(role.RoleName);

DropDownList4.Items[i].Value = role.RolesTableID.ToString();

i++;

}

DropDownList4.Visible = true;

Label24.Visible = true;

//////

if (userTable.AccessLevel == 10)

{

Gridview3.Visible = true;

Label27.Visible = true;

}

else if (userTable.AccessLevel == 9)

{

}

}

else if (DropDownList5.SelectedIndex == 2)

{

Gridview1.Visible = true;

Label25.Visible = true;

Gridview2.Visible = true;

Label26.Visible = true;

Gridview3.Visible = true;

Label27.Visible = true;

Gridview3.Columns[1].Visible = false;

Gridview3.Columns[3].Visible = false;

List<RolesTable> Roles = (from a in kPiDataContext.RolesTable

where a.Active == true

&& a.IsHead == true

select a).ToList();

int i = 1;

DropDownList4.Items.Clear();

DropDownList4.Items.Add("Выберите шаблон");

foreach (RolesTable role in Roles)

{

DropDownList4.Items.Add(role.RoleName);

DropDownList4.Items[i].Value = role.RolesTableID.ToString();

i++;

}

DropDownList4.Visible = true;

Label24.Visible = true;

////записали роли в дроп даун

}

}

protected void Gridview3\_SelectedIndexChanged(object sender, EventArgs e)

{

}

protected void DropDownList2\_SelectedIndexChanged1(object sender, EventArgs e)

{

DropDownList3.Items.Clear();

KPIWebDataContext kPiDataContext = new KPIWebDataContext();

int SelectedValue = -1;

if (int.TryParse(DropDownList2.SelectedValue, out SelectedValue) && SelectedValue != -1)

{

List<ThirdLevelSubdivisionTable> third\_stage = (from item in kPiDataContext.ThirdLevelSubdivisionTable

where item.FK\_SecondLevelSubdivisionTable == SelectedValue

select item).OrderBy(mc => mc.ThirdLevelSubdivisionTableID).ToList();

if (third\_stage != null && third\_stage.Count() > 0)

{

var dictionary = new Dictionary<int, string>();

dictionary.Add(-1, "Выберите значение");

foreach (var item in third\_stage)

dictionary.Add(item.ThirdLevelSubdivisionTableID, item.Name);

DropDownList3.DataTextField = "Value";

DropDownList3.DataValueField = "Key";

DropDownList3.DataSource = dictionary;

DropDownList3.DataBind();

}

else

{

//Page.ClientScript.RegisterClientScriptBlock(typeof(Page), "Script", "alert('Произошла ошибка.');", true);

}

}

else

{

Page.ClientScript.RegisterClientScriptBlock(typeof(Page), "Script", "alert('Произошла ошибка.');", true);

}

}

protected void Button1\_Click(object sender, EventArgs e)

{

KPIWebDataContext kPiDataContext = new KPIWebDataContext();

if (((from a in kPiDataContext.UsersTable where a.Login == UserNameText.Text select a).ToList().Count > 0)&&(UserNameText.Text.Length>2))

{

Page.ClientScript.RegisterClientScriptBlock(typeof(Page), "Script", "alert('Пользователь с таким логином уже существует, выберите другой логин!');", true);

}

else if ((from a in kPiDataContext.UsersTable where a.Email == EmailText.Text select a).ToList().Count > 0)

{

Page.ClientScript.RegisterClientScriptBlock(typeof (Page), "Script",

"alert('Введенный адрес электронной почты уже зарегестрирован, введите другой');", true);

}

else

{

#region check for user with the same basic parametrs allouence

int match\_cnt\_sum = 0;

int rowIndex = 0;

if (Gridview3.Rows.Count > 0)

{

for (int k = 1; k <= Gridview3.Rows.Count; k++)

{

CheckBox canEdit = (CheckBox) Gridview3.Rows[rowIndex].FindControl("BasicParametrsEditCheckBox");

CheckBox canConfirm = (CheckBox) Gridview3.Rows[rowIndex].FindControl("BasicParametrsConfirmCheckBox");

Label label = (Label) Gridview3.Rows[rowIndex].FindControl("BasicParametrsID");

//BasicParametrsAndUsersMapping BasicAndUsers = new BasicParametrsAndUsersMapping();

int first = 0;

int second = 0;

int third = 0;

if ((DropDownList1.SelectedIndex!=null)&&(DropDownList1.SelectedIndex > 0))

{

first = Convert.ToInt32(DropDownList1.Items[DropDownList1.SelectedIndex].Value);

}

if ((DropDownList2.SelectedIndex!=null)&&(DropDownList2.SelectedIndex > 0))

{

second = Convert.ToInt32(DropDownList2.Items[DropDownList2.SelectedIndex].Value);

}

if ((DropDownList3.SelectedIndex!=null)&&(DropDownList3.SelectedIndex > 0))

{

third = Convert.ToInt32(DropDownList3.Items[DropDownList3.SelectedIndex].Value);

}

int match\_cnt = (from a in kPiDataContext.BasicParametrsAndUsersMapping

join b in kPiDataContext.UsersTable

on a.FK\_UsersTable equals b.UsersTableID

where

(((a.CanConfirm ==true)&&(canConfirm.Checked==true))||((a.CanEdit == true)&&(canEdit.Checked==true)))

&& a.Active == true

&& b.Active == true

&& b.FK\_ZeroLevelSubdivisionTable == 1

&& a.FK\_ParametrsTable == Convert.ToInt32(label.Text)

&& ((b.FK\_FirstLevelSubdivisionTable == first))// || (b.FK\_FirstLevelSubdivisionTable == null))

&& ((b.FK\_SecondLevelSubdivisionTable == second))// || (b.FK\_SecondLevelSubdivisionTable == null))

&& ((b.FK\_ThirdLevelSubdivisionTable == third))// || (b.FK\_ThirdLevelSubdivisionTable == null))

select a).ToList().Count();

match\_cnt\_sum += match\_cnt;

rowIndex++;

}

}

#endregion

if (match\_cnt\_sum > 0)

{

Page.ClientScript.RegisterClientScriptBlock(typeof (Page), "Script",

"alert('Пользователь для заданного подразделения с " + match\_cnt\_sum +

" совпадениями возможностей редактирования поджтвержения существует');", true);

}

else

{

UsersTable user = new UsersTable();

user.Active = true;

user.Login = UserNameText.Text;

user.Password = PasswordText.Text;

user.Email = EmailText.Text;

int selectedValue = -1;

if (int.TryParse(DropDownList1.SelectedValue, out selectedValue) && selectedValue > 0)

user.FK\_FirstLevelSubdivisionTable = selectedValue;

selectedValue = -1;

if (int.TryParse(DropDownList2.SelectedValue, out selectedValue) && selectedValue > 0)

user.FK\_SecondLevelSubdivisionTable = selectedValue;

selectedValue = -1;

if (int.TryParse(DropDownList3.SelectedValue, out selectedValue) && selectedValue > 0)

user.FK\_ThirdLevelSubdivisionTable = selectedValue;

user.AccessLevel = 0; ///////НАДО ПРОДУМАТЬ

if (DropDownList5.SelectedIndex == 2)

{

user.AccessLevel = 5;

}

if (DropDownList5.SelectedIndex == 3)

{

user.AccessLevel = 9;

}

user.FK\_ZeroLevelSubdivisionTable = 1;

string passCode = RandomString(25);

user.PassCode = passCode;

user.Confirmed = false;

kPiDataContext.UsersTable.InsertOnSubmit(user);

kPiDataContext.SubmitChanges();

//// ПОЛЬЗОВАТЕЛЬ СОЗДАН

///

int userID = user.UsersTableID;

///////////////////////////////////////////шаблон//////////////////////////////////

rowIndex = 0;

if (Gridview3.Rows.Count > 0)

{

for (int k = 1; k <= Gridview3.Rows.Count; k++)

{

CheckBox canEdit =

(CheckBox) Gridview3.Rows[rowIndex].FindControl("BasicParametrsEditCheckBox");

CheckBox canView =

(CheckBox) Gridview3.Rows[rowIndex].FindControl("BasicParametrsViewCheckBox");

CheckBox canConfirm =

(CheckBox) Gridview3.Rows[rowIndex].FindControl("BasicParametrsConfirmCheckBox");

Label label = (Label) Gridview3.Rows[rowIndex].FindControl("BasicParametrsID");

BasicParametrsAndUsersMapping BasicAndUsers = new BasicParametrsAndUsersMapping();

BasicAndUsers.Active = true;

BasicAndUsers.FK\_ParametrsTable = Convert.ToInt32(label.Text);

BasicAndUsers.CanConfirm = canConfirm.Checked;

BasicAndUsers.CanEdit = canEdit.Checked;

BasicAndUsers.CanView = canView.Checked;

BasicAndUsers.FK\_UsersTable = userID;

kPiDataContext.BasicParametrsAndUsersMapping.InsertOnSubmit(BasicAndUsers);

kPiDataContext.SubmitChanges();

rowIndex++;

}

if (DropDownList5.SelectedIndex == 2) // если человек руководитель

{

rowIndex = 0;

if (Gridview1.Rows.Count > 0)

{

for (int k = 1; k <= Gridview1.Rows.Count; k++)

{

/\*CheckBox canEdit =

(CheckBox) Gridview1.Rows[rowIndex].FindControl("IndicatorEditCheckBox");\*/

CheckBox canView =

(CheckBox) Gridview1.Rows[rowIndex].FindControl("IndicatorViewCheckBox");

CheckBox canConfirm =

(CheckBox) Gridview1.Rows[rowIndex].FindControl("IndicatorConfirmCheckBox");

Label label = (Label) Gridview1.Rows[rowIndex].FindControl("IndicatorID");

IndicatorsAndUsersMapping indAndUser = new IndicatorsAndUsersMapping();

indAndUser.Active = true;

indAndUser.FK\_IndicatorsTable = Convert.ToInt32(label.Text);

indAndUser.CanConfirm = canConfirm.Checked;

//indAndUser.CanEdit = canEdit.Checked;

indAndUser.CanView = canView.Checked;

indAndUser.FK\_UsresTable = userID;

kPiDataContext.IndicatorsAndUsersMapping.InsertOnSubmit(indAndUser);

kPiDataContext.SubmitChanges();

rowIndex++;

}

}

rowIndex = 0;

if (Gridview2.Rows.Count > 0)

{

for (int k = 1; k <= Gridview2.Rows.Count; k++)

{

/\*CheckBox canEdit =

(CheckBox)

Gridview2.Rows[rowIndex].FindControl("CalculatedParametrsEditCheckBox");\*/

CheckBox canView =

(CheckBox)

Gridview2.Rows[rowIndex].FindControl("CalculatedParametrsViewCheckBox");

CheckBox canConfirm =

(CheckBox)

Gridview2.Rows[rowIndex].FindControl("CalculatedParametrsConfirmCheckBox");

Label label = (Label) Gridview2.Rows[rowIndex].FindControl("CalculatedParametrsID");

CalculatedParametrsAndUsersMapping calcAndUser =

new CalculatedParametrsAndUsersMapping();

calcAndUser.Active = true;

calcAndUser.FK\_CalculatedParametrsTable = Convert.ToInt32(label.Text);

calcAndUser.CanConfirm = canConfirm.Checked;

// calcAndUser.CanEdit = canEdit.Checked;

calcAndUser.CanView = canView.Checked;

calcAndUser.FK\_UsersTable = userID;

kPiDataContext.CalculatedParametrsAndUsersMapping.InsertOnSubmit(calcAndUser);

kPiDataContext.SubmitChanges();

rowIndex++;

}

}

}

}

EmailTemplate EmailParams = (from a in kPiDataContext.EmailTemplate

where a.Name == "InviteToRegister"

&& a.Active == true

select a).FirstOrDefault();

Action.MassMailing(user.Email, EmailParams.EmailTitle,

EmailParams.EmailContent.Replace("#LINK#", ConfigurationManager.AppSettings.Get("SiteName") + "/Account/UserRegister?&id=" + passCode), null);

LogHandler.LogWriter.WriteLog(LogCategory.INFO, "0RN0: Admin(mon) " + (string)ViewState["Login"] + " has registered a new user: " + EmailText.Text + "from ip: " + Dns.GetHostEntry(Dns.GetHostName()).AddressList.Where(ip => ip.AddressFamily == System.Net.Sockets.AddressFamily.InterNetwork).Select(ip => ip.ToString()).FirstOrDefault());

/\*

Action.MassMailing(user.Email,"Ваш почтовый адресс был зарегистрирован в системе ИАС 'КФУ-Программа развития'",

"Здравствуйте!"+Environment.NewLine+

"Ваш почтовый адрес был указан при регистрации в системе ИАС 'КФУ-Программа развития!'"+Environment.NewLine+

"Пожалуйста, проигнорируйте это письмо, если оно попало к вам по ошибке." + Environment.NewLine +

"Для продолжения регистрации перейдите по ссылке ниже:" + Environment.NewLine +

ConfigurationManager.AppSettings.Get("SiteName") + "/Account/UserRegister?&id=" + passCode + Environment.NewLine +

"Спасибо!"

, null);

\*/

}

}

}

protected void DropDownList4\_SelectedIndexChanged1(object sender, EventArgs e)

{

KPIWebDataContext kpiWebDataContext = new KPIWebDataContext();

var vrCountry = (from b in kpiWebDataContext.BasicParametersTable select b);

DataTable dataTable = new DataTable();

dataTable.Columns.Add(new DataColumn("BasicParametrsConfirmCheckBox", typeof(bool)));

dataTable.Columns.Add(new DataColumn("BasicParametrsEditCheckBox", typeof(bool)));

dataTable.Columns.Add(new DataColumn("BasicParametrsViewCheckBox", typeof(bool)));

dataTable.Columns.Add(new DataColumn("BasicParametrsName", typeof(string)));

dataTable.Columns.Add(new DataColumn("BasicParametrsID", typeof(string)));

#region

foreach (var obj in vrCountry)

{

DataRow dataRow = dataTable.NewRow();

BasicParametersAndRolesMappingTable roleAndBasicMapping =

(from a in kpiWebDataContext.BasicParametersAndRolesMappingTable

where a.FK\_BasicParametersTable == obj.BasicParametersTableID

&& a.FK\_RolesTable == Convert.ToInt32(DropDownList4.Items[DropDownList4.SelectedIndex].Value)

select a).FirstOrDefault();

if (roleAndBasicMapping != null)

{

dataRow["BasicParametrsEditCheckBox"] = roleAndBasicMapping.CanEdit;

dataRow["BasicParametrsViewCheckBox"] = roleAndBasicMapping.CanView;

dataRow["BasicParametrsConfirmCheckBox"] = roleAndBasicMapping.CanConfirm;

}

else

{

dataRow["BasicParametrsEditCheckBox"] = false;

dataRow["BasicParametrsViewCheckBox"] = false;

dataRow["BasicParametrsConfirmCheckBox"] = false;

}

dataRow["BasicParametrsID"] = obj.BasicParametersTableID.ToString();

dataRow["BasicParametrsName"] = obj.Name;

dataTable.Rows.Add(dataRow);

}

//ViewState["BasicRoleMapping"] = dataTable;

Gridview3.DataSource = dataTable;

Gridview3.DataBind();

#endregion

RolesTable role = (from a in kpiWebDataContext.RolesTable

where a.RolesTableID == Convert.ToInt32(DropDownList4.Items[DropDownList4.SelectedIndex].Value)

select a).FirstOrDefault();

ViewState["IsHead"] = role.IsHead;

if ((bool)role.IsHead)

{

//////////////////

var calcParam = (from a in kpiWebDataContext.CalculatedParametrs select a);

var Indicators = (from c in kpiWebDataContext.IndicatorsTable select c);

DataTable calcTable = new DataTable();

calcTable.Columns.Add(new DataColumn("CalculatedParametrsConfirmCheckBox", typeof(bool)));

calcTable.Columns.Add(new DataColumn("CalculatedParametrsViewCheckBox", typeof(bool)));

calcTable.Columns.Add(new DataColumn("CalculatedParametrsName", typeof(string)));

calcTable.Columns.Add(new DataColumn("CalculatedParametrsID", typeof(string)));

DataTable indicatorTable = new DataTable();

indicatorTable.Columns.Add(new DataColumn("IndicatorConfirmCheckBox", typeof(bool)));

indicatorTable.Columns.Add(new DataColumn("IndicatorViewCheckBox", typeof(bool)));

indicatorTable.Columns.Add(new DataColumn("IndicatorName", typeof(string)));

indicatorTable.Columns.Add(new DataColumn("IndicatorID", typeof(string)));

#region

foreach (var obj in calcParam)

{

DataRow dataRow = calcTable.NewRow();

CalculatedParametrsAndRolesMappingTable roleAndCalcMapping =

(from a in kpiWebDataContext.CalculatedParametrsAndRolesMappingTable

where a.FK\_CalculatedParametrs == obj.CalculatedParametrsID

&& a.FK\_RolesTable == Convert.ToInt32(DropDownList4.Items[DropDownList4.SelectedIndex].Value)

select a).FirstOrDefault();

if (roleAndCalcMapping != null)

{

dataRow["CalculatedParametrsViewCheckBox"] = roleAndCalcMapping.CanView;

dataRow["CalculatedParametrsConfirmCheckBox"] = roleAndCalcMapping.CanConfirm;

}

else

{

dataRow["CalculatedParametrsViewCheckBox"] = false;

dataRow["CalculatedParametrsConfirmCheckBox"] = false;

}

dataRow["CalculatedParametrsID"] = obj.CalculatedParametrsID.ToString();

dataRow["CalculatedParametrsName"] = obj.Name;

calcTable.Rows.Add(dataRow);

}

Gridview2.DataSource = calcTable;

Gridview2.DataBind();

// ViewState["CalcRoleMapping"] = CalcGrid;

#endregion

#region

foreach (var obj in Indicators)

{

DataRow dataRow = indicatorTable.NewRow();

IndicatorsAndRolesMappingTable roleAndIndMapping =

(from a in kpiWebDataContext.IndicatorsAndRolesMappingTable

where a.FK\_Indicators == obj.IndicatorsTableID

&& a.FK\_RolesTable == Convert.ToInt32(DropDownList4.Items[DropDownList4.SelectedIndex].Value)

select a).FirstOrDefault();

if (roleAndIndMapping != null)

{

dataRow["IndicatorViewCheckBox"] = roleAndIndMapping.CanView;

dataRow["IndicatorConfirmCheckBox"] = roleAndIndMapping.CanConfirm;

}

else

{

dataRow["IndicatorViewCheckBox"] = false;

dataRow["IndicatorConfirmCheckBox"] = false;

}

dataRow["IndicatorID"] = obj.IndicatorsTableID.ToString();

dataRow["IndicatorName"] = obj.Name;

indicatorTable.Rows.Add(dataRow);

}

Gridview1.DataSource = indicatorTable;

Gridview1.DataBind();

// ViewState["IndRoleMapping"] = IndicatorGrid;

#endregion

}

}

protected void Gridview3\_RowDataBound(object sender, GridViewRowEventArgs e)

{/\*

bool IsHead = (bool) ViewState["IsHead"];

if (IsHead)

{

gri

}

\*/

}

protected void DropDownList3\_SelectedIndexChanged(object sender, EventArgs e)

{

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Filename: Regisration.aspx.designer.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//------------------------------------------------------------------------------

// <auto-generated>

// This code was generated by a tool.

//

// Changes to this file may cause incorrect behavior and will be lost if

// the code is regenerated.

// </auto-generated>

//------------------------------------------------------------------------------

namespace KPIWeb.AutomationDepartment {

public partial class Regisration {

/// <summary>

/// Label1 control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.Label Label1;

/// <summary>

/// DropDownList1 control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.DropDownList DropDownList1;

/// <summary>

/// Label2 control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.Label Label2;

/// <summary>

/// DropDownList2 control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.DropDownList DropDownList2;

/// <summary>

/// Label3 control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.Label Label3;

/// <summary>

/// DropDownList3 control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.DropDownList DropDownList3;

/// <summary>

/// Label4 control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.Label Label4;

/// <summary>

/// DropDownList5 control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.DropDownList DropDownList5;

/// <summary>

/// Label24 control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.Label Label24;

/// <summary>

/// DropDownList4 control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.DropDownList DropDownList4;

/// <summary>

/// Label25 control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.Label Label25;

/// <summary>

/// Gridview1 control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.GridView Gridview1;

/// <summary>

/// Label26 control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.Label Label26;

/// <summary>

/// Gridview2 control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.GridView Gridview2;

/// <summary>

/// Label27 control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.Label Label27;

/// <summary>

/// Gridview3 control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.GridView Gridview3;

/// <summary>

/// UserNameLabel control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.Label UserNameLabel;

/// <summary>

/// UserNameText control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.TextBox UserNameText;

/// <summary>

/// errorNoName control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.RequiredFieldValidator errorNoName;

/// <summary>

/// EmailLabel control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.Label EmailLabel;

/// <summary>

/// EmailText control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.TextBox EmailText;

/// <summary>

/// errorNoEmailText control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.RequiredFieldValidator errorNoEmailText;

/// <summary>

/// errorInvalidEmail control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.RegularExpressionValidator errorInvalidEmail;

/// <summary>

/// PassLabel control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.Label PassLabel;

/// <summary>

/// PasswordText control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.TextBox PasswordText;

/// <summary>

/// errorNoPass control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.RequiredFieldValidator errorNoPass;

/// <summary>

/// ConfPassLabel control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.Label ConfPassLabel;

/// <summary>

/// ConfirmPasswordText control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.TextBox ConfirmPasswordText;

/// <summary>

/// errorNoConfirm control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.RequiredFieldValidator errorNoConfirm;

/// <summary>

/// ErrorWrongConfirm control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.CompareValidator ErrorWrongConfirm;

/// <summary>

/// Button1 control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.Button Button1;

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Filename: RoleMapping.aspx.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using System;

using System.Collections.Generic;

using System.Configuration;

using System.Data;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

using Microsoft.Ajax.Utilities;

namespace KPIWeb

{

public partial class WebForm1 : System.Web.UI.Page

{

int indexkorel = 1;

protected void Page\_Load(object sender, EventArgs e)

{

Serialization UserSer = (Serialization)Session["UserID"];

if (UserSer == null)

{

Response.Redirect("~/Default.aspx");

}

int userID = UserSer.Id;

KPIWebDataContext kPiDataContext = new KPIWebDataContext();

UsersTable userTable =

(from a in kPiDataContext.UsersTable where a.UsersTableID == userID select a).FirstOrDefault();

if (userTable.AccessLevel != 10)

{

Response.Redirect("~/Default.aspx");

}

//////////////////////////////////////////////////////////////////////////////////////////////////

if (!IsPostBack)

{

KPIWebDataContext kpiWebDataContext = new KPIWebDataContext();

List<string> role = new List<string>();

role = (from roles in kpiWebDataContext.RolesTable

select roles.RoleName).ToList();

foreach (string name in role)

{

DropDownList1.Items.Add(name);

}

}

}

protected void DropDownList1\_SelectedIndexChanged(object sender, EventArgs e)

{

RefreshGridView();

Button1.Visible = true;

}

private void RefreshGridView()

{

KPIWebDataContext kpiWebDataContext = new KPIWebDataContext();

var vrCountry = (from a in kpiWebDataContext.BasicParametersTable select a).Except(

from b in kpiWebDataContext.BasicParametersTable

join c in kpiWebDataContext.BasicParametersAndRolesMappingTable on b.BasicParametersTableID equals

c.FK\_BasicParametersTable

select b);

DataTable dataTable = new DataTable();

dataTable.Columns.Add(new DataColumn("EditChecked", typeof (bool)));

dataTable.Columns.Add(new DataColumn("ViewChecked", typeof (bool)));

dataTable.Columns.Add(new DataColumn("Name", typeof (string)));

int i = 1;

foreach (var obj in vrCountry)

{

DataRow dataRow = dataTable.NewRow();

dataRow["EditChecked"] = false;

dataRow["ViewChecked"] = false;

dataRow["Name"] = " " + obj.BasicParametersTableID + ". " + obj.Name;

dataTable.Rows.Add(dataRow);

i++;

}

ViewState["GridviewRoleMapping"] = dataTable;

GridviewRoles.DataSource = dataTable;

GridviewRoles.DataBind();

}

protected void Button1\_Click(object sender, EventArgs e)

{

int rowIndex = 0;

KPIWebDataContext kpiWebDataContext2 = new KPIWebDataContext();

var vrCountry = (from a in kpiWebDataContext2.BasicParametersTable select a).Except(

from b in kpiWebDataContext2.BasicParametersTable

join c in kpiWebDataContext2.BasicParametersAndRolesMappingTable on

b.BasicParametersTableID equals c.FK\_BasicParametersTable

select b).ToList();

var vrRoleName = (from a in kpiWebDataContext2.RolesTable

where a.RoleName == DropDownList1.SelectedValue.ToString()

select new { a.RoleName, a.RolesTableID }).ToList();

if (ViewState["GridviewRoleMapping"] != null)

{

DataTable dataTable = (DataTable) ViewState["GridviewRoleMapping"];

if (dataTable.Rows.Count > 0)

{

for (int i = 1; i <= dataTable.Rows.Count; i++)

{

CheckBox checkBoxCanEdit = (CheckBox)GridviewRoles.Rows[rowIndex].FindControl("CheckBoxCanEdit");

CheckBox checkBoxCanView = (CheckBox)GridviewRoles.Rows[rowIndex].FindControl("CheckBoxCanView");

if (checkBoxCanView != null && checkBoxCanEdit != null)

{

if (checkBoxCanEdit.Checked == true || checkBoxCanView.Checked == true)

{

using (KPIWebDataContext kpiWebDataContext = new KPIWebDataContext())

{

BasicParametersAndRolesMappingTable bparmTables = new BasicParametersAndRolesMappingTable();

bparmTables.Active = true;

bparmTables.FK\_RolesTable = vrRoleName[0].RolesTableID;

bparmTables.FK\_BasicParametersTable = vrCountry[rowIndex].BasicParametersTableID;

indexkorel++;

if (checkBoxCanEdit.Checked == true)

bparmTables.CanEdit = true;

else

{

bparmTables.CanEdit = false;

}

if (checkBoxCanView.Checked == true)

bparmTables.CanView = true;

else

{

bparmTables.CanView = false;

}

kpiWebDataContext.BasicParametersAndRolesMappingTable.InsertOnSubmit(bparmTables);

kpiWebDataContext.SubmitChanges();

}

}

rowIndex++;

}

}

}

}

RefreshGridView();

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Filename: RoleMapping.aspx.designer.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//------------------------------------------------------------------------------

// <автоматически создаваемое>

// Этот код создан программой.

//

// Изменения в этом файле могут привести к неправильной работе и будут потеряны в случае

// повторной генерации кода.

// </автоматически создаваемое>

//------------------------------------------------------------------------------

namespace KPIWeb {

public partial class WebForm1 {

/// <summary>

/// DropDownList1 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.DropDownList DropDownList1;

/// <summary>

/// GridviewRoles элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.GridView GridviewRoles;

/// <summary>

/// Button1 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Button Button1;

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Filename: SendInvite.aspx.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using System;

using System.Collections.Generic;

using System.Linq;

using System.Configuration;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

namespace KPIWeb.AutomationDepartment

{

public partial class SendInvite : System.Web.UI.Page

{

protected void Page\_Load(object sender, EventArgs e)

{

KPIWebDataContext kpiWebDataContext = new KPIWebDataContext();

List<FirstLevelSubdivisionTable> FirstList = (from a in kpiWebDataContext.FirstLevelSubdivisionTable

where a.Active == true

select a).ToList();

int i = 0;

foreach (FirstLevelSubdivisionTable current in FirstList)

{

CheckBoxList1.Items.Add(current.Name);

CheckBoxList1.Items[i].Value = current.FirstLevelSubdivisionTableID.ToString();

i++;

}

}

protected void Button1\_Click(object sender, EventArgs e)

{

KPIWebDataContext kPiDataContext = new KPIWebDataContext();

var users = (from a in kPiDataContext.UsersTable where a.Active && a.Confirmed == false select a).ToList();

EmailTemplate EmailParams = (from a in kPiDataContext.EmailTemplate

where a.Name == "InviteToRegister"

&& a.Active == true

select a).FirstOrDefault();

foreach (var user in users)

{

Action.MassMailing(user.Email, EmailParams.EmailTitle,

EmailParams.EmailContent.Replace("#LINK#", ConfigurationManager.AppSettings.Get("SiteName") + "/Account/UserRegister?&id=" + user.PassCode).

Replace("#SiteName#", ConfigurationManager.AppSettings.Get("SiteName")), null);

}

}

protected void Button2\_Click(object sender, EventArgs e)

{

KPIWebDataContext kPiDataContext = new KPIWebDataContext();

foreach (ListItem currentAcademy in CheckBoxList1.Items)

{

if (currentAcademy.Selected == true)

{

var users = (from a in kPiDataContext.UsersTable where a.Active

&& a.Confirmed == false

&& a.FK\_FirstLevelSubdivisionTable == Convert.ToInt32(currentAcademy.Value)

select a).ToList();

EmailTemplate EmailParams = (from a in kPiDataContext.EmailTemplate

where a.Name == "InviteToRegister"

&& a.Active == true

select a).FirstOrDefault();

foreach (var user in users)

{

Action.MassMailing(user.Email, EmailParams.EmailTitle,

EmailParams.EmailContent.Replace("#LINK#", ConfigurationManager.AppSettings.Get("SiteName") + "/Account/UserRegister?&id=" + user.PassCode).

Replace("#SiteName#", ConfigurationManager.AppSettings.Get("SiteName")), null);

}

}

}

}

protected void Button4\_Click(object sender, EventArgs e)

{

KPIWebDataContext kPiDataContext = new KPIWebDataContext();

List<String> emails = new List<string>();

if (ViewState["Mails"] != null)

{

emails = (List<String>)ViewState["Mails"];

}

var user =

(from a in kPiDataContext.UsersTable where a.Email.Equals(TextBox1.Text) select a).FirstOrDefault();

if (user != null)

{

ListBox1.Items.Clear();

emails.Add(TextBox1.Text);

ListBox1.DataSource = emails;

ListBox1.DataBind();

}

else DisplayAlert("Данный email не зарегистрирован в системе");

ViewState["Mails"] = emails;

}

private void DisplayAlert(string message)

{

ClientScript.RegisterStartupScript(

this.GetType(),

Guid.NewGuid().ToString(),

string.Format("alert('{0}');",

message.Replace("'", @"\'").Replace("\n", "\\n").Replace("\r", "\\r")),

true);

}

protected void Button3\_Click(object sender, EventArgs e)

{

KPIWebDataContext kPiDataContext = new KPIWebDataContext();

List<String> users = new List<string>();

if (ViewState["Mails"] != null)

{

users = (List<String>) ViewState["Mails"];

var emails = users.Distinct().ToList();

EmailTemplate EmailParams = (from a in kPiDataContext.EmailTemplate

where a.Name == "InviteToRegister"

&& a.Active == true

select a).FirstOrDefault();

foreach (var email in emails)

{

Action.MassMailing(email, EmailParams.EmailTitle,

EmailParams.EmailContent.Replace("#LINK#",

ConfigurationManager.AppSettings.Get("SiteName") + "/Account/UserRegister?&id=" +

(from a in kPiDataContext.UsersTable where a.Email.Equals(email) select a.PassCode)

.FirstOrDefault()).

Replace("#SiteName#", ConfigurationManager.AppSettings.Get("SiteName")), null);

}

}

else

{

DisplayAlert("Пустой список");

}

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Filename: SendInvite.aspx.designer.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//------------------------------------------------------------------------------

// <auto-generated>

// This code was generated by a tool.

//

// Changes to this file may cause incorrect behavior and will be lost if

// the code is regenerated.

// </auto-generated>

//------------------------------------------------------------------------------

namespace KPIWeb.AutomationDepartment {

public partial class SendInvite {

/// <summary>

/// Button1 control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.Button Button1;

/// <summary>

/// Button2 control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.Button Button2;

/// <summary>

/// Button3 control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.Button Button3;

/// <summary>

/// CheckBoxList1 control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.CheckBoxList CheckBoxList1;

/// <summary>

/// Label2 control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.Label Label2;

/// <summary>

/// ListBox1 control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.ListBox ListBox1;

/// <summary>

/// Label1 control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.Label Label1;

/// <summary>

/// TextBox1 control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.TextBox TextBox1;

/// <summary>

/// Button4 control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.Button Button4;

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Filename: WatchingParamtrsState.aspx.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using System;

using System.Collections.Generic;

using System.Data;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

using log4net.Util.TypeConverters;

namespace KPIWeb.AutomationDepartment

{

public partial class WatchingParamtrsState : System.Web.UI.Page

{

protected void Page\_Load(object sender, EventArgs e)

{

Serialization UserSer = (Serialization) Session["UserID"];

if (UserSer == null)

{

Response.Redirect("~/Default.aspx");

}

int userID = UserSer.Id;

KPIWebDataContext kPiDataContext = new KPIWebDataContext();

UsersTable userTable =

(from a in kPiDataContext.UsersTable where a.UsersTableID == userID select a).FirstOrDefault();

if (userTable.AccessLevel != 10)

{

Response.Redirect("~/Default.aspx");

}

//////////////////////////////////////////////////////////////////

if (!Page.IsPostBack)

{

List<FirstLevelSubdivisionTable> First\_stageList =

(from item in kPiDataContext.FirstLevelSubdivisionTable

select item).OrderBy(mc => mc.Name).ToList();

var dictionary = new Dictionary<int, string>();

dictionary.Add(0, "Выберите значение");

foreach (var item in First\_stageList)

dictionary.Add(item.FirstLevelSubdivisionTableID, item.Name);

// FillGridVIews(0);

DropDownList1.DataTextField = "Value";

DropDownList1.DataValueField = "Key";

DropDownList1.DataSource = dictionary;

DropDownList1.DataBind();

List<ReportArchiveTable> reportList =

(from item in kPiDataContext.ReportArchiveTable

select item).OrderBy(mc => mc.Name).ToList();

var diction = new Dictionary<int, string>();

diction.Add(0, "Выберите отчет");

foreach (var item in reportList)

diction.Add(item.ReportArchiveTableID, item.Name);

DropDownList4.DataTextField = "Value";

DropDownList4.DataValueField = "Key";

DropDownList4.DataSource = diction;

DropDownList4.DataBind();

}

}

private void RefreshGrid()

{

KPIWebDataContext kpiWebDataContext = new KPIWebDataContext();

var vrCountry = (from b in kpiWebDataContext.CollectedBasicParametersTable select b);

DataTable dataTable = new DataTable();

dataTable.Columns.Add(new DataColumn("Name", typeof (string)));

dataTable.Columns.Add(new DataColumn("Value", typeof (int)));

dataTable.Columns.Add(new DataColumn("State", typeof (string)));

dataTable.Columns.Add(new DataColumn("BasicId", typeof (int)));

CollectedBasicParametersTable parametersTable = (from a in kpiWebDataContext.CollectedBasicParametersTable

//where a.RolesTableID == Convert.ToInt32(DropDownList1.Items[DropDownList1.SelectedIndex].Value)

select a).FirstOrDefault();

List<ZeroLevelSubdivisionTable> zeroLevelSubdivisionTable =

(from a in kpiWebDataContext.ZeroLevelSubdivisionTable select a).ToList();

List<FirstLevelSubdivisionTable> firstLevelSubdivisionTable =

(from a in kpiWebDataContext.FirstLevelSubdivisionTable select a).ToList();

List<SecondLevelSubdivisionTable> secondLevelSubdivisionTable =

(from a in kpiWebDataContext.SecondLevelSubdivisionTable select a).ToList();

List<ThirdLevelSubdivisionTable> thirdLevelSubdivisionTable =

(from a in kpiWebDataContext.ThirdLevelSubdivisionTable select a).ToList();

List<FourthLevelSubdivisionTable> fourthLevelSubdivisionTable =

(from a in kpiWebDataContext.FourthLevelSubdivisionTable select a).ToList();

List<FifthLevelSubdivisionTable> fifthLevelSubdivisionTable =

(from a in kpiWebDataContext.FifthLevelSubdivisionTable select a).ToList();

foreach (var obj in vrCountry)

{

DataRow dataRow = dataTable.NewRow();

List<CollectedBasicParametersTable> parameters =

(from a in kpiWebDataContext.CollectedBasicParametersTable

//where a.FK\_BasicParametersTable == obj.BasicParametersTableID

// && a.FK\_RolesTable == Convert.ToInt32(DropDownList1.Items[DropDownList1.SelectedIndex].Value)

select a).ToList();

/\* if (obj.CollectedValue!=null)

{

dataRow["Value"] = (from a in kpiWebDataContext.CollectedBasicParametersTable

//where a.FK\_BasicParametersTable == obj.

select obj.CollectedValue).FirstOrDefault();

}

else

{

dataRow["Value"] = 0;

}\*/

// ViewState["BasicRoleMapping"] = dataTable;

dataRow["BasicId"] = (from a in kpiWebDataContext.BasicParametersTable select a.BasicParametersTableID).FirstOrDefault();

dataRow["Name"] = (from a in kpiWebDataContext.BasicParametersTable select a.Name).FirstOrDefault();

dataTable.Rows.Add(dataRow);

GridView1.DataSource = dataTable;

GridView1.DataBind();

}

}

protected void DropDownList1\_SelectedIndexChanged(object sender, EventArgs e)

{

DropDownList2.Items.Clear();

DropDownList3.Items.Clear();

KPIWebDataContext kPiDataContext = new KPIWebDataContext();

int SelectedValue = -1;

if (int.TryParse(DropDownList1.SelectedValue, out SelectedValue) && SelectedValue != -1)

{

List<SecondLevelSubdivisionTable> second\_stageList =

(from item in kPiDataContext.SecondLevelSubdivisionTable

where item.FK\_FirstLevelSubdivisionTable == SelectedValue

select item).OrderBy(mc => mc.SecondLevelSubdivisionTableID).ToList();

if (second\_stageList != null && second\_stageList.Count() > 0)

{

var dictionary = new Dictionary<int, string>();

dictionary.Add(-1, "Выберите значение");

foreach (var item in second\_stageList)

dictionary.Add(item.SecondLevelSubdivisionTableID, item.Name);

DropDownList2.DataTextField = "Value";

DropDownList2.DataValueField = "Key";

DropDownList2.DataSource = dictionary;

DropDownList2.DataBind();

}

else

{

//Page.ClientScript.RegisterClientScriptBlock(typeof(Page), "Script", "alert('Произошла ошибка.');", true);

}

}

else

{

Page.ClientScript.RegisterClientScriptBlock(typeof (Page), "Script", "alert('Произошла ошибка.');", true);

}

}

protected void DropDownList2\_SelectedIndexChanged(object sender, EventArgs e)

{

DropDownList3.Items.Clear();

KPIWebDataContext kPiDataContext = new KPIWebDataContext();

int SelectedValue = -1;

if (int.TryParse(DropDownList2.SelectedValue, out SelectedValue) && SelectedValue != -1)

{

List<ThirdLevelSubdivisionTable> third\_stage = (from item in kPiDataContext.ThirdLevelSubdivisionTable

where item.FK\_SecondLevelSubdivisionTable == SelectedValue

select item).OrderBy(mc => mc.ThirdLevelSubdivisionTableID).ToList();

if (third\_stage != null && third\_stage.Count() > 0)

{

var dictionary = new Dictionary<int, string>();

dictionary.Add(-1, "Выберите значение");

foreach (var item in third\_stage)

dictionary.Add(item.ThirdLevelSubdivisionTableID, item.Name);

DropDownList3.DataTextField = "Value";

DropDownList3.DataValueField = "Key";

DropDownList3.DataSource = dictionary;

DropDownList3.DataBind();

}

else

{

//Page.ClientScript.RegisterClientScriptBlock(typeof(Page), "Script", "alert('Произошла ошибка.');", true);

}

}

else

{

Page.ClientScript.RegisterClientScriptBlock(typeof (Page), "Script", "alert('Произошла ошибка.');", true);

}

}

protected void DropDownList3\_SelectedIndexChanged(object sender, EventArgs e)

{

}

protected void DropDownList4\_SelectedIndexChanged(object sender, EventArgs e)

{

}

protected void Button1\_Click(object sender, EventArgs e)

{

RefreshGrid();

/\* KPIWebDataContext kpiWebDataContext = new KPIWebDataContext();

var vrCountry = (from b in kpiWebDataContext.BasicParametersTable select b);

DataTable dataTable = new DataTable();

//dataTable.Columns.Add(new DataColumn("VerifyChecked", typeof(bool)));

//dataTable.Columns.Add(new DataColumn("EditChecked", typeof(bool)));

dataTable.Columns.Add(new DataColumn("Name", typeof(string)));

dataTable.Columns.Add(new DataColumn("Value", typeof(int)));

dataTable.Columns.Add(new DataColumn("State", typeof(string)));

dataTable.Columns.Add(new DataColumn("BasicId", typeof(string)));

CollectedBasicParametersTable parametersTable = (from a in kpiWebDataContext.CollectedBasicParametersTable

//where a.RolesTableID == Convert.ToInt32(DropDownList1.Items[DropDownList1.SelectedIndex].Value)

select a).FirstOrDefault();

foreach (var obj in vrCountry)

{

DataRow dataRow = dataTable.NewRow();

CollectedBasicParametersTable parameters =

(from a in kpiWebDataContext.CollectedBasicParametersTable

//where a.FK\_BasicParametersTable == obj.BasicParametersTableID

// && a.FK\_RolesTable == Convert.ToInt32(DropDownList1.Items[DropDownList1.SelectedIndex].Value)

select a).FirstOrDefault();

if (parameters != null)

/\* {

dataRow["EditChecked"] = parameters.CanEdit;

dataRow["ViewChecked"] = parameters.CanView;

dataRow["VerifyChecked"] = roleAndBasicMapping.CanConfirm;

}

else

{

dataRow["EditChecked"] = false;

dataRow["ViewChecked"] = false;

dataRow["VerifyChecked"] = false;

}

dataRow["Value"] = (from a in kpiWebDataContext.CollectedBasicParametersTable

where a.FK\_BasicParametersTable == obj.BasicParametersTableID

select a).FirstOrDefault();

dataRow["BasicId"] = obj.BasicParametersTableID.ToString();

dataRow["Name"] = obj.Name;

dataTable.Rows.Add(dataRow);

}

// ViewState["BasicRoleMapping"] = dataTable;

GridView1.DataSource = dataTable;

GridView1.DataBind();\*/

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Filename: WatchingParamtrsState.aspx.designer.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//------------------------------------------------------------------------------

// <автоматически создаваемое>

// Этот код создан программой.

//

// Изменения в этом файле могут привести к неправильной работе и будут потеряны в случае

// повторной генерации кода.

// </автоматически создаваемое>

//------------------------------------------------------------------------------

namespace KPIWeb.AutomationDepartment {

public partial class WatchingParamtrsState {

/// <summary>

/// Label4 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Label Label4;

/// <summary>

/// DropDownList4 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.DropDownList DropDownList4;

/// <summary>

/// Label1 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Label Label1;

/// <summary>

/// DropDownList1 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.DropDownList DropDownList1;

/// <summary>

/// Label2 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Label Label2;

/// <summary>

/// DropDownList2 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.DropDownList DropDownList2;

/// <summary>

/// Label3 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Label Label3;

/// <summary>

/// DropDownList3 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.DropDownList DropDownList3;

/// <summary>

/// Button1 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Button Button1;

/// <summary>

/// GridView1 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.GridView GridView1;

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Filename: DMain.aspx.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

using System.Data;

namespace KPIWeb.Director

{

public partial class DMain : System.Web.UI.Page

{

protected void Page\_Load(object sender, EventArgs e)

{

Serialization UserSer = (Serialization)Session["UserID"];

if (UserSer == null)

{

Response.Redirect("~/Default.aspx");

}

int userID = UserSer.Id;

ViewState["LocalUserID"] = userID;

KPIWebDataContext kpiWebDataContext = new KPIWebDataContext();

UsersTable userTable =

(from a in kpiWebDataContext.UsersTable where a.UsersTableID == userID select a).FirstOrDefault();

if (userTable.AccessLevel != 4)

{

Response.Redirect("~/Default.aspx");

}

if (!Page.IsPostBack)

{

#region get user reports

List<ReportArchiveTable> reportsArchiveTablesTable = (

from a in kpiWebDataContext.UsersTable

join b in kpiWebDataContext.FirstLevelSubdivisionTable

on a.FK\_FirstLevelSubdivisionTable equals b.FirstLevelSubdivisionTableID

join c in kpiWebDataContext.ReportArchiveAndLevelMappingTable

on b.FirstLevelSubdivisionTableID equals c.FK\_FirstLevelSubmisionTableId

join d in kpiWebDataContext.ReportArchiveTable

on c.FK\_ReportArchiveTableId equals d.ReportArchiveTableID

where a.UsersTableID == UserSer.Id

&& a.Active == true

&& b.Active == true

&& c.Active == true

&& d.Active == true

&& d.StartDateTime < DateTime.Now

&& d.EndDateTime > DateTime.Now

select d).ToList();

///тут мы получили список активных отччетов пользователя

/// пользователь привязан к таблице первого подразделения

/// таблица первого подразделения привязана к таблице отчётов(через таблицу связи)

/// на данный момент отчёт можно назначать только первому подразделению!!!

///

UsersTable user = (from a in kpiWebDataContext.UsersTable

where a.UsersTableID == userID

select a).FirstOrDefault();

int l\_0 = user.FK\_ZeroLevelSubdivisionTable == null ? 0 : (int)user.FK\_ZeroLevelSubdivisionTable;

int l\_1 = user.FK\_FirstLevelSubdivisionTable == null ? 0 : (int)user.FK\_FirstLevelSubdivisionTable;

int l\_2 = user.FK\_SecondLevelSubdivisionTable == null ? 0 : (int)user.FK\_SecondLevelSubdivisionTable;

int l\_3 = user.FK\_ThirdLevelSubdivisionTable == null ? 0 : (int)user.FK\_ThirdLevelSubdivisionTable;

int l\_4 = user.FK\_FourthLevelSubdivisionTable == null ? 0 : (int)user.FK\_FourthLevelSubdivisionTable;

int l\_5 = user.FK\_FifthLevelSubdivisionTable == null ? 0 : (int)user.FK\_FifthLevelSubdivisionTable;

int userLevel = 5;

userLevel = l\_5 == 0 ? 4 : userLevel;

userLevel = l\_4 == 0 ? 3 : userLevel;

userLevel = l\_3 == 0 ? 2 : userLevel;

userLevel = l\_2 == 0 ? 1 : userLevel;

userLevel = l\_1 == 0 ? 0 : userLevel;

userLevel = l\_0 == 0 ? -1 : userLevel;

DataTable dataTable = new DataTable();

dataTable.Columns.Add(new DataColumn("ReportArchiveID", typeof(string)));

dataTable.Columns.Add(new DataColumn("ReportName", typeof(string)));

dataTable.Columns.Add(new DataColumn("StartDate", typeof(string)));

dataTable.Columns.Add(new DataColumn("EndDate", typeof(string)));

dataTable.Columns.Add(new DataColumn("Status", typeof(string)));

foreach (ReportArchiveTable ReportRow in reportsArchiveTablesTable)

{

#region

int can\_view =

(from a in kpiWebDataContext.ReportArchiveAndBasicParametrsMappingTable

join b in kpiWebDataContext.BasicParametersTable

on a.FK\_BasicParametrsTable equals b.BasicParametersTableID

join c in kpiWebDataContext.BasicParametrsAndUsersMapping

on b.BasicParametersTableID equals c.FK\_ParametrsTable

join d in kpiWebDataContext.BasicParametrAdditional

on b.BasicParametersTableID equals d.BasicParametrAdditionalID

where

a.FK\_ReportArchiveTable == ReportRow.ReportArchiveTableID //из нужного отчёта

&& c.FK\_UsersTable == userID // свяный с пользователем

//&& d.SubvisionLevel == 3 //нужный уровень заполняющего

&& a.Active == true // запись в таблице связей показателя и отчёта активна

&& c.CanView == true

&& c.Active == true // запись в таблице связей показателя и пользователей активна

&& d.Calculated == false

// этот показатель нужно вводить а не считать

select b).ToList().Count;

if (can\_view > 0)

{

}

else

{

continue;

}

DataRow dataRow = dataTable.NewRow();

dataRow["ReportArchiveID"] = ReportRow.ReportArchiveTableID.ToString();

dataRow["ReportName"] = ReportRow.Name;

dataRow["StartDate"] = ReportRow.StartDateTime.ToString().Split(' ')[0];//только дата// время обрезается сплитом

dataRow["EndDate"] = ReportRow.EndDateTime.ToString().Split(' ')[0];

//нужно определить статус данных

dataRow["Status"] = "В разработке";

dataTable.Rows.Add(dataRow);

#endregion

}

GridView1.DataSource = dataTable;

GridView1.DataBind();

#endregion

}

}

protected void ButtonViewClick(object sender, EventArgs e)

{

Button button = (Button)sender;

{

Serialization paramSerialization = new Serialization(button.CommandArgument.ToString());

Session["ReportArchiveID"] = paramSerialization; // запомнили в сессии номер отчёта

KPIWebDataContext kpiWebDataContext = new KPIWebDataContext();

var login =

(from a in kpiWebDataContext.UsersTable

where a.UsersTableID == (int)ViewState["LocalUserID"]

select a.Email).FirstOrDefault();

LogHandler.LogWriter.WriteLog(LogCategory.INFO, "0CR1: Пользователь (Директор академии) " + login + " зашел на страницу просмотра отчета с ID = " + paramSerialization.ReportStr);

Response.Redirect("~/Director/DReportView.aspx");

}

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Filename: DMain.aspx.designer.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//------------------------------------------------------------------------------

// <auto-generated>

// This code was generated by a tool.

//

// Changes to this file may cause incorrect behavior and will be lost if

// the code is regenerated.

// </auto-generated>

//------------------------------------------------------------------------------

namespace KPIWeb.Director {

public partial class DMain {

/// <summary>

/// GridView1 control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.GridView GridView1;

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Filename: DReportView.aspx.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

using System.Data;

namespace KPIWeb.Director

{

public partial class DReportView : System.Web.UI.Page

{

protected void Page\_Load(object sender, EventArgs e)

{

Serialization UserSer = (Serialization)Session["UserID"];

if (UserSer == null)

{

Response.Redirect("~/Default.aspx");

}

int userID = UserSer.Id;

ViewState["LocalUserID"] = userID;

KPIWebDataContext kpiWebDataContext = new KPIWebDataContext();

UsersTable userTable =

(from a in kpiWebDataContext.UsersTable where a.UsersTableID == userID select a).FirstOrDefault();

if (userTable.AccessLevel != 4)

{

Response.Redirect("~/Default.aspx");

}

Serialization paramSerialization = (Serialization)Session["ReportArchiveID"];

if (paramSerialization == null)

{

Response.Redirect("~/Default.aspx");

}

int ReportID = Convert.ToInt32(paramSerialization.ReportStr);

if (!Page.IsPostBack)

{

//мы берем факультеты

DataTable dataTable = new DataTable();

dataTable.Columns.Add(new DataColumn("StructName", typeof(string)));

dataTable.Columns.Add(new DataColumn("StructID", typeof(string)));

dataTable.Columns.Add(new DataColumn("Status", typeof(string)));

List<SecondLevelSubdivisionTable> Faculties = (from a in kpiWebDataContext.SecondLevelSubdivisionTable

where a.FK\_FirstLevelSubdivisionTable == userTable.FK\_FirstLevelSubdivisionTable

&& a.Active == true

join b in kpiWebDataContext.UsersTable

on a.SecondLevelSubdivisionTableID equals b.FK\_SecondLevelSubdivisionTable

where b.Active == true

join c in kpiWebDataContext.BasicParametrsAndUsersMapping

on b.UsersTableID equals c.FK\_UsersTable

where c.Active == true

&& c.CanView == true

select a).Distinct().ToList();

foreach (SecondLevelSubdivisionTable CurrentSecond in Faculties)

{

DataRow dataRow = dataTable.NewRow();

dataRow["StructName"] = CurrentSecond.Name;

dataRow["StructID"] = CurrentSecond.SecondLevelSubdivisionTableID;

int All = (from a in kpiWebDataContext.ThirdLevelSubdivisionTable

where a.Active == true

&& a.FK\_SecondLevelSubdivisionTable == CurrentSecond.SecondLevelSubdivisionTableID

select a).Count();

int AllConfirmed = (from a in kpiWebDataContext.ThirdLevelSubdivisionTable

where a.Active == true

&& a.FK\_SecondLevelSubdivisionTable == CurrentSecond.SecondLevelSubdivisionTableID

join b in kpiWebDataContext.CollectedBasicParametersTable

on a.ThirdLevelSubdivisionTableID equals b.FK\_ThirdLevelSubdivisionTable

where b.Status == 4

&& b.Active == true

&& b.FK\_ReportArchiveTable == ReportID

select a).Distinct().Count();

dataRow["Status"] = "Утвердили "+AllConfirmed.ToString() +" из "+ All.ToString();

dataTable.Rows.Add(dataRow);

}

GridView1.DataSource = dataTable;

GridView1.DataBind();

}

}

protected void ButtonDetailClick(object sender, EventArgs e)

{

Button button = (Button)sender;

{

Serialization paramSerialization = (Serialization)Session["ReportArchiveID"];

paramSerialization.l2 = Convert.ToInt32(button.CommandArgument.ToString());

Session["ReportArchiveID"] = paramSerialization; // запомнили в сессии второй уровень

Response.Redirect("~/Director/DViewThird.aspx");

}

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Filename: DReportView.aspx.designer.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//------------------------------------------------------------------------------

// <auto-generated>

// This code was generated by a tool.

//

// Changes to this file may cause incorrect behavior and will be lost if

// the code is regenerated.

// </auto-generated>

//------------------------------------------------------------------------------

namespace KPIWeb.Director {

public partial class DReportView {

/// <summary>

/// GridView1 control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.GridView GridView1;

/// <summary>

/// Button1 control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.Button Button1;

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Filename: DViewThird.aspx.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

using System.Data;

namespace KPIWeb.Director

{

public partial class DViewThird : System.Web.UI.Page

{

protected void Page\_Load(object sender, EventArgs e)

{

Serialization UserSer = (Serialization)Session["UserID"];

if (UserSer == null)

{

Response.Redirect("~/Default.aspx");

}

int userID = UserSer.Id;

ViewState["LocalUserID"] = userID;

KPIWebDataContext kpiWebDataContext = new KPIWebDataContext();

UsersTable userTable =

(from a in kpiWebDataContext.UsersTable where a.UsersTableID == userID select a).FirstOrDefault();

if (userTable.AccessLevel != 4)

{

Response.Redirect("~/Default.aspx");

}

Serialization paramSerialization = (Serialization)Session["ReportArchiveID"];

if (paramSerialization == null)

{

Response.Redirect("~/Default.aspx");

}

int ReportID = Convert.ToInt32(paramSerialization.ReportStr);

int SecondLevel = paramSerialization.l2;

if (!Page.IsPostBack)

{

//мы берем кафедры

DataTable dataTable = new DataTable();

dataTable.Columns.Add(new DataColumn("StructName", typeof(string)));

dataTable.Columns.Add(new DataColumn("StructID", typeof(string)));

dataTable.Columns.Add(new DataColumn("Status", typeof(string)));

List<ThirdLevelSubdivisionTable> Kafedras = (from a in kpiWebDataContext.ThirdLevelSubdivisionTable

where a.FK\_SecondLevelSubdivisionTable == SecondLevel

&& a.Active == true

join b in kpiWebDataContext.UsersTable

on a.ThirdLevelSubdivisionTableID equals b.FK\_ThirdLevelSubdivisionTable

where b.Active == true

join c in kpiWebDataContext.BasicParametrsAndUsersMapping

on b.UsersTableID equals c.FK\_UsersTable

where c.Active == true

&& c.CanView == true

select a).Distinct().ToList();

foreach (ThirdLevelSubdivisionTable CurrentThird in Kafedras)

{

DataRow dataRow = dataTable.NewRow();

dataRow["StructName"] = CurrentThird.Name;

dataRow["StructID"] = CurrentThird.ThirdLevelSubdivisionTableID;

int Statusn = (int) (from a in kpiWebDataContext.CollectedBasicParametersTable

where a.FK\_ReportArchiveTable == ReportID

&& a.Active == true

&& a.FK\_ThirdLevelSubdivisionTable == CurrentThird.ThirdLevelSubdivisionTableID

select a.Status).FirstOrDefault();

string status = "Нет данных";

if (Statusn == 4)

{

status = "Данные утверждены";

}

else if (Statusn == 3)

{

status = "Данные ожидают утверждения";

}

else if (Statusn == 2)

{

status = "Данные в процессе заполнения";

}

else if (Statusn == 1)

{

status = "Данные возвращены на доработку";

}

else if (Statusn == 0)

{

status = "Данные в процессе заполнения";

}

else

{

//error

}

dataRow["Status"] = status;

dataTable.Rows.Add(dataRow);

}

GridView1.DataSource = dataTable;

GridView1.DataBind();

}

}

protected void ButtonDetailClick(object sender, EventArgs e)

{

Button button = (Button)sender;

{

Serialization paramSerialization = (Serialization)Session["ReportArchiveID"];

paramSerialization.l3 = Convert.ToInt32(button.CommandArgument.ToString());

Session["ReportArchiveID"] = paramSerialization; // запомнили в сессии второй уровень

Response.Redirect("~/Director/ViewBasicParams.aspx");

}

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Filename: DViewThird.aspx.designer.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//------------------------------------------------------------------------------

// <auto-generated>

// This code was generated by a tool.

//

// Changes to this file may cause incorrect behavior and will be lost if

// the code is regenerated.

// </auto-generated>

//------------------------------------------------------------------------------

namespace KPIWeb.Director {

public partial class DViewThird {

/// <summary>

/// GridView1 control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.GridView GridView1;

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Filename: ViewBasicParams.aspx.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

using System.Data;

namespace KPIWeb.Director

{

public partial class ViewBasicParams : System.Web.UI.Page

{

protected void Page\_Load(object sender, EventArgs e)

{

Serialization UserSer = (Serialization)Session["UserID"];

if (UserSer == null)

{

Response.Redirect("~/Default.aspx");

}

int userID = UserSer.Id;

ViewState["LocalUserID"] = userID;

KPIWebDataContext kpiWebDataContext = new KPIWebDataContext();

UsersTable userTable =

(from a in kpiWebDataContext.UsersTable where a.UsersTableID == userID select a).FirstOrDefault();

if (userTable.AccessLevel != 4)

{

Response.Redirect("~/Default.aspx");

}

Serialization paramSerialization = (Serialization)Session["ReportArchiveID"];

if (paramSerialization == null)

{

Response.Redirect("~/Default.aspx");

}

int ReportID = Convert.ToInt32(paramSerialization.ReportStr);

int SecondLevel = paramSerialization.l2;

int ThirdLevel = paramSerialization.l3;

if (!Page.IsPostBack)

{

#region

List<string> columnNames = new List<string>(); // сюда сохраняем названия колонок

List<string> basicNames = new List<string>(); // сюда названия параметров для excel

/////создаем дататейбл

DataTable dataTable = new DataTable();

dataTable.Columns.Add(new DataColumn("BasicParametersTableID", typeof(string)));

dataTable.Columns.Add(new DataColumn("Name", typeof(string)));

dataTable.Columns.Add(new DataColumn("Comment", typeof(string)));

for (int k = 0; k <= 40; k++) //создаем кучу полей

{

dataTable.Columns.Add(new DataColumn("Value" + k.ToString(), typeof(string)));

}

#endregion

#region

/\* List<BasicParametersTable> KafBasicParams =

(from a in kpiWebDataContext.ReportArchiveAndBasicParametrsMappingTable

join b in kpiWebDataContext.BasicParametersTable

on a.FK\_BasicParametrsTable equals b.BasicParametersTableID

join c in kpiWebDataContext.BasicParametrsAndUsersMapping

on b.BasicParametersTableID equals c.FK\_ParametrsTable

join d in kpiWebDataContext.BasicParametrAdditional

on b.BasicParametersTableID equals d.BasicParametrAdditionalID

join f in kpiWebDataContext.CollectedBasicParametersTable

on a.FK\_BasicParametrsTable equals f.FK\_BasicParametersTable

where

a.FK\_ReportArchiveTable == ReportID //из нужного отчёта

&& f.FK\_ReportArchiveTable == ReportID

&& c.FK\_UsersTable == userID // свяный с пользователем

&& d.SubvisionLevel == 3 //нужный уровень заполняющего

&& a.Active == true // запись в таблице связей показателя и отчёта активна

&& c.CanView == true

&& c.Active == true

&& d.Calculated == false

select b).Distinct().ToList();\*/

List<BasicParametersTable> KafBasicParams = (from a in kpiWebDataContext.BasicParametersTable

where a.Active == true

join b in kpiWebDataContext.BasicParametrAdditional

on a.BasicParametersTableID equals b.BasicParametrAdditionalID

where b.Calculated == false

join c in kpiWebDataContext.CollectedBasicParametersTable

on a.BasicParametersTableID equals c.FK\_BasicParametersTable

where c.Active == true

&& c.CollectedValue != null

&& c.FK\_ThirdLevelSubdivisionTable == ThirdLevel

&& c.FK\_FourthLevelSubdivisionTable == null

select a).Distinct().ToList();

//узнали показатели кафедры(отчёт,разрешенияПользователя,Уровеньвводяшего,вводящийся показатель)

foreach (BasicParametersTable basicParam in KafBasicParams) //пройдемся по показателям

{

//если этото параметр и эта кафедра дружат

ThirdLevelParametrs thirdParametrs =

(from a in kpiWebDataContext.ThirdLevelParametrs

where a.ThirdLevelParametrsID == ThirdLevel

select a).FirstOrDefault();

// узнали параметры специальности

BasicParametrAdditional basicParametrs =

(from a in kpiWebDataContext.BasicParametrAdditional

where

a.BasicParametrAdditionalID == basicParam.BasicParametersTableID

select a).FirstOrDefault();

//узнали параметры базового показателя

if ((thirdParametrs.CanGraduate == true) || (basicParametrs.IsGraduating == false))

//фильтруем базовые показатели для невыпускающих кафедр

{

DataRow dataRow = dataTable.NewRow();

dataRow["BasicParametersTableID"] = basicParam.BasicParametersTableID;

dataRow["Name"] = basicParam.Name;

string comment\_ = (from a in kpiWebDataContext.BasicParametrAdditional

where a.BasicParametrAdditionalID == basicParam.BasicParametersTableID

&& a.Active == true

select a.Comment).FirstOrDefault();

if (comment\_ != null)

{

if (comment\_.Length > 3)

{

dataRow["Comment"] = comment\_;

}

else

{

dataRow["Comment"] = " ";

}

}

else

{

dataRow["Comment"] = " ";

}

basicNames.Add(basicParam.Name);

CollectedBasicParametersTable collectedBasicTmp =

(from a in kpiWebDataContext.CollectedBasicParametersTable

where a.FK\_ThirdLevelSubdivisionTable == ThirdLevel

&& a.FK\_BasicParametersTable == basicParam.BasicParametersTableID

&& a.FK\_ReportArchiveTable == ReportID

select a).FirstOrDefault();

if (collectedBasicTmp != null)

{

dataRow["Value0"] = collectedBasicTmp.CollectedValue.ToString();

}

else

{

dataRow["Value0"] = " ";

}

dataTable.Rows.Add(dataRow);

}

}

columnNames.Add((from a in kpiWebDataContext.ThirdLevelSubdivisionTable

where a.ThirdLevelSubdivisionTableID == ThirdLevel

select a.Name).FirstOrDefault());

#endregion

int additionalColumnCount = 1;

#region

if ((from zz in kpiWebDataContext.ThirdLevelParametrs

where zz.ThirdLevelParametrsID == ThirdLevel

select zz.CanGraduate).FirstOrDefault() == true)

// кафедра выпускающая значит специальности есть

{

List<BasicParametersTable> SpecBasicParams =

(from a in kpiWebDataContext.ReportArchiveAndBasicParametrsMappingTable

join b in kpiWebDataContext.BasicParametersTable

on a.FK\_BasicParametrsTable equals b.BasicParametersTableID

join c in kpiWebDataContext.BasicParametrsAndUsersMapping

on b.BasicParametersTableID equals c.FK\_ParametrsTable

join d in kpiWebDataContext.BasicParametrAdditional

on b.BasicParametersTableID equals d.BasicParametrAdditionalID

where a.FK\_ReportArchiveTable == ReportID //для отчёта

&& d.SubvisionLevel == 4 // для уровня заполняющего

&& d.Calculated == false //только вводимые параметры

&& c.FK\_UsersTable == userID // связаннаые с пользователем

&& a.Active == true

&& c.CanView == true

&& c.Active == true

select b).ToList();

//Получили показатели разрешенные пользователю в данном отчёте

List<FourthLevelSubdivisionTable> Specialzations =

(from a in kpiWebDataContext.FourthLevelSubdivisionTable

where a.FK\_ThirdLevelSubdivisionTable == ThirdLevel

&& a.Active == true

select a).ToList();

//Получили список специальностей для кафедры под пользователем

foreach (FourthLevelSubdivisionTable spec in Specialzations)

{

/\*

columnNames.Add("Направление подготовки\r" +

(from a in kpiWebDataContext.SpecializationTable

where a.SpecializationTableID == spec.FK\_Specialization

select a.Name).FirstOrDefault().ToString() +" : "+

(from a in kpiWebDataContext.SpecializationTable

where a.SpecializationTableID == spec.FK\_Specialization

select a.SpecializationNumber).FirstOrDefault().ToString());

\*/

string CurrentColumnName = "<div style=\"transform:rotate(90deg);\">" + (from a in kpiWebDataContext.SpecializationTable

where a.SpecializationTableID == spec.FK\_Specialization

select a.SpecializationNumber).FirstOrDefault().ToString() + "</div>";

columnNames.Add(CurrentColumnName);

//запомнили название специальности // оно нам пригодится)

}

foreach (BasicParametersTable specBasicParam in SpecBasicParams)

{

int i = additionalColumnCount;

DataRow dataRow = dataTable.NewRow();

BasicParametrAdditional basicParametrs =

(from a in kpiWebDataContext.BasicParametrAdditional

where

a.BasicParametrAdditionalID == specBasicParam.BasicParametersTableID

select a).FirstOrDefault();

//узнали параметры базового показателя

int j = 0;

//если хоть одной специальности базовый показатель нужен то мы его выведем

foreach (FourthLevelSubdivisionTable spec in Specialzations)

{

FourthLevelParametrs fourthParametrs =

(from a in kpiWebDataContext.FourthLevelParametrs

where a.FourthLevelParametrsID == spec.FourthLevelSubdivisionTableID

select a).FirstOrDefault();

// узнали параметры специальности

//если этото параметр и эта специальность дружат

if (((fourthParametrs.IsForeignStudentsAccept == true) ||

(basicParametrs.ForForeignStudents == false)) //это для иностранцев

&&

((fourthParametrs.SpecType == basicParametrs.SpecType) ||

(basicParametrs.SpecType == 0)))

// это для деления на магистров аспирантов итд

{

j++; //потом проверка и следовательно БП нуно выводить

CollectedBasicParametersTable collectedBasicTmp =

(from a in kpiWebDataContext.CollectedBasicParametersTable

where

a.FK\_BasicParametersTable ==

specBasicParam.BasicParametersTableID

&& a.FK\_ReportArchiveTable == ReportID

&&

(a.FK\_ThirdLevelSubdivisionTable ==

ThirdLevel)

&&

(a.FK\_FourthLevelSubdivisionTable ==

spec.FourthLevelSubdivisionTableID)

select a).FirstOrDefault();

dataRow["Value" + i] = collectedBasicTmp.CollectedValue.ToString();

}

i++;

}

if (j > 0)

{

basicNames.Add(specBasicParam.Name);

dataRow["Name"] = specBasicParam.Name;

dataRow["BasicParametersTableID"] = specBasicParam.BasicParametersTableID;

string comment\_ = (from a in kpiWebDataContext.BasicParametrAdditional

where a.BasicParametrAdditionalID == specBasicParam.BasicParametersTableID

&& a.Active == true

select a.Comment).FirstOrDefault();

if (comment\_ != null)

{

if (comment\_.Length > 3)

{

dataRow["Comment"] = comment\_;

}

else

{

dataRow["Comment"] = " ";

}

}

else

{

dataRow["Comment"] = " ";

}

dataTable.Rows.Add(dataRow);

}

///////////////////////закинули все в дататейбл

}

additionalColumnCount += Specialzations.Count;

}

#endregion

GridviewCollectedBasicParameters.DataSource = dataTable;

for (int j = 0; j < additionalColumnCount; j++)

{

GridviewCollectedBasicParameters.Columns[j + 3].Visible = true;

GridviewCollectedBasicParameters.Columns[j + 3].HeaderText = columnNames[j];

}

GridviewCollectedBasicParameters.DataBind();

}

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Filename: ViewBasicParams.aspx.designer.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//------------------------------------------------------------------------------

// <auto-generated>

// This code was generated by a tool.

//

// Changes to this file may cause incorrect behavior and will be lost if

// the code is regenerated.

// </auto-generated>

//------------------------------------------------------------------------------

namespace KPIWeb.Director {

public partial class ViewBasicParams {

/// <summary>

/// GridviewCollectedBasicParameters control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.GridView GridviewCollectedBasicParameters;

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Filename: OtdelChooseReport.aspx.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using System;

using System.Collections.Generic;

using System.Data;

using System.Linq;

using System.Web.UI.WebControls;

namespace KPIWeb.FinKadr

{

public partial class OtdelChooseReport : System.Web.UI.Page

{

protected void Page\_Load(object sender, EventArgs e)

{

Serialization UserSer = (Serialization)Session["UserID"];

if (UserSer == null)

{

Response.Redirect("~/Default.aspx");

}

int userID = UserSer.Id;

KPIWebDataContext kPiDataContext = new KPIWebDataContext();

UsersTable userTable =

(from a in kPiDataContext.UsersTable where a.UsersTableID == userID select a).FirstOrDefault();

ViewState["login"] =

(from a in kPiDataContext.UsersTable

where a.UsersTableID == userID

select a.Email).FirstOrDefault();

if (userTable.AccessLevel != 3)

{

Response.Redirect("~/Default.aspx");

}

OtdelHistorySession OtdelHistory = (OtdelHistorySession)Session["OtdelHistory"];

if (OtdelHistory == null)

{

GoForwardButton.Enabled = false;

}

//////////////////////////////////////////////////////////////////////////

if (!Page.IsPostBack)

{

ParametrType paramType = new ParametrType(1);

if (paramType == null)

{

Response.Redirect("~/Default.aspx");

}

if (paramType.paramType == 0) //смотрим индикацелевой показатель

{

PageName.Text = "Работа с целевыми показателями";

}

else

{

PageName.Text = "Работа с первичными данными";

}

KPIWebDataContext kpiWebDataContext = new KPIWebDataContext();

List<ReportArchiveTable> reportsArchiveTablesTable = null;

reportsArchiveTablesTable = (from a in kpiWebDataContext.ReportArchiveTable

select a).ToList();

DataTable dataTable = new DataTable();

dataTable.Columns.Add(new DataColumn("ReportArchiveID", typeof(string)));

dataTable.Columns.Add(new DataColumn("ReportName", typeof(string)));

dataTable.Columns.Add(new DataColumn("StartDate", typeof(string)));

dataTable.Columns.Add(new DataColumn("EndDate", typeof(string)));

foreach (ReportArchiveTable ReportRow in reportsArchiveTablesTable)

{

DataRow dataRow = dataTable.NewRow();

dataRow["ReportArchiveID"] = ReportRow.ReportArchiveTableID.ToString();

dataRow["ReportName"] = ReportRow.Name;

dataRow["StartDate"] = ReportRow.StartDateTime.ToString().Split(' ')[0];

dataRow["EndDate"] = ReportRow.EndDateTime.ToString().Split(' ')[0];

dataTable.Rows.Add(dataRow);

}

GridView1.DataSource = dataTable;

GridView1.DataBind();

}

}

protected void ButtonViewClick(object sender, EventArgs e)

{

Button button = (Button)sender;

{

ParametrType paramType = new ParametrType(1);

if (paramType == null)

{

Response.Redirect("~/Default.aspx");

}

if (paramType.paramType == 0) //смотрим индцелевой показатель

{

OtdelResult.Struct mainStruct = new OtdelResult.Struct(1, "");

OtdelSession OtdelResultSession = new OtdelSession(mainStruct, 1, 0, 0,

Convert.ToInt32(button.CommandArgument), 0);

OtdelHistorySession OtdelHistory = new OtdelHistorySession();

OtdelHistory.SessionCount = 1;

OtdelHistory.CurrentSession = 0;

OtdelHistory.Visible = false;

OtdelHistory.OtdelSession[OtdelHistory.CurrentSession] = OtdelResultSession;

Session["OtdelHistory"] = OtdelHistory;

LogHandler.LogWriter.WriteLog(LogCategory.INFO, "OCR0: Проректор " + (string)ViewState["login"] + " перешел к работе с отчетом, ID = " + button.CommandArgument);

Response.Redirect("~/FinKadr/OtdelResult.aspx");

}

else // смотрим рассчетные

{

OtdelResult.Struct mainStruct = new OtdelResult.Struct(1, "");

OtdelSession OtdelResultSession = new OtdelSession(mainStruct, 1, 0, 1,

Convert.ToInt32(button.CommandArgument), 0);

OtdelHistorySession OtdelHistory = new OtdelHistorySession();

OtdelHistory.SessionCount = 1;

OtdelHistory.CurrentSession = 0;

OtdelHistory.Visible = false;

OtdelHistory.OtdelSession[OtdelHistory.CurrentSession] = OtdelResultSession;

Session["OtdelHistory"] = OtdelHistory;

LogHandler.LogWriter.WriteLog(LogCategory.INFO, "0OCR1: Проректор " + (string)ViewState["login"] + " перешел к работе с отчетом, ID = " + button.CommandArgument);

Response.Redirect("~/FinKadr/OtdelResult.aspx");

}

}

}

protected void GoBackButton\_Click(object sender, EventArgs e)

{

// Response.Redirect("~/Rector/RectorMain.aspx");

}

protected void GoForwardButton\_Click(object sender, EventArgs e)

{

Response.Redirect("~/FinKadr/OtdelResult.aspx");

}

protected void Button4\_Click(object sender, EventArgs e)

{

//Response.Redirect("~/Rector/RectorMain.aspx");

}

protected void Button5\_Click(object sender, EventArgs e)

{

//Response.Redirect("~/Rector/ViewDocument.aspx");

}

protected void Button6\_Click(object sender, EventArgs e)

{

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Filename: OtdelChooseReport.aspx.designer.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//------------------------------------------------------------------------------

// <auto-generated>

// This code was generated by a tool.

//

// Changes to this file may cause incorrect behavior and will be lost if

// the code is regenerated.

// </auto-generated>

//------------------------------------------------------------------------------

namespace KPIWeb.FinKadr {

public partial class OtdelChooseReport {

/// <summary>

/// top\_panel2 control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.Panel top\_panel2;

/// <summary>

/// GoBackButton control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.Button GoBackButton;

/// <summary>

/// GoForwardButton control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.Button GoForwardButton;

/// <summary>

/// Button2 control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.Button Button2;

/// <summary>

/// PageName control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.Label PageName;

/// <summary>

/// GridView1 control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.GridView GridView1;

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Filename: OtdelResult.aspx.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using System;

using System.Collections.Generic;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Web.UI.WebControls;

using System.Web.WebPages;

using Button = System.Web.UI.WebControls.Button;

using DataTable = System.Data.DataTable;

using Label = System.Web.UI.WebControls.Label;

namespace KPIWeb.FinKadr

{

public partial class OtdelResult : System.Web.UI.Page

{

[Serializable]

public class Struct // класс описываюший структурные подразделения

{

public int Lv\_0 { get; set; }

public int Lv\_1 { get; set; }

public int Lv\_2 { get; set; }

public int Lv\_3 { get; set; }

public int Lv\_4 { get; set; }

public int Lv\_5 { get; set; }

public string Name { get; set; }

public Struct(int lv0, int lv1, int lv2, int lv3, int lv4, int lv5, string name)

{

Lv\_0 = lv0;

Lv\_1 = lv1;

Lv\_2 = lv2;

Lv\_3 = lv3;

Lv\_4 = lv4;

Lv\_5 = lv5;

Name = name;

}

public Struct(int lv0, int lv1, int lv2, int lv3, int lv4, string name)

{

Lv\_0 = lv0;

Lv\_1 = lv1;

Lv\_2 = lv2;

Lv\_3 = lv3;

Lv\_4 = lv4;

Lv\_5 = 0;

Name = name;

}

public Struct(int lv0, int lv1, int lv2, int lv3, string name)

{

Lv\_0 = lv0;

Lv\_1 = lv1;

Lv\_2 = lv2;

Lv\_3 = lv3;

Lv\_4 = 0;

Lv\_5 = 0;

Name = name;

}

public Struct(int lv0, int lv1, int lv2, string name)

{

Lv\_0 = lv0;

Lv\_1 = lv1;

Lv\_2 = lv2;

Lv\_3 = 0;

Lv\_4 = 0;

Lv\_5 = 0;

Name = name;

}

public Struct(int lv0, int lv1, string name)

{

Lv\_0 = lv0;

Lv\_1 = lv1;

Lv\_2 = 0;

Lv\_3 = 0;

Lv\_4 = 0;

Lv\_5 = 0;

Name = name;

}

public Struct(int lv0, string name)

{

Lv\_0 = lv0;

Lv\_1 = 0;

Lv\_2 = 0;

Lv\_3 = 0;

Lv\_4 = 0;

Lv\_5 = 0;

Name = name;

}

}

public List<Struct> GetChildStructList(Struct ParentStruct)

{

List<Struct> tmpStrucList = new List<Struct>();

int Level = 5;

Level = ParentStruct.Lv\_5 == 0 ? 4 : Level;

Level = ParentStruct.Lv\_4 == 0 ? 3 : Level;

Level = ParentStruct.Lv\_3 == 0 ? 2 : Level;

Level = ParentStruct.Lv\_2 == 0 ? 1 : Level;

Level = ParentStruct.Lv\_1 == 0 ? 0 : Level;

Level = ParentStruct.Lv\_0 == 0 ? -1 : Level;

KPIWebDataContext kpiWebDataContext = new KPIWebDataContext();

switch (Level)

{

case -1: // возвращаем все нулевого уровня, хотя там должен быть только КФУ

{

tmpStrucList = (from a in kpiWebDataContext.ZeroLevelSubdivisionTable

where a.Active == true

select new Struct(1, "")

{

Lv\_0 = (int)a.ZeroLevelSubdivisionTableID,

Lv\_1 = 0,

Lv\_2 = 0,

Lv\_3 = 0,

Lv\_4 = 0,

Lv\_5 = 0,

Name = a.Name

}).ToList();

break;

}

case 0: // возвращаем все универы

{

tmpStrucList = (from a in kpiWebDataContext.FirstLevelSubdivisionTable

where a.Active == true

&& a.FK\_ZeroLevelSubvisionTable == ParentStruct.Lv\_0

select new Struct(1, "")

{

Lv\_0 = (int)a.FK\_ZeroLevelSubvisionTable,

Lv\_1 = (int)a.FirstLevelSubdivisionTableID,

Lv\_2 = 0,

Lv\_3 = 0,

Lv\_4 = 0,

Lv\_5 = 0,

Name = a.Name

}).ToList();

break;

}

case 1: // возвращаем все факультеты

{

tmpStrucList = (from a in kpiWebDataContext.FirstLevelSubdivisionTable

join b in kpiWebDataContext.SecondLevelSubdivisionTable

on a.FirstLevelSubdivisionTableID equals b.FK\_FirstLevelSubdivisionTable

where a.Active == true

&& b.Active == true

&& a.FK\_ZeroLevelSubvisionTable == ParentStruct.Lv\_0

&& b.FK\_FirstLevelSubdivisionTable == ParentStruct.Lv\_1

select new Struct(1, "")

{

Lv\_0 = (int)a.FK\_ZeroLevelSubvisionTable,

Lv\_1 = (int)a.FirstLevelSubdivisionTableID,

Lv\_2 = (int)b.SecondLevelSubdivisionTableID,

Lv\_3 = 0,

Lv\_4 = 0,

Lv\_5 = 0,

Name = b.Name

}).ToList();

break;

}

case 2: // возвращаем все кафедры

{

tmpStrucList = (from a in kpiWebDataContext.FirstLevelSubdivisionTable

join b in kpiWebDataContext.SecondLevelSubdivisionTable

on a.FirstLevelSubdivisionTableID equals b.FK\_FirstLevelSubdivisionTable

join c in kpiWebDataContext.ThirdLevelSubdivisionTable

on b.SecondLevelSubdivisionTableID equals c.FK\_SecondLevelSubdivisionTable

where a.Active == true

&& b.Active == true

&& c.Active == true

&& a.FK\_ZeroLevelSubvisionTable == ParentStruct.Lv\_0

&& b.FK\_FirstLevelSubdivisionTable == ParentStruct.Lv\_1

&& c.FK\_SecondLevelSubdivisionTable == ParentStruct.Lv\_2

select new Struct(1, "")

{

Lv\_0 = (int)a.FK\_ZeroLevelSubvisionTable,

Lv\_1 = (int)a.FirstLevelSubdivisionTableID,

Lv\_2 = (int)b.SecondLevelSubdivisionTableID,

Lv\_3 = (int)c.ThirdLevelSubdivisionTableID,

Lv\_4 = 0,

Lv\_5 = 0,

Name = c.Name

}).ToList();

break;

}

case 3: // возвращаем все специальности

{

tmpStrucList = (from a in kpiWebDataContext.FirstLevelSubdivisionTable

join b in kpiWebDataContext.SecondLevelSubdivisionTable

on a.FirstLevelSubdivisionTableID equals b.FK\_FirstLevelSubdivisionTable

join c in kpiWebDataContext.ThirdLevelSubdivisionTable

on b.SecondLevelSubdivisionTableID equals c.FK\_SecondLevelSubdivisionTable

join d in kpiWebDataContext.FourthLevelSubdivisionTable

on c.ThirdLevelSubdivisionTableID equals d.FK\_ThirdLevelSubdivisionTable

where a.Active == true

&& b.Active == true

&& c.Active == true

&& a.FK\_ZeroLevelSubvisionTable == ParentStruct.Lv\_0

&& b.FK\_FirstLevelSubdivisionTable == ParentStruct.Lv\_1

&& c.FK\_SecondLevelSubdivisionTable == ParentStruct.Lv\_2

&& d.FK\_ThirdLevelSubdivisionTable == ParentStruct.Lv\_3

select new Struct(1, "")

{

Lv\_0 = (int)a.FK\_ZeroLevelSubvisionTable,

Lv\_1 = (int)a.FirstLevelSubdivisionTableID,

Lv\_2 = (int)b.SecondLevelSubdivisionTableID,

Lv\_3 = (int)c.ThirdLevelSubdivisionTableID,

Lv\_4 = (int)d.FourthLevelSubdivisionTableID,

Lv\_5 = 0,

Name = d.Name

}).ToList();

break;

}

default:

{

//error не будем раскладывать до специальностей

break;

}

}

return tmpStrucList;

}

public List<Struct> GetChildStructList(Struct ParentStruct, int SpecID)

{

List<Struct> tmpStrucList = new List<Struct>();

int Level = 5;

Level = ParentStruct.Lv\_5 == 0 ? 4 : Level;

Level = ParentStruct.Lv\_4 == 0 ? 3 : Level;

Level = ParentStruct.Lv\_3 == 0 ? 2 : Level;

Level = ParentStruct.Lv\_2 == 0 ? 1 : Level;

Level = ParentStruct.Lv\_1 == 0 ? 0 : Level;

Level = ParentStruct.Lv\_0 == 0 ? -1 : Level;

KPIWebDataContext kpiWebDataContext = new KPIWebDataContext();

switch (Level)

{

case -1: // возвращаем все нулевого уровня, хотя там должен быть только КФУ

{

tmpStrucList = (from a in kpiWebDataContext.ZeroLevelSubdivisionTable

where a.Active == true

select new Struct(1,"") {

Lv\_0 = (int)a.ZeroLevelSubdivisionTableID,

Lv\_1 = 0,

Lv\_2 = 0,

Lv\_3 = 0,

Lv\_4 = 0,

Lv\_5 = 0,

Name = a.Name

}).OrderBy(x => x.Lv\_0).ToList();

break;

}

case 0: // возвращаем все универы

{

tmpStrucList = (from a in kpiWebDataContext.FirstLevelSubdivisionTable

join b in kpiWebDataContext.SecondLevelSubdivisionTable

on a.FirstLevelSubdivisionTableID equals b.FK\_FirstLevelSubdivisionTable

join c in kpiWebDataContext.ThirdLevelSubdivisionTable

on b.SecondLevelSubdivisionTableID equals c.FK\_SecondLevelSubdivisionTable

join d in kpiWebDataContext.FourthLevelSubdivisionTable

on c.ThirdLevelSubdivisionTableID equals d.FK\_ThirdLevelSubdivisionTable

where a.Active == true

&& a.FK\_ZeroLevelSubvisionTable == ParentStruct.Lv\_0

&& ((SpecID == 0) || (SpecID == d.FK\_Specialization))

select new Struct(1,"")

{

Lv\_0 = (int)a.FK\_ZeroLevelSubvisionTable,

Lv\_1 = (int)a.FirstLevelSubdivisionTableID,

Lv\_2 = 0,

Lv\_3 = 0,

Lv\_4 = 0,

Lv\_5 = 0,

Name = a.Name

}).OrderBy(x => x.Lv\_1).ToList();

break;

}

case 1: // возвращаем все факультеты

{

tmpStrucList = (from a in kpiWebDataContext.FirstLevelSubdivisionTable

join b in kpiWebDataContext.SecondLevelSubdivisionTable

on a.FirstLevelSubdivisionTableID equals b.FK\_FirstLevelSubdivisionTable

join c in kpiWebDataContext.ThirdLevelSubdivisionTable

on b.SecondLevelSubdivisionTableID equals c.FK\_SecondLevelSubdivisionTable

join d in kpiWebDataContext.FourthLevelSubdivisionTable

on c.ThirdLevelSubdivisionTableID equals d.FK\_ThirdLevelSubdivisionTable

where a.Active == true

&& b.Active == true

&& a.FK\_ZeroLevelSubvisionTable == ParentStruct.Lv\_0

&& b.FK\_FirstLevelSubdivisionTable == ParentStruct.Lv\_1

&& ((SpecID == 0) || (SpecID == d.FK\_Specialization))

select new Struct(1,"")

{

Lv\_0 = (int)a.FK\_ZeroLevelSubvisionTable,

Lv\_1 = (int)a.FirstLevelSubdivisionTableID,

Lv\_2 = (int)b.SecondLevelSubdivisionTableID,

Lv\_3 = 0,

Lv\_4 = 0,

Lv\_5 = 0,

Name = b.Name

}).OrderBy(x => x.Lv\_2).ToList();

break;

}

case 2: // возвращаем все кафедры

{

tmpStrucList = (from a in kpiWebDataContext.FirstLevelSubdivisionTable

join b in kpiWebDataContext.SecondLevelSubdivisionTable

on a.FirstLevelSubdivisionTableID equals b.FK\_FirstLevelSubdivisionTable

join c in kpiWebDataContext.ThirdLevelSubdivisionTable

on b.SecondLevelSubdivisionTableID equals c.FK\_SecondLevelSubdivisionTable

join e in kpiWebDataContext.FourthLevelSubdivisionTable

on c.ThirdLevelSubdivisionTableID equals e.FK\_ThirdLevelSubdivisionTable

where a.Active == true

&& b.Active == true

&& c.Active == true

&& a.FK\_ZeroLevelSubvisionTable == ParentStruct.Lv\_0

&& b.FK\_FirstLevelSubdivisionTable == ParentStruct.Lv\_1

&& c.FK\_SecondLevelSubdivisionTable == ParentStruct.Lv\_2

&& ((SpecID == 0) || (SpecID == e.FK\_Specialization))

select new Struct(1,"")

{

Lv\_0 = (int)a.FK\_ZeroLevelSubvisionTable,

Lv\_1 = (int)a.FirstLevelSubdivisionTableID,

Lv\_2 = (int)b.SecondLevelSubdivisionTableID,

Lv\_3 = (int)c.ThirdLevelSubdivisionTableID,

Lv\_4 = 0,

Lv\_5 = 0,

Name = c.Name

}).OrderBy(x => x.Lv\_3).ToList();

break;

}

default:

{

//error не будем раскладывать до специальностей

break;

}

}

List<Struct> uniqeStruct = new List<Struct>();

foreach (Struct curStruct in tmpStrucList)

{

if (uniqeStruct.Count == 0)

{

uniqeStruct.Add(curStruct);

}

else

{

if ((uniqeStruct[uniqeStruct.Count - 1].Lv\_0 != curStruct.Lv\_0)||

(uniqeStruct[uniqeStruct.Count - 1].Lv\_1 != curStruct.Lv\_1)||

(uniqeStruct[uniqeStruct.Count - 1].Lv\_2 != curStruct.Lv\_2)||

(uniqeStruct[uniqeStruct.Count - 1].Lv\_3 != curStruct.Lv\_3)||

(uniqeStruct[uniqeStruct.Count - 1].Lv\_4 != curStruct.Lv\_4))

{

uniqeStruct.Add(curStruct);

}

}

}

return uniqeStruct;

}

public class MyObject

{

public int Id;

public int ParentId;

public string Name;

public string UrlAddr;

public int Active;

}

public float GetCalculatedWithParams(Struct StructToCalcFor, int ParamType, int ParamID,int ReportID, int SpecID) // читает показатель

{

KPIWebDataContext kpiWebDataContext = new KPIWebDataContext();

float result = 0;

if (ParamType == 0) // считаем целевой показатель

{

IndicatorsTable Indicator = (from a in kpiWebDataContext.IndicatorsTable

where a.IndicatorsTableID == ParamID

select a).FirstOrDefault();

return

(float) CalculateAbb.CalculateForLevel(1, Abbreviature.CalculatedAbbToFormula(Indicator.Formula)

, ReportID, SpecID, StructToCalcFor.Lv\_0

, StructToCalcFor.Lv\_1, StructToCalcFor.Lv\_2, StructToCalcFor.Lv\_3, StructToCalcFor.Lv\_4,

StructToCalcFor.Lv\_5, 0);

}

else if (ParamType == 1) // считаем рассчетный

{

CalculatedParametrs Calculated = (from a in kpiWebDataContext.CalculatedParametrs

where a.CalculatedParametrsID == ParamID

select a).FirstOrDefault();

return (float)CalculateAbb.CalculateForLevel(1, Calculated.Formula, ReportID, SpecID , StructToCalcFor.Lv\_0

, StructToCalcFor.Lv\_1, StructToCalcFor.Lv\_2, StructToCalcFor.Lv\_3, StructToCalcFor.Lv\_4,

StructToCalcFor.Lv\_5, 0);

}

else if (ParamType == 2) // суммируем базовый

{

return (float) CalculateAbb.SumForLevel(ParamID, ReportID,SpecID, StructToCalcFor.Lv\_0

, StructToCalcFor.Lv\_1, StructToCalcFor.Lv\_2, StructToCalcFor.Lv\_3, StructToCalcFor.Lv\_4,

StructToCalcFor.Lv\_5);

}

else

{

//error

}

return result;

}

public int StructDeepness(Struct CurrentStruct)

{

int tmp=0;

if (CurrentStruct.Lv\_0 != 0) tmp++;

if (CurrentStruct.Lv\_1 != 0) tmp++;

if (CurrentStruct.Lv\_2 != 0) tmp++;

if (CurrentStruct.Lv\_3 != 0) tmp++;

if (CurrentStruct.Lv\_4 != 0) tmp++;

if (CurrentStruct.Lv\_5 != 0) tmp++;

return tmp;

}

public Struct StructDeeper(Struct parentStruct, int nextID)

{

KPIWebDataContext kpiWebDataContext = new KPIWebDataContext();

int lv0 = parentStruct.Lv\_0;

int lv1 = parentStruct.Lv\_1;

int lv2 = parentStruct.Lv\_2;

int lv3 = parentStruct.Lv\_3;

int lv4 = parentStruct.Lv\_4;

int lv5 = parentStruct.Lv\_5;

string name = parentStruct.Name;

Struct tmp = new Struct(lv0,lv1,lv2,lv3,lv4,lv5,name);

if (tmp.Lv\_0 == 0)

{

tmp.Lv\_0 = nextID;

tmp.Name = (from a in kpiWebDataContext.ZeroLevelSubdivisionTable

where a.ZeroLevelSubdivisionTableID == nextID

select a.Name).FirstOrDefault();

return tmp;

}

if (tmp.Lv\_1 == 0)

{

tmp.Lv\_1 = nextID;

tmp.Name = (from a in kpiWebDataContext.FirstLevelSubdivisionTable

where a.FirstLevelSubdivisionTableID == nextID

select a.Name).FirstOrDefault();

return tmp;

}

if (tmp.Lv\_2 == 0)

{

tmp.Lv\_2 = nextID;

tmp.Name = (from a in kpiWebDataContext.SecondLevelSubdivisionTable

where a.SecondLevelSubdivisionTableID == nextID

select a.Name).FirstOrDefault();

return tmp;

}

if (tmp.Lv\_3 == 0)

{

tmp.Lv\_3 = nextID;

tmp.Name = (from a in kpiWebDataContext.ThirdLevelSubdivisionTable

where a.ThirdLevelSubdivisionTableID == nextID

select a.Name).FirstOrDefault();

return tmp;

}

if (tmp.Lv\_4 == 0)

{

tmp.Lv\_4 = nextID;

tmp.Name = (from a in kpiWebDataContext.FourthLevelSubdivisionTable

where a.FourthLevelSubdivisionTableID == nextID

select a.Name).FirstOrDefault();

return tmp;

}

if (tmp.Lv\_5 == 0)

{

tmp.Lv\_5 = nextID;

tmp.Name = (from a in kpiWebDataContext.FifthLevelSubdivisionTable

where a.FifthLevelSubdivisionTableID == nextID

select a.Name).FirstOrDefault();

return tmp;

}

return tmp;

} //добавляет ID к первому в структуре нулю

public int GetLastID(Struct currentStruct)

{

if (currentStruct.Lv\_0 == 0)

{

return 0;

}

if (currentStruct.Lv\_1 == 0)

{

return currentStruct.Lv\_0;

}

if (currentStruct.Lv\_2 == 0)

{

return currentStruct.Lv\_1;

}

if (currentStruct.Lv\_3 == 0)

{

return currentStruct.Lv\_2;

}

if (currentStruct.Lv\_4 == 0)

{

return currentStruct.Lv\_3;

}

return 0;

} //определяет последнее не нулевое значение в структуре

public float CalculatedForDB(float input)

{

float tmp = (float) input;

if ((tmp < -(float)1E+20) || (tmp > (float)1E+20)

|| (tmp == null) || (float.IsNaN(tmp))

|| (float.IsInfinity(tmp)) || (float.IsNegativeInfinity(tmp))

|| (float.IsPositiveInfinity(tmp)) || (!tmp.ToString().IsFloat()))

{

tmp = (float)1E+20;

}

return tmp;

}

protected void Page\_Load(object sender, EventArgs e)

{

#region get user data

Panel5.Style.Add("background-color", "rgba(0, 255, 0, 0.3)");

Panel7.Style.Add("background-color", "rgba(255, 0, 0, 0.3)");

Panel6.Style.Add("background-color", "rgba(255, 255, 0, 0.3)");

string parameter = Request["\_\_EVENTARGUMENT"];

if (parameter != null)

{

int ParamId = -1;

if (int.TryParse(parameter, out ParamId) && ParamId > 0)

{

DoConfirm(ParamId);

}

}

KPIWebDataContext kpiWebDataContext = new KPIWebDataContext();

Serialization UserSer = (Serialization)Session["UserID"];

if (UserSer == null)

{

Response.Redirect("~/Default.aspx");

}

int userID = UserSer.Id;

UsersTable userTable\_ =

(from a in kpiWebDataContext.UsersTable where a.UsersTableID == userID select a).FirstOrDefault();

ViewState["login"] = (from a in kpiWebDataContext.UsersTable where a.UsersTableID == userID select a.Email).FirstOrDefault();

if (userTable\_.AccessLevel != 3)

{

Response.Redirect("~/Default.aspx");

}

#endregion

if (!IsPostBack)

{

#region session

OtdelHistorySession OtdelHistory = (OtdelHistorySession)Session["OtdelHistory"];

if (OtdelHistory == null)

{

Response.Redirect("~/Default.aspx");

}

OtdelShowUnConfirmed unConfirmed = (OtdelShowUnConfirmed)Session["unConfirmed"];

bool ShowUnconfirmed = true;

if (unConfirmed == null)

{

ShowUnconfirmed = false;

}

else

{

if (unConfirmed.DoShowUnConfirmed == false)

{

ShowUnconfirmed = false;

}

else

{

unConfirmed.DoShowUnConfirmed = false;

Session["unConfirmed"] = unConfirmed;

}

}

#region check for get

string val = this.Request.QueryString["HLevel"]; //hisoty level сова придумал)

if (val != null)

{

OtdelHistory.CurrentSession = Convert.ToInt32(val);

Session["OtdelHistory"] = OtdelHistory;

}

#endregion

if ((OtdelHistory.SessionCount - OtdelHistory.CurrentSession) < 2)

{

GoForwardButton.Enabled = false;

}

OtdelSession CurrentOtdelSession = OtdelHistory.OtdelSession[OtdelHistory.CurrentSession];

Struct mainStruct = CurrentOtdelSession.sesStruct;

int ViewType = CurrentOtdelSession.sesViewType;

int ParamID = CurrentOtdelSession.sesParamID;

int ParamType = CurrentOtdelSession.sesParamType;

int ReportID = CurrentOtdelSession.sesReportID;

int SpecID = CurrentOtdelSession.sesSpecID;

#endregion

#region DataTable init

DataTable dataTable = new DataTable();

dataTable.Columns.Add(new DataColumn("ID", typeof (string)));

dataTable.Columns.Add(new DataColumn("Number", typeof (string)));

dataTable.Columns.Add(new DataColumn("Abb", typeof (string)));

dataTable.Columns.Add(new DataColumn("Name", typeof (string)));

dataTable.Columns.Add(new DataColumn("StartDate", typeof (string)));

dataTable.Columns.Add(new DataColumn("EndDate", typeof (string)));

dataTable.Columns.Add(new DataColumn("Value", typeof (string)));

dataTable.Columns.Add(new DataColumn("Title", typeof (string)));

dataTable.Columns.Add(new DataColumn("PlannedValue", typeof (string)));

dataTable.Columns.Add(new DataColumn("Progress", typeof (string)));

dataTable.Columns.Add(new DataColumn("Color", typeof (string)));

///color table

/// 0 - no color // can't confirm

/// 1 - green (confirmed)

/// 2 - red (unconfirmed but calculated)

/// 3 - orange (can confirm)

///

dataTable.Columns.Add(new DataColumn("CanWatchWhoOws", typeof(bool)));

dataTable.Columns.Add(new DataColumn("CanConfirm", typeof (bool)));

dataTable.Columns.Add(new DataColumn("ShowLable", typeof (bool)));

dataTable.Columns.Add(new DataColumn("LableText", typeof (string)));

dataTable.Columns.Add(new DataColumn("LableColor", typeof (string)));

#endregion

#region global page settings

ReportArchiveTable ReportTable = (from a in kpiWebDataContext.ReportArchiveTable

where a.ReportArchiveTableID == ReportID

select a).FirstOrDefault();

double daysLeft = ((DateTime) ReportTable.EndDateTime - DateTime.Now).TotalDays;

ReportTitle.Text = ReportTable.Name + " " + ReportTable.StartDateTime.ToString().Split(' ')[0] + " - " +

ReportTable.EndDateTime.ToString().Split(' ')[0];

#endregion

#region Show Uncinfirmed Button

#endregion

if (ViewType == 0) // просмотр для структурных подразделений

{

#region преднастройка страницы

string title = "";

PageFullName.Text = "";

if (ParamType == 0)

{

PageFullName.Text += "<b>";

PageFullName.Text += (from a in kpiWebDataContext.IndicatorsTable

where a.IndicatorsTableID == ParamID

select a.Name).FirstOrDefault();

PageFullName.Text += "</b> </br>";

}

else if (ParamType == 1)

{

PageFullName.Text += "<b>";

PageFullName.Text += (from a in kpiWebDataContext.CalculatedParametrs

where a.CalculatedParametrsID == ParamID

select a.Name).FirstOrDefault();

PageFullName.Text += "</b> </br>";

}

else if (ParamType == 2)

{

PageFullName.Text += "<b>";

PageFullName.Text += (from a in kpiWebDataContext.BasicParametersTable

where a.BasicParametersTableID == ParamID

select a.Name).FirstOrDefault();

PageFullName.Text += "</b> </br>";

}

int Deep = StructDeepness(mainStruct);

if (Deep == 1)

{

}

if (Deep == 2)

{

PageFullName.Text += (from a in kpiWebDataContext.FirstLevelSubdivisionTable

where a.FirstLevelSubdivisionTableID == mainStruct.Lv\_1

select a.Name).FirstOrDefault();

PageFullName.Text += "</br>";

}

if (Deep == 3)

{

Grid.Columns[10].Visible = false; // В общем скрываем кнопку детализации до уровня специальностей

PageFullName.Text += (from a in kpiWebDataContext.FirstLevelSubdivisionTable

where a.FirstLevelSubdivisionTableID == mainStruct.Lv\_1

select a.Name).FirstOrDefault();

PageFullName.Text += "</br>";

PageFullName.Text += (from a in kpiWebDataContext.SecondLevelSubdivisionTable

where a.SecondLevelSubdivisionTableID == mainStruct.Lv\_2

select a.Name).FirstOrDefault();

PageFullName.Text += "</br>";

}

if (Deep == 4)

{

PageFullName.Text += (from a in kpiWebDataContext.FirstLevelSubdivisionTable

where a.FirstLevelSubdivisionTableID == mainStruct.Lv\_1

select a.Name).FirstOrDefault();

PageFullName.Text += "</br>";

PageFullName.Text += (from a in kpiWebDataContext.SecondLevelSubdivisionTable

where a.SecondLevelSubdivisionTableID == mainStruct.Lv\_2

select a.Name).FirstOrDefault();

PageFullName.Text += "</br>";

PageFullName.Text += (from a in kpiWebDataContext.ThirdLevelSubdivisionTable

where a.ThirdLevelSubdivisionTableID == mainStruct.Lv\_3

select a.Name).FirstOrDefault();

PageFullName.Text += "</br>";

}

if (SpecID != 0)

{

PageFullName.Text += "Направление подготовки \"" + (from a in kpiWebDataContext.SpecializationTable

where a.SpecializationTableID == SpecID

select a.Name).FirstOrDefault() + "\" </br>";

}

if (mainStruct.Lv\_1 == 0)

{

OtdelSession tmpses = new OtdelSession(mainStruct, ViewType, ParamID, ParamType, ReportID,

SpecID, "КФУ");

OtdelHistory.OtdelSession[OtdelHistory.CurrentSession] = tmpses;

Session["OtdelHistory"] = OtdelHistory;

}

else

{

OtdelSession tmpses = new OtdelSession(mainStruct, ViewType, ParamID, ParamType, ReportID,

SpecID, mainStruct.Name);

OtdelHistory.OtdelSession[OtdelHistory.CurrentSession] = tmpses;

Session["OtdelHistory"] = OtdelHistory;

}

//задади имя текущей сессии

title = "Подразделения";

if (StructDeepness(mainStruct) > 3)

{

title = "Направления подготовки";

}

#endregion

#region fill grid

int BasicParamLevel = 0;

if (ParamType == 2)

{

BasicParamLevel = (int)(from a in kpiWebDataContext.BasicParametrAdditional

select a.SubvisionLevel).FirstOrDefault();

}

List<Struct> currentStructList = new List<Struct>();

if (SpecID != 0)

{

currentStructList = GetChildStructList(mainStruct, SpecID);

}

else

{

currentStructList = GetChildStructList(mainStruct);

}

foreach (Struct currentStruct in currentStructList)

{

DataRow dataRow = dataTable.NewRow();

dataRow["ID"] = GetLastID(currentStruct).ToString();

dataRow["Number"] = "num";

dataRow["Name"] = currentStruct.Name;

dataRow["StartDate"] = "nun";

dataRow["EndDate"] = "nun";

dataRow["CanConfirm"] = true;

dataRow["ShowLable"] = false;

dataRow["CanWatchWhoOws"] = false;

dataRow["LableText"] = "";

dataRow["LableColor"] = "#000000";

dataRow["Value"] =

GetCalculatedWithParams(currentStruct, ParamType, ParamID, ReportID, SpecID).ToString();

dataTable.Rows.Add(dataRow);

}

#endregion

#region DataGridBind

Grid.DataSource = dataTable;

Grid.Columns[3].HeaderText = title;

Grid.DataBind();

#endregion

#region постнастройка страницы

Grid.Columns[12].Visible = false;

Grid.Columns[11].Visible = false;

Grid.Columns[9].Visible = false;

Grid.Columns[8].Visible = false;

Grid.Columns[7].Visible = false;

Grid.Columns[5].Visible = false;

Grid.Columns[4].Visible = false;

Grid.Columns[2].Visible = false;

Grid.Columns[1].Visible = false;

if ((StructDeepness(mainStruct) > (BasicParamLevel-1))||

(StructDeepness(mainStruct) > 2 )&&(SpecID!=0)) // дальше углубляться нельзя

{

Grid.Columns[10].Visible = false;

}

#endregion

}

else if (ViewType == 1) // просмотр для показателей (верхние 3 шт)

{

#region преднастройка страницы

string name\_text = "";

string value\_text = "";

string progress\_text = "";

string confirm\_text = "";

string detalize\_text = "";

if (ParamType == 0)

{

OtdelSession tmpses = new OtdelSession(mainStruct, ViewType, ParamID, ParamType, ReportID,

SpecID, "Значения целевых показателей для КФУ");

OtdelHistory.OtdelSession[OtdelHistory.CurrentSession] = tmpses;

Session["OtdelHistory"] = OtdelHistory;

PageFullName.Text = "Значения целевых показателей (ЦП) для КФУ";

//PageName.Text = "Значения индикторов для КФУ";

name\_text = "Название ЦП";

value\_text = "Значение ЦП";

progress\_text = "Степень готовности первичных данных";

confirm\_text = "Утвердить ЦП";

detalize\_text = "Просмотреть первичные данные для ЦП";

}

else if (ParamType == 1)

{

if (ParamID == 0)

{

OtdelSession tmpses = new OtdelSession(mainStruct, ViewType, ParamID, ParamType, ReportID,

SpecID, "Значения первичных данных (ПД) для КФУ");

OtdelHistory.OtdelSession[OtdelHistory.CurrentSession] = tmpses;

Session["OtdelHistory"] = OtdelHistory;

PageFullName.Text = "Значения первичных данных (ПД) для КФУ";

}

else

{

string tmp = (from a in kpiWebDataContext.IndicatorsTable

where a.IndicatorsTableID == ParamID

select a.Name).FirstOrDefault();

OtdelSession tmpses = new OtdelSession(mainStruct, ViewType, ParamID, ParamType, ReportID,

SpecID, "Первичные данные для целевого показателя: " + tmp);

OtdelHistory.OtdelSession[OtdelHistory.CurrentSession] = tmpses;

Session["OtdelHistory"] = OtdelHistory;

PageFullName.Text = "Первичные данные (ПД) целевого показателя <b> \"" + tmp + "\"</b> для КФУ";

}

name\_text = "Названия ПД";

value\_text = "Значение ПД";

progress\_text = "Степень готовности базовых показателей";

confirm\_text = "Утвердить ПД";

detalize\_text = "Просмотреть базовые показатели для ПД";

}

else if (ParamType == 2)

{

string tmp = (from a in kpiWebDataContext.CalculatedParametrs

where a.CalculatedParametrsID == ParamID

select a.Name).FirstOrDefault();

OtdelSession tmpses = new OtdelSession(mainStruct, ViewType, ParamID, ParamType, ReportID,

SpecID, "Базовые показатели для первич: " + tmp);

OtdelHistory.OtdelSession[OtdelHistory.CurrentSession] = tmpses;

Session["OtdelHistory"] = OtdelHistory;

PageFullName.Text = "Базовые показатели первичного показателя <b> \"" + tmp + "\"</b> для КФУ";

name\_text = "Названия БП";

value\_text = "Значение БП";

//progress\_text = "Степень готовности базовых показателей";

//confirm\_text = "Утвердить ПД";

//Ыdetalize\_text = "Просмотреть базовые показатели для ПД";

}

#endregion

#region fill grid

if (ParamType == 0) //считаем целевой показатель

{

#region indicator

List<IndicatorsTable> Indicators = (

from a in kpiWebDataContext.IndicatorsTable

join b in kpiWebDataContext.IndicatorsAndUsersMapping

on a.IndicatorsTableID equals b.FK\_IndicatorsTable

join c in kpiWebDataContext.ReportArchiveAndIndicatorsMappingTable

on a.IndicatorsTableID equals c.FK\_IndicatorsTable

where

a.Active == true

&& b.CanView == true

&& b.FK\_UsresTable == userID

&& c.FK\_ReportArchiveTable == ReportID

select a).OrderBy(c => c.SortID).ToList();

//нашли все целевой показатель привязанные к пользователю

foreach (IndicatorsTable CurrentIndicator in Indicators)

{

DataRow dataRow = dataTable.NewRow();

dataRow["ID"] = CurrentIndicator.IndicatorsTableID; //GetLastID(currentStruct).ToString();

dataRow["Number"] = "num";

dataRow["Name"] = CurrentIndicator.Name;

dataRow["CanWatchWhoOws"] = false;

dataRow["StartDate"] = "nun";

dataRow["EndDate"] = "nun";

PlannedIndicator plannedValue = (from a in kpiWebDataContext.PlannedIndicator

where a.FK\_IndicatorsTable == CurrentIndicator.IndicatorsTableID

&& a.Date > DateTime.Now

//Q1

select a).OrderBy(x => x.Date).FirstOrDefault();

if (plannedValue != null)

{

dataRow["PlannedValue"] = plannedValue.Value;

}

else

{

dataRow["PlannedValue"] = "Не определено";

}

#region user can confirm

bool canConfirm = (bool) (from a in kpiWebDataContext.IndicatorsAndUsersMapping

where a.FK\_IndicatorsTable == CurrentIndicator.IndicatorsTableID

&& a.FK\_UsresTable == userID

select a.CanConfirm).FirstOrDefault();

#endregion

# region are calculated confirmed

List<CalculatedParametrs> CalculatedList =

Abbreviature.GetCalculatedList(CurrentIndicator.Formula);

bool CalcAreConfirmed = true;

int AllCalculated = 0;

int AllConfirmedCalculated = 0;

foreach (CalculatedParametrs CurrentCalculated in CalculatedList)

{

AllCalculated++;

CollectedCalculatedParametrs tmp\_ =

(from a in kpiWebDataContext.CollectedCalculatedParametrs

where

a.FK\_CalculatedParametrs == CurrentCalculated.CalculatedParametrsID

&& a.FK\_ReportArchiveTable == ReportID

select a).FirstOrDefault();

if (tmp\_ == null)

{

CalcAreConfirmed = false;

}

else

{

if (tmp\_.Confirmed == null)

{

CalcAreConfirmed = false;

}

else if (tmp\_.Confirmed == false)

{

CalcAreConfirmed = false;

}

else

{

AllConfirmedCalculated++;

}

}

}

dataRow["Progress"] = AllConfirmedCalculated.ToString() + " из " + AllCalculated.ToString();

#endregion

#region get calculated if confirmed; calculate if not confirmed

string value\_ = "";

CollectedIndocators collected = (from a in kpiWebDataContext.CollectedIndocators

where a.FK\_ReportArchiveTable == ReportID

&& a.FK\_Indicators == CurrentIndicator.IndicatorsTableID

select a).FirstOrDefault();

if (collected == null)

{

collected = new CollectedIndocators();

collected.FK\_Indicators = CurrentIndicator.IndicatorsTableID;

collected.FK\_ReportArchiveTable = ReportID;

collected.FK\_UsersTable = userID;

collected.Confirmed = false;

collected.LastChangeDateTime = DateTime.Now;

collected.Active = true;

collected.CollectedValue = 0;

kpiWebDataContext.CollectedIndocators.InsertOnSubmit(collected);

kpiWebDataContext.SubmitChanges();

}

if (collected.Confirmed == true) // данные подтверждены

{

dataRow["CanConfirm"] = false;

dataRow["ShowLable"] = true;

dataRow["LableText"] = "Утверждено";

dataRow["Color"] = "1"; // confirmed

value\_ = ((float) collected.CollectedValue).ToString("0.00");

}

else // данные уже есть но еще не подтверждены

{

if (canConfirm == false)

{

dataRow["CanConfirm"] = false;

dataRow["ShowLable"] = true;

dataRow["LableText"] = " "; // нет права утверждать

dataRow["LableColor"] = "#101010";

value\_ = "Недостаточно данных";

if (ShowUnconfirmed)

{

dataRow["Color"] = "2";

/// 0 - no color // can't confirm

/// 1 - green (confirmed)

/// 2 - red (unconfirmed but calculated)

/// 3 - orange (can confirm)

///

float tmp =

CalculatedForDB(GetCalculatedWithParams(mainStruct, ParamType,

CurrentIndicator.IndicatorsTableID, ReportID, SpecID));

if (tmp == (float) 1E+20)

{

value\_ = "Рассчет невозможен";

}

else

{

value\_ = tmp.ToString("0.00");

}

}

}

else if (!CalcAreConfirmed)

{

/\*dataRow["CanConfirm"] = false;

dataRow["ShowLable"] = true;

dataRow["LableText"] = "Не все расчетные утверждены";

\*/

dataRow["CanConfirm"] = true;

dataRow["ShowLable"] = false;

dataRow["LableText"] = "";

dataRow["LableColor"] = "#101010";

value\_ = "Недостаточно данных";

if (ShowUnconfirmed)

{

dataRow["Color"] = "2";

float tmp =

CalculatedForDB(GetCalculatedWithParams(mainStruct, ParamType,

CurrentIndicator.IndicatorsTableID, ReportID, SpecID));

if (tmp == (float) 1E+20)

{

value\_ = "Рассчет невозможен";

}

else

{

value\_ = tmp.ToString("0.00");

}

}

}

else

{

dataRow["CanConfirm"] = true;

dataRow["ShowLable"] = false;

dataRow["LableText"] = "";

dataRow["LableColor"] = "#FFFFFF";

dataRow["Color"] = "3";

collected.CollectedValue =

CalculatedForDB(GetCalculatedWithParams(mainStruct, ParamType,

CurrentIndicator.IndicatorsTableID, ReportID, SpecID));

//12;

kpiWebDataContext.SubmitChanges();

value\_ = ((float) collected.CollectedValue).ToString("0.00");

}

}

dataRow["Value"] = value\_;

#endregion

dataTable.Rows.Add(dataRow);

}

#endregion indicator

}

if (ParamType == 1) //показываем рассчетный входящий в ID целевой показатель

{

#region calculated

//ID - это айди Индиктора

List<CalculatedParametrs> CalculatedList;

if (ParamID != 0)

{

IndicatorsTable Indicator = (from a in kpiWebDataContext.IndicatorsTable

where a.IndicatorsTableID == ParamID

select a).FirstOrDefault();

CalculatedList = Abbreviature.GetCalculatedList(Indicator.Formula);

}

else

{

CalculatedList = (from a in kpiWebDataContext.CalculatedParametrs

join b in kpiWebDataContext.CalculatedParametrsAndUsersMapping

on a.CalculatedParametrsID equals b.FK\_CalculatedParametrsTable

join c in kpiWebDataContext.ReportArchiveAndCalculatedParametrsMappingTable

on a.CalculatedParametrsID equals c.FK\_CalculatedParametrsTable

where a.Active == true

&& b.CanView == true

&& b.FK\_UsersTable == userID

&& c.FK\_ReportArchiveTable == ReportID

select a).ToList();

}

foreach (CalculatedParametrs CurrentCalculated in CalculatedList)

{

DataRow dataRow = dataTable.NewRow();

dataRow["ID"] = CurrentCalculated.CalculatedParametrsID;

//GetLastID(currentStruct).ToString();

dataRow["Number"] = "num";

dataRow["Name"] = CurrentCalculated.Name;

dataRow["StartDate"] = "nun";

dataRow["EndDate"] = "nun";

// dataRow["CanWatchWhoOws"] = "false";

// dataRow["CanConfirm"] = "true";

// dataRow["ShowLable"] = "false";

dataRow["Abb"] = CurrentCalculated.AbbreviationEN;

#region get calculated if confirmed; calculate if not confirmed

#region user can edit

bool canConfirm = (bool) (from a in kpiWebDataContext.CalculatedParametrsAndUsersMapping

where a.FK\_CalculatedParametrsTable == CurrentCalculated.CalculatedParametrsID

&& a.FK\_UsersTable == userID

select a.CanConfirm).FirstOrDefault();

#endregion

#region check if all users confirmed basics

List<BasicParametersTable> BasicList = Abbreviature.GetBasicList(CurrentCalculated.Formula);

int AllBasicsUsersCanEdit = 0;

int AllConfirmedBasics = 0;

foreach (BasicParametersTable Basic in BasicList)

{

/\*

AllBasicsUsersCanEdit += (from a in kpiWebDataContext.BasicParametrsAndUsersMapping

join b in kpiWebDataContext.UsersTable

on a.FK\_UsersTable equals b.UsersTableID

where a.CanEdit == true

&& b.Active == true

&& a.FK\_ParametrsTable == Basic.BasicParametersTableID

select a).Count();

AllConfirmedBasics += (from a in kpiWebDataContext.CollectedBasicParametersTable

where a.FK\_ReportArchiveTable == ReportID

&& a.FK\_BasicParametersTable == Basic.BasicParametersTableID

&& a.Status == 4

select a).Count();\*/

AllBasicsUsersCanEdit += (from a in kpiWebDataContext.CollectedBasicParametersTable

where a.FK\_ReportArchiveTable == ReportID

&& a.FK\_BasicParametersTable == Basic.BasicParametersTableID

select a).Count();

AllConfirmedBasics += (from a in kpiWebDataContext.CollectedBasicParametersTable

where a.FK\_ReportArchiveTable == ReportID

&& a.FK\_BasicParametersTable == Basic.BasicParametersTableID

&& a.Status == 4

select a).Count();

}

bool BasicsAreConfirmed = true;

if (AllBasicsUsersCanEdit != AllConfirmedBasics)

{

BasicsAreConfirmed = false;

}

#endregion

dataRow["Progress"] =

((((float) AllConfirmedBasics)\*100)/((float) AllBasicsUsersCanEdit)).ToString("0.0") +

"%";

string value\_ = "";

CollectedCalculatedParametrs collected =

(from a in kpiWebDataContext.CollectedCalculatedParametrs

where a.FK\_ReportArchiveTable == ReportID

&& a.FK\_CalculatedParametrs == CurrentCalculated.CalculatedParametrsID

select a).FirstOrDefault();

if (collected == null) // данных нет

{

collected = new CollectedCalculatedParametrs();

collected.FK\_CalculatedParametrs = CurrentCalculated.CalculatedParametrsID;

collected.FK\_ReportArchiveTable = ReportID;

collected.FK\_UsersTable = userID;

collected.Confirmed = false;

collected.LastChangeDateTime = DateTime.Now;

collected.Active = true;

collected.CollectedValue = GetCalculatedWithParams(mainStruct, ParamType,

CurrentCalculated.CalculatedParametrsID, ReportID, SpecID); //11;

kpiWebDataContext.CollectedCalculatedParametrs.InsertOnSubmit(collected);

kpiWebDataContext.SubmitChanges();

}

UsersTable ConfirmUser = (from a in kpiWebDataContext.UsersTable

join b in kpiWebDataContext.CalculatedParametrsAndUsersMapping

on a.UsersTableID equals b.FK\_UsersTable

where a.Active == true

&& b.FK\_CalculatedParametrsTable == CurrentCalculated.CalculatedParametrsID

&& b.Active == true

select a).FirstOrDefault();

string UserName = "";

if (ConfirmUser.Position != null)

{

if (ConfirmUser.Position.Length > 2)

{

UserName = ConfirmUser.Position;

}

else

{

UserName = ConfirmUser.Email;

}

}

else

{

UserName = ConfirmUser.Email;

}

if (collected.Confirmed == true) //данные подтверждены

{

dataRow["CanWatchWhoOws"] = false;

dataRow["CanConfirm"] = false;

dataRow["ShowLable"] = true;

dataRow["LableText"] = "Утверждено ";

dataRow["LableColor"] = Color.LawnGreen;

dataRow["Color"] = "1";

value\_ = ((float) collected.CollectedValue).ToString("0.00");

}

else // данные есть но не подтверждены

{

if (canConfirm == false) //

{

dataRow["CanWatchWhoOws"] = false;

dataRow["CanConfirm"] = false;

dataRow["ShowLable"] = true;

dataRow["LableText"] = "Ответственный: " + UserName;

value\_ = "Недостаточно данных";

if (ShowUnconfirmed)

{

dataRow["Color"] = "2";

float tmp =

CalculatedForDB(GetCalculatedWithParams(mainStruct, ParamType,

CurrentCalculated.CalculatedParametrsID, ReportID, SpecID));

if (tmp == (float) 1E+20)

{

value\_ = "Рассчет невозможен";

}

else

{

value\_ = tmp.ToString("0.00");

}

}

}

else if (BasicsAreConfirmed == false)

{

dataRow["CanWatchWhoOws"] = true;

dataRow["CanConfirm"] = false;

dataRow["ShowLable"] = false;

dataRow["LableText"] = "";

/\*

dataRow["CanConfirm"] = false;

dataRow["ShowLable"] = true;

dataRow["LableText"] = "Не все базовые показатели утверждены";\*/

dataRow["LableColor"] = Color.LightBlue;

value\_ = "Недостаточно данных";

if (ShowUnconfirmed)

{

dataRow["Color"] = "2";

float tmp =

CalculatedForDB(GetCalculatedWithParams(mainStruct, ParamType,

CurrentCalculated.CalculatedParametrsID, ReportID, SpecID));

if (tmp == (float) 1E+20)

{

value\_ = "Рассчет невозможен";

}

else

{

value\_ = tmp.ToString("0.00");

}

}

}

else

{

dataRow["CanConfirm"] = true;

dataRow["CanWatchWhoOws"] = false;

dataRow["ShowLable"] = false;

dataRow["LableText"] = "";

dataRow["Color"] = "3";

dataRow["LableColor"] = "#000000";

collected.CollectedValue =

CalculatedForDB(GetCalculatedWithParams(mainStruct, ParamType,

CurrentCalculated.CalculatedParametrsID, ReportID, SpecID));

kpiWebDataContext.SubmitChanges();

value\_ = ((float) collected.CollectedValue).ToString("0.00");

}

}

dataRow["Value"] = value\_;

#endregion

dataTable.Rows.Add(dataRow);

}

#endregion

}

if (ParamType == 2) //

{

#region basic

//ID - Рассчетного айдишник

CalculatedParametrs Calculated = (from a in kpiWebDataContext.CalculatedParametrs

where a.CalculatedParametrsID == ParamID

select a).FirstOrDefault();

List<BasicParametersTable> BasicList = Abbreviature.GetBasicList(Calculated.Formula);

foreach (BasicParametersTable CurrebtBasic in BasicList)

{

DataRow dataRow = dataTable.NewRow();

dataRow["ID"] = CurrebtBasic.BasicParametersTableID; //GetLastID(currentStruct).ToString();

dataRow["Number"] = "num";

dataRow["Name"] = CurrebtBasic.Name;

dataRow["StartDate"] = "nun";

dataRow["EndDate"] = "nun";

dataRow["Abb"] = CurrebtBasic.AbbreviationEN;

dataRow["CanWatchWhoOws"] = false;

dataRow["CanConfirm"] = true;

dataRow["ShowLable"] = false;

dataRow["LableText"] = "";

dataRow["LableColor"] = "#000000";

dataRow["Value"] =

GetCalculatedWithParams(mainStruct, ParamType, CurrebtBasic.BasicParametersTableID,

ReportID, SpecID).ToString();

dataTable.Rows.Add(dataRow);

}

#endregion

}

#endregion

#region DataGridBind

Grid.DataSource = dataTable;

Grid.Columns[3].HeaderText = name\_text;

Grid.Columns[6].HeaderText = value\_text;

Grid.Columns[8].HeaderText = progress\_text;

Grid.Columns[9].HeaderText = confirm\_text;

Grid.Columns[11].HeaderText = detalize\_text;

Grid.DataBind();

#endregion

#region постнастройки страницы

Grid.Columns[5].Visible = false;

Grid.Columns[4].Visible = false;

Grid.Columns[1].Visible = false;

if (ParamType == 0)

{

Grid.Columns[2].Visible = false;

Grid.Columns[10].Visible = false; //

Grid.Columns[12].Visible = false; //

}

if (ParamType == 1)

{

if (ParamID == 0)

{

Grid.Columns[2].Visible = false;

}

Grid.Columns[10].Visible = false; //

Grid.Columns[12].Visible = false; //

Grid.Columns[7].Visible = false;

}

if (ParamType == 2) // дальше углубляться нельзя

{

Grid.Columns[7].Visible = false;

Grid.Columns[9].Visible = false;

Grid.Columns[8].Visible = false;

Grid.Columns[11].Visible = false;

}

#endregion

}

else if (ViewType == 2) // просмотр по специальностям

{

#region преднастройка страницы

if (ParamType == 0)

{

OtdelSession tmpses = new OtdelSession(mainStruct, ViewType, ParamID, ParamType, ReportID,

SpecID, "Целевой показатель (ЦП) для направления подготовки");

OtdelHistory.OtdelSession[OtdelHistory.CurrentSession] = tmpses;

Session["OtdelHistory"] = OtdelHistory;

}

else if (ParamType == 1)

{

OtdelSession tmpses = new OtdelSession(mainStruct, ViewType, ParamID, ParamType, ReportID,

SpecID, "Первичные данные (ПД) для направления подготовки");

OtdelHistory.OtdelSession[OtdelHistory.CurrentSession] = tmpses;

Session["OtdelHistory"] = OtdelHistory;

}

else if (ParamType == 2)

{

OtdelSession tmpses = new OtdelSession(mainStruct, ViewType, ParamID, ParamType, ReportID,

SpecID, "Базовый показатель (БП) для направления подготовки");

OtdelHistory.OtdelSession[OtdelHistory.CurrentSession] = tmpses;

Session["OtdelHistory"] = OtdelHistory;

string tmp = (from a in kpiWebDataContext.BasicParametersTable

where a.BasicParametersTableID == ParamID

select a.Name).FirstOrDefault();

PageFullName.Text = "Значения базового показателя (БП) <b> \"" + tmp + "\" </b> по направлениям подготовки для КФУ";

}

string title = "Направления подготовки";

#endregion

#region fill grid

List<SpecializationTable> SpecTable = (from a in kpiWebDataContext.SpecializationTable

join b in kpiWebDataContext.FourthLevelSubdivisionTable

on a.SpecializationTableID equals b.FK\_Specialization

where a.Active == true

&& b.Active == true

select a).OrderBy(mc => mc.SpecializationTableID).ToList();

//взяли все специальности которые привязаны к кафедрам

int old = 0;

foreach (SpecializationTable currentSpec in SpecTable)

{

if (currentSpec.SpecializationTableID != old)

{

DataRow dataRow = dataTable.NewRow();

dataRow["ID"] = currentSpec.SpecializationTableID; //GetLastID(currentStruct).ToString();

dataRow["Number"] = "num";

dataRow["Name"] = currentSpec.Name + ": " + (from a in kpiWebDataContext.SpecializationTable where a.SpecializationTableID == currentSpec.SpecializationTableID select a.SpecializationNumber).FirstOrDefault().ToString() +" "+ Action.EncodeToStr((from a in kpiWebDataContext.SpecializationTable where a.SpecializationTableID == currentSpec.SpecializationTableID select a.SpecializationNumber).FirstOrDefault().ToString()); //currentStruct.Name; // Шифр добавить!!

dataRow["StartDate"] = "nun";

dataRow["EndDate"] = "nun";

dataRow["CanWatchWhoOws"] = false;

dataRow["CanConfirm"] = true;

dataRow["ShowLable"] = false;

dataRow["LableText"] = "";

dataRow["LableColor"] = "#000000";

dataRow["Value"] =

GetCalculatedWithParams(mainStruct, ParamType, ParamID, ReportID,

currentSpec.SpecializationTableID).ToString();

dataTable.Rows.Add(dataRow);

}

else

{

}

old = currentSpec.SpecializationTableID;

}

#endregion

#region DataGridBind

Grid.DataSource = dataTable;

Grid.Columns[3].HeaderText = title;

Grid.DataBind();

#endregion

#region постнастройка страницы

Grid.Columns[12].Visible = false;

Grid.Columns[11].Visible = false;

Grid.Columns[7].Visible = false;

Grid.Columns[9].Visible = false;

Grid.Columns[8].Visible = false;

Grid.Columns[5].Visible = false;

Grid.Columns[4].Visible = false;

Grid.Columns[2].Visible = false;

Grid.Columns[1].Visible = false;

#endregion

}

else if (ViewType == 3)

{

#region

PageFullName.Text = "";

PageFullName.Text += "<b>";

PageFullName.Text += (from a in kpiWebDataContext.CalculatedParametrs

where a.CalculatedParametrsID == ParamID

select a.Name).FirstOrDefault();

PageFullName.Text += "</b> </br>";

int Deep = StructDeepness(mainStruct);

if (Deep == 1)

{

}

if (Deep == 2)

{

PageFullName.Text += (from a in kpiWebDataContext.FirstLevelSubdivisionTable

where a.FirstLevelSubdivisionTableID == mainStruct.Lv\_1

select a.Name).FirstOrDefault();

PageFullName.Text += "</br>";

}

if (Deep == 3)

{

PageFullName.Text += (from a in kpiWebDataContext.FirstLevelSubdivisionTable

where a.FirstLevelSubdivisionTableID == mainStruct.Lv\_1

select a.Name).FirstOrDefault();

PageFullName.Text += "</br>";

PageFullName.Text += (from a in kpiWebDataContext.SecondLevelSubdivisionTable

where a.SecondLevelSubdivisionTableID == mainStruct.Lv\_2

select a.Name).FirstOrDefault();

PageFullName.Text += "</br>";

}

if (Deep == 4)

{

PageFullName.Text += (from a in kpiWebDataContext.FirstLevelSubdivisionTable

where a.FirstLevelSubdivisionTableID == mainStruct.Lv\_1

select a.Name).FirstOrDefault();

PageFullName.Text += "</br>";

PageFullName.Text += (from a in kpiWebDataContext.SecondLevelSubdivisionTable

where a.SecondLevelSubdivisionTableID == mainStruct.Lv\_2

select a.Name).FirstOrDefault();

PageFullName.Text += "</br>";

PageFullName.Text += (from a in kpiWebDataContext.ThirdLevelSubdivisionTable

where a.ThirdLevelSubdivisionTableID == mainStruct.Lv\_3

select a.Name).FirstOrDefault();

PageFullName.Text += "</br>";

}

#endregion

#region fill grid

CalculatedParametrs Calculated = (from a in kpiWebDataContext.CalculatedParametrs

where a.CalculatedParametrsID == ParamID

select a).FirstOrDefault();

List<BasicParametersTable> BasicList = Abbreviature.GetBasicList(Calculated.Formula);

List<Struct> currentStructList = new List<Struct>();

currentStructList = GetChildStructList(mainStruct);

foreach (Struct currentStruct in currentStructList)

{

DataRow dataRow = dataTable.NewRow();

dataRow["ID"] = GetLastID(currentStruct).ToString();

dataRow["Number"] = "num";

dataRow["Name"] = currentStruct.Name;

dataRow["StartDate"] = "nun";

dataRow["EndDate"] = "nun";

dataRow["CanConfirm"] = false;

dataRow["ShowLable"] = false;

dataRow["CanWatchWhoOws"] = false;

dataRow["LableText"] = "";

dataRow["LableColor"] = "#000000";

dataRow["Value"] = "nun";

#region check if all users confirmed basics

int AllBasicsUsersCanEdit = 0;

int AllConfirmedBasics = 0;

foreach (BasicParametersTable Basic in BasicList)

{

/\*

List<UsersTable> UserTableList = (from a in kpiWebDataContext.UsersTable

join b in kpiWebDataContext.BasicParametrsAndUsersMapping

on a.UsersTableID equals b.FK\_UsersTable

join c in kpiWebDataContext.BasicParametrAdditional

on b.FK\_ParametrsTable equals c.BasicParametrAdditionalID

where a.Active == true

&& c.Calculated == false

&& b.Active == true

&& b.CanEdit == true

&& b.FK\_ParametrsTable == Basic.BasicParametersTableID

&& ((a.FK\_ZeroLevelSubdivisionTable == currentStruct.Lv\_0) || (currentStruct.Lv\_0 == 0))

&& ((a.FK\_FirstLevelSubdivisionTable == currentStruct.Lv\_1) || (currentStruct.Lv\_1 == 0))

&& ((a.FK\_SecondLevelSubdivisionTable == currentStruct.Lv\_2) || (currentStruct.Lv\_2 == 0))

&& ((a.FK\_ThirdLevelSubdivisionTable == currentStruct.Lv\_3) || (currentStruct.Lv\_3 == 0))

// && ((a.FK\_FourthLevelSubdivisionTable == currentStruct.Lv\_4) || (currentStruct.Lv\_4 == 0))

select a).ToList();

int SpecCnt = 0;

BasicParametrAdditional basicAdditional =

(from a in kpiWebDataContext.BasicParametrAdditional

where a.BasicParametrAdditionalID == Basic.BasicParametersTableID

select a).FirstOrDefault();

if (basicAdditional.SubvisionLevel != 4)

{

AllBasicsUsersCanEdit += UserTableList.Count();

}

else

{

foreach (UsersTable CurUSer in UserTableList)

{

SpecCnt += (from a in kpiWebDataContext.FourthLevelSubdivisionTable

where a.FK\_ThirdLevelSubdivisionTable == CurUSer.FK\_ThirdLevelSubdivisionTable

&& a.Active == true

select a).Count();

}

AllBasicsUsersCanEdit += (UserTableList.Count() \* SpecCnt);

}

\*/

AllBasicsUsersCanEdit += (from a in kpiWebDataContext.CollectedBasicParametersTable

where a.FK\_ReportArchiveTable == ReportID

&& a.FK\_BasicParametersTable == Basic.BasicParametersTableID

&& ((a.FK\_ZeroLevelSubdivisionTable == currentStruct.Lv\_0) || (currentStruct.Lv\_0 == 0))

&& ((a.FK\_FirstLevelSubdivisionTable == currentStruct.Lv\_1) || (currentStruct.Lv\_1 == 0))

&& ((a.FK\_SecondLevelSubdivisionTable == currentStruct.Lv\_2) || (currentStruct.Lv\_2 == 0))

&& ((a.FK\_ThirdLevelSubdivisionTable == currentStruct.Lv\_3) || (currentStruct.Lv\_3 == 0))

&& ((a.FK\_FourthLevelSubdivisionTable == currentStruct.Lv\_4) || (currentStruct.Lv\_4 == 0))

select a).Count();

AllConfirmedBasics += (from a in kpiWebDataContext.CollectedBasicParametersTable

where a.FK\_ReportArchiveTable == ReportID

&& a.FK\_BasicParametersTable == Basic.BasicParametersTableID

&& a.Status == 4

&& ((a.FK\_ZeroLevelSubdivisionTable == currentStruct.Lv\_0) || (currentStruct.Lv\_0 == 0))

&& ((a.FK\_FirstLevelSubdivisionTable == currentStruct.Lv\_1) || (currentStruct.Lv\_1 == 0))

&& ((a.FK\_SecondLevelSubdivisionTable == currentStruct.Lv\_2) || (currentStruct.Lv\_2 == 0))

&& ((a.FK\_ThirdLevelSubdivisionTable == currentStruct.Lv\_3) || (currentStruct.Lv\_3 == 0))

&& ((a.FK\_FourthLevelSubdivisionTable == currentStruct.Lv\_4) || (currentStruct.Lv\_4 == 0))

select a).Count();

}

bool BasicsAreConfirmed = true;

if (AllBasicsUsersCanEdit != AllConfirmedBasics)

{

BasicsAreConfirmed = false;

}

#endregion

if (AllBasicsUsersCanEdit == 0)

{

dataRow["Progress"] = "";

}

else

{

dataRow["Progress"] =

((((float) AllConfirmedBasics)\*100)/((float) AllBasicsUsersCanEdit)).ToString("0.0") +"%";

dataTable.Rows.Add(dataRow);

}

// dataTable.Rows.Add(dataRow);

}

#endregion

#region DataGridBind

Grid.DataSource = dataTable;

Grid.Columns[3].HeaderText = "Подразделения";

if (StructDeepness(mainStruct) > 3)

{

Grid.Columns[3].HeaderText = "Направления подготовки";

}

Grid.DataBind();

#endregion

#region постнастройка страницы

Grid.Columns[12].Visible = false;

Grid.Columns[11].Visible = false;

Grid.Columns[9].Visible = false;

// Grid.Columns[8].Visible = false;

Grid.Columns[7].Visible = false;

Grid.Columns[6].Visible = false;

Grid.Columns[5].Visible = false;

Grid.Columns[4].Visible = false;

Grid.Columns[2].Visible = false;

Grid.Columns[1].Visible = false;

if (StructDeepness(mainStruct) >2)

{

Grid.Columns[10].Visible = false;

}

#endregion

}

else

{

//error // wrong ViewType

}

RefreshHistory();

if ((ViewType == 1) && ((ParamType == 0) || (ParamType == 1)))

{

ClientScript.RegisterStartupScript(this.GetType(), "ShowLegend", "ShowLegend()", true);

}

}

}

protected void ButtonConfirmClick(object sender, EventArgs e)

{

/\*

KPIWebDataContext kpiWebDataContext = new KPIWebDataContext();

OtdelHistorySession OtdelHistory = (OtdelHistorySession)Session["OtdelHistory"];

if (OtdelHistory == null)

{

Response.Redirect("~/Default.aspx");

}

OtdelSession CurrentOtdelSession = OtdelHistory.OtdelSession[OtdelHistory.CurrentSession];

int ParamType = CurrentOtdelSession.sesParamType;

if (ParamType == 0) // indicator

{

CollectedIndocators Indicator = (from a in kpiWebDataContext.CollectedIndocators

where a.FK\_Indicators == Convert.ToInt32(button.CommandArgument)

select a).FirstOrDefault();

Indicator.Confirmed = true;

kpiWebDataContext.SubmitChanges();

Response.Redirect("~/Otdel/Result.aspx");

}

else if (ParamType == 1) // calculated

{

CollectedCalculatedParametrs Calculated = (from a in kpiWebDataContext.CollectedCalculatedParametrs

where a.FK\_CalculatedParametrs == Convert.ToInt32(button.CommandArgument)

select a).FirstOrDefault();

Calculated.Confirmed = true;

kpiWebDataContext.SubmitChanges();

Response.Redirect("~/Otdel/Result.aspx");

}

}\*/

}

protected void Button1Click(object sender, EventArgs e) //по структуре

{

Button button = (Button)sender;

{

OtdelHistorySession OtdelHistory = (OtdelHistorySession)Session["OtdelHistory"];

if (OtdelHistory == null)

{

Response.Redirect("~/Default.aspx");

}

OtdelSession CurrentOtdelSession = OtdelHistory.OtdelSession[OtdelHistory.CurrentSession];

Struct mainStruct = CurrentOtdelSession.sesStruct;

int ViewType = CurrentOtdelSession.sesViewType;

int ParamID = CurrentOtdelSession.sesParamID;

int ParamType = CurrentOtdelSession.sesParamType;

int ReportID = CurrentOtdelSession.sesReportID;

int SpecID = CurrentOtdelSession.sesSpecID;

OtdelSession currentOtdelSession = new OtdelSession(mainStruct, ViewType, ParamID, ParamType, ReportID, SpecID,"");

if (currentOtdelSession.sesViewType == 1)

{//впервые перешли на разложение по структуре сразу после показателя

currentOtdelSession.sesParamID = Convert.ToInt32(button.CommandArgument);

currentOtdelSession.sesViewType = 0;

currentOtdelSession.sesStruct.Lv\_0 = 1;

currentOtdelSession.sesStruct.Lv\_1 = 0;

currentOtdelSession.sesStruct.Lv\_2 = 0;

currentOtdelSession.sesStruct.Lv\_3 = 0;

currentOtdelSession.sesStruct.Lv\_4 = 0;

currentOtdelSession.sesStruct.Lv\_5 = 0;

}

else if (currentOtdelSession.sesViewType == 2)

{//впервые перешли на разложение по структуре после выбора специальности

currentOtdelSession.sesSpecID = Convert.ToInt32(button.CommandArgument);

currentOtdelSession.sesViewType = 0;

currentOtdelSession.sesStruct.Lv\_0 = 1;

currentOtdelSession.sesStruct.Lv\_1 = 0;

currentOtdelSession.sesStruct.Lv\_2 = 0;

currentOtdelSession.sesStruct.Lv\_3 = 0;

currentOtdelSession.sesStruct.Lv\_4 = 0;

currentOtdelSession.sesStruct.Lv\_5 = 0;

}

else if (currentOtdelSession.sesViewType == 3)

{

currentOtdelSession.sesStruct = StructDeeper(currentOtdelSession.sesStruct, Convert.ToInt32(button.CommandArgument));

}

else

{

currentOtdelSession.sesStruct = StructDeeper(currentOtdelSession.sesStruct, Convert.ToInt32(button.CommandArgument));

}

OtdelHistory.CurrentSession++;

OtdelHistory.SessionCount = OtdelHistory.CurrentSession + 1;

OtdelHistory.OtdelSession[OtdelHistory.CurrentSession] = currentOtdelSession;

Session["OtdelHistory"] = OtdelHistory;

Response.Redirect("~/FinKadr/OtdelResult.aspx");

}

}

protected void Button2Click(object sender, EventArgs e) // по составляющим

{

Button button = (Button)sender;

{

OtdelHistorySession OtdelHistory = (OtdelHistorySession)Session["OtdelHistory"];

if (OtdelHistory == null)

{

Response.Redirect("~/Default.aspx");

}

OtdelSession CurrentOtdelSession = OtdelHistory.OtdelSession[OtdelHistory.CurrentSession];

Struct mainStruct = CurrentOtdelSession.sesStruct;

int ViewType = CurrentOtdelSession.sesViewType;

int ParamID = CurrentOtdelSession.sesParamID;

int ParamType = CurrentOtdelSession.sesParamType;

int ReportID = CurrentOtdelSession.sesReportID;

int SpecID = CurrentOtdelSession.sesSpecID;

OtdelSession currentOtdelSession = new OtdelSession(mainStruct, ViewType, ParamID, ParamType, ReportID, SpecID,"");

//OtdelSession currentOtdelSession = OtdelHistory.OtdelSession[OtdelHistory.CurrentSession];

currentOtdelSession.sesParamID = Convert.ToInt32(button.CommandArgument);

currentOtdelSession.sesParamType++;

OtdelHistory.CurrentSession++;

OtdelHistory.SessionCount = OtdelHistory.CurrentSession + 1;

OtdelHistory.OtdelSession[OtdelHistory.CurrentSession] = currentOtdelSession;

Session["OtdelHistory"] = OtdelHistory;

Response.Redirect("~/FinKadr/OtdelResult.aspx");

}

}

protected void Button3Click(object sender, EventArgs e) //по специальности

{

Button button = (Button)sender;

{

OtdelHistorySession OtdelHistory = (OtdelHistorySession) Session["OtdelHistory"];

if (OtdelHistory == null)

{

Response.Redirect("~/Default.aspx");

}

OtdelSession CurrentOtdelSession = OtdelHistory.OtdelSession[OtdelHistory.CurrentSession];

Struct mainStruct = CurrentOtdelSession.sesStruct;

int ViewType = CurrentOtdelSession.sesViewType;

int ParamID = CurrentOtdelSession.sesParamID;

int ParamType = CurrentOtdelSession.sesParamType;

int ReportID = CurrentOtdelSession.sesReportID;

int SpecID = CurrentOtdelSession.sesSpecID;

OtdelSession currentOtdelSession = new OtdelSession(mainStruct, ViewType, ParamID, ParamType, ReportID, SpecID,"");

currentOtdelSession.sesParamID = Convert.ToInt32(button.CommandArgument);

currentOtdelSession.sesViewType = 2;

OtdelHistory.CurrentSession++;

OtdelHistory.SessionCount = OtdelHistory.CurrentSession + 1;

OtdelHistory.OtdelSession[OtdelHistory.CurrentSession] = currentOtdelSession;

Session["OtdelHistory"] = OtdelHistory;

Response.Redirect("~/FinKadr/OtdelResult.aspx");

}

}

protected void GoBackButton\_Click(object sender, EventArgs e)

{

OtdelHistorySession OtdelHistory = (OtdelHistorySession)Session["OtdelHistory"];

if (OtdelHistory == null)

{

Response.Redirect("~/Default.aspx");

}

if (OtdelHistory.CurrentSession == 0)

{

Response.Redirect("~/FinKadr/OtdelChooseReport.aspx");

}

OtdelHistory.CurrentSession--;

Session["OtdelHistory"] = OtdelHistory;

Response.Redirect("~/FinKadr/OtdelResult.aspx");

}

protected void GoForwardButton\_Click(object sender, EventArgs e)

{

OtdelHistorySession OtdelHistory = (OtdelHistorySession)Session["OtdelHistory"];

if (OtdelHistory == null)

{

Response.Redirect("~/Default.aspx");

}

if (OtdelHistory.CurrentSession < OtdelHistory.SessionCount) // есть куда переходить

{

OtdelHistory.CurrentSession++;

Session["OtdelHistory"] = OtdelHistory;

Response.Redirect("~/FinKadr/OtdelResult.aspx");

}

}

protected void Button4\_Click(object sender, EventArgs e)

{

Response.Redirect("~/FinKadr/OtdelChooseReport.aspx");

}

protected void Button5\_Click(object sender, EventArgs e)

{

Response.Redirect("~/FinKadr/OtdelChooseReport.aspx");

}

private void BindTree(IEnumerable<MyObject> list, TreeNode parentNode)

{

var nodes = list.Where(x => parentNode == null ? x.ParentId == 0 : x.ParentId == int.Parse(parentNode.Value));

foreach (var node in nodes)

{

TreeNode newNode = new TreeNode(node.Name, node.Id.ToString());

if (node.Active == 1)

{

newNode.NavigateUrl = node.UrlAddr;

}

else

{

newNode.SelectAction = TreeNodeSelectAction.None;

}

if (parentNode == null)

{

TreeView1.Nodes.Add(newNode);

}

else

{

parentNode.ChildNodes.Add(newNode);

}

BindTree(list, newNode);

}

}

public void RefreshHistory()

{

/\*

#region history

OtdelHistorySession OtdelHistory\_ = (OtdelHistorySession)Session["OtdelHistory"];

if (OtdelHistory\_ == null)

{

Response.Redirect("~/Default.aspx");

}

if (OtdelHistory\_.Visible == true)

{

Button6.Text = "Скрыть историю";

TreeView1.Visible = true;

List<MyObject> list = new List<MyObject>();

for (int i = 0; i < OtdelHistory\_.SessionCount; i++)

{

OtdelSession curSesion = OtdelHistory\_.OtdelSession[i];

int tmp = OtdelHistory\_.CurrentSession == i ? 0:1;

list.Add(new MyObject() { Id = i+1, Name = curSesion.sesName, ParentId = i, UrlAddr = "Result?&HLevel="+i,Active=tmp });

}

BindTree(list, null);

TreeView1.ExpandAll();

}

else

{

Button6.Text = "Показать историю";

TreeView1.Visible = false;

TreeView1.Nodes.Clear();

TreeView1.DataBind();

}

#endregion

\*/

}

protected void Button6\_Click(object sender, EventArgs e)

{

OtdelHistorySession OtdelHistory\_ = (OtdelHistorySession)Session["OtdelHistory"];

if (OtdelHistory\_ == null)

{

Response.Redirect("~/Default.aspx");

}

if (OtdelHistory\_.Visible == true)

{

OtdelHistory\_.Visible = false;

}

else

{

OtdelHistory\_.Visible = true;

}

Session["OtdelHistory"] = OtdelHistory\_;

RefreshHistory();

}

protected void Grid\_SelectedIndexChanged(object sender, EventArgs e)

{

}

protected void Button7\_Click(object sender, EventArgs e)

{

ShowUnConfirmed unConfirmed =new ShowUnConfirmed(true);

Session["unConfirmed"] = unConfirmed;

Response.Redirect("~/FinKadr/OtdelResult.aspx");

}

protected void Grid\_RowDataBound(object sender, GridViewRowEventArgs e)

{

#region

var ColorLable = e.Row.FindControl("Color") as Label;

var PageConfirmButton = e.Row.FindControl("ConfirmButton") as Button;

var PageButton2 = e.Row.FindControl("Button2") as Button;

//// костыль 0%

var Button1\_ = e.Row.FindControl("Button1") as Button;

var PLable\_ = e.Row.FindControl("ProgressLable") as Label;

if ((Button1\_ != null) && (PLable\_ != null))

{

if (PLable\_.Text == "0,0%")

{

Button1\_.Enabled = false;

}

}

//end костыль 0%

if (ColorLable != null)

{

OtdelHistorySession OtdelHistory = (OtdelHistorySession)Session["OtdelHistory"];

if (OtdelHistory == null)

{

Response.Redirect("~/Default.aspx");

}

OtdelSession CurrentOtdelSession = OtdelHistory.OtdelSession[OtdelHistory.CurrentSession];

PageConfirmButton.Enabled = false;

PageButton2.Enabled = false;

int ColorNumber = -1;

if (int.TryParse(ColorLable.Text, out ColorNumber) && ColorNumber > -1)

{

switch (ColorNumber)

{

case 0:

{

break;

}

case 1: // утверждено

{

e.Row.Style.Add("background-color", "rgba(0, 255, 0, 0.3)");

PageButton2.Enabled = true;

break;

}

case 2: // можно утвердить

{

e.Row.Style.Add("background-color", "rgba(255, 0, 0, 0.3)");

PageButton2.Enabled = true;

break;

}

case 3: // рассчитано на неутвержденных данных

{

e.Row.Style.Add("background-color", "rgba(255, 255, 0, 0.3)");

PageConfirmButton.Enabled = true;

PageButton2.Enabled = true;

break;

}

default:

{

break;

}

}

}

if ((CurrentOtdelSession.sesViewType == 1) || (CurrentOtdelSession.sesParamType == 0))

{

PageButton2.Enabled = true;

}

}

#endregion

var ConfirmButton = e.Row.FindControl("ConfirmButton") as Button;

if (ConfirmButton != null)

{

ConfirmButton.OnClientClick = "javascript:return showCommentSection(" + ConfirmButton.CommandArgument+ ");";

}

}

protected void Button8\_Click(object sender, EventArgs e)

{

}

public void DoConfirm(int ParamId)

{

KPIWebDataContext kpiWebDataContext = new KPIWebDataContext();

OtdelHistorySession OtdelHistory = (OtdelHistorySession)Session["OtdelHistory"];

if (OtdelHistory == null)

{

Response.Redirect("~/Default.aspx");

}

Serialization UserSer = (Serialization)Session["UserID"];

if (UserSer == null)

{

Response.Redirect("~/Default.aspx");

}

int userID = UserSer.Id;

OtdelSession CurrentOtdelSession = OtdelHistory.OtdelSession[OtdelHistory.CurrentSession];

int ParamType = CurrentOtdelSession.sesParamType;

if (ParamType == 0) // indicator

{

CollectedIndocators Indicator = (from a in kpiWebDataContext.CollectedIndocators

where a.FK\_Indicators == ParamId

select a).FirstOrDefault();

Indicator.Confirmed = true;

kpiWebDataContext.SubmitChanges();

#region save params in DB with comment

ConfirmationHistory ConfirmParam = new ConfirmationHistory();

ConfirmParam.Date = DateTime.Now;

ConfirmParam.FK\_IndicatorsTable = ParamId;

ConfirmParam.FK\_ReportTable = CurrentOtdelSession.sesReportID;

ConfirmParam.FK\_UsersTable = userID;

ConfirmParam.Name = "Подтверждение ЦП проректором";

ConfirmParam.Comment = TextBox1.Text;

kpiWebDataContext.ConfirmationHistory.InsertOnSubmit(ConfirmParam);

kpiWebDataContext.SubmitChanges();

#endregion

Response.Redirect("~/FinKadr/OtdelResult.aspx");

}

else if (ParamType == 1) // calculated

{

CollectedCalculatedParametrs Calculated = (from a in kpiWebDataContext.CollectedCalculatedParametrs

where a.FK\_CalculatedParametrs == ParamId

select a).FirstOrDefault();

Calculated.Confirmed = true;

kpiWebDataContext.SubmitChanges();

#region save params in DB with comment

ConfirmationHistory ConfirmParam = new ConfirmationHistory();

ConfirmParam.Date = DateTime.Now;

ConfirmParam.FK\_CalculatedParamTable = ParamId;

ConfirmParam.FK\_ReportTable = CurrentOtdelSession.sesReportID;

ConfirmParam.FK\_UsersTable = userID;

ConfirmParam.Name = "Подтверждение ПД проректором";

ConfirmParam.Comment = TextBox1.Text;

kpiWebDataContext.ConfirmationHistory.InsertOnSubmit(ConfirmParam);

kpiWebDataContext.SubmitChanges();

#endregion

Response.Redirect("~/FinKadr/OtdelResult.aspx");

}

}

protected void ButtonOweClick(object sender, EventArgs e)

{

Button button = (Button)sender;

{

OtdelHistorySession OtdelHistory = (OtdelHistorySession)Session["OtdelHistory"];

if (OtdelHistory == null)

{

Response.Redirect("~/Default.aspx");

}

OtdelSession CurrentOtdelSession = OtdelHistory.OtdelSession[OtdelHistory.CurrentSession];

Struct mainStruct = CurrentOtdelSession.sesStruct;

int ViewType = CurrentOtdelSession.sesViewType;

int ParamID = CurrentOtdelSession.sesParamID;

int ParamType = CurrentOtdelSession.sesParamType;

int ReportID = CurrentOtdelSession.sesReportID;

int SpecID = CurrentOtdelSession.sesSpecID;

OtdelSession currentOtdelSession = new OtdelSession(mainStruct, ViewType, ParamID, ParamType, ReportID, SpecID, "");

currentOtdelSession.sesParamID = Convert.ToInt32(button.CommandArgument);

currentOtdelSession.sesViewType = 3;

currentOtdelSession.sesStruct.Lv\_0 = 1;

currentOtdelSession.sesStruct.Lv\_1 = 0;

currentOtdelSession.sesStruct.Lv\_2 = 0;

currentOtdelSession.sesStruct.Lv\_3 = 0;

currentOtdelSession.sesStruct.Lv\_4 = 0;

currentOtdelSession.sesStruct.Lv\_5 = 0;

OtdelHistory.CurrentSession++;

OtdelHistory.SessionCount = OtdelHistory.CurrentSession + 1;

OtdelHistory.OtdelSession[OtdelHistory.CurrentSession] = currentOtdelSession;

Session["OtdelHistory"] = OtdelHistory;

Response.Redirect("~/FinKadr/OtdelResult.aspx");

}

}

protected void Button8\_Click1(object sender, EventArgs e)

{

Response.Redirect("~/FinKadr/OtdelChooseReport.aspx");

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Filename: OtdelResult.aspx.designer.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//------------------------------------------------------------------------------

// <auto-generated>

// This code was generated by a tool.

//

// Changes to this file may cause incorrect behavior and will be lost if

// the code is regenerated.

// </auto-generated>

//------------------------------------------------------------------------------

namespace KPIWeb.FinKadr {

public partial class OtdelResult {

/// <summary>

/// Label7 control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.Label Label7;

/// <summary>

/// Panel5 control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.Panel Panel5;

/// <summary>

/// Label11 control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.Label Label11;

/// <summary>

/// Panel6 control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.Panel Panel6;

/// <summary>

/// Label12 control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.Label Label12;

/// <summary>

/// Panel7 control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.Panel Panel7;

/// <summary>

/// Label13 control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.Label Label13;

/// <summary>

/// TextBox1 control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.TextBox TextBox1;

/// <summary>

/// top\_panel2 control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.Panel top\_panel2;

/// <summary>

/// GoBackButton control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.Button GoBackButton;

/// <summary>

/// GoForwardButton control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.Button GoForwardButton;

/// <summary>

/// Button8 control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.Button Button8;

/// <summary>

/// ReportTitle control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.Label ReportTitle;

/// <summary>

/// PageFullName control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.Label PageFullName;

/// <summary>

/// TreeView1 control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.TreeView TreeView1;

/// <summary>

/// Grid control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.GridView Grid;

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Filename: LogHandler.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

namespace KPIWeb

{

public class LogHandler

{

private static LogWriter \_LogWriter;

public static LogWriter LogWriter

{

get

{

if (\_LogWriter == null)

\_LogWriter = InitLogger();

return \_LogWriter;

}

}

private static LogWriter InitLogger()

{

try

{

string loggingConfig = AppDomain.CurrentDomain.SetupInformation.ConfigurationFile;

return new LogWriter(loggingConfig, false, null);

}

catch (Exception e)

{

throw new Exception("Error creating log4net handler.", e);

}

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Filename: LogWriter.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using log4net;

using System.Diagnostics;

using System.Reflection;

using System.IO;

using log4net.Config;

using log4net.Repository;

namespace KPIWeb

{

public class LogWriter

{

private readonly ILog \_log;

public LogWriter()

{

XmlConfigurator.Configure();

\_log = LogManager.GetLogger(MethodBase.GetCurrentMethod().DeclaringType);

}

public LogWriter(string fileName, bool watchChanges, Type loggerType)

{

ILoggerRepository repository = LogManager.GetRepository();

if (loggerType == null)

loggerType = MethodBase.GetCurrentMethod().DeclaringType;

var fileInfo = new FileInfo(fileName);

if (watchChanges)

XmlConfigurator.ConfigureAndWatch(repository, fileInfo);

else

XmlConfigurator.Configure(repository, fileInfo);

\_log = LogManager.GetLogger(repository.Name, loggerType);

}

public void WriteLog(LogCategory category, string message, params object[] args)

{

WriteLog(category, string.Format(message, args));

}

public void WriteError(Exception ex)

{

WriteLog(LogCategory.ERROR, PrepareException(ex));

}

public void WriteError(Exception ex, string message)

{

WriteLog(LogCategory.ERROR, string.Format("Exception:{0} \n Message:{1}", PrepareException(ex), message));

}

public void WriteLog(LogCategory category, string message)

{

switch (category)

{

case LogCategory.DEBUG:

case LogCategory.INFO:

case LogCategory.WARN:

case LogCategory.ERROR:

case LogCategory.FATAL:

message = PrepareMessageString(message);

break;

default:

break;

}

switch (category)

{

case LogCategory.DEBUG:

if (\_log.IsDebugEnabled)

\_log.Debug(message);

break;

case LogCategory.INFO:

if (\_log.IsInfoEnabled)

\_log.Info(message);

break;

case LogCategory.WARN:

if (\_log.IsWarnEnabled)

\_log.Warn(message);

break;

case LogCategory.ERROR:

if (\_log.IsErrorEnabled)

\_log.Error(message);

break;

case LogCategory.FATAL:

if (\_log.IsFatalEnabled)

\_log.Fatal(message);

break;

default:

break;

}

}

private static string PrepareException(Exception ex)

{

if (ex == null)

return string.Empty;

string res = ex.ToString();

if (ex.Data != null && ex.Data.Count > 0)

{

res += "\n Additional Info: \n";

foreach (var k in ex.Data.Keys)

{

res += string.Format("{0} : {1}\n", k, ex.Data[k]);

}

}

try

{

StackTrace stackTrace = new StackTrace();

StackFrame stackFrame = stackTrace.GetFrame(1); // back to once step before

MethodBase methodBase = stackFrame.GetMethod();

int lineNumber = stackFrame.GetFileLineNumber();

if (methodBase.Name.Equals("Write"))

{

// back to 2 step before

stackFrame = stackTrace.GetFrame(2);

methodBase = stackFrame.GetMethod();

lineNumber = stackFrame.GetFileLineNumber();

}

res += string.Format("{0} {1} ", methodBase.ReflectedType, methodBase);

res += "Line Number: " + lineNumber;

}

catch

{ }

return res;

}

private static string PrepareMessageString(string message)

{

if (!String.IsNullOrEmpty(message))

{

message = message.Replace(Environment.NewLine, " ");

var forbiddenSimbols = new[] { ';', '\n', '\r' };

foreach (Char forbiddenChar in forbiddenSimbols)

message = message.Replace(forbiddenChar, ' ');

}

return message;

}

}

public enum LogCategory

{

DEBUG,

INFO,

WARN,

ERROR,

FATAL

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Filename: IdentityModels.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using Microsoft.AspNet.Identity;

using Microsoft.AspNet.Identity.EntityFramework;

using Microsoft.Owin.Security;

using System.Web;

using System;

using KPIWeb.Models;

namespace KPIWeb.Models

{

// You can add User data for the user by adding more properties to your User class, please visit http://go.microsoft.com/fwlink/?LinkID=317594 to learn more.

public class ApplicationUser : IdentityUser

{

}

public class ApplicationDbContext : IdentityDbContext<ApplicationUser>

{

public ApplicationDbContext()

: base("DefaultConnection")

{

}

}

#region Helpers

public class UserManager : UserManager<ApplicationUser>

{

public UserManager()

: base(new UserStore<ApplicationUser>(new ApplicationDbContext()))

{

}

}

}

namespace KPIWeb

{

public static class IdentityHelper

{

// Used for XSRF when linking external logins

public const string XsrfKey = "XsrfId";

public static void SignIn(UserManager manager, ApplicationUser user, bool isPersistent)

{

IAuthenticationManager authenticationManager = HttpContext.Current.GetOwinContext().Authentication;

authenticationManager.SignOut(DefaultAuthenticationTypes.ExternalCookie);

var identity = manager.CreateIdentity(user, DefaultAuthenticationTypes.ApplicationCookie);

authenticationManager.SignIn(new AuthenticationProperties() { IsPersistent = isPersistent }, identity);

}

public const string ProviderNameKey = "providerName";

public static string GetProviderNameFromRequest(HttpRequest request)

{

return request[ProviderNameKey];

}

public static string GetExternalLoginRedirectUrl(string accountProvider)

{

return "/Account/RegisterExternalLogin?" + ProviderNameKey + "=" + accountProvider;

}

private static bool IsLocalUrl(string url)

{

return !string.IsNullOrEmpty(url) && ((url[0] == '/' && (url.Length == 1 || (url[1] != '/' && url[1] != '\\'))) || (url.Length > 1 && url[0] == '~' && url[1] == '/'));

}

public static void RedirectToReturnUrl(string returnUrl, HttpResponse response)

{

if (!String.IsNullOrEmpty(returnUrl) && IsLocalUrl(returnUrl))

{

response.Redirect(returnUrl);

}

else

{

response.Redirect("~/");

}

}

}

#endregion

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Filename: TemporaryGeneratedFile\_036C0B5B-1481-4323-8D20-8F5ADCB23D92.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Filename: TemporaryGeneratedFile\_5937a670-0e60-4077-877b-f7221da3dda1.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Filename: TemporaryGeneratedFile\_E7A71F73-0F8D-4B9B-B56E-8E70B10BC5D3.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Filename: AssemblyInfo.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using System.Reflection;

using System.Runtime.CompilerServices;

using System.Runtime.InteropServices;

// General Information about an assembly is controlled through the following

// set of attributes. Change these attribute values to modify the information

// associated with an assembly.

[assembly: AssemblyTitle("KPIWeb")]

[assembly: AssemblyDescription("")]

[assembly: AssemblyConfiguration("")]

[assembly: AssemblyCompany("Крымский федеральный университет имени В. И. Вернадского")]

[assembly: AssemblyProduct("KPIWeb")]

[assembly: AssemblyCopyright("Крымский федеральный университет имени В. И. Вернадского © 2015")]

[assembly: AssemblyTrademark("")]

[assembly: AssemblyCulture("")]

// Setting ComVisible to false makes the types in this assembly not visible

// to COM components. If you need to access a type in this assembly from

// COM, set the ComVisible attribute to true on that type.

[assembly: ComVisible(false)]

// The following GUID is for the ID of the typelib if this project is exposed to COM

[assembly: Guid("78aba763-d0a6-4202-9366-88f63d83e0d3")]

// Version information for an assembly consists of the following four values:

//

// Major Version

// Minor Version

// Build Number

// Revision

//

// You can specify all the values or you can default the Revision and Build Numbers

// by using the '\*' as shown below:

[assembly: AssemblyVersion("1.0.0.0")]

[assembly: AssemblyFileVersion("1.0.0.0")]

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Filename: ClassForCharts.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

namespace KPIWeb.Rector

{

[Serializable]

public class ChartOneValue

{

public string name { get; set; }

public float value { get; set; }

public float planned { get; set; }

public ChartOneValue(String name\_, float value\_, float planned\_)

{

this.name = name\_;

this.value = value\_;

this.planned = planned\_;

}

public ChartOneValue(String name\_, float value\_)

{

this.name = name\_;

this.value = value\_;

}

}

public class ChartValueArray

{

public string chartName;

public List<ChartOneValue> ChartValues;

public ChartValueArray(string chartName\_)

{

this.chartName = chartName\_;

ChartValues = new List<ChartOneValue>();

}

public ChartValueArray(List<ChartOneValue> ChartValues\_, string chartName\_ )

{

this.chartName = chartName\_;

this.ChartValues = ChartValues\_;

}

}

public class ClassForCharts

{

}

public class ChartPlannedValue

{

public float PlannedValue;

public float RealValue;

public DateTime Date;

}

public class ChartValueWithAllPlanned

{

public int IndicatorID;

public string IndicatorName;

public List<ChartPlannedValue> PlannedAndRealValuesList;

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Filename: Documents.aspx.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

namespace KPIWeb.Rector

{

public partial class Documents : System.Web.UI.Page

{

protected void Page\_Load(object sender, EventArgs e)

{

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Filename: Documents.aspx.designer.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//------------------------------------------------------------------------------

// <автоматически создаваемое>

// Этот код создан программой.

//

// Изменения в этом файле могут привести к неправильной работе и будут потеряны в случае

// повторной генерации кода.

// </автоматически создаваемое>

//------------------------------------------------------------------------------

namespace KPIWeb.Rector {

public partial class Documents {

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Filename: ForRCalc.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.WebPages;

namespace KPIWeb.Rector

{

public class ForRCalc

{

[Serializable]

public class Struct // класс описываюший структурные подразделения

{

public int Lv\_0 { get; set; }

public int Lv\_1 { get; set; }

public int Lv\_2 { get; set; }

public int Lv\_3 { get; set; }

public int Lv\_4 { get; set; }

public int Lv\_5 { get; set; }

public string Name { get; set; }

public Struct(int lv0, int lv1, int lv2, int lv3, int lv4, int lv5, string name)

{

Lv\_0 = lv0;

Lv\_1 = lv1;

Lv\_2 = lv2;

Lv\_3 = lv3;

Lv\_4 = lv4;

Lv\_5 = lv5;

Name = name;

}

public Struct(int lv0, int lv1, int lv2, int lv3, int lv4, string name)

{

Lv\_0 = lv0;

Lv\_1 = lv1;

Lv\_2 = lv2;

Lv\_3 = lv3;

Lv\_4 = lv4;

Lv\_5 = 0;

Name = name;

}

public Struct(int lv0, int lv1, int lv2, int lv3, string name)

{

Lv\_0 = lv0;

Lv\_1 = lv1;

Lv\_2 = lv2;

Lv\_3 = lv3;

Lv\_4 = 0;

Lv\_5 = 0;

Name = name;

}

public Struct(int lv0, int lv1, int lv2, string name)

{

Lv\_0 = lv0;

Lv\_1 = lv1;

Lv\_2 = lv2;

Lv\_3 = 0;

Lv\_4 = 0;

Lv\_5 = 0;

Name = name;

}

public Struct(int lv0, int lv1, string name)

{

Lv\_0 = lv0;

Lv\_1 = lv1;

Lv\_2 = 0;

Lv\_3 = 0;

Lv\_4 = 0;

Lv\_5 = 0;

Name = name;

}

public Struct(int lv0, string name)

{

Lv\_0 = lv0;

Lv\_1 = 0;

Lv\_2 = 0;

Lv\_3 = 0;

Lv\_4 = 0;

Lv\_5 = 0;

Name = name;

}

}

public static List<Struct> GetAllSecondLevel()

{

List<Struct> tmpStrucList = new List<Struct>();

KPIWebDataContext kpiWebDataContext = new KPIWebDataContext();

tmpStrucList = (from a in kpiWebDataContext.FirstLevelSubdivisionTable

join b in kpiWebDataContext.SecondLevelSubdivisionTable

on a.FirstLevelSubdivisionTableID equals b.FK\_FirstLevelSubdivisionTable

where a.Active == true

&& b.Active == true

select new Struct(1, "")

{

Lv\_0 = (int)a.FK\_ZeroLevelSubvisionTable,

Lv\_1 = (int)a.FirstLevelSubdivisionTableID,

Lv\_2 = (int)b.SecondLevelSubdivisionTableID,

Lv\_3 = 0,

Lv\_4 = 0,

Lv\_5 = 0,

Name = b.Name

}).ToList();

return tmpStrucList;

}

public static List<Struct> GetChildStructList(Struct ParentStruct, int ReportID)

{

List<Struct> tmpStrucList = new List<Struct>();

int Level = 5;

Level = ParentStruct.Lv\_5 == 0 ? 4 : Level;

Level = ParentStruct.Lv\_4 == 0 ? 3 : Level;

Level = ParentStruct.Lv\_3 == 0 ? 2 : Level;

Level = ParentStruct.Lv\_2 == 0 ? 1 : Level;

Level = ParentStruct.Lv\_1 == 0 ? 0 : Level;

Level = ParentStruct.Lv\_0 == 0 ? -1 : Level;

KPIWebDataContext kpiWebDataContext = new KPIWebDataContext();

switch (Level)

{

case -1: // возвращаем все нулевого уровня, хотя там должен быть только КФУ

{

tmpStrucList = (from a in kpiWebDataContext.ZeroLevelSubdivisionTable

where a.Active == true

select new Struct(1, "")

{

Lv\_0 = (int)a.ZeroLevelSubdivisionTableID,

Lv\_1 = 0,

Lv\_2 = 0,

Lv\_3 = 0,

Lv\_4 = 0,

Lv\_5 = 0,

Name = a.Name

}).ToList();

break;

}

case 0: // возвращаем все универы

{

tmpStrucList = (from a in kpiWebDataContext.FirstLevelSubdivisionTable

join b in kpiWebDataContext.ReportArchiveAndLevelMappingTable

on a.FirstLevelSubdivisionTableID equals b.FK\_FirstLevelSubmisionTableId

where a.Active == true

&& b.FK\_ReportArchiveTableId == ReportID

&& b.Active == true

&& a.FK\_ZeroLevelSubvisionTable == ParentStruct.Lv\_0

select new Struct(1, "")

{

Lv\_0 = (int)a.FK\_ZeroLevelSubvisionTable,

Lv\_1 = (int)a.FirstLevelSubdivisionTableID,

Lv\_2 = 0,

Lv\_3 = 0,

Lv\_4 = 0,

Lv\_5 = 0,

Name = a.Name

}).ToList();

break;

}

case 1: // возвращаем все факультеты

{

tmpStrucList = (from a in kpiWebDataContext.FirstLevelSubdivisionTable

join b in kpiWebDataContext.SecondLevelSubdivisionTable

on a.FirstLevelSubdivisionTableID equals b.FK\_FirstLevelSubdivisionTable

where a.Active == true

&& b.Active == true

&& a.FK\_ZeroLevelSubvisionTable == ParentStruct.Lv\_0

&& b.FK\_FirstLevelSubdivisionTable == ParentStruct.Lv\_1

select new Struct(1, "")

{

Lv\_0 = (int)a.FK\_ZeroLevelSubvisionTable,

Lv\_1 = (int)a.FirstLevelSubdivisionTableID,

Lv\_2 = (int)b.SecondLevelSubdivisionTableID,

Lv\_3 = 0,

Lv\_4 = 0,

Lv\_5 = 0,

Name = b.Name

}).ToList();

break;

}

case 2: // возвращаем все кафедры

{

tmpStrucList = (from a in kpiWebDataContext.FirstLevelSubdivisionTable

join b in kpiWebDataContext.SecondLevelSubdivisionTable

on a.FirstLevelSubdivisionTableID equals b.FK\_FirstLevelSubdivisionTable

join c in kpiWebDataContext.ThirdLevelSubdivisionTable

on b.SecondLevelSubdivisionTableID equals c.FK\_SecondLevelSubdivisionTable

where a.Active == true

&& b.Active == true

&& c.Active == true

&& a.FK\_ZeroLevelSubvisionTable == ParentStruct.Lv\_0

&& b.FK\_FirstLevelSubdivisionTable == ParentStruct.Lv\_1

&& c.FK\_SecondLevelSubdivisionTable == ParentStruct.Lv\_2

select new Struct(1, "")

{

Lv\_0 = (int)a.FK\_ZeroLevelSubvisionTable,

Lv\_1 = (int)a.FirstLevelSubdivisionTableID,

Lv\_2 = (int)b.SecondLevelSubdivisionTableID,

Lv\_3 = (int)c.ThirdLevelSubdivisionTableID,

Lv\_4 = 0,

Lv\_5 = 0,

Name = c.Name

}).ToList();

break;

}

case 3: // возвращаем все специальности

{

tmpStrucList = (from a in kpiWebDataContext.FirstLevelSubdivisionTable

join b in kpiWebDataContext.SecondLevelSubdivisionTable

on a.FirstLevelSubdivisionTableID equals b.FK\_FirstLevelSubdivisionTable

join c in kpiWebDataContext.ThirdLevelSubdivisionTable

on b.SecondLevelSubdivisionTableID equals c.FK\_SecondLevelSubdivisionTable

join d in kpiWebDataContext.FourthLevelSubdivisionTable

on c.ThirdLevelSubdivisionTableID equals d.FK\_ThirdLevelSubdivisionTable

where a.Active == true

&& b.Active == true

&& c.Active == true

&& a.FK\_ZeroLevelSubvisionTable == ParentStruct.Lv\_0

&& b.FK\_FirstLevelSubdivisionTable == ParentStruct.Lv\_1

&& c.FK\_SecondLevelSubdivisionTable == ParentStruct.Lv\_2

&& d.FK\_ThirdLevelSubdivisionTable == ParentStruct.Lv\_3

select new Struct(1, "")

{

Lv\_0 = (int)a.FK\_ZeroLevelSubvisionTable,

Lv\_1 = (int)a.FirstLevelSubdivisionTableID,

Lv\_2 = (int)b.SecondLevelSubdivisionTableID,

Lv\_3 = (int)c.ThirdLevelSubdivisionTableID,

Lv\_4 = (int)d.FourthLevelSubdivisionTableID,

Lv\_5 = 0,

Name = d.Name

}).ToList();

break;

}

default:

{

//error не будем раскладывать до специальностей

break;

}

}

return tmpStrucList;

}

public static List<Struct> GetChildStructList(Struct ParentStruct, int ReportID, int SpecID)

{

List<Struct> tmpStrucList = new List<Struct>();

int Level = 5;

Level = ParentStruct.Lv\_5 == 0 ? 4 : Level;

Level = ParentStruct.Lv\_4 == 0 ? 3 : Level;

Level = ParentStruct.Lv\_3 == 0 ? 2 : Level;

Level = ParentStruct.Lv\_2 == 0 ? 1 : Level;

Level = ParentStruct.Lv\_1 == 0 ? 0 : Level;

Level = ParentStruct.Lv\_0 == 0 ? -1 : Level;

KPIWebDataContext kpiWebDataContext = new KPIWebDataContext();

switch (Level)

{

case -1: // возвращаем все нулевого уровня, хотя там должен быть только КФУ

{

tmpStrucList = (from a in kpiWebDataContext.ZeroLevelSubdivisionTable

where a.Active == true

select new Struct(1, "")

{

Lv\_0 = (int)a.ZeroLevelSubdivisionTableID,

Lv\_1 = 0,

Lv\_2 = 0,

Lv\_3 = 0,

Lv\_4 = 0,

Lv\_5 = 0,

Name = a.Name

}).OrderBy(x => x.Lv\_0).ToList();

break;

}

case 0: // возвращаем все универы

{

tmpStrucList = (from a in kpiWebDataContext.FirstLevelSubdivisionTable

join z in kpiWebDataContext.ReportArchiveAndLevelMappingTable

on a.FirstLevelSubdivisionTableID equals z.FK\_FirstLevelSubmisionTableId

join b in kpiWebDataContext.SecondLevelSubdivisionTable

on a.FirstLevelSubdivisionTableID equals b.FK\_FirstLevelSubdivisionTable

join c in kpiWebDataContext.ThirdLevelSubdivisionTable

on b.SecondLevelSubdivisionTableID equals c.FK\_SecondLevelSubdivisionTable

join d in kpiWebDataContext.FourthLevelSubdivisionTable

on c.ThirdLevelSubdivisionTableID equals d.FK\_ThirdLevelSubdivisionTable

where a.Active == true

&& a.FK\_ZeroLevelSubvisionTable == ParentStruct.Lv\_0

&& z.FK\_ReportArchiveTableId == ReportID

&& z.Active == true

&& ((SpecID == 0) || (SpecID == d.FK\_Specialization))

select new Struct(1, "")

{

Lv\_0 = (int)a.FK\_ZeroLevelSubvisionTable,

Lv\_1 = (int)a.FirstLevelSubdivisionTableID,

Lv\_2 = 0,

Lv\_3 = 0,

Lv\_4 = 0,

Lv\_5 = 0,

Name = a.Name

}).OrderBy(x => x.Lv\_1).ToList();

break;

}

case 1: // возвращаем все факультеты

{

tmpStrucList = (from a in kpiWebDataContext.FirstLevelSubdivisionTable

join b in kpiWebDataContext.SecondLevelSubdivisionTable

on a.FirstLevelSubdivisionTableID equals b.FK\_FirstLevelSubdivisionTable

join c in kpiWebDataContext.ThirdLevelSubdivisionTable

on b.SecondLevelSubdivisionTableID equals c.FK\_SecondLevelSubdivisionTable

join d in kpiWebDataContext.FourthLevelSubdivisionTable

on c.ThirdLevelSubdivisionTableID equals d.FK\_ThirdLevelSubdivisionTable

where a.Active == true

&& b.Active == true

&& a.FK\_ZeroLevelSubvisionTable == ParentStruct.Lv\_0

&& b.FK\_FirstLevelSubdivisionTable == ParentStruct.Lv\_1

&& ((SpecID == 0) || (SpecID == d.FK\_Specialization))

select new Struct(1, "")

{

Lv\_0 = (int)a.FK\_ZeroLevelSubvisionTable,

Lv\_1 = (int)a.FirstLevelSubdivisionTableID,

Lv\_2 = (int)b.SecondLevelSubdivisionTableID,

Lv\_3 = 0,

Lv\_4 = 0,

Lv\_5 = 0,

Name = b.Name

}).OrderBy(x => x.Lv\_2).ToList();

break;

}

case 2: // возвращаем все кафедры

{

tmpStrucList = (from a in kpiWebDataContext.FirstLevelSubdivisionTable

join b in kpiWebDataContext.SecondLevelSubdivisionTable

on a.FirstLevelSubdivisionTableID equals b.FK\_FirstLevelSubdivisionTable

join c in kpiWebDataContext.ThirdLevelSubdivisionTable

on b.SecondLevelSubdivisionTableID equals c.FK\_SecondLevelSubdivisionTable

join e in kpiWebDataContext.FourthLevelSubdivisionTable

on c.ThirdLevelSubdivisionTableID equals e.FK\_ThirdLevelSubdivisionTable

where a.Active == true

&& b.Active == true

&& c.Active == true

&& a.FK\_ZeroLevelSubvisionTable == ParentStruct.Lv\_0

&& b.FK\_FirstLevelSubdivisionTable == ParentStruct.Lv\_1

&& c.FK\_SecondLevelSubdivisionTable == ParentStruct.Lv\_2

&& ((SpecID == 0) || (SpecID == e.FK\_Specialization))

select new Struct(1, "")

{

Lv\_0 = (int)a.FK\_ZeroLevelSubvisionTable,

Lv\_1 = (int)a.FirstLevelSubdivisionTableID,

Lv\_2 = (int)b.SecondLevelSubdivisionTableID,

Lv\_3 = (int)c.ThirdLevelSubdivisionTableID,

Lv\_4 = 0,

Lv\_5 = 0,

Name = c.Name

}).OrderBy(x => x.Lv\_3).ToList();

break;

}

default:

{

//error не будем раскладывать до специальностей

break;

}

}

List<Struct> uniqeStruct = new List<Struct>();

foreach (Struct curStruct in tmpStrucList)

{

if (uniqeStruct.Count == 0)

{

uniqeStruct.Add(curStruct);

}

else

{

if ((uniqeStruct[uniqeStruct.Count - 1].Lv\_0 != curStruct.Lv\_0) ||

(uniqeStruct[uniqeStruct.Count - 1].Lv\_1 != curStruct.Lv\_1) ||

(uniqeStruct[uniqeStruct.Count - 1].Lv\_2 != curStruct.Lv\_2) ||

(uniqeStruct[uniqeStruct.Count - 1].Lv\_3 != curStruct.Lv\_3) ||

(uniqeStruct[uniqeStruct.Count - 1].Lv\_4 != curStruct.Lv\_4))

{

uniqeStruct.Add(curStruct);

}

}

}

return uniqeStruct;

}

public static float GetCalculatedWithParams(Struct StructToCalcFor, int ParamType, int ParamID, int ReportID, int SpecID) // читает показатель

{

KPIWebDataContext kpiWebDataContext = new KPIWebDataContext();

float result = 0;

if (ParamType == 0) // считаем целевой показатель

{

IndicatorsTable Indicator = (from a in kpiWebDataContext.IndicatorsTable

where a.IndicatorsTableID == ParamID

select a).FirstOrDefault();

return

(float)CalculateAbb.CalculateForLevel(1, Abbreviature.CalculatedAbbToFormula(Indicator.Formula)

, ReportID, SpecID, StructToCalcFor.Lv\_0

, StructToCalcFor.Lv\_1, StructToCalcFor.Lv\_2, StructToCalcFor.Lv\_3, StructToCalcFor.Lv\_4,

StructToCalcFor.Lv\_5, 0);

}

else if (ParamType == 1) // считаем рассчетный

{

CalculatedParametrs Calculated = (from a in kpiWebDataContext.CalculatedParametrs

where a.CalculatedParametrsID == ParamID

select a).FirstOrDefault();

return (float)CalculateAbb.CalculateForLevel(1, Calculated.Formula, ReportID, SpecID, StructToCalcFor.Lv\_0

, StructToCalcFor.Lv\_1, StructToCalcFor.Lv\_2, StructToCalcFor.Lv\_3, StructToCalcFor.Lv\_4,

StructToCalcFor.Lv\_5, 0);

}

else if (ParamType == 2) // суммируем базовый

{

return (float)CalculateAbb.SumForLevel(ParamID, ReportID, SpecID, StructToCalcFor.Lv\_0

, StructToCalcFor.Lv\_1, StructToCalcFor.Lv\_2, StructToCalcFor.Lv\_3, StructToCalcFor.Lv\_4,

StructToCalcFor.Lv\_5);

}

else

{

//error

}

return result;

}

public static int StructDeepness(Struct CurrentStruct)

{

int tmp = 0;

if (CurrentStruct.Lv\_0 != 0) tmp++;

if (CurrentStruct.Lv\_1 != 0) tmp++;

if (CurrentStruct.Lv\_2 != 0) tmp++;

if (CurrentStruct.Lv\_3 != 0) tmp++;

if (CurrentStruct.Lv\_4 != 0) tmp++;

if (CurrentStruct.Lv\_5 != 0) tmp++;

return tmp;

}

public static Struct StructDeeper(Struct parentStruct, int nextID)

{

KPIWebDataContext kpiWebDataContext = new KPIWebDataContext();

int lv0 = parentStruct.Lv\_0;

int lv1 = parentStruct.Lv\_1;

int lv2 = parentStruct.Lv\_2;

int lv3 = parentStruct.Lv\_3;

int lv4 = parentStruct.Lv\_4;

int lv5 = parentStruct.Lv\_5;

string name = parentStruct.Name;

Struct tmp = new Struct(lv0, lv1, lv2, lv3, lv4, lv5, name);

if (tmp.Lv\_0 == 0)

{

tmp.Lv\_0 = nextID;

tmp.Name = (from a in kpiWebDataContext.ZeroLevelSubdivisionTable

where a.ZeroLevelSubdivisionTableID == nextID

select a.Name).FirstOrDefault();

return tmp;

}

if (tmp.Lv\_1 == 0)

{

tmp.Lv\_1 = nextID;

tmp.Name = (from a in kpiWebDataContext.FirstLevelSubdivisionTable

where a.FirstLevelSubdivisionTableID == nextID

select a.Name).FirstOrDefault();

return tmp;

}

if (tmp.Lv\_2 == 0)

{

tmp.Lv\_2 = nextID;

tmp.Name = (from a in kpiWebDataContext.SecondLevelSubdivisionTable

where a.SecondLevelSubdivisionTableID == nextID

select a.Name).FirstOrDefault();

return tmp;

}

if (tmp.Lv\_3 == 0)

{

tmp.Lv\_3 = nextID;

tmp.Name = (from a in kpiWebDataContext.ThirdLevelSubdivisionTable

where a.ThirdLevelSubdivisionTableID == nextID

select a.Name).FirstOrDefault();

return tmp;

}

if (tmp.Lv\_4 == 0)

{

tmp.Lv\_4 = nextID;

tmp.Name = (from a in kpiWebDataContext.FourthLevelSubdivisionTable

where a.FourthLevelSubdivisionTableID == nextID

select a.Name).FirstOrDefault();

return tmp;

}

if (tmp.Lv\_5 == 0)

{

tmp.Lv\_5 = nextID;

tmp.Name = (from a in kpiWebDataContext.FifthLevelSubdivisionTable

where a.FifthLevelSubdivisionTableID == nextID

select a.Name).FirstOrDefault();

return tmp;

}

return tmp;

} //добавляет ID к первому в структуре нулю

public static int GetLastID(Struct currentStruct)

{

if (currentStruct.Lv\_0 == 0)

{

return 0;

}

if (currentStruct.Lv\_1 == 0)

{

return currentStruct.Lv\_0;

}

if (currentStruct.Lv\_2 == 0)

{

return currentStruct.Lv\_1;

}

if (currentStruct.Lv\_3 == 0)

{

return currentStruct.Lv\_2;

}

if (currentStruct.Lv\_4 == 0)

{

return currentStruct.Lv\_3;

}

return 0;

} //определяет последнее не нулевое значение в структуре

public static float CalculatedForDB(float input)

{

float tmp = (float)input;

if ((tmp < -(float)1E+20) || (tmp > (float)1E+20)

|| (tmp == null) || (float.IsNaN(tmp))

|| (float.IsInfinity(tmp)) || (float.IsNegativeInfinity(tmp))

|| (float.IsPositiveInfinity(tmp)) || (!tmp.ToString().IsFloat()))

{

tmp = (float)1E+20;

}

return tmp;

}

public static ChartOneValue GetCalculatedIndicator(int ReportID, IndicatorsTable Indicator, FirstLevelSubdivisionTable Academy, SecondLevelSubdivisionTable Faculty) // academyID == null && facultyID==null значит для всего КФУ

{

KPIWebDataContext kpiWebDataContext = new KPIWebDataContext();

float Planned\_Value = 0;

string Name\_ = "";

float Value\_ = 0;

#region plannedIndicator

PlannedIndicator plannedValue = (from a in kpiWebDataContext.PlannedIndicator

where a.FK\_IndicatorsTable == Indicator.IndicatorsTableID

&& a.Date > DateTime.Now

select a).OrderBy(x => x.Date).FirstOrDefault();

if (plannedValue != null)

{

Planned\_Value = (float)plannedValue.Value;

}

#endregion

#region Name

if ((Academy == null) && (Faculty == null))

{

if (Indicator.Measure != null)

{

if (Indicator.Measure.Length > 0)

{

Name\_ = Indicator.Name + " (" + Indicator.Measure + ")";

}

else

{

Name\_ = Indicator.Name;

}

}

else

{

Name\_ = Indicator.Name;

}

}

else if (Faculty != null)

{

Name\_ = Faculty.Name;

}

else if (Academy != null)

{

Name\_ = Academy.Name;

}

#endregion

#region

//ForRCalc.Struct mainStruct = mainStruct = new ForRCalc.Struct(1, 0, 0, 0, 0, "N");

CollectedIndicatorsForR collected = new CollectedIndicatorsForR();

if ((Academy == null) && (Faculty == null))

{

//mainStruct = new ForRCalc.Struct(1, 0, 0, 0, 0, "N");

collected = (from a in kpiWebDataContext.CollectedIndicatorsForR

where a.FK\_ReportArchiveTable == ReportID

&& a.FK\_IndicatorsTable == Indicator.IndicatorsTableID

&& a.FK\_FirstLevelSubdivisionTable == null

&& a.FK\_SecondLevelSubdivisionTable == null

select a).FirstOrDefault();

}

else if (Faculty != null)

{

//mainStruct = new ForRCalc.Struct(1, Faculty.FK\_FirstLevelSubdivisionTable, Faculty.SecondLevelSubdivisionTableID, 0, 0, "N");

collected = (from a in kpiWebDataContext.CollectedIndicatorsForR

where a.FK\_ReportArchiveTable == ReportID

&& a.FK\_IndicatorsTable == Indicator.IndicatorsTableID

&& a.FK\_FirstLevelSubdivisionTable == Faculty.FK\_FirstLevelSubdivisionTable

&& a.FK\_SecondLevelSubdivisionTable == Faculty.SecondLevelSubdivisionTableID

select a).FirstOrDefault();

}

else if (Academy != null)

{

//mainStruct = new ForRCalc.Struct(1, Academy.FirstLevelSubdivisionTableID, 0, 0, 0, "N");

collected = (from a in kpiWebDataContext.CollectedIndicatorsForR

where a.FK\_ReportArchiveTable == ReportID

&& a.FK\_IndicatorsTable == Indicator.IndicatorsTableID

&& a.FK\_FirstLevelSubdivisionTable == Academy.FirstLevelSubdivisionTableID

&& a.FK\_SecondLevelSubdivisionTable == null

select a).FirstOrDefault();

}

/\*

float tmp = ForRCalc.CalculatedForDB(ForRCalc.GetCalculatedWithParams(mainStruct, 0, Indicator.IndicatorsTableID, ReportID, 0));

if (tmp == (float)1E+20)

{

Value\_ = 0;

}

else

{

Value\_ = tmp;

}

\*/

if (collected == null)

{

Value\_ = 0;

}

else

if (collected.Value == null)

{

Value\_ = 0;

}

else

{

Value\_ = (float)collected.Value;

}

#endregion

ChartOneValue DataRowForChart = new ChartOneValue(Name\_, Value\_, Planned\_Value);

return DataRowForChart;

}

public static ChartValueWithAllPlanned GetAllPlannedForIndicator(int IndicatorID)

{

ChartValueWithAllPlanned tmp = new ChartValueWithAllPlanned();

KPIWebDataContext kpiWebDataContext = new KPIWebDataContext();

IndicatorsTable CurrentIndicator = (from a in kpiWebDataContext.IndicatorsTable

where a.IndicatorsTableID == IndicatorID

select a).FirstOrDefault();

tmp.IndicatorID = CurrentIndicator.IndicatorsTableID;

tmp.IndicatorName = CurrentIndicator.Name;

List<PlannedIndicator> PlannedForIndicatorList = (from a in kpiWebDataContext.PlannedIndicator

where a.Active == true

&& a.FK\_IndicatorsTable == CurrentIndicator.IndicatorsTableID

select a).OrderBy(c => c.Date).ToList();

List<ChartPlannedValue> PlannedValues = new List<ChartPlannedValue>();

PlannedIndicator prev = null;

foreach (PlannedIndicator currentPlanned in PlannedForIndicatorList)

{

ChartPlannedValue PlannedTemp = new ChartPlannedValue();

PlannedTemp.Date = (DateTime)currentPlanned.Date;

PlannedTemp.PlannedValue = (float)currentPlanned.Value;

CollectedIndicatorsForR ValuesForIndicatorList = new CollectedIndicatorsForR();

float tmp2 =0;

if (prev == null)

{

CollectedIndicatorsForR collected = (from a in kpiWebDataContext.CollectedIndicatorsForR

where a.Active == true

&& a.FK\_FirstLevelSubdivisionTable == null

&& a.FK\_IndicatorsTable == CurrentIndicator.IndicatorsTableID

&& a.CreatedDateTime < currentPlanned.Date

select a).FirstOrDefault();

if(collected!=null )

{

tmp2 = (float) collected.Value;

}

}

else

{

CollectedIndicatorsForR collected = (from a in kpiWebDataContext.CollectedIndicatorsForR

where a.Active == true

&& a.FK\_FirstLevelSubdivisionTable == null

&& a.FK\_IndicatorsTable == CurrentIndicator.IndicatorsTableID

&& a.CreatedDateTime < currentPlanned.Date

&& a.CreatedDateTime > prev.Date

select a).FirstOrDefault();

if (collected != null)

{

tmp2 = (float)collected.Value;

}

}

PlannedTemp.RealValue = tmp2;

PlannedValues.Add(PlannedTemp);

prev = currentPlanned;

}

tmp.PlannedAndRealValuesList = PlannedValues;

return tmp;

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Filename: RAnalitics.aspx.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Data;

using System.Web.UI;

using System.Web.UI.WebControls;

using Microsoft.Ajax.Utilities;

namespace KPIWeb.Rector

{

public partial class RAnalitics : System.Web.UI.Page

{

public class Struct // класс описываюший структурные подразделения

{

public int Lv\_0 { get; set; }

public int Lv\_1 { get; set; }

public int Lv\_2 { get; set; }

public int Lv\_3 { get; set; }

public int Lv\_4 { get; set; }

public int Lv\_5 { get; set; }

public string Name { get; set; }

public Struct(int lv0, int lv1, int lv2, int lv3, int lv4, int lv5, string name)

{

Lv\_0 = lv0;

Lv\_1 = lv1;

Lv\_2 = lv2;

Lv\_3 = lv3;

Lv\_4 = lv4;

Lv\_5 = lv5;

Name = name;

}

public Struct(int lv0, int lv1, int lv2, int lv3, int lv4, string name)

{

Lv\_0 = lv0;

Lv\_1 = lv1;

Lv\_2 = lv2;

Lv\_3 = lv3;

Lv\_4 = lv4;

Lv\_5 = 0;

Name = name;

}

public Struct(int lv0, int lv1, int lv2, int lv3, string name)

{

Lv\_0 = lv0;

Lv\_1 = lv1;

Lv\_2 = lv2;

Lv\_3 = lv3;

Lv\_4 = 0;

Lv\_5 = 0;

Name = name;

}

public Struct(int lv0, int lv1, int lv2, string name)

{

Lv\_0 = lv0;

Lv\_1 = lv1;

Lv\_2 = lv2;

Lv\_3 = 0;

Lv\_4 = 0;

Lv\_5 = 0;

Name = name;

}

public Struct(int lv0, int lv1, string name)

{

Lv\_0 = lv0;

Lv\_1 = lv1;

Lv\_2 = 0;

Lv\_3 = 0;

Lv\_4 = 0;

Lv\_5 = 0;

Name = name;

}

public Struct(int lv0, string name)

{

Lv\_0 = lv0;

Lv\_1 = 0;

Lv\_2 = 0;

Lv\_3 = 0;

Lv\_4 = 0;

Lv\_5 = 0;

Name = name;

}

}

protected void Page\_Load(object sender, EventArgs e)

{

KPIWebDataContext kpiWebDataContext = new KPIWebDataContext();

Serialization UserSer = (Serialization)Session["UserID"];

if (UserSer == null)

{

Response.Redirect("~/Default.aspx");

}

int userID = UserSer.Id;

UsersTable userTable =

(from a in kpiWebDataContext.UsersTable where a.UsersTableID == userID select a).FirstOrDefault();

if (userTable.AccessLevel != 7)

{

Response.Redirect("~/Default.aspx");

}

List<UsersTable> ProrectorList = (from a in kpiWebDataContext.UsersTable

where a.Active == true

&& a.AccessLevel == 5

select a).ToList();

if (Page.IsPostBack)

{

}

else

{

DataTable dataTable = new DataTable();

dataTable.Columns.Add(new DataColumn("ProrectorID", typeof(string)));

dataTable.Columns.Add(new DataColumn("ProrectorName", typeof(string)));

foreach (UsersTable Prorector in ProrectorList)

{

DataRow dataRow = dataTable.NewRow();

dataRow["ProrectorID"] = Prorector.UsersTableID;

if (Prorector.Position != null)

{

if (Prorector.Position.Length <2)

{

dataRow["ProrectorName"] = Prorector.Email;

}

else

{

dataRow["ProrectorName"] = Prorector.Position;

}

}

else

{

dataRow["ProrectorName"] = Prorector.Email;

}

dataTable.Rows.Add(dataRow);

}

GridView2.DataSource = dataTable;

GridView2.DataBind();

/////////////////////////////////////////////////////////////////////////////////////////////////////////////

DataTable dataTable2 = new DataTable();

dataTable2.Columns.Add(new DataColumn("IndicatorClassID", typeof(string)));

dataTable2.Columns.Add(new DataColumn("IndicatorClassName", typeof(string)));

List<IndicatorClass> IndicatorClassList = (from a in kpiWebDataContext.IndicatorClass

select a).ToList();

foreach (IndicatorClass IndicatorClass in IndicatorClassList)

{

DataRow dataRow = dataTable2.NewRow();

dataRow["IndicatorClassID"] = IndicatorClass.IndicatorClassID;

dataRow["IndicatorClassName"] = IndicatorClass.ClassName;

dataTable2.Rows.Add(dataRow);

}

GridView1.DataSource = dataTable2;

GridView1.DataBind();

/////////////////////////////////////////////////////////////////////////////////////////////////////////////

List<IndicatorsTable> IndicatorsTableList = (from a in kpiWebDataContext.IndicatorsTable

where a.Active == true

join b in kpiWebDataContext.IndicatorsAndUsersMapping

on a.IndicatorsTableID equals b.FK\_IndicatorsTable

where b.CanView == true

&& b.Active == true

&& b.FK\_UsresTable == userID

select a).OrderBy(c => c.SortID).ToList();

CheckBoxList1.Items.Clear();

foreach (IndicatorsTable currentIndicator in IndicatorsTableList)

{

ListItem Item1 = new ListItem();

Item1.Text = currentIndicator.Name;

Item1.Value = currentIndicator.IndicatorsTableID.ToString();

CheckBoxList1.Items.Add(Item1);

}

}

}

protected void ButtonClassClick(object sender, EventArgs e) // По типам индикаторов

{

LinkButton button = (LinkButton)sender;

{

KPIWebDataContext kpiWebDataContext = new KPIWebDataContext();

List<IndicatorsTable> IndicatorList\_0 = (from a in kpiWebDataContext.IndicatorsTable

where a.Active == true

join b in kpiWebDataContext.IndicatorClass

on a.FK\_IndicatorClass equals b.IndicatorClassID

where

b.IndicatorClassID == Convert.ToInt32(button.CommandArgument)

select a).Distinct().OrderBy(c => c.SortID).ToList();

List<int> IndicatorList = new List<int>();

foreach (IndicatorsTable current in IndicatorList\_0)

{

IndicatorList.Add(current.IndicatorsTableID);

}

if (IndicatorList.Count() > 0)

{

RectorChartSession RectorChart = new RectorChartSession();

RectorChart.IndicatorsList = IndicatorList;

Session["RectorChart"] = RectorChart;

Response.Redirect("~/Rector/RShowChart.aspx");

}

}

}

protected void ButtonProrectorClick(object sender, EventArgs e) //По проректорам

{

LinkButton button = (LinkButton)sender;

{

KPIWebDataContext kpiWebDataContext = new KPIWebDataContext();

List<IndicatorsTable> IndicatorList\_0 = (from a in kpiWebDataContext.IndicatorsTable

where a.Active == true

join b in kpiWebDataContext.IndicatorsAndUsersMapping

on a.IndicatorsTableID equals b.FK\_IndicatorsTable

where b.Active == true

&& b.CanConfirm == true

&& b.FK\_UsresTable == Convert.ToInt32(button.CommandArgument)

select a).Distinct().OrderBy(c => c.SortID).ToList();

List<int> IndicatorList = new List<int>();

foreach (IndicatorsTable current in IndicatorList\_0)

{

IndicatorList.Add(current.IndicatorsTableID);

}

if (IndicatorList.Count() > 0)

{

RectorChartSession RectorChart = new RectorChartSession();

RectorChart.IndicatorsList = IndicatorList;

Session["RectorChart"] = RectorChart;

Response.Redirect("~/Rector/RShowChart.aspx");

}

}

}

protected void Button4\_Click(object sender, EventArgs e) // по всем показателям

{

KPIWebDataContext kpiWebDataContext = new KPIWebDataContext();

Serialization UserSer = (Serialization)Session["UserID"];

if (UserSer == null)

{

Response.Redirect("~/Default.aspx");

}

int userID = UserSer.Id;

List<IndicatorsTable> IndicatorList\_0 = (from a in kpiWebDataContext.IndicatorsTable

where a.Active == true

join b in kpiWebDataContext.IndicatorsAndUsersMapping

on a.IndicatorsTableID equals b.FK\_IndicatorsTable

where b.CanView == true

&& b.Active == true

&& b.FK\_UsresTable == userID

select a).OrderBy(c => c.SortID).ToList();

List<int> IndicatorList = new List<int>();

foreach (IndicatorsTable current in IndicatorList\_0)

{

IndicatorList.Add(current.IndicatorsTableID);

}

RectorChartSession RectorChart = new RectorChartSession();

RectorChart.IndicatorsList = IndicatorList;

Session["RectorChart"] = RectorChart;

Response.Redirect("~/Rector/RShowChart.aspx");

}

protected void Button5\_Click(object sender, EventArgs e) // по отмеченным в чекбоксах галочкам

{

List<int> IndicatorList = new List<int>();

foreach (ListItem Item in CheckBoxList1.Items)

{

if (Item.Selected)

{

IndicatorList.Add(Convert.ToInt32(Item.Value));

}

}

if (IndicatorList.Count()>0)

{

RectorChartSession RectorChart = new RectorChartSession();

RectorChart.IndicatorsList = IndicatorList;

Session["RectorChart"] = RectorChart;

Response.Redirect("~/Rector/RShowChart.aspx");

}

}

protected void CheckBoxList1\_SelectedIndexChanged(object sender, EventArgs e)

{

}

protected void Button2\_Click(object sender, EventArgs e)

{

Response.Redirect("~/Rector/RectorMain.aspx");

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Filename: RAnalitics.aspx.designer.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//------------------------------------------------------------------------------

// <автоматически создаваемое>

// Этот код создан программой.

//

// Изменения в этом файле могут привести к неправильной работе и будут потеряны в случае

// повторной генерации кода.

// </автоматически создаваемое>

//------------------------------------------------------------------------------

namespace KPIWeb.Rector {

public partial class RAnalitics {

/// <summary>

/// top\_panel2 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Panel top\_panel2;

/// <summary>

/// GoBackButton элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Button GoBackButton;

/// <summary>

/// GoForwardButton элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Button GoForwardButton;

/// <summary>

/// Button2 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Button Button2;

/// <summary>

/// Button1 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Button Button1;

/// <summary>

/// Button6 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Button Button6;

/// <summary>

/// Button4 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Button Button4;

/// <summary>

/// GridView1 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.GridView GridView1;

/// <summary>

/// GridView2 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.GridView GridView2;

/// <summary>

/// CheckBoxList1 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.CheckBoxList CheckBoxList1;

/// <summary>

/// Button5 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Button Button5;

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Filename: RCalculate.aspx.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

namespace KPIWeb.Rector

{

public partial class RCalculate : System.Web.UI.Page

{

protected void Page\_Load(object sender, EventArgs e)

{

Serialization UserSer = (Serialization)Session["UserID"];

if (UserSer == null)

{

Response.Redirect("~/Default.aspx");

}

int userID = UserSer.Id;

KPIWebDataContext kPiDataContext = new KPIWebDataContext();

UsersTable userTable =

(from a in kPiDataContext.UsersTable where a.UsersTableID == userID select a).FirstOrDefault();

if (userTable.AccessLevel != 10)

{

Response.Redirect("~/Default.aspx");

}

}

protected void Button1\_Click(object sender, EventArgs e)

{

KPIWebDataContext kPiDataContext = new KPIWebDataContext();

int ReportID = 1; // пока только один отчет и надо сделать быстро)

#region init

//сначала берем все показатели целевые которые включены в выбранный отчет

//находим все структурные подразделения 1-го уровня которые участвуют в этом отчете

//находим все структурные подразделения 2-го уровня которые участвуют в этом отчете

List<IndicatorsTable> IndicatorsToCalculateList = (from a in kPiDataContext.IndicatorsTable

join b in kPiDataContext.ReportArchiveAndIndicatorsMappingTable

on a.IndicatorsTableID equals b.FK\_IndicatorsTable

where b.FK\_ReportArchiveTable == ReportID

&& b.Active == true

select a).OrderBy(c => c.SortID).ToList();

List<FirstLevelSubdivisionTable> FirstLevelToCalculate = (from a in kPiDataContext.FirstLevelSubdivisionTable

join b in kPiDataContext.ReportArchiveAndLevelMappingTable

on a.FirstLevelSubdivisionTableID equals b.FK\_FirstLevelSubmisionTableId

where b.FK\_ReportArchiveTableId == ReportID

select a).ToList();

List<SecondLevelSubdivisionTable> SecondLevelToCalculate = (from a in kPiDataContext.SecondLevelSubdivisionTable

join b in kPiDataContext.FirstLevelSubdivisionTable

on a.FK\_FirstLevelSubdivisionTable equals b.FirstLevelSubdivisionTableID

join c in kPiDataContext.ReportArchiveAndLevelMappingTable

on b.FirstLevelSubdivisionTableID equals c.FK\_FirstLevelSubmisionTableId

where c.FK\_ReportArchiveTableId == ReportID

select a).ToList();

//теперь пройдемся поочереди по каждому показателю

//первым делом посчитаем показатель для КФУ

//потом посчитаем показатель для каждого структурного 1-го уровня

//потом посчитаем показатель для каждого структурного 2-го уровня

//поехали:)

#endregion

foreach (IndicatorsTable CurrentIndicator in IndicatorsToCalculateList) //считаем каждый показатель для каждого факультета, каждой академии и всего КФУ

{

#region calcForCFU

{

//считай для КФУ

ForRCalc.Struct mainStruct = mainStruct = new ForRCalc.Struct(1, 0, 0, 0, 0, "N");

float tmp = ForRCalc.CalculatedForDB(ForRCalc.GetCalculatedWithParams(mainStruct, 0, CurrentIndicator.IndicatorsTableID, ReportID, 0));

CollectedIndicatorsForR newCollected = new CollectedIndicatorsForR();

newCollected.Active = true;

newCollected.CreatedDateTime = DateTime.Now;

newCollected.FK\_ReportArchiveTable = ReportID;

newCollected.FK\_IndicatorsTable = CurrentIndicator.IndicatorsTableID;

newCollected.FK\_FirstLevelSubdivisionTable = null;

newCollected.FK\_SecondLevelSubdivisionTable = null;

newCollected.FK\_ThirdLevelSubdivisionTable = null;

newCollected.FK\_FourthLelevlSubdivisionTable = null;

newCollected.FK\_FifthLevelSubdivisionTable = null;

if (tmp == (float)1E+20)

{

newCollected.Value = null;

}

else

{

newCollected.Value = tmp;

}

kPiDataContext.CollectedIndicatorsForR.InsertOnSubmit(newCollected);

}

#endregion

#region calcForAcademys

foreach (FirstLevelSubdivisionTable CurrentFirstLevel in FirstLevelToCalculate)

{

//считай для академий

ForRCalc.Struct mainStruct = mainStruct = new ForRCalc.Struct(1, CurrentFirstLevel.FirstLevelSubdivisionTableID, 0, 0, 0, "N");

float tmp = ForRCalc.CalculatedForDB(ForRCalc.GetCalculatedWithParams(mainStruct, 0, CurrentIndicator.IndicatorsTableID, ReportID, 0));

CollectedIndicatorsForR newCollected = new CollectedIndicatorsForR();

newCollected.Active = true;

newCollected.CreatedDateTime = DateTime.Now;

newCollected.FK\_ReportArchiveTable = ReportID;

newCollected.FK\_IndicatorsTable = CurrentIndicator.IndicatorsTableID;

newCollected.FK\_FirstLevelSubdivisionTable = CurrentFirstLevel.FirstLevelSubdivisionTableID;

newCollected.FK\_SecondLevelSubdivisionTable = null;

newCollected.FK\_ThirdLevelSubdivisionTable = null;

newCollected.FK\_FourthLelevlSubdivisionTable = null;

newCollected.FK\_FifthLevelSubdivisionTable = null;

if (tmp == (float)1E+20)

{

newCollected.Value = null;

}

else

{

newCollected.Value = tmp;

}

kPiDataContext.CollectedIndicatorsForR.InsertOnSubmit(newCollected);

}

#endregion

#region CalcForFaculys

foreach (SecondLevelSubdivisionTable CurrentSecondLevel in SecondLevelToCalculate)

{

// считай для кафедр

ForRCalc.Struct mainStruct = mainStruct = new ForRCalc.Struct(1, CurrentSecondLevel.FK\_FirstLevelSubdivisionTable, CurrentSecondLevel.SecondLevelSubdivisionTableID, 0, 0, "N");

float tmp = ForRCalc.CalculatedForDB(ForRCalc.GetCalculatedWithParams(mainStruct, 0, CurrentIndicator.IndicatorsTableID, ReportID, 0));

CollectedIndicatorsForR newCollected = new CollectedIndicatorsForR();

newCollected.Active = true;

newCollected.CreatedDateTime = DateTime.Now;

newCollected.FK\_ReportArchiveTable = ReportID;

newCollected.FK\_IndicatorsTable = CurrentIndicator.IndicatorsTableID;

newCollected.FK\_FirstLevelSubdivisionTable = CurrentSecondLevel.FK\_FirstLevelSubdivisionTable;

newCollected.FK\_SecondLevelSubdivisionTable = CurrentSecondLevel.SecondLevelSubdivisionTableID;

newCollected.FK\_ThirdLevelSubdivisionTable = null;

newCollected.FK\_FourthLelevlSubdivisionTable = null;

newCollected.FK\_FifthLevelSubdivisionTable = null;

if (tmp == (float)1E+20)

{

newCollected.Value = null;

}

else

{

newCollected.Value = tmp;

}

kPiDataContext.CollectedIndicatorsForR.InsertOnSubmit(newCollected);

}

#endregion

}

kPiDataContext.SubmitChanges();

}

#region patterns

protected double pattern1(UsersTable user, int ReportArchiveID, int spectype\_, string basicAbb, string basicAbb2) // по областям знаний

{

KPIWebDataContext kpiWebDataContext = new KPIWebDataContext();

return Convert.ToDouble(

(from a in kpiWebDataContext.CollectedBasicParametersTable

join z in kpiWebDataContext.BasicParametersTable

on a.FK\_BasicParametersTable equals z.BasicParametersTableID

join b in kpiWebDataContext.FourthLevelParametrs

on a.FK\_FourthLevelSubdivisionTable equals b.FourthLevelParametrsID

join c in kpiWebDataContext.ThirdLevelParametrs

on a.FK\_ThirdLevelSubdivisionTable equals c.ThirdLevelParametrsID

join d in kpiWebDataContext.FourthLevelSubdivisionTable

on a.FK\_FourthLevelSubdivisionTable equals d.FourthLevelSubdivisionTableID

join e in kpiWebDataContext.SpecializationTable

on d.FK\_Specialization equals e.SpecializationTableID

where

a.FK\_ZeroLevelSubdivisionTable == user.FK\_ZeroLevelSubdivisionTable

&& a.FK\_FirstLevelSubdivisionTable == user.FK\_FirstLevelSubdivisionTable

&& a.FK\_SecondLevelSubdivisionTable == user.FK\_SecondLevelSubdivisionTable

&& a.FK\_ThirdLevelSubdivisionTable == user.FK\_ThirdLevelSubdivisionTable

&& ((z.AbbreviationEN == basicAbb) || ((basicAbb2 != null) && (z.AbbreviationEN == basicAbb2)))

&& a.FK\_ReportArchiveTable == ReportArchiveID

&& b.SpecType == spectype\_

&& a.Active == true

&& d.Active == true

&& (e.FK\_FieldOfExpertise == 10 || e.FK\_FieldOfExpertise == 11 || e.FK\_FieldOfExpertise == 12)

select a.CollectedValue).Sum());

}

protected double pattern2(UsersTable user, int ReportArchiveID, string basicAbb) // для инстранцев

{

KPIWebDataContext kpiWebDataContext = new KPIWebDataContext();

return Convert.ToDouble(

(from a in kpiWebDataContext.CollectedBasicParametersTable

join z in kpiWebDataContext.BasicParametersTable

on a.FK\_BasicParametersTable equals z.BasicParametersTableID

join b in kpiWebDataContext.FourthLevelParametrs

on a.FK\_FourthLevelSubdivisionTable equals b.FourthLevelParametrsID

where

a.FK\_ZeroLevelSubdivisionTable == user.FK\_ZeroLevelSubdivisionTable

&& a.FK\_FirstLevelSubdivisionTable == user.FK\_FirstLevelSubdivisionTable

&& a.FK\_SecondLevelSubdivisionTable == user.FK\_SecondLevelSubdivisionTable

&& a.FK\_ThirdLevelSubdivisionTable == user.FK\_ThirdLevelSubdivisionTable

&& z.AbbreviationEN == basicAbb

&& a.FK\_ReportArchiveTable == ReportArchiveID

&& b.IsForeignStudentsAccept == true

&& a.Active == true

&& z.Active == true

&& b.Active == true

select a.CollectedValue).Sum());

}

protected double pattern3(UsersTable user, int ReportArchiveID, int SpecType) // Кол-во ООП // считает кол-во прикрепленных специальностьей

{

KPIWebDataContext kpiWebDataContext = new KPIWebDataContext();

return Convert.ToDouble(

(from a in kpiWebDataContext.ThirdLevelSubdivisionTable

join b in kpiWebDataContext.FourthLevelSubdivisionTable

on a.ThirdLevelSubdivisionTableID equals b.FK\_ThirdLevelSubdivisionTable

join c in kpiWebDataContext.FourthLevelParametrs

on b.FourthLevelSubdivisionTableID equals c.FourthLevelParametrsID

join d in kpiWebDataContext.ThirdLevelParametrs

on a.ThirdLevelSubdivisionTableID equals d.ThirdLevelParametrsID

where c.SpecType == SpecType

&& a.ThirdLevelSubdivisionTableID == user.FK\_ThirdLevelSubdivisionTable

&& d.CanGraduate == true

&& a.Active == true

// && z.Active == true

&& b.Active == true

select b).ToList().Count);

}

protected double pattern4(UsersTable user, int ReportArchiveID, int SpecType) // Кол-во ООП с условиями для инвалидов

{

KPIWebDataContext kpiWebDataContext = new KPIWebDataContext();

return Convert.ToDouble(

(from a in kpiWebDataContext.ThirdLevelSubdivisionTable

join b in kpiWebDataContext.FourthLevelSubdivisionTable

on a.ThirdLevelSubdivisionTableID equals b.FK\_ThirdLevelSubdivisionTable

join c in kpiWebDataContext.FourthLevelParametrs

on b.FourthLevelSubdivisionTableID equals c.FourthLevelParametrsID

join d in kpiWebDataContext.ThirdLevelParametrs

on a.ThirdLevelSubdivisionTableID equals d.ThirdLevelParametrsID

where c.SpecType == SpecType

&& a.ThirdLevelSubdivisionTableID == user.FK\_ThirdLevelSubdivisionTable

&& d.CanGraduate == true

&& c.IsInvalidStudentsFacilities == true

&& a.Active == true

&& b.Active == true

select b).ToList().Count);

}

protected double pattern5(UsersTable user, int ReportArchiveID, int SpecType)

{

KPIWebDataContext kpiWebDataContext = new KPIWebDataContext();

return Convert.ToDouble(

(from a in kpiWebDataContext.ThirdLevelSubdivisionTable

join b in kpiWebDataContext.FourthLevelSubdivisionTable

on a.ThirdLevelSubdivisionTableID equals b.FK\_ThirdLevelSubdivisionTable

join c in kpiWebDataContext.FourthLevelParametrs

on b.FourthLevelSubdivisionTableID equals c.FourthLevelParametrsID

join d in kpiWebDataContext.ThirdLevelParametrs

on a.ThirdLevelSubdivisionTableID equals d.ThirdLevelParametrsID

where c.SpecType == SpecType

&& a.ThirdLevelSubdivisionTableID == user.FK\_ThirdLevelSubdivisionTable

&& d.CanGraduate == true

&& c.IsNetworkComunication == true

&& a.Active == true

&& b.Active == true

select b).ToList().Count);

}

protected double pattern6(UsersTable user, int ReportArchiveID, int SpecType)

{

KPIWebDataContext kpiWebDataContext = new KPIWebDataContext();

return Convert.ToDouble(

(from a in kpiWebDataContext.ThirdLevelSubdivisionTable

join b in kpiWebDataContext.FourthLevelSubdivisionTable

on a.ThirdLevelSubdivisionTableID equals b.FK\_ThirdLevelSubdivisionTable

join c in kpiWebDataContext.FourthLevelParametrs

on b.FourthLevelSubdivisionTableID equals c.FourthLevelParametrsID

join d in kpiWebDataContext.ThirdLevelParametrs

on a.ThirdLevelSubdivisionTableID equals d.ThirdLevelParametrsID

where c.SpecType == SpecType

&& a.ThirdLevelSubdivisionTableID == user.FK\_ThirdLevelSubdivisionTable

&& d.CanGraduate == true

&& a.Active == true

&& b.Active == true

&& c.IsModernEducationTechnologies == true

select b).ToList().Count);

}

protected double pattern7(int SpecID, int typeOfCost, int ReportID, FourthLevelSubdivisionTable Fourth, int SpecType) // type 0 очное // 1 очное иностранцы // 2 заочное // 3 вечернее

{

KPIWebDataContext kpiWebDataContext = new KPIWebDataContext();

if (typeOfCost == 0)

{

return Convert.ToDouble((from a in kpiWebDataContext.EducationCostTable

where a.Active == true

&& a.FK\_Specialization == SpecID

select a.CostOfCommercOch).FirstOrDefault()

\*

(from a in kpiWebDataContext.CollectedBasicParametersTable

where a.FK\_ReportArchiveTable == ReportID

&& a.FK\_FourthLevelSubdivisionTable == Fourth.FourthLevelSubdivisionTableID

join c in kpiWebDataContext.BasicParametersTable

on a.FK\_BasicParametersTable equals c.BasicParametersTableID

where

((c.AbbreviationEN == "a\_Och\_M\_Kom" && SpecType == 3)

|| (c.AbbreviationEN == "a\_Och\_B\_Kom" && SpecType == 1)

|| (c.AbbreviationEN == "a\_Och\_S\_Kom" && SpecType == 2)

|| (c.AbbreviationEN == "a\_Och\_A\_Kom" && SpecType == 4))

select a.CollectedValue).Sum());

}

else if (typeOfCost == 1)

{

return Convert.ToDouble((from a in kpiWebDataContext.EducationCostTable

where a.Active == true

&& a.FK\_Specialization == SpecID

select a.CostOfCommercOchIn).FirstOrDefault()

\*

(from a in kpiWebDataContext.CollectedBasicParametersTable

where a.FK\_ReportArchiveTable == ReportID

&& a.FK\_FourthLevelSubdivisionTable == Fourth.FourthLevelSubdivisionTableID

join c in kpiWebDataContext.BasicParametersTable

on a.FK\_BasicParametersTable equals c.BasicParametersTableID

where

((c.AbbreviationEN == "a\_Och\_In\_M"&&SpecType == 3)

|| (c.AbbreviationEN == "a\_Och\_In\_B"&&SpecType == 1)

|| (c.AbbreviationEN == "a\_Och\_In\_S"&&SpecType == 2)

|| (c.AbbreviationEN == "a\_Och\_In\_A"&&SpecType == 4))

select a.CollectedValue).Sum());

}

else if (typeOfCost == 2)

{

return Convert.ToDouble((from a in kpiWebDataContext.EducationCostTable

where a.Active == true

&& a.FK\_Specialization == SpecID

select a.CostOfCommercZaoch).FirstOrDefault()

\*

(from a in kpiWebDataContext.CollectedBasicParametersTable

where a.FK\_ReportArchiveTable == ReportID

&& a.FK\_FourthLevelSubdivisionTable == Fourth.FourthLevelSubdivisionTableID

join c in kpiWebDataContext.BasicParametersTable

on a.FK\_BasicParametersTable equals c.BasicParametersTableID

where

((c.AbbreviationEN == "c\_Z\_A\_Kom" && SpecType == 3)

|| (c.AbbreviationEN == "c\_Z\_B\_Kom" && SpecType == 1)

|| (c.AbbreviationEN == "c\_Z\_S\_Kom" && SpecType == 2)

|| (c.AbbreviationEN == "c\_Z\_M\_Kom" && SpecType == 4))

select a.CollectedValue).Sum());

}

else if (typeOfCost == 3)

{

return Convert.ToDouble((from a in kpiWebDataContext.EducationCostTable

where a.Active == true

&& a.FK\_Specialization == SpecID

select a.CostOfCommercEvening).FirstOrDefault()

\*

(from a in kpiWebDataContext.CollectedBasicParametersTable

where a.FK\_ReportArchiveTable == ReportID

&& a.FK\_FourthLevelSubdivisionTable == Fourth.FourthLevelSubdivisionTableID

join c in kpiWebDataContext.BasicParametersTable

on a.FK\_BasicParametersTable equals c.BasicParametersTableID

where

((c.AbbreviationEN == "b\_OchZ\_S\_Kom" && SpecType == 3)

|| (c.AbbreviationEN == "b\_OchZ\_M\_Kom" && SpecType == 1)

|| (c.AbbreviationEN == "b\_OchZ\_A\_Kom" && SpecType == 2)

|| (c.AbbreviationEN == "b\_OchZ\_B\_Kom" && SpecType == 4))

select a.CollectedValue).Sum());

}

return 0;

}

public void patternSwitch(int ReportArchiveID, BasicParametersTable basicParam, FourthLevelSubdivisionTable FourthLevel, int fourthCnt, UsersTable user)

{

KPIWebDataContext kpiWebDataContext = new KPIWebDataContext();

double tmp = 1000000000001;

if (fourthCnt > 0)

{

if (FourthLevel == null)

{

if (basicParam.AbbreviationEN == "a\_Och\_M\_IZO") tmp = pattern1(user, ReportArchiveID, 3, "a\_Och\_M", "a\_Och\_M\_Kom");

if (basicParam.AbbreviationEN == "b\_OchZ\_M\_IZO") tmp = pattern1(user, ReportArchiveID, 3, "b\_OchZ\_M", "b\_OchZ\_M\_Kom");

if (basicParam.AbbreviationEN == "c\_Z\_M\_IZO") tmp = pattern1(user, ReportArchiveID, 3, "c\_Z\_M", "c\_Z\_M\_Kom");

if (basicParam.AbbreviationEN == "d\_E\_M\_IZO") tmp = pattern1(user, ReportArchiveID, 3, "d\_E\_M", "d\_E\_M\_Kom");

if (basicParam.AbbreviationEN == "a\_Och\_M\_NoIn") tmp = pattern2(user, ReportArchiveID, "a\_Och\_M");

if (basicParam.AbbreviationEN == "b\_OchZ\_M\_NoIn") tmp = pattern2(user, ReportArchiveID, "b\_OchZ\_M");

if (basicParam.AbbreviationEN == "c\_Z\_M\_NoIn") tmp = pattern2(user, ReportArchiveID, "c\_Z\_M");

if (basicParam.AbbreviationEN == "d\_E\_M\_NoIn") tmp = pattern2(user, ReportArchiveID, "d\_E\_M");

if (basicParam.AbbreviationEN == "a\_Och\_M\_NoIn\_Kom") tmp = pattern2(user, ReportArchiveID, "a\_Och\_M\_Kom");

if (basicParam.AbbreviationEN == "b\_OchZ\_M\_NoIn\_Kom") tmp = pattern2(user, ReportArchiveID, "b\_OchZ\_M\_Kom");

if (basicParam.AbbreviationEN == "c\_Z\_M\_NoIn\_Kom") tmp = pattern2(user, ReportArchiveID, "c\_Z\_M\_Kom");

if (basicParam.AbbreviationEN == "d\_E\_M\_NoIn\_Kom") tmp = pattern2(user, ReportArchiveID, "d\_E\_M\_Kom");

if (basicParam.AbbreviationEN == "a\_Och\_S\_IZO") tmp = pattern1(user, ReportArchiveID, 2, "a\_Och\_S", "a\_Och\_S\_Kom");

if (basicParam.AbbreviationEN == "b\_OchZ\_S\_IZO") tmp = pattern1(user, ReportArchiveID, 2, "b\_OchZ\_S", "b\_OchZ\_S\_Kom");

if (basicParam.AbbreviationEN == "c\_Z\_S\_IZO") tmp = pattern1(user, ReportArchiveID, 2, "c\_Z\_S", "c\_Z\_S\_Kom");

if (basicParam.AbbreviationEN == "d\_E\_S\_IZO") tmp = pattern1(user, ReportArchiveID, 2, "d\_E\_S", "c\_Z\_S\_Kom");

if (basicParam.AbbreviationEN == "a\_Och\_S\_NoIn") tmp = pattern2(user, ReportArchiveID, "a\_Och\_S");

if (basicParam.AbbreviationEN == "b\_OchZ\_S\_NoIn") tmp = pattern2(user, ReportArchiveID, "b\_OchZ\_S");

if (basicParam.AbbreviationEN == "c\_Z\_S\_NoIn") tmp = pattern2(user, ReportArchiveID, "c\_Z\_S");

if (basicParam.AbbreviationEN == "d\_E\_S\_NoIn") tmp = pattern2(user, ReportArchiveID, "d\_E\_S");

if (basicParam.AbbreviationEN == "a\_Och\_S\_NoIn\_Kom") tmp = pattern2(user, ReportArchiveID, "a\_Och\_S\_Kom");

if (basicParam.AbbreviationEN == "b\_OchZ\_S\_NoIn\_Kom") tmp = pattern2(user, ReportArchiveID, "b\_OchZ\_S\_Kom");

if (basicParam.AbbreviationEN == "c\_Z\_S\_NoIn\_Kom") tmp = pattern2(user, ReportArchiveID, "c\_Z\_S\_Kom");

if (basicParam.AbbreviationEN == "d\_E\_S\_NoIn\_Kom") tmp = pattern2(user, ReportArchiveID, "d\_E\_S\_Kom");

if (basicParam.AbbreviationEN == "a\_Och\_B\_IZO") tmp = pattern1(user, ReportArchiveID, 1, "a\_Och\_B", "a\_Och\_B\_Kom");

if (basicParam.AbbreviationEN == "b\_OchZ\_B\_IZO") tmp = pattern1(user, ReportArchiveID, 1, "b\_OchZ\_B", "b\_OchZ\_B\_Kom");

if (basicParam.AbbreviationEN == "c\_Z\_B\_IZO") tmp = pattern1(user, ReportArchiveID, 1, "c\_Z\_B", "c\_Z\_B\_Kom");

if (basicParam.AbbreviationEN == "d\_E\_B\_IZO") tmp = pattern1(user, ReportArchiveID, 1, "d\_E\_B", "d\_E\_B\_Kom");

if (basicParam.AbbreviationEN == "a\_Och\_B\_NoIn") tmp = pattern2(user, ReportArchiveID, "a\_Och\_B");

if (basicParam.AbbreviationEN == "b\_OchZ\_B\_NoIn") tmp = pattern2(user, ReportArchiveID, "b\_OchZ\_B");

if (basicParam.AbbreviationEN == "c\_Z\_B\_NoIn") tmp = pattern2(user, ReportArchiveID, "c\_Z\_B");

if (basicParam.AbbreviationEN == "d\_E\_B\_NoIn") tmp = pattern2(user, ReportArchiveID, "d\_E\_B");

if (basicParam.AbbreviationEN == "a\_Och\_B\_NoIn\_Kom") tmp = pattern2(user, ReportArchiveID, "a\_Och\_B\_Kom");

if (basicParam.AbbreviationEN == "b\_OchZ\_B\_NoIn\_Kom") tmp = pattern2(user, ReportArchiveID, "b\_OchZ\_B\_Kom");

if (basicParam.AbbreviationEN == "c\_Z\_B\_NoIn\_Kom") tmp = pattern2(user, ReportArchiveID, "c\_Z\_B\_Kom");

if (basicParam.AbbreviationEN == "d\_E\_B\_NoIn\_Kom") tmp = pattern2(user, ReportArchiveID, "d\_E\_B\_Kom");

if (basicParam.AbbreviationEN == "OOP\_M") tmp = pattern3(user, ReportArchiveID, 3);

if (basicParam.AbbreviationEN == "kol\_M\_OP") tmp = pattern4(user, ReportArchiveID, 3);

if (basicParam.AbbreviationEN == "kol\_M\_OP\_SV") tmp = pattern5(user, ReportArchiveID, 3);

if (basicParam.AbbreviationEN == "OOP\_M\_SOT") tmp = pattern6(user, ReportArchiveID, 3);

if (basicParam.AbbreviationEN == "OOP\_S") tmp = pattern3(user, ReportArchiveID, 2);

if (basicParam.AbbreviationEN == "kol\_S\_OP") tmp = pattern4(user, ReportArchiveID, 2);

if (basicParam.AbbreviationEN == "kol\_S\_OP\_SV") tmp = pattern5(user, ReportArchiveID, 2);

if (basicParam.AbbreviationEN == "OOP\_S\_SOT") tmp = pattern6(user, ReportArchiveID, 2);

if (basicParam.AbbreviationEN == "OOP\_B") tmp = pattern3(user, ReportArchiveID, 1);

if (basicParam.AbbreviationEN == "kol\_B\_OP") tmp = pattern4(user, ReportArchiveID, 1);

if (basicParam.AbbreviationEN == "kol\_B\_OP\_SV") tmp = pattern5(user, ReportArchiveID, 1);

if (basicParam.AbbreviationEN == "OOP\_B\_SOT") tmp = pattern6(user, ReportArchiveID, 1);

if (basicParam.AbbreviationEN == "OOP\_A") tmp = pattern3(user, ReportArchiveID, 4);

if (basicParam.AbbreviationEN == "kol\_A\_OP") tmp = pattern4(user, ReportArchiveID, 4);

if (basicParam.AbbreviationEN == "kol\_A\_OP\_SV") tmp = pattern5(user, ReportArchiveID, 4);

if (basicParam.AbbreviationEN == "OOP\_A\_SOT") tmp = pattern6(user, ReportArchiveID, 4);

//новые показатели 13.06.2015

if (basicParam.AbbreviationEN == "a\_Och\_M\_CO\_R") tmp =

pattern1(user, ReportArchiveID, 3, "a\_Och\_M\_C", null);

if (basicParam.AbbreviationEN == "b\_OchZ\_M\_CO\_R") tmp =

pattern1(user, ReportArchiveID, 3, "b\_OchZ\_M\_C", null);

if (basicParam.AbbreviationEN == "c\_Z\_M\_CO\_R") tmp =

pattern1(user, ReportArchiveID, 3, "c\_Z\_M\_C", null);

if (basicParam.AbbreviationEN == "d\_E\_M\_CO\_R") tmp =

pattern1(user, ReportArchiveID, 3, "d\_E\_M\_C", null);

if (basicParam.AbbreviationEN == "a\_Och\_B\_CO\_R") tmp =

pattern1(user, ReportArchiveID, 1, "a\_Och\_B\_C", null);

if (basicParam.AbbreviationEN == "d\_E\_B\_CO\_R") tmp =

pattern1(user, ReportArchiveID, 1, "d\_E\_B\_C", null);

if (basicParam.AbbreviationEN == "c\_Z\_B\_CO\_R") tmp =

pattern1(user, ReportArchiveID, 1, "c\_Z\_B\_C", null);

if (basicParam.AbbreviationEN == "d\_E\_B\_CO\_R") tmp =

pattern1(user, ReportArchiveID, 1, "d\_E\_B\_C", null);

if (basicParam.AbbreviationEN == "a\_Och\_S\_CO\_R") tmp =

pattern1(user, ReportArchiveID, 2, "a\_Och\_S\_C", null);

if (basicParam.AbbreviationEN == "b\_OchZ\_S\_CO\_R") tmp =

pattern1(user, ReportArchiveID, 2, "b\_OchZ\_S\_C", null);

if (basicParam.AbbreviationEN == "c\_Z\_S\_CO\_R") tmp =

pattern1(user, ReportArchiveID, 2, "c\_Z\_S\_C", null);

if (basicParam.AbbreviationEN == "d\_E\_S\_CO\_R") tmp =

pattern1(user, ReportArchiveID, 2, "d\_E\_S\_C", null);

//новые показатели

}

else

{

//новейшие показатели 19.06.2015

//// type 0 очное // 1 очное иностранцы // 2 заочное // 3 вечернее

if (basicParam.AbbreviationEN == "a\_Och\_B\_Kom\_money") tmp =

pattern7(FourthLevel.FK\_Specialization, 0, ReportArchiveID, FourthLevel,1);

if (basicParam.AbbreviationEN == "a\_OchZ\_B\_Kom\_money") tmp =

pattern7(FourthLevel.FK\_Specialization, 3, ReportArchiveID, FourthLevel, 1);

if (basicParam.AbbreviationEN == "a\_Z\_B\_Kom\_money") tmp =

pattern7(FourthLevel.FK\_Specialization, 2, ReportArchiveID, FourthLevel, 1);

if (basicParam.AbbreviationEN == "a\_IN\_B\_Kom\_money") tmp =

pattern7(FourthLevel.FK\_Specialization, 1, ReportArchiveID, FourthLevel, 1);

if (basicParam.AbbreviationEN == "a\_Och\_S\_Kom\_money") tmp =

pattern7(FourthLevel.FK\_Specialization, 0, ReportArchiveID, FourthLevel, 2);

if (basicParam.AbbreviationEN == "a\_OchZ\_S\_Kom\_money") tmp =

pattern7(FourthLevel.FK\_Specialization, 3, ReportArchiveID, FourthLevel, 2);

if (basicParam.AbbreviationEN == "a\_Z\_S\_Kom\_money") tmp =

pattern7(FourthLevel.FK\_Specialization, 2, ReportArchiveID, FourthLevel, 2);

if (basicParam.AbbreviationEN == "a\_IN\_S\_Kom\_money") tmp =

pattern7(FourthLevel.FK\_Specialization, 1, ReportArchiveID, FourthLevel, 2);

if (basicParam.AbbreviationEN == "a\_Och\_M\_Kom\_money") tmp =

pattern7(FourthLevel.FK\_Specialization, 0, ReportArchiveID, FourthLevel, 3);

if (basicParam.AbbreviationEN == "a\_OchZ\_M\_Kom\_money") tmp =

pattern7(FourthLevel.FK\_Specialization, 3, ReportArchiveID, FourthLevel, 3);

if (basicParam.AbbreviationEN == "a\_Z\_M\_Kom\_money") tmp =

pattern7(FourthLevel.FK\_Specialization, 2, ReportArchiveID, FourthLevel, 3);

if (basicParam.AbbreviationEN == "a\_IN\_M\_Kom\_money") tmp =

pattern7(FourthLevel.FK\_Specialization, 1, ReportArchiveID, FourthLevel, 3);

if (basicParam.AbbreviationEN == "a\_Och\_A\_Kom\_money") tmp =

pattern7(FourthLevel.FK\_Specialization, 0, ReportArchiveID, FourthLevel, 4);

if (basicParam.AbbreviationEN == "a\_OchZ\_A\_Kom\_money") tmp =

pattern7(FourthLevel.FK\_Specialization, 3, ReportArchiveID, FourthLevel, 4);

if (basicParam.AbbreviationEN == "a\_Z\_A\_Kom\_money") tmp =

pattern7(FourthLevel.FK\_Specialization, 2, ReportArchiveID, FourthLevel, 4);

if (basicParam.AbbreviationEN == "a\_IN\_A\_Kom\_money") tmp =

pattern7(FourthLevel.FK\_Specialization, 1, ReportArchiveID, FourthLevel, 4);

}

//новейшие показатели

}

else

{

if (basicParam.AbbreviationEN == "Kol\_Kaf\_R")

tmp = Convert.ToDouble((from a in kpiWebDataContext.ThirdLevelParametrs

where a.ThirdLevelParametrsID == user.FK\_ThirdLevelSubdivisionTable

select a.IsBasic).FirstOrDefault());

}

if (tmp < 1000000000000)

{

CollectedBasicParametersTable collectedBasicTmp = new CollectedBasicParametersTable();

int basicParamLevel = (int)(from a in kpiWebDataContext.BasicParametrAdditional

where a.BasicParametrAdditionalID == basicParam.BasicParametersTableID

select a.SubvisionLevel).FirstOrDefault();

if (basicParamLevel == 3)

{

collectedBasicTmp =

(from a in kpiWebDataContext.CollectedBasicParametersTable

where a.FK\_ZeroLevelSubdivisionTable == user.FK\_ZeroLevelSubdivisionTable

&& a.FK\_FirstLevelSubdivisionTable == user.FK\_FirstLevelSubdivisionTable

&& a.FK\_SecondLevelSubdivisionTable == user.FK\_SecondLevelSubdivisionTable

&& a.FK\_ThirdLevelSubdivisionTable == user.FK\_ThirdLevelSubdivisionTable

&& a.FK\_BasicParametersTable == basicParam.BasicParametersTableID

&& a.FK\_ReportArchiveTable == ReportArchiveID

select a).FirstOrDefault();

}

else

{

collectedBasicTmp =

(from a in kpiWebDataContext.CollectedBasicParametersTable

where a.FK\_ZeroLevelSubdivisionTable == user.FK\_ZeroLevelSubdivisionTable

&& a.FK\_FirstLevelSubdivisionTable == user.FK\_FirstLevelSubdivisionTable

&& a.FK\_SecondLevelSubdivisionTable == user.FK\_SecondLevelSubdivisionTable

&& a.FK\_ThirdLevelSubdivisionTable == user.FK\_ThirdLevelSubdivisionTable

&& a.FK\_FourthLevelSubdivisionTable == FourthLevel.FourthLevelSubdivisionTableID

&& a.FK\_BasicParametersTable == basicParam.BasicParametersTableID

&& a.FK\_ReportArchiveTable == ReportArchiveID

select a).FirstOrDefault();

}

if (collectedBasicTmp == null) // надо создать

{

collectedBasicTmp = new CollectedBasicParametersTable();

collectedBasicTmp.Active = true;

collectedBasicTmp.Status = 0;

collectedBasicTmp.FK\_UsersTable = user.UsersTableID;

collectedBasicTmp.FK\_BasicParametersTable = basicParam.BasicParametersTableID;

collectedBasicTmp.FK\_ReportArchiveTable = ReportArchiveID;

collectedBasicTmp.CollectedValue = tmp;

collectedBasicTmp.UserIP = "0.0.0.0";//Dns.GetHostEntry(Dns.GetHostName()).AddressList.Where(ip => ip.AddressFamily == System.Net.Sockets.AddressFamily.InterNetwork).Select(ip => ip.ToString()).FirstOrDefault() ?? "";

collectedBasicTmp.LastChangeDateTime = DateTime.Now;

collectedBasicTmp.SavedDateTime = DateTime.Now;

collectedBasicTmp.FK\_ZeroLevelSubdivisionTable = user.FK\_ZeroLevelSubdivisionTable;

collectedBasicTmp.FK\_FirstLevelSubdivisionTable = user.FK\_FirstLevelSubdivisionTable;

collectedBasicTmp.FK\_SecondLevelSubdivisionTable = user.FK\_SecondLevelSubdivisionTable;

collectedBasicTmp.FK\_ThirdLevelSubdivisionTable = user.FK\_ThirdLevelSubdivisionTable;

if (basicParamLevel == 4)

{

collectedBasicTmp.FK\_FourthLevelSubdivisionTable = FourthLevel.FourthLevelSubdivisionTableID;

}

kpiWebDataContext.CollectedBasicParametersTable.InsertOnSubmit(collectedBasicTmp);

kpiWebDataContext.SubmitChanges();

}

else

{

collectedBasicTmp.CollectedValue = tmp;

kpiWebDataContext.SubmitChanges();

}

}

}

/\*

protected void ConfCalculate(int ReportArchiveID, UsersTable user)

{

KPIWebDataContext kpiWebDataContext = new KPIWebDataContext();

List<BasicParametersTable> calcBasicParams =

(from a in kpiWebDataContext.ReportArchiveAndBasicParametrsMappingTable

join b in kpiWebDataContext.BasicParametersTable

on a.FK\_BasicParametrsTable equals b.BasicParametersTableID

join c in kpiWebDataContext.BasicParametrsAndUsersMapping

on b.BasicParametersTableID equals c.FK\_ParametrsTable

join d in kpiWebDataContext.BasicParametrAdditional

on b.BasicParametersTableID equals d.BasicParametrAdditionalID

where

a.FK\_ReportArchiveTable == ReportArchiveID //из нужного отчёта

&& c.FK\_UsersTable == user.UsersTableID // свяный с пользователем

&& (d.SubvisionLevel == 3 || d.SubvisionLevel == 4)//нужный уровень заполняющего

&& a.Active == true // запись в таблице связей показателя и отчёта активна

&& c.Active == true // запись в таблице связей показателя и пользователей активна

&& d.Calculated == true // этот показатель нужно считать

select b).ToList();

//узнали показатели кафедры(отчёт,разрешенияПользователя,Уровеньвводяшего,вводящийся показатель)

foreach (BasicParametersTable basicParam in calcBasicParams) //пройдемся по показателям

{

CollectedBasicParametersTable collectedBasicTmp =

(from a in kpiWebDataContext.CollectedBasicParametersTable

where a.FK\_ZeroLevelSubdivisionTable == user.FK\_ZeroLevelSubdivisionTable

&& a.FK\_FirstLevelSubdivisionTable == user.FK\_FirstLevelSubdivisionTable

&& a.FK\_SecondLevelSubdivisionTable == user.FK\_SecondLevelSubdivisionTable

&& a.FK\_ThirdLevelSubdivisionTable == user.FK\_ThirdLevelSubdivisionTable

&& a.FK\_BasicParametersTable == basicParam.BasicParametersTableID

&& a.FK\_ReportArchiveTable == ReportArchiveID

select a).FirstOrDefault();

if (collectedBasicTmp != null) // надо создать

{

collectedBasicTmp.Status = 4;

kpiWebDataContext.SubmitChanges();

}

}

}

\*/

protected void CalcCalculate(int ReportArchiveID, UsersTable user)

{

KPIWebDataContext kpiWebDataContext = new KPIWebDataContext();

List<BasicParametersTable> calcBasicParams =

(from a in kpiWebDataContext.ReportArchiveAndBasicParametrsMappingTable

join b in kpiWebDataContext.BasicParametersTable

on a.FK\_BasicParametrsTable equals b.BasicParametersTableID

join c in kpiWebDataContext.BasicParametrsAndUsersMapping

on b.BasicParametersTableID equals c.FK\_ParametrsTable

join d in kpiWebDataContext.BasicParametrAdditional

on b.BasicParametersTableID equals d.BasicParametrAdditionalID

where

a.FK\_ReportArchiveTable == ReportArchiveID //из нужного отчёта

&& c.FK\_UsersTable == user.UsersTableID // свяный с пользователем

&& (d.SubvisionLevel == 3 || d.SubvisionLevel == 4)//нужный уровень заполняющего

&& a.Active == true // запись в таблице связей показателя и отчёта активна

&& c.Active == true // запись в таблице связей показателя и пользователей активна

&& d.Calculated == true // этот показатель нужно считать

select b).ToList();

#region toDelete

/\*

ThirdLevelParametrs Cangrad = (from a in kpiWebDataContext.ThirdLevelParametrs

where a.CanGraduate == true

&& a.ThirdLevelParametrsID == user.FK\_ThirdLevelSubdivisionTable

select a).FirstOrDefault();

if (Cangrad != null) // кафедра выпускает

{

//определим какого типа специальности есть на данной кафедре

bool AnyB = (from a in kpiWebDataContext.FourthLevelSubdivisionTable

join b in kpiWebDataContext.FourthLevelParametrs

on a.FourthLevelSubdivisionTableID equals b.FourthLevelParametrsID

where

a.FK\_ThirdLevelSubdivisionTable == user.FK\_ThirdLevelSubdivisionTable

&& b.SpecType == 1

select a).ToList().Count() > 0 ? true : false;

bool AnyS = (from a in kpiWebDataContext.FourthLevelSubdivisionTable

join b in kpiWebDataContext.FourthLevelParametrs

on a.FourthLevelSubdivisionTableID equals b.FourthLevelParametrsID

where

a.FK\_ThirdLevelSubdivisionTable == user.FK\_ThirdLevelSubdivisionTable

&& b.SpecType == 2

select a).ToList().Count() > 0 ? true : false;

bool AnyM = (from a in kpiWebDataContext.FourthLevelSubdivisionTable

join b in kpiWebDataContext.FourthLevelParametrs

on a.FourthLevelSubdivisionTableID equals b.FourthLevelParametrsID

where

a.FK\_ThirdLevelSubdivisionTable == user.FK\_ThirdLevelSubdivisionTable

&& b.SpecType == 3

select a).ToList().Count() > 0 ? true : false;

bool AnyA = (from a in kpiWebDataContext.FourthLevelSubdivisionTable

join b in kpiWebDataContext.FourthLevelParametrs

on a.FourthLevelSubdivisionTableID equals b.FourthLevelParametrsID

where

a.FK\_ThirdLevelSubdivisionTable == user.FK\_ThirdLevelSubdivisionTable

&& b.SpecType == 4

select a).ToList().Count() > 0 ? true : false;

//узнали показатели кафедры(отчёт,разрешенияПользователя,Уровеньвводяшего,вводящийся показатель)

\*/

#endregion

List<FourthLevelSubdivisionTable> FourtLevels = new List<FourthLevelSubdivisionTable>();

if (user.FK\_ThirdLevelSubdivisionTable != null)

{

FourtLevels = (from a in kpiWebDataContext.FourthLevelSubdivisionTable

where a.FK\_ThirdLevelSubdivisionTable == user.FK\_ThirdLevelSubdivisionTable

&& a.Active == true

select a).ToList();

}

foreach (BasicParametersTable basicParam in calcBasicParams) //пройдемся по показателям

{

int ii = (int)(from a in kpiWebDataContext.BasicParametrAdditional

where a.BasicParametrAdditionalID == basicParam.BasicParametersTableID

select a.SubvisionLevel).FirstOrDefault();

if (ii == 4)

{

if (FourtLevels.Count() > 0) // есть хоть одна специальность

{

foreach (FourthLevelSubdivisionTable CurrentFourth in FourtLevels)

{

patternSwitch(ReportArchiveID, basicParam, CurrentFourth, FourtLevels.Count(), user); // считаем для каждой специальности

}

}

}

else

{

patternSwitch(ReportArchiveID, basicParam, null, FourtLevels.Count(), user);

}

}

// }

}

#endregion

protected void Button2\_Click(object sender, EventArgs e)

{

KPIWebDataContext kPiDataContext = new KPIWebDataContext();

List<ThirdLevelSubdivisionTable> ThirdLevelList = (from a in kPiDataContext.ThirdLevelSubdivisionTable

join b in kPiDataContext.CollectedBasicParametersTable

on a.ThirdLevelSubdivisionTableID equals b.FK\_ThirdLevelSubdivisionTable

where

b.CollectedValue !=null

&&

b.FK\_ReportArchiveTable == 1

select a).Distinct().ToList();

/\* UsersTable UUUUUSER = (from a in kPiDataContext.UsersTable

where a.UsersTableID == 12489

select a).FirstOrDefault();

CalcCalculate(1, UUUUUSER);

\*/

// ThirdLevelList.Clear();

// ThirdLevelList.Add((from a in kPiDataContext.ThirdLevelSubdivisionTable where a.ThirdLevelSubdivisionTableID == 4684 select a).FirstOrDefault());

foreach (ThirdLevelSubdivisionTable thirdLevel in ThirdLevelList)

{

UsersTable UsersToCalculate = (from a in kPiDataContext.UsersTable

join b in kPiDataContext.BasicParametrsAndUsersMapping

on a.UsersTableID equals b.FK\_UsersTable

//join c in kPiDataContext.CollectedBasicParametersTable

// on b.FK\_ParametrsTable equals c.FK\_BasicParametersTable

where a.FK\_ThirdLevelSubdivisionTable == thirdLevel.ThirdLevelSubdivisionTableID

&& b.CanEdit == true select a).FirstOrDefault();

if (UsersToCalculate != null)

CalcCalculate(1, UsersToCalculate);

}

}

protected void Button3\_Click(object sender, EventArgs e)

{

KPIWebDataContext kPiDataContext = new KPIWebDataContext();

List<ThirdLevelSubdivisionTable> ThirdLevelList = (from a in kPiDataContext.ThirdLevelSubdivisionTable

where a.Active == true

select a).ToList();

TextBox1.Text = "";

foreach (ThirdLevelSubdivisionTable CurrentThird in ThirdLevelList)

{

int ConfirmedCnt = (from a in kPiDataContext.CollectedBasicParametersTable

where a.FK\_ThirdLevelSubdivisionTable == CurrentThird.ThirdLevelSubdivisionTableID

&& a.Active == true

&& a.Status == 4

&& a.FK\_ReportArchiveTable == 1

select a).Count();

CollectedBasicParametersTable FirstCollected = (from a in kPiDataContext.CollectedBasicParametersTable

where a.FK\_ThirdLevelSubdivisionTable == CurrentThird.ThirdLevelSubdivisionTableID

&& a.FK\_ReportArchiveTable == 1

&& a.Active == true

select a).FirstOrDefault();

if ((ConfirmedCnt>10)&&(FirstCollected.Status == 4))

{

TextBox1.Text += "утверждено" + ConfirmedCnt.ToString();

TextBox1.Text += " всего" + (from a in kPiDataContext.CollectedBasicParametersTable

where a.FK\_ThirdLevelSubdivisionTable == CurrentThird.ThirdLevelSubdivisionTableID

&& a.FK\_ReportArchiveTable == 1

&& a.Active == true

select a).Count().ToString();

List<CollectedBasicParametersTable> CollectedToChange = (from a in kPiDataContext.CollectedBasicParametersTable

where a.FK\_ThirdLevelSubdivisionTable == CurrentThird.ThirdLevelSubdivisionTableID

&& a.FK\_ReportArchiveTable == 1

&& a.Active == true

select a).ToList();

foreach (CollectedBasicParametersTable CollectedBasic in CollectedToChange)

{

CollectedBasic.Status = 4;

}

kPiDataContext.SubmitChanges();

TextBox1.Text += " сделано";

TextBox1.Text += Environment.NewLine;

}

else if (ConfirmedCnt > 0)

{

TextBox1.Text += "ОШИБКА утверждено" + ConfirmedCnt.ToString();

TextBox1.Text += " не утверждено" + (from a in kPiDataContext.CollectedBasicParametersTable

where a.FK\_ThirdLevelSubdivisionTable == CurrentThird.ThirdLevelSubdivisionTableID

&& a.FK\_ReportArchiveTable == 1

&& a.Active == true

select a).Count().ToString();

TextBox1.Text += " ID 3-го уровня" + CurrentThird.ThirdLevelSubdivisionTableID.ToString();

TextBox1.Text += Environment.NewLine;

}

else if ((FirstCollected!=null)&&(FirstCollected.Status == 4))

{

TextBox1.Text += "ОШИБКА утверждено" + ConfirmedCnt.ToString();

TextBox1.Text += " не утверждено" + (from a in kPiDataContext.CollectedBasicParametersTable

where a.FK\_ThirdLevelSubdivisionTable == CurrentThird.ThirdLevelSubdivisionTableID

&& a.FK\_ReportArchiveTable == 1

&& a.Active == true

select a).Count().ToString();

TextBox1.Text += " ID 3-го уровня" + CurrentThird.ThirdLevelSubdivisionTableID.ToString();

TextBox1.Text += Environment.NewLine;

}

else

{

}

}

}

public void gogo (int ParamID,int UserID)

{

KPIWebDataContext kPiDataContext = new KPIWebDataContext();

//////////////////////////////////////////////////////////////////////////////////////////////////////////

BasicParametrsAndUsersMapping newconnection = (from a in kPiDataContext.BasicParametrsAndUsersMapping

where a.Active == true

&& a.FK\_ParametrsTable == ParamID

&& a.FK\_UsersTable == UserID

select a).FirstOrDefault();

if (newconnection == null)

{

newconnection = new BasicParametrsAndUsersMapping();

newconnection.FK\_UsersTable = UserID;

newconnection.FK\_ParametrsTable = ParamID;

newconnection.Active = true;

newconnection.CanView = true;

kPiDataContext.BasicParametrsAndUsersMapping.InsertOnSubmit(newconnection);

}

else

{

newconnection.Active = true;

newconnection.CanView = true;

}

kPiDataContext.SubmitChanges();

//////////////////////////////////////////////////////////////////////////////////////////////////////////

}

public void gogo2(int ParamID, int UserID)

{

KPIWebDataContext kPiDataContext = new KPIWebDataContext();

//////////////////////////////////////////////////////////////////////////////////////////////////////////

BasicParametrsAndUsersMapping newconnection = (from a in kPiDataContext.BasicParametrsAndUsersMapping

where a.Active == true

&& a.FK\_ParametrsTable == ParamID

&& a.FK\_UsersTable == UserID

select a).FirstOrDefault();

if (newconnection == null)

{

newconnection = new BasicParametrsAndUsersMapping();

newconnection.FK\_UsersTable = UserID;

newconnection.FK\_ParametrsTable = ParamID;

newconnection.Active = true;

newconnection.CanEdit = true;

kPiDataContext.BasicParametrsAndUsersMapping.InsertOnSubmit(newconnection);

}

else

{

newconnection.Active = true;

newconnection.CanEdit = true;

}

kPiDataContext.SubmitChanges();

//////////////////////////////////////////////////////////////////////////////////////////////////////////

}

public void gogo3(int ParamID, int UserID)

{

KPIWebDataContext kPiDataContext = new KPIWebDataContext();

//////////////////////////////////////////////////////////////////////////////////////////////////////////

BasicParametrsAndUsersMapping newconnection = (from a in kPiDataContext.BasicParametrsAndUsersMapping

where a.Active == true

&& a.FK\_ParametrsTable == ParamID

&& a.FK\_UsersTable == UserID

select a).FirstOrDefault();

if (newconnection == null)

{

newconnection = new BasicParametrsAndUsersMapping();

newconnection.FK\_UsersTable = UserID;

newconnection.FK\_ParametrsTable = ParamID;

newconnection.Active = true;

newconnection.CanConfirm = true;

kPiDataContext.BasicParametrsAndUsersMapping.InsertOnSubmit(newconnection);

}

else

{

newconnection.Active = true;

newconnection.CanConfirm = true;

}

kPiDataContext.SubmitChanges();

//////////////////////////////////////////////////////////////////////////////////////////////////////////

}

protected void Button4\_Click(object sender, EventArgs e)

{

KPIWebDataContext kPiDataContext = new KPIWebDataContext();

List<UsersTable> UsersList = (from a in kPiDataContext.UsersTable

join b in kPiDataContext.BasicParametrsAndUsersMapping

on a.UsersTableID equals b.FK\_UsersTable

where a.Active == true

&& b.Active == true

&& b.CanView == true

&& b.FK\_ParametrsTable == 3605

select a).Distinct().ToList();

foreach (UsersTable currentUser in UsersList)

{

gogo(3895, currentUser.UsersTableID);

gogo(3896, currentUser.UsersTableID);

gogo(3897, currentUser.UsersTableID);

gogo(3898, currentUser.UsersTableID);

gogo(3899, currentUser.UsersTableID);

gogo(3900, currentUser.UsersTableID);

gogo(3901, currentUser.UsersTableID);

gogo(3902, currentUser.UsersTableID);

gogo(3903, currentUser.UsersTableID);

gogo(3904, currentUser.UsersTableID);

gogo(3905, currentUser.UsersTableID);

gogo(3906, currentUser.UsersTableID);

gogo(3907, currentUser.UsersTableID);

gogo(3908, currentUser.UsersTableID);

gogo(3909, currentUser.UsersTableID);

gogo(3910, currentUser.UsersTableID);

}

List<UsersTable> UsersList2 = (from a in kPiDataContext.UsersTable

join b in kPiDataContext.BasicParametrsAndUsersMapping

on a.UsersTableID equals b.FK\_UsersTable

where a.Active == true

&& b.Active == true

&& b.CanEdit == true

&& b.FK\_ParametrsTable == 3605

select a).Distinct().ToList();

foreach (UsersTable currentUser in UsersList2)

{

gogo2(3895, currentUser.UsersTableID);

gogo2(3896, currentUser.UsersTableID);

gogo2(3897, currentUser.UsersTableID);

gogo2(3898, currentUser.UsersTableID);

gogo2(3899, currentUser.UsersTableID);

gogo2(3900, currentUser.UsersTableID);

gogo2(3901, currentUser.UsersTableID);

gogo2(3902, currentUser.UsersTableID);

gogo2(3903, currentUser.UsersTableID);

gogo2(3904, currentUser.UsersTableID);

gogo2(3905, currentUser.UsersTableID);

gogo2(3906, currentUser.UsersTableID);

gogo2(3907, currentUser.UsersTableID);

gogo2(3908, currentUser.UsersTableID);

gogo2(3909, currentUser.UsersTableID);

gogo2(3910, currentUser.UsersTableID);

}

List<UsersTable> UsersList3 = (from a in kPiDataContext.UsersTable

join b in kPiDataContext.BasicParametrsAndUsersMapping

on a.UsersTableID equals b.FK\_UsersTable

where a.Active == true

&& b.Active == true

&& b.CanConfirm == true

&& b.FK\_ParametrsTable == 3605

select a).Distinct().ToList();

foreach (UsersTable currentUser in UsersList3)

{

gogo3(3895, currentUser.UsersTableID);

gogo3(3896, currentUser.UsersTableID);

gogo3(3897, currentUser.UsersTableID);

gogo3(3898, currentUser.UsersTableID);

gogo3(3899, currentUser.UsersTableID);

gogo3(3900, currentUser.UsersTableID);

gogo3(3901, currentUser.UsersTableID);

gogo3(3902, currentUser.UsersTableID);

gogo3(3903, currentUser.UsersTableID);

gogo3(3904, currentUser.UsersTableID);

gogo3(3905, currentUser.UsersTableID);

gogo3(3906, currentUser.UsersTableID);

gogo3(3907, currentUser.UsersTableID);

gogo3(3908, currentUser.UsersTableID);

gogo3(3909, currentUser.UsersTableID);

gogo3(3910, currentUser.UsersTableID);

}

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Filename: RCalculate.aspx.designer.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//------------------------------------------------------------------------------

// <auto-generated>

// This code was generated by a tool.

//

// Changes to this file may cause incorrect behavior and will be lost if

// the code is regenerated.

// </auto-generated>

//------------------------------------------------------------------------------

namespace KPIWeb.Rector {

public partial class RCalculate {

/// <summary>

/// Button1 control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.Button Button1;

/// <summary>

/// Button2 control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.Button Button2;

/// <summary>

/// Button3 control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.Button Button3;

/// <summary>

/// TextBox1 control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.TextBox TextBox1;

/// <summary>

/// Button4 control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.Button Button4;

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Filename: RectorChooseReport.aspx.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using System;

using System.Collections.Generic;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

namespace KPIWeb.Rector

{

public partial class RectorChooseReport : System.Web.UI.Page

{

protected void Page\_Load(object sender, EventArgs e)

{

Serialization UserSer = (Serialization)Session["UserID"];

if (UserSer == null)

{

Response.Redirect("~/Default.aspx");

}

int userID = UserSer.Id;

KPIWebDataContext kPiDataContext = new KPIWebDataContext();

UsersTable userTable =

(from a in kPiDataContext.UsersTable where a.UsersTableID == userID select a).FirstOrDefault();

ViewState["login"] =

(from a in kPiDataContext.UsersTable

where a.UsersTableID == userID

select a.Email).FirstOrDefault();

/\*

\* if (userTable.AccessLevel != 3)

{

Response.Redirect("~/Default.aspx");

}\*/

RectorHistorySession RectorHistory = (RectorHistorySession) Session["rectorHistory"];

if (RectorHistory == null)

{

GoForwardButton.Enabled = false;

}

//////////////////////////////////////////////////////////////////////////

if (!Page.IsPostBack)

{

//ParametrType paramType = new ParametrType(1);;

ParametrType paramType = (ParametrType)Session["paramType"];

if (paramType == null)

{

Response.Redirect("~/Default.aspx");

}

if (paramType.paramType == 0) //смотрим индикацелевой показатель

{

PageName.Text = "Работа с целевыми показателями";

}

else

{

PageName.Text = "Работа с первичными данными";

}

KPIWebDataContext kpiWebDataContext = new KPIWebDataContext();

List<ReportArchiveTable> reportsArchiveTablesTable = null;

reportsArchiveTablesTable = (from a in kpiWebDataContext.ReportArchiveTable

select a).ToList();

DataTable dataTable = new DataTable();

dataTable.Columns.Add(new DataColumn("ReportArchiveID", typeof(string)));

dataTable.Columns.Add(new DataColumn("ReportName", typeof(string)));

dataTable.Columns.Add(new DataColumn("StartDate", typeof(string)));

dataTable.Columns.Add(new DataColumn("EndDate", typeof(string)));

foreach (ReportArchiveTable ReportRow in reportsArchiveTablesTable)

{

DataRow dataRow = dataTable.NewRow();

dataRow["ReportArchiveID"] = ReportRow.ReportArchiveTableID.ToString();

dataRow["ReportName"] = ReportRow.Name;

dataRow["StartDate"] = ReportRow.StartDateTime.ToString().Split(' ')[0];

dataRow["EndDate"] = ReportRow.EndDateTime.ToString().Split(' ')[0];

dataTable.Rows.Add(dataRow);

}

GridView1.DataSource = dataTable;

GridView1.DataBind();

}

}

protected void ButtonViewClick(object sender, EventArgs e)

{

Button button = (Button)sender;

{

ParametrType paramType = (ParametrType)Session["paramType"];

if (paramType == null)

{

Response.Redirect("~/Default.aspx");

}

if (paramType.paramType == 0) //смотрим индцелевой показатель

{

ForRCalc.Struct mainStruct = new ForRCalc.Struct(1, "");

RectorSession rectorResultSession = new RectorSession(mainStruct, 1, 0, 0,

Convert.ToInt32(button.CommandArgument), 0);

RectorHistorySession RectorHistory = new RectorHistorySession();

RectorHistory.SessionCount = 1;

RectorHistory.CurrentSession = 0;

RectorHistory.Visible = false;

RectorHistory.RectorSession[RectorHistory.CurrentSession] = rectorResultSession;

Session["rectorHistory"] = RectorHistory;

LogHandler.LogWriter.WriteLog(LogCategory.INFO, "0RCR0: Prorector " + (string)ViewState["login"] + " pereshel k rabote s othetom, ID = " + button.CommandArgument);

Response.Redirect("~/Rector/Result.aspx");

}

else // смотрим рассчетные

{

ForRCalc.Struct mainStruct = new ForRCalc.Struct(1, "");

RectorSession rectorResultSession = new RectorSession(mainStruct, 1, 0, 1,

Convert.ToInt32(button.CommandArgument), 0);

RectorHistorySession RectorHistory = new RectorHistorySession();

RectorHistory.SessionCount = 1;

RectorHistory.CurrentSession = 0;

RectorHistory.Visible = false;

RectorHistory.RectorSession[RectorHistory.CurrentSession] = rectorResultSession;

Session["rectorHistory"] = RectorHistory;

LogHandler.LogWriter.WriteLog(LogCategory.INFO, "0RCR1: Prorector " + (string)ViewState["login"] + " pereshel k rabote s othetom, ID = " + button.CommandArgument);

Response.Redirect("~/Rector/Result.aspx");

}

}

}

protected void GoBackButton\_Click(object sender, EventArgs e)

{

Response.Redirect("~/Rector/RectorMain.aspx");

}

protected void GoForwardButton\_Click(object sender, EventArgs e)

{

Response.Redirect("~/Rector/Result.aspx");

}

protected void Button4\_Click(object sender, EventArgs e)

{

Response.Redirect("~/Rector/RectorMain.aspx");

}

protected void Button5\_Click(object sender, EventArgs e)

{

Response.Redirect("~/Rector/ViewDocument.aspx");

}

protected void Button6\_Click(object sender, EventArgs e)

{

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Filename: RectorChooseReport.aspx.designer.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//------------------------------------------------------------------------------

// <auto-generated>

// This code was generated by a tool.

//

// Changes to this file may cause incorrect behavior and will be lost if

// the code is regenerated.

// </auto-generated>

//------------------------------------------------------------------------------

namespace KPIWeb.Rector {

public partial class RectorChooseReport {

/// <summary>

/// top\_panel2 control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.Panel top\_panel2;

/// <summary>

/// GoBackButton control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.Button GoBackButton;

/// <summary>

/// GoForwardButton control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.Button GoForwardButton;

/// <summary>

/// Button2 control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.Button Button2;

/// <summary>

/// Button5 control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.Button Button5;

/// <summary>

/// PageName control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.Label PageName;

/// <summary>

/// GridView1 control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.GridView GridView1;

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Filename: RectorMain.aspx.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

namespace KPIWeb.Rector

{

public partial class RectorMain : System.Web.UI.Page

{

protected void Page\_Load(object sender, EventArgs e)

{

Serialization UserSer = (Serialization)Session["UserID"];

if (UserSer == null)

{

Response.Redirect("~/Default.aspx");

}

int userID = UserSer.Id;

ViewState["LocalUserID"] = userID;

KPIWebDataContext kPiDataContext = new KPIWebDataContext();

UsersTable userTable =

(from a in kPiDataContext.UsersTable where a.UsersTableID == userID select a).FirstOrDefault();

var login =

(from a in kPiDataContext.UsersTable

where a.UsersTableID == (int)ViewState["LocalUserID"]

select a.Email).FirstOrDefault();

ViewState["login"] = login;

if (!IsPostBack)

LogHandler.LogWriter.WriteLog(LogCategory.INFO, "0RM0: Prorector " + (string)ViewState["login"] + " moved to page (RectorMain)");

if (userTable.AccessLevel != 5)

{

Response.Redirect("~/Default.aspx");

}

ParametrType paramType = (ParametrType)Session["paramType"];

if (paramType == null)

{

GoForwardButton.Enabled=false;

}

}

protected void Button1\_Click(object sender, EventArgs e)

{

ParametrType paramType = new ParametrType(0);

Session["paramType"] = paramType;

LogHandler.LogWriter.WriteLog(LogCategory.INFO, "0RM1: Prorector " + (string)ViewState["login"] + " vibral raboty s Celevimi Pokazatelyami i pereshel na stranicy vibora otcheta");

Response.Redirect("~/Rector/RectorChooseReport.aspx");

}

protected void Button3\_Click(object sender, EventArgs e)

{

ParametrType paramType = new ParametrType(1);

Session["paramType"] = paramType;

LogHandler.LogWriter.WriteLog(LogCategory.INFO, "0RM2: Prorector " + (string)ViewState["login"] + " vibral raboty s Pervichnimi dannimy i pereshel na stranicy vibora otcheta");

Response.Redirect("~/Rector/RectorChooseReport.aspx");

}

protected void Button4\_Click(object sender, EventArgs e)

{

Response.Redirect("~/Rector/RectorMain.aspx");

}

protected void GoBackButton\_Click(object sender, EventArgs e)

{

}

protected void GoForwardButton\_Click(object sender, EventArgs e)

{

Response.Redirect("~/Rector/RectorChooseReport.aspx");

}

protected void Button5\_Click(object sender, EventArgs e)

{

Response.Redirect("~/Rector/ViewDocument.aspx");

}

protected void Button6\_Click(object sender, EventArgs e)

{

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Filename: RectorMain.aspx.designer.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//------------------------------------------------------------------------------

// <автоматически создаваемое>

// Этот код создан программой.

//

// Изменения в этом файле могут привести к неправильной работе и будут потеряны в случае

// повторной генерации кода.

// </автоматически создаваемое>

//------------------------------------------------------------------------------

namespace KPIWeb.Rector {

public partial class RectorMain {

/// <summary>

/// top\_panel2 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Panel top\_panel2;

/// <summary>

/// GoBackButton элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Button GoBackButton;

/// <summary>

/// GoForwardButton элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Button GoForwardButton;

/// <summary>

/// Button2 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Button Button2;

/// <summary>

/// Button5 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Button Button5;

/// <summary>

/// Button6 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Button Button6;

/// <summary>

/// PageName элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Label PageName;

/// <summary>

/// Button1 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Button Button1;

/// <summary>

/// Button3 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Button Button3;

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Filename: Result.aspx.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using System;

using System.Collections.Generic;

using System.ComponentModel.Design;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

using System.Web.WebPages;

using System.Net;

using Microsoft.Ajax.Utilities;

using Button = System.Web.UI.WebControls.Button;

using DataTable = System.Data.DataTable;

using Label = System.Web.UI.WebControls.Label;

namespace KPIWeb.Rector

{

public partial class Result : System.Web.UI.Page

{

public class MyObject

{

public int Id;

public int ParentId;

public string Name;

public string UrlAddr;

public int Active;

}

protected void Page\_Load(object sender, EventArgs e)

{

#region get user data

Panel5.Style.Add("background-color", "rgba(0, 255, 0, 0.3)");

Panel7.Style.Add("background-color", "rgba(255, 0, 0, 0.3)");

Panel6.Style.Add("background-color", "rgba(255, 255, 0, 0.3)");

string parameter = Request["\_\_EVENTARGUMENT"];

if (parameter != null)

{

int ParamId = -1;

if (int.TryParse(parameter, out ParamId) && ParamId > 0)

{

DoConfirm(ParamId);

}

}

KPIWebDataContext kpiWebDataContext = new KPIWebDataContext();

Serialization UserSer = (Serialization)Session["UserID"];

if (UserSer == null)

{

Response.Redirect("~/Default.aspx");

}

int userID = UserSer.Id;

UsersTable userTable\_ =

(from a in kpiWebDataContext.UsersTable where a.UsersTableID == userID select a).FirstOrDefault();

ViewState["login"] = (from a in kpiWebDataContext.UsersTable where a.UsersTableID == userID select a.Email).FirstOrDefault();

if (userTable\_.AccessLevel != 5)

{

Response.Redirect("~/Default.aspx");

}

#endregion

if (!IsPostBack)

{

#region session

RectorHistorySession rectorHistory = (RectorHistorySession) Session["rectorHistory"];

if (rectorHistory == null)

{

Response.Redirect("~/Default.aspx");

}

ShowUnConfirmed unConfirmed = (ShowUnConfirmed) Session["unConfirmed"];

bool ShowUnconfirmed = true;

if (unConfirmed == null)

{

ShowUnconfirmed = false;

}

else

{

if (unConfirmed.DoShowUnConfirmed == false)

{

ShowUnconfirmed = false;

}

else

{

Button7.Enabled = false;

unConfirmed.DoShowUnConfirmed = false;

Session["unConfirmed"] = unConfirmed;

}

}

#region check for get

string val = this.Request.QueryString["HLevel"]; //hisoty level сова придумал)

if (val != null)

{

rectorHistory.CurrentSession = Convert.ToInt32(val);

Session["rectorHistory"] = rectorHistory;

}

#endregion

if ((rectorHistory.SessionCount - rectorHistory.CurrentSession) < 2)

{

GoForwardButton.Enabled = false;

}

RectorSession CurrentRectorSession = rectorHistory.RectorSession[rectorHistory.CurrentSession];

ForRCalc.Struct mainStruct = CurrentRectorSession.sesStruct;

int ViewType = CurrentRectorSession.sesViewType;

int ParamID = CurrentRectorSession.sesParamID;

int ParamType = CurrentRectorSession.sesParamType;

int ReportID = CurrentRectorSession.sesReportID;

int SpecID = CurrentRectorSession.sesSpecID;

#endregion

#region DataTable init

DataTable dataTable = new DataTable();

dataTable.Columns.Add(new DataColumn("ID", typeof (string)));

dataTable.Columns.Add(new DataColumn("Number", typeof (string)));

dataTable.Columns.Add(new DataColumn("Abb", typeof (string)));

dataTable.Columns.Add(new DataColumn("Name", typeof (string)));

dataTable.Columns.Add(new DataColumn("StartDate", typeof (string)));

dataTable.Columns.Add(new DataColumn("EndDate", typeof (string)));

dataTable.Columns.Add(new DataColumn("Value", typeof (string)));

dataTable.Columns.Add(new DataColumn("Title", typeof (string)));

dataTable.Columns.Add(new DataColumn("PlannedValue", typeof (string)));

dataTable.Columns.Add(new DataColumn("Progress", typeof (string)));

dataTable.Columns.Add(new DataColumn("Color", typeof (string)));

///color table

/// 0 - no color // can't confirm

/// 1 - green (confirmed)

/// 2 - red (unconfirmed but calculated)

/// 3 - orange (can confirm)

///

dataTable.Columns.Add(new DataColumn("CanWatchWhoOws", typeof(bool)));

dataTable.Columns.Add(new DataColumn("CanConfirm", typeof (bool)));

dataTable.Columns.Add(new DataColumn("ShowLable", typeof (bool)));

dataTable.Columns.Add(new DataColumn("LableText", typeof (string)));

dataTable.Columns.Add(new DataColumn("LableColor", typeof (string)));

dataTable.Columns.Add(new DataColumn("Comment", typeof(string)));

dataTable.Columns.Add(new DataColumn("CommentEnabled", typeof(string)));

#endregion

#region global page settings

ReportArchiveTable ReportTable = (from a in kpiWebDataContext.ReportArchiveTable

where a.ReportArchiveTableID == ReportID

select a).FirstOrDefault();

double daysLeft = ((DateTime) ReportTable.EndDateTime - DateTime.Now).TotalDays;

ReportTitle.Text = ReportTable.Name + " " + ReportTable.StartDateTime.ToString().Split(' ')[0] + " - " +

ReportTable.EndDateTime.ToString().Split(' ')[0];

#endregion

#region Show Uncinfirmed Button

Button7.Visible = false;

if ((daysLeft < ReportTable.DaysBeforeToCalcForRector) &&

(ReportTable.DaysBeforeToCalcForRector != 0))

{

Button7.Enabled = true;

}

else

{

Button7.Enabled = false;

}

if (ShowUnconfirmed == true)

{

Button7.Enabled = false;

}

#endregion

if (ViewType == 0) // просмотр для структурных подразделений

{

#region преднастройка страницы

string title = "";

PageFullName.Text = "";

if (ParamType == 0)

{

PageFullName.Text += "<b>";

PageFullName.Text += (from a in kpiWebDataContext.IndicatorsTable

where a.IndicatorsTableID == ParamID

select a.Name).FirstOrDefault();

PageFullName.Text += "</b> </br>";

}

else if (ParamType == 1)

{

PageFullName.Text += "<b>";

PageFullName.Text += (from a in kpiWebDataContext.CalculatedParametrs

where a.CalculatedParametrsID == ParamID

select a.Name).FirstOrDefault();

PageFullName.Text += "</b> </br>";

}

else if (ParamType == 2)

{

PageFullName.Text += "<b>";

PageFullName.Text += (from a in kpiWebDataContext.BasicParametersTable

where a.BasicParametersTableID == ParamID

select a.Name).FirstOrDefault();

PageFullName.Text += "</b> </br>";

}

int Deep = ForRCalc.StructDeepness(mainStruct);

if (Deep == 1)

{

}

if (Deep == 2)

{

PageFullName.Text += (from a in kpiWebDataContext.FirstLevelSubdivisionTable

where a.FirstLevelSubdivisionTableID == mainStruct.Lv\_1

select a.Name).FirstOrDefault();

PageFullName.Text += "</br>";

}

if (Deep == 3)

{

PageFullName.Text += (from a in kpiWebDataContext.FirstLevelSubdivisionTable

where a.FirstLevelSubdivisionTableID == mainStruct.Lv\_1

select a.Name).FirstOrDefault();

PageFullName.Text += "</br>";

PageFullName.Text += (from a in kpiWebDataContext.SecondLevelSubdivisionTable

where a.SecondLevelSubdivisionTableID == mainStruct.Lv\_2

select a.Name).FirstOrDefault();

PageFullName.Text += "</br>";

}

if (Deep == 4)

{

PageFullName.Text += (from a in kpiWebDataContext.FirstLevelSubdivisionTable

where a.FirstLevelSubdivisionTableID == mainStruct.Lv\_1

select a.Name).FirstOrDefault();

PageFullName.Text += "</br>";

PageFullName.Text += (from a in kpiWebDataContext.SecondLevelSubdivisionTable

where a.SecondLevelSubdivisionTableID == mainStruct.Lv\_2

select a.Name).FirstOrDefault();

PageFullName.Text += "</br>";

PageFullName.Text += (from a in kpiWebDataContext.ThirdLevelSubdivisionTable

where a.ThirdLevelSubdivisionTableID == mainStruct.Lv\_3

select a.Name).FirstOrDefault();

PageFullName.Text += "</br>";

}

if (SpecID != 0)

{

PageFullName.Text += "Направление подготовки \"" + (from a in kpiWebDataContext.SpecializationTable

where a.SpecializationTableID == SpecID

select a.Name).FirstOrDefault() + "\" </br>";

}

if (mainStruct.Lv\_1 == 0)

{

RectorSession tmpses = new RectorSession(mainStruct, ViewType, ParamID, ParamType, ReportID,

SpecID, "КФУ");

rectorHistory.RectorSession[rectorHistory.CurrentSession] = tmpses;

Session["rectorHistory"] = rectorHistory;

}

else

{

RectorSession tmpses = new RectorSession(mainStruct, ViewType, ParamID, ParamType, ReportID,

SpecID, mainStruct.Name);

rectorHistory.RectorSession[rectorHistory.CurrentSession] = tmpses;

Session["rectorHistory"] = rectorHistory;

}

//задади имя текущей сессии

title = "Подразделения";

if (ForRCalc.StructDeepness(mainStruct) > 3)

{

title = "Направления подготовки";

}

#endregion

#region fill grid

int BasicParamLevel = 0;

if (ParamType == 2)

{

BasicParamLevel = (int)(from a in kpiWebDataContext.BasicParametrAdditional

select a.SubvisionLevel).FirstOrDefault();

}

List<ForRCalc.Struct> currentStructList = new List<ForRCalc.Struct>();

if (SpecID != 0)

{

currentStructList = ForRCalc.GetChildStructList(mainStruct, ReportID, SpecID);

}

else

{

currentStructList = ForRCalc.GetChildStructList(mainStruct, ReportID);

}

foreach (ForRCalc.Struct currentStruct in currentStructList)

{

DataRow dataRow = dataTable.NewRow();

dataRow["ID"] = ForRCalc.GetLastID(currentStruct).ToString();

dataRow["Number"] = "num";

dataRow["Name"] = currentStruct.Name;

dataRow["StartDate"] = "nun";

dataRow["EndDate"] = "nun";

dataRow["Comment"] = "nun";

dataRow["CommentEnabled"] = "hidden";

dataRow["CanConfirm"] = true;

dataRow["ShowLable"] = false;

dataRow["CanWatchWhoOws"] = false;

dataRow["LableText"] = "";

dataRow["LableColor"] = "#000000";

dataRow["Value"] =

ForRCalc.GetCalculatedWithParams(currentStruct, ParamType, ParamID, ReportID, SpecID).ToString();

dataTable.Rows.Add(dataRow);

}

#endregion

#region DataGridBind

Grid.DataSource = dataTable;

Grid.Columns[3].HeaderText = title;

Grid.DataBind();

#endregion

#region постнастройка страницы

Grid.Columns[12].Visible = false;

Grid.Columns[11].Visible = false;

Grid.Columns[9].Visible = false;

Grid.Columns[8].Visible = false;

Grid.Columns[7].Visible = false;

Grid.Columns[5].Visible = false;

Grid.Columns[4].Visible = false;

Grid.Columns[2].Visible = false;

Grid.Columns[1].Visible = false;

if ((ForRCalc.StructDeepness(mainStruct) > (BasicParamLevel - 1)) ||

(ForRCalc.StructDeepness(mainStruct) > 2) && (SpecID != 0)) // дальше углубляться нельзя

{

Grid.Columns[10].Visible = false;

}

#endregion

}

else if (ViewType == 1) // просмотр для показателей (верхние 3 шт)

{

#region преднастройка страницы

string name\_text = "";

string value\_text = "";

string progress\_text = "";

string confirm\_text = "";

string detalize\_text = "";

if (ParamType == 0)

{

RectorSession tmpses = new RectorSession(mainStruct, ViewType, ParamID, ParamType, ReportID,

SpecID, "Значения целевых показателей для КФУ");

rectorHistory.RectorSession[rectorHistory.CurrentSession] = tmpses;

Session["rectorHistory"] = rectorHistory;

PageFullName.Text = "Значения целевых показателей (ЦП) для КФУ";

//PageName.Text = "Значения индикторов для КФУ";

name\_text = "Название ЦП";

value\_text = "Значение ЦП";

progress\_text = "Степень готовности первичных данных";

confirm\_text = "Утвердить ЦП";

detalize\_text = "Просмотреть первичные данные для ЦП";

}

else if (ParamType == 1)

{

if (ParamID == 0)

{

RectorSession tmpses = new RectorSession(mainStruct, ViewType, ParamID, ParamType, ReportID,

SpecID, "Значения первичных данных (ПД) для КФУ");

rectorHistory.RectorSession[rectorHistory.CurrentSession] = tmpses;

Session["rectorHistory"] = rectorHistory;

PageFullName.Text = "Значения первичных данных (ПД) для КФУ";

}

else

{

string tmp = (from a in kpiWebDataContext.IndicatorsTable

where a.IndicatorsTableID == ParamID

select a.Name).FirstOrDefault();

RectorSession tmpses = new RectorSession(mainStruct, ViewType, ParamID, ParamType, ReportID,

SpecID, "Первичные данные для целевого показателя: " + tmp);

rectorHistory.RectorSession[rectorHistory.CurrentSession] = tmpses;

Session["rectorHistory"] = rectorHistory;

PageFullName.Text = "Первичные данные (ПД) целевого показателя <b> \"" + tmp + "\"</b> для КФУ";

FormulaLable.Text = (from a in kpiWebDataContext.IndicatorsTable

where a.IndicatorsTableID == ParamID

select a.Formula).FirstOrDefault();

FormulaLable.Visible = true;

}

name\_text = "Названия ПД";

value\_text = "Значение ПД";

progress\_text = "Степень готовности базовых показателей";

confirm\_text = "Утвердить ПД";

detalize\_text = "Просмотреть базовые показатели для ПД";

}

else if (ParamType == 2)

{

string tmp = (from a in kpiWebDataContext.CalculatedParametrs

where a.CalculatedParametrsID == ParamID

select a.Name).FirstOrDefault();

RectorSession tmpses = new RectorSession(mainStruct, ViewType, ParamID, ParamType, ReportID,

SpecID, "Базовые показатели для первич: " + tmp);

rectorHistory.RectorSession[rectorHistory.CurrentSession] = tmpses;

Session["rectorHistory"] = rectorHistory;

PageFullName.Text = "Базовые показатели первичного показателя <b> \"" + tmp + "\"</b> для КФУ";

name\_text = "Названия БП";

value\_text = "Значение БП";

//progress\_text = "Степень готовности базовых показателей";

//confirm\_text = "Утвердить ПД";

//Ыdetalize\_text = "Просмотреть базовые показатели для ПД";

FormulaLable.Text = (from a in kpiWebDataContext.CalculatedParametrs

where a.CalculatedParametrsID == ParamID

select a.Formula).FirstOrDefault();

FormulaLable.Visible = true;

}

#endregion

#region fill grid

if (ParamType == 0) //считаем целевой показатель

{

#region indicator

List<IndicatorsTable> IndicatorsNotUnique = (

from a in kpiWebDataContext.IndicatorsTable

join b in kpiWebDataContext.IndicatorsAndUsersMapping

on a.IndicatorsTableID equals b.FK\_IndicatorsTable

join c in kpiWebDataContext.ReportArchiveAndIndicatorsMappingTable

on a.IndicatorsTableID equals c.FK\_IndicatorsTable

where

a.Active == true

&& b.CanView == true

&& b.FK\_UsresTable == userID

&& c.FK\_ReportArchiveTable == ReportID

select a).OrderBy(mc => mc.SortID).ToList();

////для уникальнности

int IDForUnique=0;

List<IndicatorsTable> Indicators = new List<IndicatorsTable>() ;

foreach (IndicatorsTable CurrentIndicator in IndicatorsNotUnique)

{

if (CurrentIndicator.IndicatorsTableID != IDForUnique)

{

Indicators.Add(CurrentIndicator);

}

IDForUnique = CurrentIndicator.IndicatorsTableID;

}

// теперь повторяющихся индикаторов нет

//нашли все целевой показатель привязанные к пользователю

foreach (IndicatorsTable CurrentIndicator in Indicators)

{

DataRow dataRow = dataTable.NewRow();

dataRow["ID"] = CurrentIndicator.IndicatorsTableID; //GetLastID(currentStruct).ToString();

dataRow["Number"] = "num";

if (CurrentIndicator.Measure!=null)

{

if (CurrentIndicator.Measure.Length>0)

{

dataRow["Name"] = CurrentIndicator.Name + " (" + CurrentIndicator.Measure + ")";

}

else

{

dataRow["Name"] = CurrentIndicator.Name;

}

}

else

{

dataRow["Name"] = CurrentIndicator.Name;

}

dataRow["CanWatchWhoOws"] = false;

dataRow["StartDate"] = "nun";

dataRow["EndDate"] = "nun";

ConfirmationHistory CommentRow = (from a in kpiWebDataContext.ConfirmationHistory

where a.FK\_IndicatorsTable == CurrentIndicator.IndicatorsTableID

&& a.FK\_ReportTable == ReportID

select a).OrderByDescending(mc => mc.Date).FirstOrDefault();

if (CommentRow != null)

{

if (CommentRow.Comment.Length > 0)

{

dataRow["Comment"] = "От: " + CommonCode.GetUserById(Convert.ToInt32(CommentRow.FK\_UsersTable)) +"</br>"+ CommentRow.Comment;

dataRow["CommentEnabled"] = "visible";

}

else

{

dataRow["Comment"] = "nun";

dataRow["CommentEnabled"] = "hidden";

}

}

else

{

dataRow["Comment"] = "nun";

dataRow["CommentEnabled"] = "hidden";

}

PlannedIndicator plannedValue = (from a in kpiWebDataContext.PlannedIndicator

where a.FK\_IndicatorsTable == CurrentIndicator.IndicatorsTableID

&& a.Date > DateTime.Now

//Q1

select a).OrderBy(x => x.Date).FirstOrDefault();

if (plannedValue != null)

{

dataRow["PlannedValue"] = plannedValue.Value;

}

else

{

dataRow["PlannedValue"] = "Не определено";

}

#region user can confirm

bool canConfirm = (bool) (from a in kpiWebDataContext.IndicatorsAndUsersMapping

where a.FK\_IndicatorsTable == CurrentIndicator.IndicatorsTableID

&& a.FK\_UsresTable == userID

select a.CanConfirm).FirstOrDefault();

#endregion

# region are calculated confirmed

List<CalculatedParametrs> CalculatedList =

Abbreviature.GetCalculatedList(CurrentIndicator.Formula);

bool CalcAreConfirmed = true;

int AllCalculated = 0;

int AllConfirmedCalculated = 0;

foreach (CalculatedParametrs CurrentCalculated in CalculatedList)

{

AllCalculated++;

CollectedCalculatedParametrs tmp\_ =

(from a in kpiWebDataContext.CollectedCalculatedParametrs

where

a.FK\_CalculatedParametrs == CurrentCalculated.CalculatedParametrsID

&& a.FK\_ReportArchiveTable == ReportID

select a).FirstOrDefault();

if (tmp\_ == null)

{

CalcAreConfirmed = false;

}

else

{

if (tmp\_.Confirmed == null)

{

CalcAreConfirmed = false;

}

else if (tmp\_.Confirmed == false)

{

CalcAreConfirmed = false;

}

else

{

AllConfirmedCalculated++;

}

}

}

dataRow["Progress"] = AllConfirmedCalculated.ToString() + " из " + AllCalculated.ToString();

#endregion

#region get calculated if confirmed; calculate if not confirmed

string value\_ = "";

CollectedIndocators collected = (from a in kpiWebDataContext.CollectedIndocators

where a.FK\_ReportArchiveTable == ReportID

&& a.FK\_Indicators == CurrentIndicator.IndicatorsTableID

select a).FirstOrDefault();

if (collected == null)

{

collected = new CollectedIndocators();

collected.FK\_Indicators = CurrentIndicator.IndicatorsTableID;

collected.FK\_ReportArchiveTable = ReportID;

collected.FK\_UsersTable = userID;

collected.Confirmed = false;

collected.LastChangeDateTime = DateTime.Now;

collected.Active = true;

collected.CollectedValue = 0;

kpiWebDataContext.CollectedIndocators.InsertOnSubmit(collected);

kpiWebDataContext.SubmitChanges();

}

if (collected.Confirmed == true) // данные подтверждены

{

dataRow["CanConfirm"] = false;

dataRow["ShowLable"] = true;

dataRow["LableText"] = "Утверждено";

dataRow["Color"] = "1"; // confirmed

value\_ = ((float) collected.CollectedValue).ToString("0.00");

}

else // данные уже есть но еще не подтверждены

{

if (canConfirm == false)

{

dataRow["CanConfirm"] = false;

dataRow["ShowLable"] = true;

dataRow["LableText"] = " "; // нет права утверждать

dataRow["LableColor"] = "#101010";

value\_ = "Недостаточно данных";

if (ShowUnconfirmed)

{

dataRow["Color"] = "2";

/// 0 - no color // can't confirm

/// 1 - green (confirmed)

/// 2 - red (unconfirmed but calculated)

/// 3 - orange (can confirm)

///

float tmp =

ForRCalc.CalculatedForDB(ForRCalc.GetCalculatedWithParams(mainStruct, ParamType,

CurrentIndicator.IndicatorsTableID, ReportID, SpecID));

if (tmp == (float) 1E+20)

{

value\_ = "Рассчет невозможен";

}

else

{

value\_ = tmp.ToString("0.00");

}

}

}

else if (!CalcAreConfirmed)

{

/\*dataRow["CanConfirm"] = false;

dataRow["ShowLable"] = true;

dataRow["LableText"] = "Не все расчетные утверждены";

\*/

dataRow["CanConfirm"] = true;

dataRow["ShowLable"] = false;

dataRow["LableText"] = "";

dataRow["LableColor"] = "#101010";

value\_ = "Недостаточно данных";

if (ShowUnconfirmed)

{

dataRow["Color"] = "2";

float tmp =

ForRCalc.CalculatedForDB(ForRCalc.GetCalculatedWithParams(mainStruct, ParamType,

CurrentIndicator.IndicatorsTableID, ReportID, SpecID));

if (tmp == (float) 1E+20)

{

value\_ = "Рассчет невозможен";

}

else

{

value\_ = tmp.ToString("0.00");

}

}

}

else

{

dataRow["CanConfirm"] = true;

dataRow["ShowLable"] = false;

dataRow["LableText"] = "";

dataRow["LableColor"] = "#FFFFFF";

dataRow["Color"] = "3";

collected.CollectedValue =

ForRCalc.CalculatedForDB(ForRCalc.GetCalculatedWithParams(mainStruct, ParamType,

CurrentIndicator.IndicatorsTableID, ReportID, SpecID));

//12;

kpiWebDataContext.SubmitChanges();

value\_ = ((float) collected.CollectedValue).ToString("0.00");

}

}

dataRow["Value"] = value\_;

#endregion

dataTable.Rows.Add(dataRow);

}

#endregion indicator

}

if (ParamType == 1) //показываем рассчетный входящий в ID целевой показатель

{

#region calculated

//ID - это айди Индиктора

List<CalculatedParametrs> CalculatedList;

if (ParamID != 0)

{

IndicatorsTable Indicator = (from a in kpiWebDataContext.IndicatorsTable

where a.IndicatorsTableID == ParamID

select a).FirstOrDefault();

CalculatedList = Abbreviature.GetCalculatedList(Indicator.Formula);

}

else

{

CalculatedList = (from a in kpiWebDataContext.CalculatedParametrs

join b in kpiWebDataContext.CalculatedParametrsAndUsersMapping

on a.CalculatedParametrsID equals b.FK\_CalculatedParametrsTable

join c in kpiWebDataContext.ReportArchiveAndCalculatedParametrsMappingTable

on a.CalculatedParametrsID equals c.FK\_CalculatedParametrsTable

where a.Active == true

&& b.CanView == true

&& b.FK\_UsersTable == userID

&& c.FK\_ReportArchiveTable == ReportID

select a).ToList();

}

////для уникальнности

CalculatedList = CalculatedList.OrderBy(o => o.CalculatedParametrsID).ToList();

// List<CalculatedParametrs> SortedList = CalculatedList.OrderBy(o => o.CalculatedParametrsID).ToList();

int IDForUnique = 0;

List<CalculatedParametrs> CalculatedListUnique = new List<CalculatedParametrs>();

foreach (CalculatedParametrs CurrentCalc in CalculatedList)

{

if (CurrentCalc.CalculatedParametrsID != IDForUnique)

{

CalculatedListUnique.Add(CurrentCalc);

}

IDForUnique = CurrentCalc.CalculatedParametrsID;

}

// теперь повторяющихся нет

foreach (CalculatedParametrs CurrentCalculated in CalculatedListUnique)

{

DataRow dataRow = dataTable.NewRow();

dataRow["ID"] = CurrentCalculated.CalculatedParametrsID;

//GetLastID(currentStruct).ToString();

dataRow["Number"] = "num";

if (CurrentCalculated.Measure != null)

{

if (CurrentCalculated.Measure.Length > 0)

{

dataRow["Name"] = CurrentCalculated.Name + " (" + CurrentCalculated.Measure + ")";

}

else

{

dataRow["Name"] = CurrentCalculated.Name;

}

}

else

{

dataRow["Name"] = CurrentCalculated.Name;

}

//

//dataRow["Name"] = CurrentCalculated.Name + " (" + CurrentCalculated.Measure + ")";

dataRow["StartDate"] = "nun";

dataRow["EndDate"] = "nun";

ConfirmationHistory CommentRow = (from a in kpiWebDataContext.ConfirmationHistory

where a.FK\_CalculatedParamTable == CurrentCalculated.CalculatedParametrsID

&& a.FK\_ReportTable == ReportID

select a).OrderByDescending(mc => mc.Date).FirstOrDefault();

if (CommentRow != null)

{

if (CommentRow.Comment.Length > 0)

{

dataRow["Comment"] = CommentRow.Comment;

dataRow["CommentEnabled"] = "visible";

}

else

{

dataRow["Comment"] = "nun";

dataRow["CommentEnabled"] = "hidden";

}

}

else

{

dataRow["Comment"] = "nun";

dataRow["CommentEnabled"] = "hidden";

}

// dataRow["CanWatchWhoOws"] = "false";

// dataRow["CanConfirm"] = "true";

// dataRow["ShowLable"] = "false";

dataRow["Abb"] = CurrentCalculated.AbbreviationEN;

#region get calculated if confirmed; calculate if not confirmed

#region user can edit

CalculatedParametrsAndUsersMapping Calc = (from a in kpiWebDataContext.CalculatedParametrsAndUsersMapping

where a.FK\_CalculatedParametrsTable == CurrentCalculated.CalculatedParametrsID

&& a.FK\_UsersTable == userID

select a).FirstOrDefault();

bool canConfirm;

if (Calc != null)

{

canConfirm = (bool)Calc.CanConfirm;

}

else

{

canConfirm = false;

}

#endregion

#region check if all users confirmed basics

List<BasicParametersTable> BasicList = Abbreviature.GetBasicList(CurrentCalculated.Formula);

int AllBasicsUsersCanEdit = 0;

int AllConfirmedBasics = 0;

int AllConnectedToReportAndUser = 0;

/\*

List<UsersTable> UsersWithNoCollected = (from a in kpiWebDataContext.UsersTable

join b in kpiWebDataContext.BasicParametrsAndUsersMapping

on a.UsersTableID equals b.FK\_UsersTable

where a.Active == true

&& b.Active == true

&& b.CanEdit == true

join c in kpiWebDataContext.ReportArchiveAndLevelMappingTable

on a.FK\_FirstLevelSubdivisionTable equals c.FK\_FirstLevelSubmisionTableId

where c.Active == true

&& c.FK\_ReportArchiveTableId == ReportID

join d in kpiWebDataContext.ReportArchiveAndBasicParametrsMappingTable

on b.FK\_ParametrsTable equals d.FK\_BasicParametrsTable

where d.Active == true

&& d.FK\_ReportArchiveTable == ReportID

select a).Distinct().ToList();\*/

foreach (BasicParametersTable Basic in BasicList)

{

AllBasicsUsersCanEdit += (from a in kpiWebDataContext.CollectedBasicParametersTable

where a.FK\_ReportArchiveTable == ReportID

&& a.FK\_BasicParametersTable == Basic.BasicParametersTableID

select a).Count();

AllConfirmedBasics += (from a in kpiWebDataContext.CollectedBasicParametersTable

where a.FK\_ReportArchiveTable == ReportID

&& a.FK\_BasicParametersTable == Basic.BasicParametersTableID

&& a.Status == 4

select a).Count();

}

bool BasicsAreConfirmed = true;

if (AllBasicsUsersCanEdit != AllConfirmedBasics)

{

BasicsAreConfirmed = false;

}

#endregion

dataRow["Progress"] =

((((float) AllConfirmedBasics)\*100)/((float) AllBasicsUsersCanEdit)).ToString("0.0") +

"%";

if (float.IsNaN((((float)AllConfirmedBasics) \* 100) / ((float)AllBasicsUsersCanEdit)))

{

dataRow["Progress"] = "100,0%";

}

string value\_ = "";

CollectedCalculatedParametrs collected =

(from a in kpiWebDataContext.CollectedCalculatedParametrs

where a.FK\_ReportArchiveTable == ReportID

&& a.FK\_CalculatedParametrs == CurrentCalculated.CalculatedParametrsID

select a).FirstOrDefault();

if (collected == null) // данных нет

{

collected = new CollectedCalculatedParametrs();

collected.FK\_CalculatedParametrs = CurrentCalculated.CalculatedParametrsID;

collected.FK\_ReportArchiveTable = ReportID;

collected.FK\_UsersTable = userID;

collected.Confirmed = false;

collected.LastChangeDateTime = DateTime.Now;

collected.Active = true;

collected.CollectedValue = ForRCalc.GetCalculatedWithParams(mainStruct, ParamType,

CurrentCalculated.CalculatedParametrsID, ReportID, SpecID); //11;

kpiWebDataContext.CollectedCalculatedParametrs.InsertOnSubmit(collected);

kpiWebDataContext.SubmitChanges();

}

UsersTable ConfirmUser = (from a in kpiWebDataContext.UsersTable

join b in kpiWebDataContext.CalculatedParametrsAndUsersMapping

on a.UsersTableID equals b.FK\_UsersTable

where

b.CanConfirm == true

&& a.Active == true

&& b.FK\_CalculatedParametrsTable == CurrentCalculated.CalculatedParametrsID

&& b.Active == true

select a).FirstOrDefault();

string UserName = "";

if (ConfirmUser != null)

{

if (ConfirmUser.Position != null)

{

if (ConfirmUser.Position.Length > 2)

{

UserName = ConfirmUser.Position;

}

else

{

UserName = ConfirmUser.Email;

}

}

else

{

UserName = ConfirmUser.Email;

}

}

else

{

UserName = "не определено";

}

if (collected.Confirmed == true) //данные подтверждены

{

dataRow["CanWatchWhoOws"] = false;

dataRow["CanConfirm"] = false;

dataRow["ShowLable"] = true;

dataRow["LableText"] = "Утверждено ";

dataRow["LableColor"] = Color.LawnGreen;

dataRow["Color"] = "1";

value\_ = ((float) collected.CollectedValue).ToString("0.00");

}

else // данные есть но не подтверждены

{

if (canConfirm == false) //

{

dataRow["CanWatchWhoOws"] = false;

dataRow["CanConfirm"] = false;

dataRow["ShowLable"] = true;

dataRow["LableText"] = "Ответственный: " + UserName;

value\_ = "Недостаточно данных";

if (ShowUnconfirmed)

{

dataRow["Color"] = "2";

float tmp =

ForRCalc.CalculatedForDB(ForRCalc.GetCalculatedWithParams(mainStruct, ParamType,

CurrentCalculated.CalculatedParametrsID, ReportID, SpecID));

if (tmp == (float) 1E+20)

{

value\_ = "Рассчет невозможен";

}

else

{

value\_ = tmp.ToString("0.00");

}

}

}

else if (BasicsAreConfirmed == false)

{

dataRow["CanWatchWhoOws"] = true;

dataRow["CanConfirm"] = false;

dataRow["ShowLable"] = false;

dataRow["LableText"] = "";

/\*

dataRow["CanConfirm"] = false;

dataRow["ShowLable"] = true;

dataRow["LableText"] = "Не все базовые показатели утверждены";\*/

dataRow["LableColor"] = Color.LightBlue;

value\_ = "Недостаточно данных";

if (ShowUnconfirmed)

{

dataRow["Color"] = "2";

float tmp =

ForRCalc.CalculatedForDB(ForRCalc.GetCalculatedWithParams(mainStruct, ParamType,

CurrentCalculated.CalculatedParametrsID, ReportID, SpecID));

if (tmp == (float) 1E+20)

{

value\_ = "Рассчет невозможен";

}

else

{

value\_ = tmp.ToString("0.00");

}

}

}

else

{

dataRow["CanConfirm"] = true;

dataRow["CanWatchWhoOws"] = false;

dataRow["ShowLable"] = false;

dataRow["LableText"] = "";

dataRow["Color"] = "3";

dataRow["LableColor"] = "#000000";

collected.CollectedValue =

ForRCalc.CalculatedForDB(ForRCalc.GetCalculatedWithParams(mainStruct, ParamType,

CurrentCalculated.CalculatedParametrsID, ReportID, SpecID));

kpiWebDataContext.SubmitChanges();

value\_ = ((float) collected.CollectedValue).ToString("0.00");

}

}

dataRow["Value"] = value\_;

#endregion

dataTable.Rows.Add(dataRow);

}

#endregion

}

if (ParamType == 2) //

{

#region basic

//ID - Рассчетного айдишник

CalculatedParametrs Calculated = (from a in kpiWebDataContext.CalculatedParametrs

where a.CalculatedParametrsID == ParamID

select a).FirstOrDefault();

List<BasicParametersTable> BasicList = Abbreviature.GetBasicList(Calculated.Formula);

////для уникальнности

BasicList = BasicList.OrderBy(mc => mc.BasicParametersTableID).ToList();

int IDForUnique = 0;

List<BasicParametersTable> BasicListUnique = new List<BasicParametersTable>();

// List<CalculatedParametrs> CalculatedListUnique = new List<CalculatedParametrs>();

foreach (BasicParametersTable CurrebtBasic in BasicList)

{

if (CurrebtBasic.BasicParametersTableID != IDForUnique)

{

BasicListUnique.Add(CurrebtBasic);

}

IDForUnique = CurrebtBasic.BasicParametersTableID;

}

// теперь повторяющихся нет

foreach (BasicParametersTable CurrebtBasic in BasicListUnique)

{

DataRow dataRow = dataTable.NewRow();

dataRow["ID"] = CurrebtBasic.BasicParametersTableID; //GetLastID(currentStruct).ToString();

dataRow["Number"] = "num";

if (CurrebtBasic.Measure != null)

{

if (CurrebtBasic.Measure.Length > 0)

{

dataRow["Name"] = CurrebtBasic.Name + " (" + CurrebtBasic.Measure + ")";

}

else

{

dataRow["Name"] = CurrebtBasic.Name;

}

}

else

{

dataRow["Name"] = CurrebtBasic.Name;

}

// dataRow["Name"] = CurrebtBasic.Name + " (" + CurrebtBasic.Measure + ")";

dataRow["StartDate"] = "nun";

dataRow["EndDate"] = "nun";

dataRow["Abb"] = CurrebtBasic.AbbreviationEN;

dataRow["CanWatchWhoOws"] = false;

dataRow["CanConfirm"] = true;

dataRow["ShowLable"] = false;

dataRow["LableText"] = "";

dataRow["LableColor"] = "#000000";

dataRow["Comment"] = "nun";

dataRow["CommentEnabled"] = "hidden";

dataRow["Value"] =

ForRCalc.GetCalculatedWithParams(mainStruct, ParamType, CurrebtBasic.BasicParametersTableID,

ReportID, SpecID).ToString();

dataTable.Rows.Add(dataRow);

}

#endregion

}

#endregion

#region DataGridBind

Grid.DataSource = dataTable;

Grid.Columns[3].HeaderText = name\_text;

Grid.Columns[6].HeaderText = value\_text;

Grid.Columns[8].HeaderText = progress\_text;

Grid.Columns[9].HeaderText = confirm\_text;

Grid.Columns[11].HeaderText = detalize\_text;

Grid.DataBind();

#endregion

#region постнастройки страницы

Grid.Columns[5].Visible = false;

Grid.Columns[4].Visible = false;

Grid.Columns[1].Visible = false;

if (ParamType == 0)

{

Grid.Columns[2].Visible = false;

Grid.Columns[10].Visible = false; //

Grid.Columns[12].Visible = false; //

Button7.Visible = true;

}

if (ParamType == 1)

{

Button7.Visible = true;

if (ParamID == 0)

{

Grid.Columns[2].Visible = false;

}

Grid.Columns[10].Visible = false; //

Grid.Columns[12].Visible = false; //

Grid.Columns[7].Visible = false;

}

if (ParamType == 2) // дальше углубляться нельзя

{

Grid.Columns[7].Visible = false;

Grid.Columns[9].Visible = false;

Grid.Columns[8].Visible = false;

Grid.Columns[11].Visible = false;

}

#endregion

}

else if (ViewType == 2) // просмотр по специальностям

{

#region преднастройка страницы

if (ParamType == 0)

{

RectorSession tmpses = new RectorSession(mainStruct, ViewType, ParamID, ParamType, ReportID,

SpecID, "Целевой показатель (ЦП) для направления подготовки");

rectorHistory.RectorSession[rectorHistory.CurrentSession] = tmpses;

Session["rectorHistory"] = rectorHistory;

}

else if (ParamType == 1)

{

RectorSession tmpses = new RectorSession(mainStruct, ViewType, ParamID, ParamType, ReportID,

SpecID, "Первичные данные (ПД) для направления подготовки");

rectorHistory.RectorSession[rectorHistory.CurrentSession] = tmpses;

Session["rectorHistory"] = rectorHistory;

}

else if (ParamType == 2)

{

RectorSession tmpses = new RectorSession(mainStruct, ViewType, ParamID, ParamType, ReportID,

SpecID, "Базовый показатель (БП) для направления подготовки");

rectorHistory.RectorSession[rectorHistory.CurrentSession] = tmpses;

Session["rectorHistory"] = rectorHistory;

string tmp = (from a in kpiWebDataContext.BasicParametersTable

where a.BasicParametersTableID == ParamID

select a.Name).FirstOrDefault();

PageFullName.Text = "Значения базового показателя (БП) <b> \"" + tmp + "\" </b> по направлениям подготовки для КФУ";

}

string title = "Направления подготовки";

#endregion

#region fill grid

List<SpecializationTable> SpecTable = (from a in kpiWebDataContext.SpecializationTable

join b in kpiWebDataContext.FourthLevelSubdivisionTable

on a.SpecializationTableID equals b.FK\_Specialization

where a.Active == true

&& b.Active == true

select a).OrderBy(mc => mc.SpecializationTableID).ToList();

//взяли все специальности которые привязаны к кафедрам

int old = 0;

foreach (SpecializationTable currentSpec in SpecTable)

{

if (currentSpec.SpecializationTableID != old)

{

DataRow dataRow = dataTable.NewRow();

dataRow["ID"] = currentSpec.SpecializationTableID; //GetLastID(currentStruct).ToString();

dataRow["Number"] = "num";

dataRow["Name"] = currentSpec.Name + ": " + (from a in kpiWebDataContext.SpecializationTable where a.SpecializationTableID == currentSpec.SpecializationTableID select a.SpecializationNumber).FirstOrDefault().ToString() +" "+ Action.EncodeToStr((from a in kpiWebDataContext.SpecializationTable where a.SpecializationTableID == currentSpec.SpecializationTableID select a.SpecializationNumber).FirstOrDefault().ToString()); //currentStruct.Name; // Шифр добавить!!

dataRow["StartDate"] = "nun";

dataRow["EndDate"] = "nun";

dataRow["CanWatchWhoOws"] = false;

dataRow["CanConfirm"] = true;

dataRow["ShowLable"] = false;

dataRow["LableText"] = "";

dataRow["LableColor"] = "#000000";

dataRow["Comment"] = "nun";

dataRow["CommentEnabled"] = "hidden";

dataRow["Value"] =

ForRCalc.GetCalculatedWithParams(mainStruct, ParamType, ParamID, ReportID,

currentSpec.SpecializationTableID).ToString();

dataTable.Rows.Add(dataRow);

}

else

{

}

old = currentSpec.SpecializationTableID;

}

#endregion

#region DataGridBind

Grid.DataSource = dataTable;

Grid.Columns[3].HeaderText = title;

Grid.DataBind();

#endregion

#region постнастройка страницы

Grid.Columns[12].Visible = false;

Grid.Columns[11].Visible = false;

Grid.Columns[7].Visible = false;

Grid.Columns[9].Visible = false;

Grid.Columns[8].Visible = false;

Grid.Columns[5].Visible = false;

Grid.Columns[4].Visible = false;

Grid.Columns[2].Visible = false;

Grid.Columns[1].Visible = false;

#endregion

}

else if (ViewType == 3) // Должники

{

#region

PageFullName.Text = "";

PageFullName.Text += "<b>";

PageFullName.Text += (from a in kpiWebDataContext.CalculatedParametrs

where a.CalculatedParametrsID == ParamID

select a.Name).FirstOrDefault();

PageFullName.Text += "</b> </br>";

int Deep = ForRCalc.StructDeepness(mainStruct);

if (Deep == 1)

{

}

if (Deep == 2)

{

PageFullName.Text += (from a in kpiWebDataContext.FirstLevelSubdivisionTable

where a.FirstLevelSubdivisionTableID == mainStruct.Lv\_1

select a.Name).FirstOrDefault();

PageFullName.Text += "</br>";

}

if (Deep == 3)

{

PageFullName.Text += (from a in kpiWebDataContext.FirstLevelSubdivisionTable

where a.FirstLevelSubdivisionTableID == mainStruct.Lv\_1

select a.Name).FirstOrDefault();

PageFullName.Text += "</br>";

PageFullName.Text += (from a in kpiWebDataContext.SecondLevelSubdivisionTable

where a.SecondLevelSubdivisionTableID == mainStruct.Lv\_2

select a.Name).FirstOrDefault();

PageFullName.Text += "</br>";

}

if (Deep == 4)

{

PageFullName.Text += (from a in kpiWebDataContext.FirstLevelSubdivisionTable

where a.FirstLevelSubdivisionTableID == mainStruct.Lv\_1

select a.Name).FirstOrDefault();

PageFullName.Text += "</br>";

PageFullName.Text += (from a in kpiWebDataContext.SecondLevelSubdivisionTable

where a.SecondLevelSubdivisionTableID == mainStruct.Lv\_2

select a.Name).FirstOrDefault();

PageFullName.Text += "</br>";

PageFullName.Text += (from a in kpiWebDataContext.ThirdLevelSubdivisionTable

where a.ThirdLevelSubdivisionTableID == mainStruct.Lv\_3

select a.Name).FirstOrDefault();

PageFullName.Text += "</br>";

}

#endregion

#region fill grid

CalculatedParametrs Calculated = (from a in kpiWebDataContext.CalculatedParametrs

where a.CalculatedParametrsID == ParamID

select a).FirstOrDefault();

List<BasicParametersTable> BasicList = Abbreviature.GetBasicList(Calculated.Formula);

List<ForRCalc.Struct> currentStructList = new List<ForRCalc.Struct>();

currentStructList = ForRCalc.GetChildStructList(mainStruct, ReportID);

foreach (ForRCalc.Struct currentStruct in currentStructList)

{

DataRow dataRow = dataTable.NewRow();

dataRow["ID"] = ForRCalc.GetLastID(currentStruct).ToString();

dataRow["Number"] = "num";

dataRow["Name"] = currentStruct.Name;

dataRow["StartDate"] = "nun";

dataRow["EndDate"] = "nun";

dataRow["CanConfirm"] = false;

dataRow["ShowLable"] = false;

dataRow["CanWatchWhoOws"] = false;

dataRow["LableText"] = "";

dataRow["LableColor"] = "#000000";

dataRow["Value"] = "nun";

dataRow["Comment"] = "nun";

dataRow["CommentEnabled"] = "hidden";

#region check if all users confirmed basics

int AllBasicsUsersCanEdit = 0;

int AllConfirmedBasics = 0;

foreach (BasicParametersTable Basic in BasicList)

{

/\*

List<UsersTable> UserTableList = (from a in kpiWebDataContext.UsersTable

join b in kpiWebDataContext.BasicParametrsAndUsersMapping

on a.UsersTableID equals b.FK\_UsersTable

join c in kpiWebDataContext.BasicParametrAdditional

on b.FK\_ParametrsTable equals c.BasicParametrAdditionalID

where a.Active == true

&& c.Calculated == false

&& b.Active == true

&& b.CanEdit == true

&& b.FK\_ParametrsTable == Basic.BasicParametersTableID

&& ((a.FK\_ZeroLevelSubdivisionTable == currentStruct.Lv\_0) || (currentStruct.Lv\_0 == 0))

&& ((a.FK\_FirstLevelSubdivisionTable == currentStruct.Lv\_1) || (currentStruct.Lv\_1 == 0))

&& ((a.FK\_SecondLevelSubdivisionTable == currentStruct.Lv\_2) || (currentStruct.Lv\_2 == 0))

&& ((a.FK\_ThirdLevelSubdivisionTable == currentStruct.Lv\_3) || (currentStruct.Lv\_3 == 0))

// && ((a.FK\_FourthLevelSubdivisionTable == currentStruct.Lv\_4) || (currentStruct.Lv\_4 == 0))

select a).ToList();

int SpecCnt = 0;

BasicParametrAdditional basicAdditional =

(from a in kpiWebDataContext.BasicParametrAdditional

where a.BasicParametrAdditionalID == Basic.BasicParametersTableID

select a).FirstOrDefault();

if (basicAdditional.SubvisionLevel != 4)

{

AllBasicsUsersCanEdit += UserTableList.Count();

}

else

{

foreach (UsersTable CurUSer in UserTableList)

{

SpecCnt += (from a in kpiWebDataContext.FourthLevelSubdivisionTable

where a.FK\_ThirdLevelSubdivisionTable == CurUSer.FK\_ThirdLevelSubdivisionTable

&& a.Active == true

select a).Count();

}

AllBasicsUsersCanEdit += (UserTableList.Count() \* SpecCnt);

}

\*/

AllBasicsUsersCanEdit += (from a in kpiWebDataContext.CollectedBasicParametersTable

where a.FK\_ReportArchiveTable == ReportID

&& a.FK\_BasicParametersTable == Basic.BasicParametersTableID

&& ((a.FK\_ZeroLevelSubdivisionTable == currentStruct.Lv\_0) || (currentStruct.Lv\_0 == 0))

&& ((a.FK\_FirstLevelSubdivisionTable == currentStruct.Lv\_1) || (currentStruct.Lv\_1 == 0))

&& ((a.FK\_SecondLevelSubdivisionTable == currentStruct.Lv\_2) || (currentStruct.Lv\_2 == 0))

&& ((a.FK\_ThirdLevelSubdivisionTable == currentStruct.Lv\_3) || (currentStruct.Lv\_3 == 0))

&& ((a.FK\_FourthLevelSubdivisionTable == currentStruct.Lv\_4) || (currentStruct.Lv\_4 == 0))

select a).Count();

AllConfirmedBasics += (from a in kpiWebDataContext.CollectedBasicParametersTable

where a.FK\_ReportArchiveTable == ReportID

&& a.FK\_BasicParametersTable == Basic.BasicParametersTableID

&& a.Status == 4

&& ((a.FK\_ZeroLevelSubdivisionTable == currentStruct.Lv\_0) || (currentStruct.Lv\_0 == 0))

&& ((a.FK\_FirstLevelSubdivisionTable == currentStruct.Lv\_1) || (currentStruct.Lv\_1 == 0))

&& ((a.FK\_SecondLevelSubdivisionTable == currentStruct.Lv\_2) || (currentStruct.Lv\_2 == 0))

&& ((a.FK\_ThirdLevelSubdivisionTable == currentStruct.Lv\_3) || (currentStruct.Lv\_3 == 0))

&& ((a.FK\_FourthLevelSubdivisionTable == currentStruct.Lv\_4) || (currentStruct.Lv\_4 == 0))

select a).Count();

}

bool BasicsAreConfirmed = true;

if (AllBasicsUsersCanEdit != AllConfirmedBasics)

{

BasicsAreConfirmed = false;

}

#endregion

if (AllBasicsUsersCanEdit == 0)

{

dataRow["Progress"] = "";

}

else

{

dataRow["Progress"] =

((((float) AllConfirmedBasics)\*100)/((float) AllBasicsUsersCanEdit)).ToString("0.0") +"%";

if (float.IsNaN((((float)AllConfirmedBasics) \* 100) / ((float)AllBasicsUsersCanEdit)))

{

dataRow["Progress"] = "100,0%";

}

dataTable.Rows.Add(dataRow);

}

// dataTable.Rows.Add(dataRow);

}

#endregion

#region DataGridBind

Grid.DataSource = dataTable;

Grid.Columns[3].HeaderText = "Подразделения";

if (ForRCalc.StructDeepness(mainStruct) > 3)

{

Grid.Columns[3].HeaderText = "Направления подготовки";

}

Grid.DataBind();

#endregion

#region постнастройка страницы

Grid.Columns[12].Visible = false;

Grid.Columns[11].Visible = false;

Grid.Columns[9].Visible = false;

// Grid.Columns[8].Visible = false;

Grid.Columns[7].Visible = false;

Grid.Columns[6].Visible = false;

Grid.Columns[5].Visible = false;

Grid.Columns[4].Visible = false;

Grid.Columns[2].Visible = false;

Grid.Columns[1].Visible = false;

if (ForRCalc.StructDeepness(mainStruct) > 2)

{

Grid.Columns[10].Visible = false;

}

#endregion

}

else

{

//error // wrong ViewType

}

RefreshHistory();

if ((ViewType == 1) && ((ParamType == 0) || (ParamType == 1)))

{

ClientScript.RegisterStartupScript(this.GetType(), "ShowLegend", "ShowLegend()", true);

}

}

}

protected void ButtonConfirmClick(object sender, EventArgs e)

{

/\*

KPIWebDataContext kpiWebDataContext = new KPIWebDataContext();

RectorHistorySession rectorHistory = (RectorHistorySession)Session["rectorHistory"];

if (rectorHistory == null)

{

Response.Redirect("~/Default.aspx");

}

RectorSession CurrentRectorSession = rectorHistory.RectorSession[rectorHistory.CurrentSession];

int ParamType = CurrentRectorSession.sesParamType;

if (ParamType == 0) // indicator

{

CollectedIndocators Indicator = (from a in kpiWebDataContext.CollectedIndocators

where a.FK\_Indicators == Convert.ToInt32(button.CommandArgument)

select a).FirstOrDefault();

Indicator.Confirmed = true;

kpiWebDataContext.SubmitChanges();

Response.Redirect("~/Rector/Result.aspx");

}

else if (ParamType == 1) // calculated

{

CollectedCalculatedParametrs Calculated = (from a in kpiWebDataContext.CollectedCalculatedParametrs

where a.FK\_CalculatedParametrs == Convert.ToInt32(button.CommandArgument)

select a).FirstOrDefault();

Calculated.Confirmed = true;

kpiWebDataContext.SubmitChanges();

Response.Redirect("~/Rector/Result.aspx");

}

}\*/

}

protected void Button1Click(object sender, EventArgs e) //по структуре

{

Button button = (Button)sender;

{

RectorHistorySession rectorHistory = (RectorHistorySession)Session["rectorHistory"];

if (rectorHistory == null)

{

Response.Redirect("~/Default.aspx");

}

RectorSession CurrentRectorSession = rectorHistory.RectorSession[rectorHistory.CurrentSession];

ForRCalc.Struct mainStruct = CurrentRectorSession.sesStruct;

int ViewType = CurrentRectorSession.sesViewType;

int ParamID = CurrentRectorSession.sesParamID;

int ParamType = CurrentRectorSession.sesParamType;

int ReportID = CurrentRectorSession.sesReportID;

int SpecID = CurrentRectorSession.sesSpecID;

RectorSession currentRectorSession = new RectorSession(mainStruct, ViewType, ParamID, ParamType, ReportID, SpecID,"");

if (currentRectorSession.sesViewType == 1)

{//впервые перешли на разложение по структуре сразу после показателя

currentRectorSession.sesParamID = Convert.ToInt32(button.CommandArgument);

currentRectorSession.sesViewType = 0;

currentRectorSession.sesStruct.Lv\_0 = 1;

currentRectorSession.sesStruct.Lv\_1 = 0;

currentRectorSession.sesStruct.Lv\_2 = 0;

currentRectorSession.sesStruct.Lv\_3 = 0;

currentRectorSession.sesStruct.Lv\_4 = 0;

currentRectorSession.sesStruct.Lv\_5 = 0;

}

else if (currentRectorSession.sesViewType == 2)

{//впервые перешли на разложение по структуре после выбора специальности

currentRectorSession.sesSpecID = Convert.ToInt32(button.CommandArgument);

currentRectorSession.sesViewType = 0;

currentRectorSession.sesStruct.Lv\_0 = 1;

currentRectorSession.sesStruct.Lv\_1 = 0;

currentRectorSession.sesStruct.Lv\_2 = 0;

currentRectorSession.sesStruct.Lv\_3 = 0;

currentRectorSession.sesStruct.Lv\_4 = 0;

currentRectorSession.sesStruct.Lv\_5 = 0;

}

else if (currentRectorSession.sesViewType == 3)

{

currentRectorSession.sesStruct = ForRCalc.StructDeeper(currentRectorSession.sesStruct, Convert.ToInt32(button.CommandArgument));

}

else

{

currentRectorSession.sesStruct = ForRCalc.StructDeeper(currentRectorSession.sesStruct, Convert.ToInt32(button.CommandArgument));

}

rectorHistory.CurrentSession++;

rectorHistory.SessionCount = rectorHistory.CurrentSession + 1;

rectorHistory.RectorSession[rectorHistory.CurrentSession] = currentRectorSession;

Session["rectorHistory"] = rectorHistory;

Response.Redirect("~/Rector/Result.aspx");

}

}

protected void Button2Click(object sender, EventArgs e) // по составляющим

{

Button button = (Button)sender;

{

RectorHistorySession rectorHistory = (RectorHistorySession)Session["rectorHistory"];

if (rectorHistory == null)

{

Response.Redirect("~/Default.aspx");

}

RectorSession CurrentRectorSession = rectorHistory.RectorSession[rectorHistory.CurrentSession];

ForRCalc.Struct mainStruct = CurrentRectorSession.sesStruct;

int ViewType = CurrentRectorSession.sesViewType;

int ParamID = CurrentRectorSession.sesParamID;

int ParamType = CurrentRectorSession.sesParamType;

int ReportID = CurrentRectorSession.sesReportID;

int SpecID = CurrentRectorSession.sesSpecID;

RectorSession currentRectorSession = new RectorSession(mainStruct, ViewType, ParamID, ParamType, ReportID, SpecID,"");

//RectorSession currentRectorSession = rectorHistory.RectorSession[rectorHistory.CurrentSession];

currentRectorSession.sesParamID = Convert.ToInt32(button.CommandArgument);

currentRectorSession.sesParamType++;

rectorHistory.CurrentSession++;

rectorHistory.SessionCount = rectorHistory.CurrentSession + 1;

rectorHistory.RectorSession[rectorHistory.CurrentSession] = currentRectorSession;

Session["rectorHistory"] = rectorHistory;

Response.Redirect("~/Rector/Result.aspx");

}

}

protected void Button3Click(object sender, EventArgs e) //по специальности

{

Button button = (Button)sender;

{

RectorHistorySession rectorHistory = (RectorHistorySession) Session["rectorHistory"];

if (rectorHistory == null)

{

Response.Redirect("~/Default.aspx");

}

RectorSession CurrentRectorSession = rectorHistory.RectorSession[rectorHistory.CurrentSession];

ForRCalc.Struct mainStruct = CurrentRectorSession.sesStruct;

int ViewType = CurrentRectorSession.sesViewType;

int ParamID = CurrentRectorSession.sesParamID;

int ParamType = CurrentRectorSession.sesParamType;

int ReportID = CurrentRectorSession.sesReportID;

int SpecID = CurrentRectorSession.sesSpecID;

RectorSession currentRectorSession = new RectorSession(mainStruct, ViewType, ParamID, ParamType, ReportID, SpecID,"");

currentRectorSession.sesParamID = Convert.ToInt32(button.CommandArgument);

currentRectorSession.sesViewType = 2;

rectorHistory.CurrentSession++;

rectorHistory.SessionCount = rectorHistory.CurrentSession + 1;

rectorHistory.RectorSession[rectorHistory.CurrentSession] = currentRectorSession;

Session["rectorHistory"] = rectorHistory;

Response.Redirect("~/Rector/Result.aspx");

}

}

protected void GoBackButton\_Click(object sender, EventArgs e)

{

RectorHistorySession rectorHistory = (RectorHistorySession)Session["rectorHistory"];

if (rectorHistory == null)

{

Response.Redirect("~/Default.aspx");

}

if (rectorHistory.CurrentSession == 0)

{

Response.Redirect("~/Rector/RectorChooseReport.aspx");

}

rectorHistory.CurrentSession--;

Session["rectorHistory"] = rectorHistory;

Response.Redirect("~/Rector/Result.aspx");

}

protected void GoForwardButton\_Click(object sender, EventArgs e)

{

RectorHistorySession rectorHistory = (RectorHistorySession)Session["rectorHistory"];

if (rectorHistory == null)

{

Response.Redirect("~/Default.aspx");

}

if (rectorHistory.CurrentSession < rectorHistory.SessionCount) // есть куда переходить

{

rectorHistory.CurrentSession++;

Session["rectorHistory"] = rectorHistory;

Response.Redirect("~/Rector/Result.aspx");

}

}

protected void Button4\_Click(object sender, EventArgs e)

{

Response.Redirect("~/Rector/RectorMain.aspx");

}

protected void Button5\_Click(object sender, EventArgs e)

{

Response.Redirect("~/Rector/ViewDocument.aspx");

}

private void BindTree(IEnumerable<MyObject> list, TreeNode parentNode)

{

var nodes = list.Where(x => parentNode == null ? x.ParentId == 0 : x.ParentId == int.Parse(parentNode.Value));

foreach (var node in nodes)

{

TreeNode newNode = new TreeNode(node.Name, node.Id.ToString());

if (node.Active == 1)

{

newNode.NavigateUrl = node.UrlAddr;

}

else

{

newNode.SelectAction = TreeNodeSelectAction.None;

}

if (parentNode == null)

{

TreeView1.Nodes.Add(newNode);

}

else

{

parentNode.ChildNodes.Add(newNode);

}

BindTree(list, newNode);

}

}

public void RefreshHistory()

{

/\*

#region history

RectorHistorySession rectorHistory\_ = (RectorHistorySession)Session["rectorHistory"];

if (rectorHistory\_ == null)

{

Response.Redirect("~/Default.aspx");

}

if (rectorHistory\_.Visible == true)

{

Button6.Text = "Скрыть историю";

TreeView1.Visible = true;

List<MyObject> list = new List<MyObject>();

for (int i = 0; i < rectorHistory\_.SessionCount; i++)

{

RectorSession curSesion = rectorHistory\_.RectorSession[i];

int tmp = rectorHistory\_.CurrentSession == i ? 0:1;

list.Add(new MyObject() { Id = i+1, Name = curSesion.sesName, ParentId = i, UrlAddr = "Result?&HLevel="+i,Active=tmp });

}

BindTree(list, null);

TreeView1.ExpandAll();

}

else

{

Button6.Text = "Показать историю";

TreeView1.Visible = false;

TreeView1.Nodes.Clear();

TreeView1.DataBind();

}

#endregion

\*/

}

protected void Button6\_Click(object sender, EventArgs e)

{

RectorHistorySession rectorHistory\_ = (RectorHistorySession)Session["rectorHistory"];

if (rectorHistory\_ == null)

{

Response.Redirect("~/Default.aspx");

}

if (rectorHistory\_.Visible == true)

{

rectorHistory\_.Visible = false;

}

else

{

rectorHistory\_.Visible = true;

}

Session["rectorHistory"] = rectorHistory\_;

RefreshHistory();

}

protected void Grid\_SelectedIndexChanged(object sender, EventArgs e)

{

}

protected void Button7\_Click(object sender, EventArgs e)

{

ShowUnConfirmed unConfirmed =new ShowUnConfirmed(true);

Session["unConfirmed"] = unConfirmed;

Response.Redirect("~/Rector/Result.aspx");

}

protected void Grid\_RowDataBound(object sender, GridViewRowEventArgs e)

{

#region

var ColorLable = e.Row.FindControl("Color") as Label;

var PageConfirmButton = e.Row.FindControl("ConfirmButton") as Button;

var PageButton2 = e.Row.FindControl("Button2") as Button;

//// костыль 0%

var Button1\_ = e.Row.FindControl("Button1") as Button;

var PLable\_ = e.Row.FindControl("ProgressLable") as Label;

if ((Button1\_ != null) && (PLable\_ != null))

{

if (PLable\_.Text == "0,0%")

{

Button1\_.Enabled = false;

}

}

//end костыль 0%

if (ColorLable != null)

{

RectorHistorySession rectorHistory = (RectorHistorySession)Session["rectorHistory"];

if (rectorHistory == null)

{

Response.Redirect("~/Default.aspx");

}

RectorSession CurrentRectorSession = rectorHistory.RectorSession[rectorHistory.CurrentSession];

PageConfirmButton.Enabled = false;

PageButton2.Enabled = false;

int ColorNumber = -1;

if (int.TryParse(ColorLable.Text, out ColorNumber) && ColorNumber > -1)

{

switch (ColorNumber)

{

case 0:

{

break;

}

case 1: // утверждено

{

e.Row.Style.Add("background-color", "rgba(0, 255, 0, 0.3)");

PageButton2.Enabled = true;

break;

}

case 2: // можно утвердить

{

e.Row.Style.Add("background-color", "rgba(255, 0, 0, 0.3)");

PageButton2.Enabled = true;

break;

}

case 3: // рассчитано на неутвержденных данных

{

e.Row.Style.Add("background-color", "rgba(255, 255, 0, 0.3)");

PageConfirmButton.Enabled = true;

PageButton2.Enabled = true;

break;

}

default:

{

break;

}

}

}

if ((CurrentRectorSession.sesViewType == 1) || (CurrentRectorSession.sesParamType == 0))

{

PageButton2.Enabled = true;

}

}

#endregion

var ConfirmButton = e.Row.FindControl("ConfirmButton") as Button;

if (ConfirmButton != null)

{

ConfirmButton.OnClientClick = "javascript:return showCommentSection(" + ConfirmButton.CommandArgument+ ");";

}

}

protected void Button8\_Click(object sender, EventArgs e)

{

}

public void DoConfirm(int ParamId)

{

KPIWebDataContext kpiWebDataContext = new KPIWebDataContext();

RectorHistorySession rectorHistory = (RectorHistorySession) Session["rectorHistory"];

if (rectorHistory == null)

{

Response.Redirect("~/Default.aspx");

}

RectorSession CurrentRectorSession = rectorHistory.RectorSession[rectorHistory.CurrentSession];

int ReportID = CurrentRectorSession.sesReportID;

Serialization UserSer = (Serialization)Session["UserID"];

if (UserSer == null)

{

Response.Redirect("~/Default.aspx");

}

int userID = UserSer.Id;

int ParamType = CurrentRectorSession.sesParamType;

if (ParamType == 0) // indicator

{

CollectedIndocators Indicator = (from a in kpiWebDataContext.CollectedIndocators

where a.FK\_Indicators == ParamId

&& a.FK\_ReportArchiveTable == ReportID

select a).FirstOrDefault();

Indicator.Confirmed = true;

kpiWebDataContext.SubmitChanges();

#region save params in DB with comment

ConfirmationHistory ConfirmParam = new ConfirmationHistory();

ConfirmParam.Date = DateTime.Now;

ConfirmParam.FK\_IndicatorsTable = ParamId;

ConfirmParam.FK\_ReportTable = CurrentRectorSession.sesReportID;

ConfirmParam.FK\_UsersTable = userID;

ConfirmParam.Name = "Подтверждение ЦП проректором";

ConfirmParam.Comment = TextBox1.Text;

kpiWebDataContext.ConfirmationHistory.InsertOnSubmit(ConfirmParam);

kpiWebDataContext.SubmitChanges();

#endregion

Response.Redirect("~/Rector/Result.aspx");

}

else if (ParamType == 1) // calculated

{

CollectedCalculatedParametrs Calculated = (from a in kpiWebDataContext.CollectedCalculatedParametrs

where a.FK\_CalculatedParametrs == ParamId

&& a.FK\_ReportArchiveTable == ReportID

select a).FirstOrDefault();

Calculated.Confirmed = true;

kpiWebDataContext.SubmitChanges();

#region save params in DB with comment

ConfirmationHistory ConfirmParam = new ConfirmationHistory();

ConfirmParam.Date = DateTime.Now;

ConfirmParam.FK\_CalculatedParamTable = ParamId;

ConfirmParam.FK\_ReportTable = CurrentRectorSession.sesReportID;

ConfirmParam.FK\_UsersTable = userID;

ConfirmParam.Name = "Подтверждение ПД проректором";

ConfirmParam.Comment = TextBox1.Text;

kpiWebDataContext.ConfirmationHistory.InsertOnSubmit(ConfirmParam);

kpiWebDataContext.SubmitChanges();

#endregion

Response.Redirect("~/Rector/Result.aspx");

}

}

protected void ButtonOweClick(object sender, EventArgs e)

{

Button button = (Button)sender;

{

RectorHistorySession rectorHistory = (RectorHistorySession)Session["rectorHistory"];

if (rectorHistory == null)

{

Response.Redirect("~/Default.aspx");

}

RectorSession CurrentRectorSession = rectorHistory.RectorSession[rectorHistory.CurrentSession];

ForRCalc.Struct mainStruct = CurrentRectorSession.sesStruct;

int ViewType = CurrentRectorSession.sesViewType;

int ParamID = CurrentRectorSession.sesParamID;

int ParamType = CurrentRectorSession.sesParamType;

int ReportID = CurrentRectorSession.sesReportID;

int SpecID = CurrentRectorSession.sesSpecID;

RectorSession currentRectorSession = new RectorSession(mainStruct, ViewType, ParamID, ParamType, ReportID, SpecID, "");

currentRectorSession.sesParamID = Convert.ToInt32(button.CommandArgument);

currentRectorSession.sesViewType = 3;

currentRectorSession.sesStruct.Lv\_0 = 1;

currentRectorSession.sesStruct.Lv\_1 = 0;

currentRectorSession.sesStruct.Lv\_2 = 0;

currentRectorSession.sesStruct.Lv\_3 = 0;

currentRectorSession.sesStruct.Lv\_4 = 0;

currentRectorSession.sesStruct.Lv\_5 = 0;

rectorHistory.CurrentSession++;

rectorHistory.SessionCount = rectorHistory.CurrentSession + 1;

rectorHistory.RectorSession[rectorHistory.CurrentSession] = currentRectorSession;

Session["rectorHistory"] = rectorHistory;

Response.Redirect("~/Rector/Result.aspx");

}

}

protected void Button8\_Click1(object sender, EventArgs e)

{

Response.Redirect("~/Rector/RectorChooseReport.aspx");

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Filename: Result.aspx.designer.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//------------------------------------------------------------------------------

// <auto-generated>

// This code was generated by a tool.

//

// Changes to this file may cause incorrect behavior and will be lost if

// the code is regenerated.

// </auto-generated>

//------------------------------------------------------------------------------

namespace KPIWeb.Rector {

public partial class Result {

/// <summary>

/// Label7 control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.Label Label7;

/// <summary>

/// Panel5 control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.Panel Panel5;

/// <summary>

/// Label11 control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.Label Label11;

/// <summary>

/// Panel6 control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.Panel Panel6;

/// <summary>

/// Label12 control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.Label Label12;

/// <summary>

/// Panel7 control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.Panel Panel7;

/// <summary>

/// Label13 control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.Label Label13;

/// <summary>

/// TextBox1 control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.TextBox TextBox1;

/// <summary>

/// top\_panel2 control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.Panel top\_panel2;

/// <summary>

/// GoBackButton control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.Button GoBackButton;

/// <summary>

/// GoForwardButton control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.Button GoForwardButton;

/// <summary>

/// Button4 control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.Button Button4;

/// <summary>

/// Button8 control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.Button Button8;

/// <summary>

/// Button5 control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.Button Button5;

/// <summary>

/// ReportTitle control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.Label ReportTitle;

/// <summary>

/// PageFullName control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.Label PageFullName;

/// <summary>

/// TreeView1 control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.TreeView TreeView1;

/// <summary>

/// Button7 control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.Button Button7;

/// <summary>

/// Grid control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.GridView Grid;

/// <summary>

/// FormulaLable control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.Label FormulaLable;

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Filename: RGetCalculated.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

namespace KPIWeb.Rector

{

public class RGetCalculated

{

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Filename: RMain.aspx.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

namespace KPIWeb.Rector

{

public partial class RMain : System.Web.UI.Page

{

protected void Page\_Load(object sender, EventArgs e)

{

Serialization UserSer = (Serialization)Session["UserID"];

if (UserSer == null)

{

Response.Redirect("~/Default.aspx");

}

int userID = UserSer.Id;

KPIWebDataContext kPiDataContext = new KPIWebDataContext();

UsersTable userTable =

(from a in kPiDataContext.UsersTable where a.UsersTableID == userID select a).FirstOrDefault();

if (userTable.AccessLevel != 7)

{

Response.Redirect("~/Default.aspx");

}

}

protected void Button1\_Click(object sender, EventArgs e)

{

Response.Redirect("~/Rector/RAnalitics.aspx");

}

protected void Button2\_Click(object sender, EventArgs e)

{

ForRCalc.Struct mainStruct = new ForRCalc.Struct(1, "");

RectorSession rectorResultSession = new RectorSession(mainStruct, 1, 0, 0,1, 0);// какой отчет ? ER

RectorHistorySession RectorHistory = new RectorHistorySession();

RectorHistory.SessionCount = 1;

RectorHistory.CurrentSession = 0;

RectorHistory.Visible = false;

RectorHistory.RectorSession[RectorHistory.CurrentSession] = rectorResultSession;

Session["rectorHistory"] = RectorHistory;

Response.Redirect("~/Rector/RRating.aspx");

}

protected void Button3\_Click(object sender, EventArgs e)

{

/\*KPIWebDataContext kpiWebDataContext = new KPIWebDataContext();

IndicatorsTable Indicator = (from a in kpiWebDataContext.IndicatorsTable

where a.Active == true

select a).FirstOrDefault();

ChartValueWithAllPlanned NewChartValue = ForRCalc.GetAllPlannedForIndicator(Indicator.IndicatorsTableID);\*/

Response.Redirect("~/Rector/RPlannedDynamics.aspx");

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Filename: RMain.aspx.designer.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//------------------------------------------------------------------------------

// <автоматически создаваемое>

// Этот код создан программой.

//

// Изменения в этом файле могут привести к неправильной работе и будут потеряны в случае

// повторной генерации кода.

// </автоматически создаваемое>

//------------------------------------------------------------------------------

namespace KPIWeb.Rector {

public partial class RMain {

/// <summary>

/// Button1 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Button Button1;

/// <summary>

/// Button2 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Button Button2;

/// <summary>

/// Button3 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Button Button3;

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Filename: RPlannedDynamics.aspx.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using System;

using System.Collections.Generic;

using System.Drawing;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.DataVisualization.Charting;

using System.Web.UI.WebControls;

namespace KPIWeb.Rector

{

public partial class RPlannedDynamics : System.Web.UI.Page

{

protected void Page\_Load(object sender, EventArgs e)

{

Serialization UserSer = (Serialization)Session["UserID"];

if (UserSer == null)

{

Response.Redirect("~/Default.aspx");

}

int userID = UserSer.Id;

KPIWebDataContext kPiDataContext = new KPIWebDataContext();

UsersTable userTable =

(from a in kPiDataContext.UsersTable where a.UsersTableID == userID select a).FirstOrDefault();

if (userTable.AccessLevel != 7)

{

Response.Redirect("~/Default.aspx");

}

if (!IsPostBack)

{

/////

KPIWebDataContext kpiWebDataContext = new KPIWebDataContext();

List<IndicatorsTable> Indicators = (from a in kpiWebDataContext.IndicatorsTable

where a.Active == true

select a).ToList();

//!=null

/////////////

var dictionary = new Dictionary<int, string>();

//dictionary.Add(0, "Выберите целевой показатель");

foreach (var item in Indicators)

dictionary.Add(item.IndicatorsTableID, item.Name);

DropDownList1.DataTextField = "Value";

DropDownList1.DataValueField = "Key";

DropDownList1.DataSource = dictionary;

DropDownList1.DataBind();

////////////

ChartDraw(Indicators.FirstOrDefault().IndicatorsTableID);

}

}

private void ChartDraw(int IndicatorID)

{

ChartValueWithAllPlanned NewChartValue = ForRCalc.GetAllPlannedForIndicator(IndicatorID);

// Формат диаграммы

Chart1.BackColor = Color.Gray;

Chart1.BackSecondaryColor = Color.WhiteSmoke;

Chart1.BackGradientStyle = GradientStyle.DiagonalRight;

Chart1.BorderlineDashStyle = ChartDashStyle.Solid;

Chart1.BorderlineColor = Color.Gray;

Chart1.BorderSkin.SkinStyle = BorderSkinStyle.Emboss;

// Формат области диаграммы

Chart1.ChartAreas[0].BackColor = Color.Gainsboro;

// формат заголовока

KPIWebDataContext kpiWebDataContext = new KPIWebDataContext();

var namemeasure = (from a in kpiWebDataContext.IndicatorsTable

where a.Active == true && a.IndicatorsTableID == IndicatorID

select a).FirstOrDefault();

Chart1.Titles.Add(namemeasure.Name + " (" + namemeasure.Measure+")");

Chart1.Titles[0].Font = new Font("Utopia", 16);

ChartItems chartItems = new ChartItems();

Chart1.ChartAreas[0].AxisX.IntervalAutoMode = IntervalAutoMode.VariableCount;

Chart1.ChartAreas[0].AxisY.IntervalAutoMode = IntervalAutoMode.VariableCount;

foreach (var item in NewChartValue.PlannedAndRealValuesList)

{

chartItems.AddChartItem(item.Date.Year.ToString(), item.RealValue);

Chart1.Series[1].Points.AddY(item.PlannedValue);

}

Chart1.DataSource = chartItems.GetDataSource();

Chart1.Series[0].XValueMember = "Name";

Chart1.Series[0].YValueMembers = "Value";

}

protected void DropDownList1\_SelectedIndexChanged(object sender, EventArgs e)

{

ChartDraw(Convert.ToInt32(DropDownList1.Items[DropDownList1.SelectedIndex].Value));

}

protected void Button2\_Click(object sender, EventArgs e)

{

Response.Redirect("~/Rector/RectorMain.aspx");

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Filename: RPlannedDynamics.aspx.designer.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//------------------------------------------------------------------------------

// <автоматически создаваемое>

// Этот код создан программой.

//

// Изменения в этом файле могут привести к неправильной работе и будут потеряны в случае

// повторной генерации кода.

// </автоматически создаваемое>

//------------------------------------------------------------------------------

namespace KPIWeb.Rector {

public partial class RPlannedDynamics {

/// <summary>

/// DropDownList1 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.DropDownList DropDownList1;

/// <summary>

/// top\_panel2 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Panel top\_panel2;

/// <summary>

/// GoBackButton элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Button GoBackButton;

/// <summary>

/// GoForwardButton элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Button GoForwardButton;

/// <summary>

/// Button2 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Button Button2;

/// <summary>

/// Button1 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Button Button1;

/// <summary>

/// Chart1 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.DataVisualization.Charting.Chart Chart1;

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Filename: RRating.aspx.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

using System.Data;

namespace KPIWeb.Rector

{

public partial class RRating : System.Web.UI.Page

{

public string FloatToStrFormat(float value,float plannedValue,int DataType)

{

if (DataType == 1)

{

string tmpValue = Math.Round(value).ToString();// value.ToString("0");

return tmpValue;

}

else if(DataType == 2)

{

string tmpValue = value.ToString();

string tmpPlanned = plannedValue.ToString();

int PlannedNumbersAftepPoint = 2;

if (tmpPlanned.IndexOf(',') != -1)

{

PlannedNumbersAftepPoint = (tmpPlanned.Length - tmpPlanned.IndexOf(',')+1);

}

int ValuePointIndex = tmpValue.IndexOf(',');

if (ValuePointIndex != -1)

{

if ((tmpValue.Length - ValuePointIndex - PlannedNumbersAftepPoint) > 0)

{

tmpValue = tmpValue.Remove(ValuePointIndex + PlannedNumbersAftepPoint, tmpValue.Length - ValuePointIndex - PlannedNumbersAftepPoint);

}

}

return tmpValue;

}

return "0";

}

public ChartOneValue GetCalculatedIndicator(int ReportID, IndicatorsTable Indicator, FirstLevelSubdivisionTable Academy, SecondLevelSubdivisionTable Faculty) // academyID == null && facultyID==null значит для всего КФУ

{

return ForRCalc.GetCalculatedIndicator(ReportID, Indicator, Academy, Faculty);

/\* KPIWebDataContext kpiWebDataContext = new KPIWebDataContext();

float Planned\_Value = 0;

string Name\_ = "";

float Value\_ = 0;

#region plannedIndicator

PlannedIndicator plannedValue = (from a in kpiWebDataContext.PlannedIndicator

where a.FK\_IndicatorsTable == Indicator.IndicatorsTableID

&& a.Date > DateTime.Now

select a).OrderBy(x => x.Date).FirstOrDefault();

if (plannedValue != null)

{

Planned\_Value = (float)plannedValue.Value;

}

#endregion

#region Name

if ((Academy == null) && (Faculty == null))

{

if (Indicator.Measure != null)

{

if (Indicator.Measure.Length > 0)

{

Name\_ = Indicator.Name + " (" + Indicator.Measure + ")";

}

else

{

Name\_ = Indicator.Name;

}

}

else

{

Name\_ = Indicator.Name;

}

}

else if (Faculty != null)

{

Name\_ = Faculty.Name;

}

else if (Academy != null)

{

Name\_ = Academy.Name;

}

#endregion

#region

//ForRCalc.Struct mainStruct = mainStruct = new ForRCalc.Struct(1, 0, 0, 0, 0, "N");

CollectedIndicatorsForR collected = new CollectedIndicatorsForR();

if ((Academy == null) && (Faculty == null))

{

//mainStruct = new ForRCalc.Struct(1, 0, 0, 0, 0, "N");

collected = (from a in kpiWebDataContext.CollectedIndicatorsForR

where a.FK\_ReportArchiveTable == ReportID

&& a.FK\_IndicatorsTable == Indicator.IndicatorsTableID

&& a.FK\_FirstLevelSubdivisionTable == null

&& a.FK\_SecondLevelSubdivisionTable == null

select a).FirstOrDefault();

}

else if (Faculty != null)

{

//mainStruct = new ForRCalc.Struct(1, Faculty.FK\_FirstLevelSubdivisionTable, Faculty.SecondLevelSubdivisionTableID, 0, 0, "N");

collected = (from a in kpiWebDataContext.CollectedIndicatorsForR

where a.FK\_ReportArchiveTable == ReportID

&& a.FK\_IndicatorsTable == Indicator.IndicatorsTableID

&& a.FK\_FirstLevelSubdivisionTable == Faculty.FK\_FirstLevelSubdivisionTable

&& a.FK\_SecondLevelSubdivisionTable == Faculty.SecondLevelSubdivisionTableID

select a).FirstOrDefault();

}

else if (Academy != null)

{

//mainStruct = new ForRCalc.Struct(1, Academy.FirstLevelSubdivisionTableID, 0, 0, 0, "N");

collected = (from a in kpiWebDataContext.CollectedIndicatorsForR

where a.FK\_ReportArchiveTable == ReportID

&& a.FK\_IndicatorsTable == Indicator.IndicatorsTableID

&& a.FK\_FirstLevelSubdivisionTable == Academy.FirstLevelSubdivisionTableID

&& a.FK\_SecondLevelSubdivisionTable == null

select a).FirstOrDefault();

}

if (collected == null)

{

Value\_ = 0;

}

else

if (collected.Value == null)

{

Value\_ = 0;

}

else

{

Value\_ = (float)collected.Value;

}

#endregion

ChartOneValue DataRowForChart = new ChartOneValue(Name\_, Value\_, Planned\_Value);

return DataRowForChart;\*/

}

protected void Page\_Load(object sender, EventArgs e)

{

Serialization UserSer = (Serialization)Session["UserID"];

if (UserSer == null)

{

Response.Redirect("~/Default.aspx");

}

int userID = UserSer.Id;

KPIWebDataContext kpiWebDataContext = new KPIWebDataContext();

UsersTable userTable =

(from a in kpiWebDataContext.UsersTable where a.UsersTableID == userID select a).FirstOrDefault();

if (userTable.AccessLevel != 7)

{

Response.Redirect("~/Default.aspx");

}

//-------------------------------------------------------------------------------------------------------------

if (!IsPostBack)

{

#region session

RectorHistorySession rectorHistory = (RectorHistorySession)Session["rectorHistory"];

if (rectorHistory == null)

{

Response.Redirect("~/Default.aspx");

}

string val = this.Request.QueryString["HLevel"]; //hisoty level сова придумал)

if (val != null)

{

rectorHistory.CurrentSession = Convert.ToInt32(val);

Session["rectorHistory"] = rectorHistory;

}

RectorSession CurrentRectorSession = rectorHistory.RectorSession[rectorHistory.CurrentSession];

ForRCalc.Struct mainStruct = CurrentRectorSession.sesStruct;

int ViewType = CurrentRectorSession.sesViewType;

int ParamID = CurrentRectorSession.sesParamID;

int ParamType = CurrentRectorSession.sesParamType;

int ReportID = CurrentRectorSession.sesReportID;

int SpecID = CurrentRectorSession.sesSpecID;

#endregion

DataTable dataTable = new DataTable();

dataTable.Columns.Add(new DataColumn("ID", typeof(string)));

dataTable.Columns.Add(new DataColumn("Name", typeof(string)));

dataTable.Columns.Add(new DataColumn("Value", typeof(string)));

dataTable.Columns.Add(new DataColumn("PlannedValue", typeof(string)));

dataTable.Columns.Add(new DataColumn("Number", typeof(string)));

ReportArchiveTable ReportTable = (from a in kpiWebDataContext.ReportArchiveTable

where a.ReportArchiveTableID == ReportID

select a).FirstOrDefault();

if (ViewType == 0) // просмотр для структурных подразделений

{

Title.Text = (from a in kpiWebDataContext.IndicatorsTable

where a.IndicatorsTableID == ParamID

select a.Name).FirstOrDefault();

List<ForRCalc.Struct> currentStructList = new List<ForRCalc.Struct>();

currentStructList = ForRCalc.GetChildStructList(mainStruct, ReportID);

foreach (ForRCalc.Struct currentStruct in currentStructList)

{

DataRow dataRow = dataTable.NewRow();

dataRow["ID"] = ForRCalc.GetLastID(currentStruct).ToString();

dataRow["Number"] = "";

dataRow["Name"] = currentStruct.Name;

FirstLevelSubdivisionTable Academy = (from a in kpiWebDataContext.FirstLevelSubdivisionTable

where a.FirstLevelSubdivisionTableID == currentStruct.Lv\_1

select a).FirstOrDefault();

IndicatorsTable CurrentIndicator = (from a in kpiWebDataContext.IndicatorsTable

where a.IndicatorsTableID == ParamID

select a).FirstOrDefault();

ChartOneValue CurrentValue = ForRCalc.GetCalculatedIndicator(ReportID, CurrentIndicator, Academy, null);

dataRow["Value"] = FloatToStrFormat(CurrentValue.value, CurrentValue.planned, (Int32)CurrentIndicator.DataType);

// ForRCalc.GetCalculatedWithParams(currentStruct, ParamType, ParamID, ReportID, SpecID).ToString("0.000");

dataTable.Rows.Add(dataRow);

}

Grid.Columns[3].Visible = false;

Grid.Columns[4].Visible = false;

Grid.Columns[5].Visible = false;

// Grid.DataSource = dataTable;

// Grid.DataBind();

}

else if (ViewType == 1)

{

Title.Text = "Целевые показатели";

if (ParamType == 0) //считаем целевой показатель

{

List<IndicatorsTable> Indicators = (

from a in kpiWebDataContext.IndicatorsTable

join b in kpiWebDataContext.IndicatorsAndUsersMapping

on a.IndicatorsTableID equals b.FK\_IndicatorsTable

join c in kpiWebDataContext.ReportArchiveAndIndicatorsMappingTable

on a.IndicatorsTableID equals c.FK\_IndicatorsTable

where

a.Active == true

&& b.CanView == true

&& b.FK\_UsresTable == userID

&& c.FK\_ReportArchiveTable == ReportID

select a).OrderBy(mc => mc.SortID).ToList();

foreach (IndicatorsTable CurrentIndicator in Indicators)

{

DataRow dataRow = dataTable.NewRow();

dataRow["ID"] = CurrentIndicator.IndicatorsTableID; //GetLastID(currentStruct).ToString();

dataRow["Number"] = "";

if (CurrentIndicator.Measure != null)

{

if (CurrentIndicator.Measure.Length > 2)

{

dataRow["Name"] = CurrentIndicator.Name + " (" + CurrentIndicator.Measure + ")";

}

else

{

dataRow["Name"] = CurrentIndicator.Name;

}

}

else

{

dataRow["Name"] = CurrentIndicator.Name;

}

PlannedIndicator plannedValue = (from a in kpiWebDataContext.PlannedIndicator

where a.FK\_IndicatorsTable == CurrentIndicator.IndicatorsTableID

&& a.Date > DateTime.Now

select a).OrderBy(x => x.Date).FirstOrDefault();

if (plannedValue != null)

{

dataRow["PlannedValue"] = plannedValue.Value;

}

else

{

dataRow["PlannedValue"] = "Не определено";

}

/\* FirstLevelSubdivisionTable Academy = (from a in kpiWebDataContext.FirstLevelSubdivisionTable

where a.FirstLevelSubdivisionTableID == mainStruct.Lv\_1

select a).FirstOrDefault();\*/

// SecondLevelSubdivisionTable Faculty = new SecondLevelSubdivisionTable();

ChartOneValue CurrentValue = ForRCalc.GetCalculatedIndicator(ReportID, CurrentIndicator, null, null);

///плановое тоже надо брать из этого экзамляра а не заново искать

float tmp = CurrentValue.value;

//ForRCalc.CalculatedForDB(ForRCalc.GetCalculatedWithParams(mainStruct, ParamType,

// CurrentIndicator.IndicatorsTableID, ReportID, SpecID));

if (tmp == (float)1E+20)

{

dataRow["Value"] = "Рассчет невозможен";

}

else

{

dataRow["Value"] = FloatToStrFormat(tmp, CurrentValue.planned, (Int32)CurrentIndicator.DataType);

}

dataTable.Rows.Add(dataRow);

}

}

Grid.Columns[0].Visible = false;

}

else if (ViewType == 5)

{

Title.Text = (from a in kpiWebDataContext.IndicatorsTable

where a.IndicatorsTableID == ParamID

select a.Name).FirstOrDefault();

List<ForRCalc.Struct> currentStructList = new List<ForRCalc.Struct>();

currentStructList = ForRCalc.GetAllSecondLevel();

foreach (ForRCalc.Struct currentStruct in currentStructList)

{

DataRow dataRow = dataTable.NewRow();

dataRow["ID"] = ForRCalc.GetLastID(currentStruct).ToString();

dataRow["Number"] = "";

dataRow["Name"] = currentStruct.Name +", "+(from a in kpiWebDataContext.FirstLevelSubdivisionTable

where a.FirstLevelSubdivisionTableID == currentStruct.Lv\_1

select a.Name).FirstOrDefault();

FirstLevelSubdivisionTable Academy = (from a in kpiWebDataContext.FirstLevelSubdivisionTable

where a.FirstLevelSubdivisionTableID == currentStruct.Lv\_1

select a).FirstOrDefault();

SecondLevelSubdivisionTable Facullty = (from a in kpiWebDataContext.SecondLevelSubdivisionTable

where a.SecondLevelSubdivisionTableID == currentStruct.Lv\_2

select a).FirstOrDefault();

IndicatorsTable CurrentIndicator = (from a in kpiWebDataContext.IndicatorsTable

where a.IndicatorsTableID == ParamID

select a).FirstOrDefault();

ChartOneValue CurrentValue = ForRCalc.GetCalculatedIndicator(ReportID, CurrentIndicator, Academy, Facullty);

dataRow["Value"] = FloatToStrFormat(CurrentValue.value, CurrentValue.planned, (Int32)CurrentIndicator.DataType);

// ForRCalc.GetCalculatedWithParams(currentStruct, ParamType, ParamID, ReportID, SpecID).ToString("0.000");

dataTable.Rows.Add(dataRow);

}

Grid.Columns[3].Visible = false;

Grid.Columns[4].Visible = false;

Grid.Columns[5].Visible = false;

}

else

{

//error // wrong ViewType

}

if (ViewType != 1)

{

DataView dv = dataTable.DefaultView;

dv.Sort = "Value desc";

dataTable = dv.ToTable();

int Number = 0;

//Number++;

foreach (DataRow row in dataTable.Rows)

row["Number"] = ++Number;

}

Grid.DataSource = dataTable;

Grid.DataBind();

}

}

protected void Button1Click(object sender, EventArgs e)

{

Button button = (Button)sender;

{

RectorHistorySession rectorHistory = (RectorHistorySession)Session["rectorHistory"];

if (rectorHistory == null)

{

Response.Redirect("~/Default.aspx");

}

RectorSession CurrentRectorSession = rectorHistory.RectorSession[rectorHistory.CurrentSession];

ForRCalc.Struct mainStruct = CurrentRectorSession.sesStruct;

int ViewType = CurrentRectorSession.sesViewType;

int ParamID = CurrentRectorSession.sesParamID;

int ParamType = CurrentRectorSession.sesParamType;

int ReportID = CurrentRectorSession.sesReportID;

int SpecID = CurrentRectorSession.sesSpecID;

RectorSession currentRectorSession = new RectorSession(mainStruct, ViewType, ParamID, ParamType, ReportID, SpecID, "");

if (currentRectorSession.sesViewType == 1)

{//впервые перешли на разложение по структуре сразу после показателя

currentRectorSession.sesParamID = Convert.ToInt32(button.CommandArgument);

currentRectorSession.sesViewType = 0;

currentRectorSession.sesStruct.Lv\_0 = 1;

currentRectorSession.sesStruct.Lv\_1 = 0;

currentRectorSession.sesStruct.Lv\_2 = 0;

currentRectorSession.sesStruct.Lv\_3 = 0;

currentRectorSession.sesStruct.Lv\_4 = 0;

currentRectorSession.sesStruct.Lv\_5 = 0;

}

else if (currentRectorSession.sesViewType == 3)

{

currentRectorSession.sesStruct = ForRCalc.StructDeeper(currentRectorSession.sesStruct, Convert.ToInt32(button.CommandArgument));

}

else

{

currentRectorSession.sesStruct = ForRCalc.StructDeeper(currentRectorSession.sesStruct, Convert.ToInt32(button.CommandArgument));

}

rectorHistory.CurrentSession++;

rectorHistory.SessionCount = rectorHistory.CurrentSession + 1;

rectorHistory.RectorSession[rectorHistory.CurrentSession] = currentRectorSession;

Session["rectorHistory"] = rectorHistory;

Response.Redirect("~/Rector/RRating.aspx");

}

}

protected void Button2Click(object sender, EventArgs e)

{

Button button = (Button)sender;

{

RectorHistorySession rectorHistory = (RectorHistorySession)Session["rectorHistory"];

if (rectorHistory == null)

{

Response.Redirect("~/Default.aspx");

}

RectorSession CurrentRectorSession = rectorHistory.RectorSession[rectorHistory.CurrentSession];

ForRCalc.Struct mainStruct = CurrentRectorSession.sesStruct;

int ViewType = CurrentRectorSession.sesViewType;

int ParamID = CurrentRectorSession.sesParamID;

int ParamType = CurrentRectorSession.sesParamType;

int ReportID = CurrentRectorSession.sesReportID;

int SpecID = CurrentRectorSession.sesSpecID;

RectorSession currentRectorSession = new RectorSession(mainStruct, ViewType, ParamID, ParamType, ReportID, SpecID, "");

currentRectorSession.sesParamID = Convert.ToInt32(button.CommandArgument);

currentRectorSession.sesViewType = 5;

currentRectorSession.sesStruct.Lv\_0 = 1;

currentRectorSession.sesStruct.Lv\_1 = 0;

currentRectorSession.sesStruct.Lv\_2 = 0;

currentRectorSession.sesStruct.Lv\_3 = 0;

currentRectorSession.sesStruct.Lv\_4 = 0;

currentRectorSession.sesStruct.Lv\_5 = 0;

rectorHistory.CurrentSession++;

rectorHistory.SessionCount = rectorHistory.CurrentSession + 1;

rectorHistory.RectorSession[rectorHistory.CurrentSession] = currentRectorSession;

Session["rectorHistory"] = rectorHistory;

Response.Redirect("~/Rector/RRating.aspx");

}

}

protected void Button22\_Click(object sender, EventArgs e)

{

Response.Redirect("~/Rector/RectorMain.aspx");

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Filename: RRating.aspx.designer.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//------------------------------------------------------------------------------

// <автоматически создаваемое>

// Этот код создан программой.

//

// Изменения в этом файле могут привести к неправильной работе и будут потеряны в случае

// повторной генерации кода.

// </автоматически создаваемое>

//------------------------------------------------------------------------------

namespace KPIWeb.Rector {

public partial class RRating {

/// <summary>

/// top\_panel2 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Panel top\_panel2;

/// <summary>

/// GoBackButton элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Button GoBackButton;

/// <summary>

/// GoForwardButton элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Button GoForwardButton;

/// <summary>

/// Button22 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Button Button22;

/// <summary>

/// Button11 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Button Button11;

/// <summary>

/// Title элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Label Title;

/// <summary>

/// Grid элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.GridView Grid;

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Filename: RShowChart.aspx.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using System;

using System.Collections.Generic;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.DataVisualization.Charting;

using System.Web.UI.WebControls;

namespace KPIWeb.Rector

{

public partial class RShowChart : System.Web.UI.Page

{

public string FloatToStrFormat(float value, float plannedValue, int DataType)

{

if (DataType == 1)

{

string tmpValue = Math.Round(value).ToString();// value.ToString("0");

return tmpValue;

}

else if (DataType == 2)

{

string tmpValue = value.ToString();

string tmpPlanned = plannedValue.ToString();

int PlannedNumbersAftepPoint = 2;

if (tmpPlanned.IndexOf(',') != -1)

{

PlannedNumbersAftepPoint = (tmpPlanned.Length - tmpPlanned.IndexOf(',') + 1);

}

int ValuePointIndex = tmpValue.IndexOf(',');

if (ValuePointIndex != -1)

{

if ((tmpValue.Length - ValuePointIndex - PlannedNumbersAftepPoint) > 0)

{

tmpValue = tmpValue.Remove(ValuePointIndex + PlannedNumbersAftepPoint, tmpValue.Length - ValuePointIndex - PlannedNumbersAftepPoint);

}

}

return tmpValue;

}

return "0";

}

protected void Page\_Load(object sender, EventArgs e)

{

Serialization UserSer = (Serialization)Session["UserID"];

if (UserSer == null)

{

Response.Redirect("~/Default.aspx");

}

int userID = UserSer.Id;

KPIWebDataContext kPiDataContext = new KPIWebDataContext();

UsersTable userTable =

(from a in kPiDataContext.UsersTable where a.UsersTableID == userID select a).FirstOrDefault();

if (userTable.AccessLevel != 7)

{

Response.Redirect("~/Default.aspx");

}

RectorChartSession RectorChart = (RectorChartSession)Session["RectorChart"];

if (RectorChart == null)

{

Response.Redirect("~/Default.aspx");

}

List<int> IndicatorsList = RectorChart.IndicatorsList;

ViewState["IndicatorsList"] = IndicatorsList;

//IndicatorsForCFU(IndicatorsList,1); // ВОЗВРАЩАЕТ ДАННЫЕ ДЛЯ ЧАРТА

if (!IsPostBack)

{

// Формируем GridView

DataTable dataTable = new DataTable();

dataTable.Columns.Add(new DataColumn("IndicatorID", typeof(string)));

dataTable.Columns.Add(new DataColumn("IndicatorName", typeof(string)));

foreach (var item in IndicatorsList)

{

string namestr =

(from i in kPiDataContext.IndicatorsTable where i.IndicatorsTableID == item select i.Name)

.FirstOrDefault();

string tmp;

if (namestr.Length > 124) tmp = namestr.Substring(0, 125) + "...";

else tmp = namestr;

DataRow dataRow = dataTable.NewRow();

dataRow["IndicatorID"] = item;

dataRow["IndicatorName"] = tmp;

dataTable.Rows.Add(dataRow);

}

GridView1.DataSource = dataTable;

GridView1.DataBind();

}

}

public ChartValueArray AllIndicatorsForAcademys(int ReportID)

{

KPIWebDataContext kpiWebDataContext = new KPIWebDataContext();

List<IndicatorsTable> Indicators = (

from a in kpiWebDataContext.IndicatorsTable

where

a.Active == true

select a).OrderBy(mc => mc.SortID).ToList();

ChartValueArray DataForChart = new ChartValueArray("График достижения плановых значений целевых показателей");

foreach (IndicatorsTable CurrentIndicator in Indicators)

{

DataForChart.ChartValues.Add(ForRCalc.GetCalculatedIndicator(1, CurrentIndicator, null, null));

}

return DataForChart;

}

public ChartValueArray IndicatorForAllAcademys(int IndicatorID, int ReportID)

{

KPIWebDataContext kpiWebDataContext = new KPIWebDataContext();

IndicatorsTable Indicator = (from a in kpiWebDataContext.IndicatorsTable

where a.IndicatorsTableID == IndicatorID

select a).FirstOrDefault();

List<FirstLevelSubdivisionTable> AcademyList = (from a in kpiWebDataContext.FirstLevelSubdivisionTable

where a.Active == true

select a).ToList();

ChartValueArray DataForChart = new ChartValueArray("Целевой показатель '" + Indicator.Name + "' в разрезе академий КФУ");

foreach (FirstLevelSubdivisionTable CurrentAcademy in AcademyList)

{

DataForChart.ChartValues.Add(ForRCalc.GetCalculatedIndicator(1, Indicator, CurrentAcademy, null));

}

return DataForChart;

}

public ChartValueArray AllIndicatorsForOneAcademy(int AcademyID, int ReportID)

{

KPIWebDataContext kpiWebDataContext = new KPIWebDataContext();

List<IndicatorsTable> Indicators = (

from a in kpiWebDataContext.IndicatorsTable

where

a.Active == true

select a).OrderBy(mc => mc.SortID).ToList();

FirstLevelSubdivisionTable FirstLevelRow = (from a in kpiWebDataContext.FirstLevelSubdivisionTable

where a.FirstLevelSubdivisionTableID == AcademyID

select a).FirstOrDefault();

ChartValueArray DataForChart = new ChartValueArray("График достижения плановых значений целевых показателей для академии " + FirstLevelRow.Name);

foreach (IndicatorsTable CurrentIndicator in Indicators)

{

DataForChart.ChartValues.Add(ForRCalc.GetCalculatedIndicator(1, CurrentIndicator, FirstLevelRow, null));

}

return DataForChart;

}

public ChartValueArray IndicatorsForCFU(List<int> Indicators, int ReportID)

{

KPIWebDataContext kpiWebDataContext = new KPIWebDataContext();

ChartValueArray DataForChart = new ChartValueArray("График достижения выбранных плановых значений целевых показателей для КФУ");

foreach (int CurrentIndicatorID in Indicators)

{

IndicatorsTable Indicator = (

from a in kpiWebDataContext.IndicatorsTable

where

a.Active == true

&& a.IndicatorsTableID == CurrentIndicatorID

select a).FirstOrDefault();

DataForChart.ChartValues.Add(ForRCalc.GetCalculatedIndicator(1, Indicator, null, null));

}

return DataForChart;

}

public ChartOneValue IndicatorsForCFUOneIndicator(int curIndicator, int ReportID)

{

KPIWebDataContext kpiWebDataContext = new KPIWebDataContext();

IndicatorsTable Indicator = (

from a in kpiWebDataContext.IndicatorsTable

where

a.Active == true

&& a.IndicatorsTableID == curIndicator

select a).FirstOrDefault();

return ForRCalc.GetCalculatedIndicator(1, Indicator, null, null);

}

public ChartValueArray IndicatorsForAllFacultys(int IndicatorID, int ReportID)

{

KPIWebDataContext kpiWebDataContext = new KPIWebDataContext();

IndicatorsTable Indicator = (from a in kpiWebDataContext.IndicatorsTable

where a.IndicatorsTableID == IndicatorID

select a).FirstOrDefault();

List<SecondLevelSubdivisionTable> FacultyList = (from a in kpiWebDataContext.SecondLevelSubdivisionTable

where a.Active == true

select a).ToList();

ChartValueArray DataForChart = new ChartValueArray("Целевой показатель '" + Indicator.Name + "' в разрезе факультетов КФУ");

foreach (SecondLevelSubdivisionTable CurrentFavulty in FacultyList)

{

FirstLevelSubdivisionTable Academy = (from a in kpiWebDataContext.FirstLevelSubdivisionTable

where a.FirstLevelSubdivisionTableID == CurrentFavulty.FK\_FirstLevelSubdivisionTable

select a).FirstOrDefault();

DataForChart.ChartValues.Add(ForRCalc.GetCalculatedIndicator(1, Indicator, Academy, CurrentFavulty));

}

return DataForChart;

}

protected void Button1\_Click(object sender, EventArgs e) //Значение каждого отмеченного показателя рассчитанное по КФУ

{

RectorChartSession RectorChart = (RectorChartSession)Session["RectorChart"]; //вытаскиваем из сессии список айдишников показателей

ChartValueArray NewDataForChart = IndicatorsForCFU(RectorChart.IndicatorsList, 1);

}

protected void GridView1\_RowDataBound(object sender, GridViewRowEventArgs e)

{

if (e.Row.RowType == DataControlRowType.DataRow)

{

//// tooltip

KPIWebDataContext kPiDataContext = new KPIWebDataContext();

List<int> IndicatorsList = (List<int>)ViewState["IndicatorsList"];

var indicator = (from ind in kPiDataContext.IndicatorsTable

where ind.IndicatorsTableID == IndicatorsList[e.Row.RowIndex]

select ind).FirstOrDefault();

// ERROR != null

string tooltip = indicator.Name;

if (tooltip.Count() > 124)

e.Row.Cells[1].ToolTip = tooltip;

///////

string indID = DataBinder.Eval(e.Row.DataItem, "IndicatorID").ToString();

Chart chart = (Chart)e.Row.FindControl("Chart3");

Button button = (Button)e.Row.FindControl("Button7");

//chart.TempDirectory = "/app/tmp";

if (chart != null)

{

ChartOneValue DataForChart = IndicatorsForCFUOneIndicator(Convert.ToInt32(indID), 1); // 1 индикатор в разрезе КФУ взятый по ID

// Формирую чарт

//chart.Series["ValueSeries"].Color = Color.CornflowerBlue; //Color.FromArgb(255, 100, 149, 0);

//chart.Series["ValueSeries"].Color = Color.FromArgb(255, 100, 149, 237);

chart.Series["ValueSeries"].Color = Color.FromArgb(255, 100, 149, 237);

chart.ChartAreas["ChartArea1"].AxisX.IntervalAutoMode = IntervalAutoMode.VariableCount; // масштабирование разметки

chart.ChartAreas["ChartArea1"].AxisY.IntervalAutoMode = IntervalAutoMode.VariableCount; // -==--

//chart.Series["TargetSeries"].Color = Color.FromArgb(122, 50, 255, 0);

chart.Series["TargetSeries"].Color = Color.FromArgb(75, 20, 30, 0);

ChartItems chartItems = new ChartItems(); // класс для работы с чартом

chartItems.AddChartItem(DataForChart.name, DataForChart.value); // добавляем по оси X value

if (DataForChart.planned > 0)

chart.Series["TargetSeries"].Points.AddY(DataForChart.planned); // добавляем по оси Y Target value

// Привязываем класс с объектом к диаграмме

chart.DataSource = chartItems.GetDataSource();

//chart.Series["ValueSeries"].XValueMember = "Name";

chart.Series["ValueSeries"].YValueMembers = "Value";

if (DataForChart.value != 0)

{

chart.Series["ValueSeries"].Label = "#VALY " + indicator.Measure;

chart.Series["ValueSeries"].Font = new Font("Arial", 10f, FontStyle.Bold);

}

chart.Series["ValueSeries"].LegendText = "#AXISLABEL (#PERCENT{P0})";

if (DataForChart.planned != 0)

{

chart.Series["ValueSeries"].ToolTip = "Целевое: " + DataForChart.planned + ". Достигнуто на: " + Convert.ToInt32(DataForChart.value / DataForChart.planned \* 100) + "%";

chart.Series["TargetSeries"].ToolTip = "Целевое: " + DataForChart.planned + ". Достигнуто на: " + Convert.ToInt32(DataForChart.value / DataForChart.planned \* 100) + "%";

}

else

{

chart.Series["ValueSeries"].ToolTip = "Целевое: " + DataForChart.planned ;

chart.Series["TargetSeries"].ToolTip = "Целевое: " + DataForChart.planned;

}

// Линния планового значения

VerticalLineAnnotation verticalLine = new VerticalLineAnnotation();

verticalLine.AxisX = chart.ChartAreas["ChartArea1"].AxisX;

verticalLine.AxisY = chart.ChartAreas["ChartArea1"].AxisY;

verticalLine.Width = 3;

verticalLine.IsInfinitive = true; // либо высоту

verticalLine.LineDashStyle = ChartDashStyle.Solid;

verticalLine.LineColor = Color.Crimson;

verticalLine.LineWidth = 3;

verticalLine.AnchorX = DataForChart.planned;

verticalLine.Name = "myLine"; // !!

verticalLine.AnchorY = 0;

verticalLine.X = DataForChart.planned; ;

verticalLine.Y = 0;

//Прямоугольник со значением

RectangleAnnotation RA = new RectangleAnnotation();

RA.AxisX = chart.ChartAreas["ChartArea1"].AxisX;

RA.IsSizeAlwaysRelative = false;

// КОСТЫЛЬ "формула" расчета масштабируемости прямоугольника

try

{

if (DataForChart.value > DataForChart.planned)

{

int tmp = Math.Round(DataForChart.value, 3).ToString().Count();

if (tmp > 3)

RA.Width = 20\*DataForChart.value/(500 - ((tmp - 3)\*60));

else

{

RA.Width = 20\*DataForChart.value/500;

}

}

else

{

int tmp = Math.Round(DataForChart.planned, 3).ToString().Count();

if (tmp > 3)

RA.Width = 20\*DataForChart.planned/(500 - ((tmp - 3)\*60));

else

{

RA.Width = 20\*DataForChart.planned/500;

}

}

}

catch

{

RA.Width = 290000000 - 1 ;

}

// END КОСТЫЛЬ

RA.Height = 8 \* 0.04;

verticalLine.Name = "myRect"; // !!

RA.LineColor = Color.Red;

RA.BackColor = Color.Red;

RA.AxisY = chart.ChartAreas["ChartArea1"].AxisY;

RA.Y = -RA.Height;

RA.X = verticalLine.X - RA.Width / 2;

RA.Text = DataForChart.planned.ToString();

RA.ForeColor = Color.White;

RA.Font = new System.Drawing.Font("Arial", 8f, FontStyle.Bold);

// Consider adding transparency so that the strip lines are lighter

// stripLine1.BackColor = Color.FromArgb(200, 200, 0, 0);

// stripLine1.BackSecondaryColor = Color.FromArgb(122, 250, 255, 0);

// stripLine1.BackGradientStyle = GradientStyle.LeftRight;

// Add the strip line to the chart

chart.Annotations.Add(verticalLine);

chart.Annotations.Add(RA);

if (Convert.ToInt32(indID) == 1026 || DataForChart.value == 0)

{

button.Enabled = false;

}

}

}

}

protected void DetailedButtonClick(object sender, EventArgs e)

{

Button button = (Button)sender;

{

var par = button.CommandArgument.ToString();

Session["IndicatorToDetailed"] = par;

Response.Redirect("~/Rector/RShowChartDetailed.aspx");

}

}

protected void Button2\_Click(object sender, EventArgs e)

{

Response.Redirect("~/Rector/RectorMain.aspx");

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Filename: RShowChart.aspx.designer.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//------------------------------------------------------------------------------

// <автоматически создаваемое>

// Этот код создан программой.

//

// Изменения в этом файле могут привести к неправильной работе и будут потеряны в случае

// повторной генерации кода.

// </автоматически создаваемое>

//------------------------------------------------------------------------------

namespace KPIWeb.Rector {

public partial class RShowChart {

/// <summary>

/// top\_panel2 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Panel top\_panel2;

/// <summary>

/// GoBackButton элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Button GoBackButton;

/// <summary>

/// GoForwardButton элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Button GoForwardButton;

/// <summary>

/// Button2 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Button Button2;

/// <summary>

/// Button1 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Button Button1;

/// <summary>

/// Button6 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Button Button6;

/// <summary>

/// GridView1 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.GridView GridView1;

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Filename: RShowChartDetailed.aspx.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using System;

using System.Collections.Generic;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.DataVisualization.Charting;

using System.Web.UI.WebControls;

namespace KPIWeb.Rector

{

public partial class RShowChartDetailed : System.Web.UI.Page

{

public ChartOneValue GetCalculatedIndicator(int ReportID, IndicatorsTable Indicator, FirstLevelSubdivisionTable Academy, SecondLevelSubdivisionTable Faculty) // academyID == null && facultyID==null значит для всего КФУ

{

KPIWebDataContext kpiWebDataContext = new KPIWebDataContext();

float Planned\_Value = 0;

string Name\_ = "";

float Value\_ = 0;

#region plannedIndicator

PlannedIndicator plannedValue = (from a in kpiWebDataContext.PlannedIndicator

where a.FK\_IndicatorsTable == Indicator.IndicatorsTableID

&& a.Date > DateTime.Now

select a).OrderBy(x => x.Date).FirstOrDefault();

if (plannedValue != null)

{

Planned\_Value = (float)plannedValue.Value;

}

#endregion

#region Name

if ((Academy == null) && (Faculty == null))

{

if (Indicator.Measure != null)

{

if (Indicator.Measure.Length > 0)

{

Name\_ = Indicator.Name + " (" + Indicator.Measure + ")";

}

else

{

Name\_ = Indicator.Name;

}

}

else

{

Name\_ = Indicator.Name;

}

}

else if (Faculty != null)

{

Name\_ = Faculty.Name;

}

else if (Academy != null)

{

Name\_ = Academy.Name;

}

#endregion

#region

//ForRCalc.Struct mainStruct = mainStruct = new ForRCalc.Struct(1, 0, 0, 0, 0, "N");

CollectedIndicatorsForR collected = new CollectedIndicatorsForR();

if ((Academy == null) && (Faculty == null))

{

//mainStruct = new ForRCalc.Struct(1, 0, 0, 0, 0, "N");

collected = (from a in kpiWebDataContext.CollectedIndicatorsForR

where a.FK\_ReportArchiveTable == ReportID

&& a.FK\_IndicatorsTable == Indicator.IndicatorsTableID

&& a.FK\_FirstLevelSubdivisionTable == null

&& a.FK\_SecondLevelSubdivisionTable == null

select a).FirstOrDefault();

}

else if (Faculty != null)

{

//mainStruct = new ForRCalc.Struct(1, Faculty.FK\_FirstLevelSubdivisionTable, Faculty.SecondLevelSubdivisionTableID, 0, 0, "N");

collected = (from a in kpiWebDataContext.CollectedIndicatorsForR

where a.FK\_ReportArchiveTable == ReportID

&& a.FK\_IndicatorsTable == Indicator.IndicatorsTableID

&& a.FK\_FirstLevelSubdivisionTable == Faculty.FK\_FirstLevelSubdivisionTable

&& a.FK\_SecondLevelSubdivisionTable == Faculty.SecondLevelSubdivisionTableID

select a).FirstOrDefault();

}

else if (Academy != null)

{

//mainStruct = new ForRCalc.Struct(1, Academy.FirstLevelSubdivisionTableID, 0, 0, 0, "N");

collected = (from a in kpiWebDataContext.CollectedIndicatorsForR

where a.FK\_ReportArchiveTable == ReportID

&& a.FK\_IndicatorsTable == Indicator.IndicatorsTableID

&& a.FK\_FirstLevelSubdivisionTable == Academy.FirstLevelSubdivisionTableID

&& a.FK\_SecondLevelSubdivisionTable == null

select a).FirstOrDefault();

}

/\*

float tmp = ForRCalc.CalculatedForDB(ForRCalc.GetCalculatedWithParams(mainStruct, 0, Indicator.IndicatorsTableID, ReportID, 0));

if (tmp == (float)1E+20)

{

Value\_ = 0;

}

else

{

Value\_ = tmp;

}

\*/

if (collected.Value == null)

{

Value\_ = 0;

}

else

{

Value\_ = (float)collected.Value;

}

#endregion

ChartOneValue DataRowForChart = new ChartOneValue(Name\_, Value\_, Planned\_Value);

return DataRowForChart;

}

public ChartValueArray AllIndicatorsForAcademys(int ReportID)

{

KPIWebDataContext kpiWebDataContext = new KPIWebDataContext();

List<IndicatorsTable> Indicators = (

from a in kpiWebDataContext.IndicatorsTable

where

a.Active == true

select a).OrderBy(mc => mc.SortID).ToList();

ChartValueArray DataForChart = new ChartValueArray("График достижения плановых значений целевых показателей");

foreach (IndicatorsTable CurrentIndicator in Indicators)

{

DataForChart.ChartValues.Add(GetCalculatedIndicator(1, CurrentIndicator, null, null));

}

return DataForChart;

}

public ChartValueArray IndicatorForAllAcademys(int IndicatorID, int ReportID)

{

KPIWebDataContext kpiWebDataContext = new KPIWebDataContext();

IndicatorsTable Indicator = (from a in kpiWebDataContext.IndicatorsTable

where a.IndicatorsTableID == IndicatorID

select a).FirstOrDefault();

List<FirstLevelSubdivisionTable> AcademyList = (from a in kpiWebDataContext.FirstLevelSubdivisionTable

where a.Active == true

select a).ToList();

ChartValueArray DataForChart = new ChartValueArray("Целевой показатель '" + Indicator.Name + "' в разрезе академий КФУ");

foreach (FirstLevelSubdivisionTable CurrentAcademy in AcademyList)

{

DataForChart.ChartValues.Add(GetCalculatedIndicator(1, Indicator, CurrentAcademy, null));

}

return DataForChart;

}

public ChartValueArray AllIndicatorsForOneAcademy(int AcademyID, int ReportID)

{

KPIWebDataContext kpiWebDataContext = new KPIWebDataContext();

List<IndicatorsTable> Indicators = (

from a in kpiWebDataContext.IndicatorsTable

where

a.Active == true

select a).OrderBy(mc => mc.SortID).ToList();

FirstLevelSubdivisionTable FirstLevelRow = (from a in kpiWebDataContext.FirstLevelSubdivisionTable

where a.FirstLevelSubdivisionTableID == AcademyID

select a).FirstOrDefault();

ChartValueArray DataForChart = new ChartValueArray("График достижения плановых значений целевых показателей для академии " + FirstLevelRow.Name);

foreach (IndicatorsTable CurrentIndicator in Indicators)

{

DataForChart.ChartValues.Add(GetCalculatedIndicator(1, CurrentIndicator, FirstLevelRow, null));

}

return DataForChart;

}

public ChartValueArray IndicatorsForCFU(List<int> Indicators, int ReportID)

{

KPIWebDataContext kpiWebDataContext = new KPIWebDataContext();

ChartValueArray DataForChart = new ChartValueArray("График достижения выбранных плановых значений целевых показателей для КФУ");

foreach (int CurrentIndicatorID in Indicators)

{

IndicatorsTable Indicator = (

from a in kpiWebDataContext.IndicatorsTable

where

a.Active == true

&& a.IndicatorsTableID == CurrentIndicatorID

select a).FirstOrDefault();

DataForChart.ChartValues.Add(GetCalculatedIndicator(1, Indicator, null, null));

}

return DataForChart;

}

public ChartOneValue IndicatorsForCFUOneIndicator(int curIndicator, int ReportID)

{

KPIWebDataContext kpiWebDataContext = new KPIWebDataContext();

IndicatorsTable Indicator = (

from a in kpiWebDataContext.IndicatorsTable

where

a.Active == true

&& a.IndicatorsTableID == curIndicator

select a).FirstOrDefault();

return GetCalculatedIndicator(1, Indicator, null, null);

}

public ChartValueArray IndicatorsForAllFacultys(int IndicatorID, int ReportID)

{

KPIWebDataContext kpiWebDataContext = new KPIWebDataContext();

IndicatorsTable Indicator = (from a in kpiWebDataContext.IndicatorsTable

where a.IndicatorsTableID == IndicatorID

select a).FirstOrDefault();

List<SecondLevelSubdivisionTable> FacultyList = (from a in kpiWebDataContext.SecondLevelSubdivisionTable

where a.Active == true

select a).ToList();

ChartValueArray DataForChart = new ChartValueArray("Целевой показатель '" + Indicator.Name + "' в разрезе факультетов КФУ");

foreach (SecondLevelSubdivisionTable CurrentFavulty in FacultyList)

{

FirstLevelSubdivisionTable Academy = (from a in kpiWebDataContext.FirstLevelSubdivisionTable

where a.FirstLevelSubdivisionTableID == CurrentFavulty.FK\_FirstLevelSubdivisionTable

select a).FirstOrDefault();

DataForChart.ChartValues.Add(GetCalculatedIndicator(1, Indicator, Academy, CurrentFavulty));

}

return DataForChart;

}

protected void Page\_Load(object sender, EventArgs e)

{

Serialization UserSer = (Serialization)Session["UserID"];

if (UserSer == null)

{

Response.Redirect("~/Default.aspx");

}

int userID = UserSer.Id;

KPIWebDataContext kPiDataContext = new KPIWebDataContext();

UsersTable userTable =

(from a in kPiDataContext.UsersTable where a.UsersTableID == userID select a).FirstOrDefault();

if (userTable.AccessLevel != 7)

{

Response.Redirect("~/Default.aspx");

}

String IndicatorID = (string)Session["IndicatorToDetailed"];

int indicator = Convert.ToInt32(IndicatorID);

List<int> Columnindicators = new List<int> {1016,1017,1024,1035};

ChartItems chartItems = new ChartItems();

ChartValueArray DataForChart = IndicatorForAllAcademys(indicator, 1);

// Формируем GridView

DataTable dataTable = new DataTable();

dataTable.Columns.Add(new DataColumn("IndicatorID", typeof(string)));

dataTable.Columns.Add(new DataColumn("Ratio", typeof(string)));

dataTable.Columns.Add(new DataColumn("IndicatorName", typeof(string)));

dataTable.Columns.Add(new DataColumn("IndicatorValue", typeof(string)));

#region график + чуток gridview

// Форматируем диаграмму

Chart1.BackColor = Color.White;

Chart1.BackSecondaryColor = Color.White;

Chart1.BackGradientStyle = GradientStyle.DiagonalRight;

Chart1.BorderlineDashStyle = ChartDashStyle.Solid;

Chart1.BorderlineColor = Color.Black;

Chart1.BorderSkin.SkinStyle = BorderSkinStyle.Emboss;

// Форматируем область диаграммы

Chart1.ChartAreas[0].BackColor = Color.White;

// заголовок

Chart1.Titles.Add(DataForChart.chartName);

Chart1.Titles[0].Font = new Font("Utopia", 16);

if (Columnindicators.Contains(indicator)) // Column для нескольки показателей из ColumnIndicators

{

Chart1.Series.Add(new Series("Value")

{

ChartType = SeriesChartType.Column,

Color = Color.SeaGreen

});

Chart1.Series.Add(new Series("Planned")

{

ChartType = SeriesChartType.FastLine,

Color = Color.Red,

BorderWidth = 4,

});

Chart1.ChartAreas[0].AxisX.IntervalAutoMode = IntervalAutoMode.VariableCount;

Chart1.ChartAreas[0].AxisY.IntervalAutoMode = IntervalAutoMode.VariableCount;

int ratio = 1;

List<ChartOneValue> sortItems = chartItems.SortReverse(DataForChart.ChartValues);

ViewState["Items"] = sortItems;

var measure = (from ind in kPiDataContext.IndicatorsTable

where ind.IndicatorsTableID == indicator

select ind.Measure).FirstOrDefault().ToString();

// chartItems.AddChartItem("", 0);

ChartOneValue DataForChartKFUvalue = IndicatorsForCFUOneIndicator(indicator, 1);

foreach (ChartOneValue item in sortItems)

{

if (item.value == 0) continue;

chartItems.AddChartItem(item.name, item.value);

Chart1.Series["Planned"].Points.AddY(DataForChartKFUvalue.value);

DataRow dataRow = dataTable.NewRow();

dataRow["IndicatorID"] = (from a in kPiDataContext.FirstLevelSubdivisionTable

where a.Name.Equals(item.name)

select a.FirstLevelSubdivisionTableID).FirstOrDefault();

// Не индикаторID а FirstLevelSubdivisionTableID

dataRow["Ratio"] = ratio; //Ratio

dataRow["IndicatorName"] = item.name;

dataRow["IndicatorValue"] = Math.Round(item.value, 3) + " " + measure;

dataTable.Rows.Add(dataRow);

ratio++;

}

Chart1.Legends.Add(new Legend("Default") {Docking = Docking.Right, Font = new Font("Arial", 11)});

// Chart1.Legends["Default"].Font = new Font("Utopia", 16);

// Привязать источник к диаграмме

Chart1.DataSource = chartItems.GetDataSource();

Chart1.Series["Value"].XValueMember = "Name";

Chart1.Series["Value"].YValueMembers = "Value";

// Random random = new Random();

//foreach (var item in Chart1.Series[0].Points)

//{

// Color c = Color.FromArgb(random.Next(0, 255), random.Next(0, 255), random.Next(0, 255));

//item.Color = c;

//}

Chart1.Series["Value"].Label = "#VALY" + " %";

Chart1.Series["Value"].LegendText = "Значение по каждой академии";

Chart1.Series["Planned"].LegendText = "Плановое значение по КФУ";

Chart1.ChartAreas["ChartArea1"].AxisX = new Axis

{

LabelStyle = new LabelStyle() {Font = new Font("Verdana", 6.5f)}

};

Chart1.ChartAreas["ChartArea1"].AxisX.LabelAutoFitMinFontSize = 5;

//Chart1.ChartAreas["ChartArea1"].AxisX.LabelStyle.Angle = 30;

Chart1.ChartAreas["ChartArea1"].AxisX.LabelAutoFitStyle = LabelAutoFitStyles.None;

Chart1.Series["Value"].ToolTip = "#VALX" + ", \n значение = "+ "#VALY" +" "+ measure;

Chart1.Series["Value"].Font = new Font("Arial", 9f, FontStyle.Bold);

GridView1.DataSource = dataTable;

GridView1.DataBind();

Chart1.Series["Value"].PostBackValue = "#INDEX";

}

else if (indicator == 1036) // Костыль для V.20

{

Chart1.Series.Add(new Series("Default")

{

ChartType = SeriesChartType.Pie,

Color = Color.CornflowerBlue

});

Chart1.ChartAreas[0].AxisX.IntervalAutoMode = IntervalAutoMode.VariableCount;

Chart1.ChartAreas[0].AxisY.IntervalAutoMode = IntervalAutoMode.VariableCount;

int ratio = 1;

List<ChartOneValue> sortItems = chartItems.SortReverse(DataForChart.ChartValues);

ViewState["Items"] = sortItems;

var measure = (from ind in kPiDataContext.IndicatorsTable

where ind.IndicatorsTableID == indicator

select ind.Measure).FirstOrDefault().ToString();

foreach (ChartOneValue item in sortItems)

{

if (item.value == 0) continue;

chartItems.AddChartItem(item.name, item.value);

DataRow dataRow = dataTable.NewRow();

dataRow["IndicatorID"] = (from a in kPiDataContext.FirstLevelSubdivisionTable

where a.Name.Equals(item.name)

select a.FirstLevelSubdivisionTableID).FirstOrDefault();

// Не индикаторID а FirstLevelSubdivisionTableID

dataRow["Ratio"] = ratio; //Ratio

dataRow["IndicatorName"] = item.name;

dataRow["IndicatorValue"] = Math.Round(item.value, 3) + " " + measure;

dataTable.Rows.Add(dataRow);

ratio++;

}

Chart1.Legends.Add(new Legend("Default") { Docking = Docking.Right, Font = new Font("Arial", 11) });

// Chart1.Legends["Default"].Font = new Font("Utopia", 16);

// Привязать источник к диаграмме

Chart1.DataSource = chartItems.GetDataSource();

Chart1.Series[0].XValueMember = "Name";

Chart1.Series[0].YValueMembers = "Value";

Chart1.Series[0].Label = "#VALY" + " %";

Chart1.Series[0].LegendText = "#AXISLABEL";

Chart1.Series[0].ToolTip = "#VALX";

Chart1.ChartAreas[0].Area3DStyle.Enable3D = true;

GridView1.DataSource = dataTable;

GridView1.DataBind();

Chart1.Series[0].PostBackValue = "#INDEX";

}

else // Pie для остальных

{

Chart1.Series.Add(new Series("Default")

{

ChartType = SeriesChartType.Pie,

Color = Color.CornflowerBlue

});

Chart1.ChartAreas[0].AxisX.IntervalAutoMode = IntervalAutoMode.VariableCount;

Chart1.ChartAreas[0].AxisY.IntervalAutoMode = IntervalAutoMode.VariableCount;

int ratio = 1;

List<ChartOneValue> sortItems = chartItems.SortReverse(DataForChart.ChartValues);

ViewState["Items"] = sortItems;

var measure = (from ind in kPiDataContext.IndicatorsTable

where ind.IndicatorsTableID == indicator

select ind.Measure).FirstOrDefault().ToString();

foreach (ChartOneValue item in sortItems)

{

if (item.value == 0) continue;

chartItems.AddChartItem(item.name, item.value);

DataRow dataRow = dataTable.NewRow();

dataRow["IndicatorID"] = (from a in kPiDataContext.FirstLevelSubdivisionTable

where a.Name.Equals(item.name)

select a.FirstLevelSubdivisionTableID).FirstOrDefault();

// Не индикаторID а FirstLevelSubdivisionTableID

dataRow["Ratio"] = ratio; //Ratio

dataRow["IndicatorName"] = item.name;

dataRow["IndicatorValue"] = Math.Round(item.value, 3) + " " + measure;

dataTable.Rows.Add(dataRow);

ratio++;

}

Chart1.Legends.Add(new Legend("Default") {Docking = Docking.Right, Font = new Font("Arial", 11)});

// Chart1.Legends["Default"].Font = new Font("Utopia", 16);

// Привязать источник к диаграмме

Chart1.DataSource = chartItems.GetDataSource();

Chart1.Series[0].XValueMember = "Name";

Chart1.Series[0].YValueMembers = "Value";

Chart1.Series[0].Label = "#PERCENT{P0}";

Chart1.Series[0].LegendText = "#AXISLABEL";

Chart1.Series[0].ToolTip = "#VALX";

Chart1.ChartAreas[0].Area3DStyle.Enable3D = true;

GridView1.DataSource = dataTable;

GridView1.DataBind();

Chart1.Series[0].PostBackValue = "#INDEX";

}

#endregion

}

protected void FacultyButtonClick(object sender, EventArgs e)

{

Button button = (Button)sender;

{

var par = button.CommandArgument.ToString();

Session["AcademyToDetailed"] = par;

Response.Redirect("~/Rector/RShowChartFaculty.aspx");

}

}

protected void GridView1\_RowDataBound(object sender, GridViewRowEventArgs e)

{

if (e.Row.RowType == DataControlRowType.DataRow)

{

List<int> ExceptItems = new List<int>

{

1022,

1023,

1025,

1012,

1026,

1013,

1027,

1020,

1018,

1017,

1034,

1021,

1015,

1028,

1029,

1030,

1019,

1033,

1031,

1032

}; // ID Академий с фейками на уровне кафедр

string indID = DataBinder.Eval(e.Row.DataItem, "IndicatorID").ToString();

if (ExceptItems.Contains(Convert.ToInt32(indID)))

{

Button button = (Button) e.Row.FindControl("Button7");

if (button != null)

button.Enabled = false;

}

}

}

protected void Button2\_Click(object sender, EventArgs e)

{

Response.Redirect("~/Rector/RectorMain.aspx");

}

protected void Chart1\_Click(object sender, ImageMapEventArgs e)

{

try

{

List<int> ExceptItems = new List<int>

{

1022,

1023,

1025,

1012,

1026,

1013,

1027,

1020,

1018,

1017,

1034,

1021,

1015,

1028,

1029,

1030,

1019,

1033,

1031,

1032

}; // ID Академий с фейками на уровне кафедр

KPIWebDataContext kPiDataContext = new KPIWebDataContext();

List<ChartOneValue> item = (List<ChartOneValue>) ViewState["Items"];

int id = (from a in kPiDataContext.FirstLevelSubdivisionTable

where a.Name.Equals(item[Convert.ToInt32(e.PostBackValue)].name)

select a.FirstLevelSubdivisionTableID).FirstOrDefault();

if (!ExceptItems.Contains(Convert.ToInt32(id)))

{

Session["AcademyToDetailed"] = id.ToString();

Response.Redirect("~/Rector/RShowChartFaculty.aspx");

}

else

{

DisplayAlert("Невозможно детализировать");

}

}

catch (Exception)

{

// error

}

}

private void DisplayAlert(string message)

{

ClientScript.RegisterStartupScript(

this.GetType(),

Guid.NewGuid().ToString(),

string.Format("alert('{0}');",

message.Replace("'", @"\'").Replace("\n", "\\n").Replace("\r", "\\r")),

true);

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Filename: RShowChartDetailed.aspx.designer.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//------------------------------------------------------------------------------

// <автоматически создаваемое>

// Этот код создан программой.

//

// Изменения в этом файле могут привести к неправильной работе и будут потеряны в случае

// повторной генерации кода.

// </автоматически создаваемое>

//------------------------------------------------------------------------------

namespace KPIWeb.Rector {

public partial class RShowChartDetailed {

/// <summary>

/// top\_panel2 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Panel top\_panel2;

/// <summary>

/// GoBackButton элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Button GoBackButton;

/// <summary>

/// GoForwardButton элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Button GoForwardButton;

/// <summary>

/// Button2 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Button Button2;

/// <summary>

/// Button1 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Button Button1;

/// <summary>

/// Button6 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Button Button6;

/// <summary>

/// Chart1 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.DataVisualization.Charting.Chart Chart1;

/// <summary>

/// GridView1 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.GridView GridView1;

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Filename: RShowChartFaculty.aspx.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using System;

using System.Collections.Generic;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.DataVisualization.Charting;

using System.Web.UI.WebControls;

namespace KPIWeb.Rector

{

public partial class RShowChartFaculty : System.Web.UI.Page

{

public ChartOneValue GetCalculatedIndicator(int ReportID, IndicatorsTable Indicator, FirstLevelSubdivisionTable Academy, SecondLevelSubdivisionTable Faculty) // academyID == null && facultyID==null значит для всего КФУ

{

KPIWebDataContext kpiWebDataContext = new KPIWebDataContext();

float Planned\_Value = 0;

string Name\_ = "";

float Value\_ = 0;

#region plannedIndicator

PlannedIndicator plannedValue = (from a in kpiWebDataContext.PlannedIndicator

where a.FK\_IndicatorsTable == Indicator.IndicatorsTableID

&& a.Date > DateTime.Now

select a).OrderBy(x => x.Date).FirstOrDefault();

if (plannedValue != null)

{

Planned\_Value = (float)plannedValue.Value;

}

#endregion

#region Name

if ((Academy == null) && (Faculty == null))

{

if (Indicator.Measure != null)

{

if (Indicator.Measure.Length > 0)

{

Name\_ = Indicator.Name + " (" + Indicator.Measure + ")";

}

else

{

Name\_ = Indicator.Name;

}

}

else

{

Name\_ = Indicator.Name;

}

}

else if (Faculty != null)

{

Name\_ = Faculty.Name;

}

else if (Academy != null)

{

Name\_ = Academy.Name;

}

#endregion

#region

//ForRCalc.Struct mainStruct = mainStruct = new ForRCalc.Struct(1, 0, 0, 0, 0, "N");

CollectedIndicatorsForR collected = new CollectedIndicatorsForR();

if ((Academy == null) && (Faculty == null))

{

//mainStruct = new ForRCalc.Struct(1, 0, 0, 0, 0, "N");

collected = (from a in kpiWebDataContext.CollectedIndicatorsForR

where a.FK\_ReportArchiveTable == ReportID

&& a.FK\_IndicatorsTable == Indicator.IndicatorsTableID

&& a.FK\_FirstLevelSubdivisionTable == null

&& a.FK\_SecondLevelSubdivisionTable == null

select a).FirstOrDefault();

}

else if (Faculty != null)

{

//mainStruct = new ForRCalc.Struct(1, Faculty.FK\_FirstLevelSubdivisionTable, Faculty.SecondLevelSubdivisionTableID, 0, 0, "N");

collected = (from a in kpiWebDataContext.CollectedIndicatorsForR

where a.FK\_ReportArchiveTable == ReportID

&& a.FK\_IndicatorsTable == Indicator.IndicatorsTableID

&& a.FK\_FirstLevelSubdivisionTable == Faculty.FK\_FirstLevelSubdivisionTable

&& a.FK\_SecondLevelSubdivisionTable == Faculty.SecondLevelSubdivisionTableID

select a).FirstOrDefault();

}

else if (Academy != null)

{

//mainStruct = new ForRCalc.Struct(1, Academy.FirstLevelSubdivisionTableID, 0, 0, 0, "N");

collected = (from a in kpiWebDataContext.CollectedIndicatorsForR

where a.FK\_ReportArchiveTable == ReportID

&& a.FK\_IndicatorsTable == Indicator.IndicatorsTableID

&& a.FK\_FirstLevelSubdivisionTable == Academy.FirstLevelSubdivisionTableID

&& a.FK\_SecondLevelSubdivisionTable == null

select a).FirstOrDefault();

}

/\*

float tmp = ForRCalc.CalculatedForDB(ForRCalc.GetCalculatedWithParams(mainStruct, 0, Indicator.IndicatorsTableID, ReportID, 0));

if (tmp == (float)1E+20)

{

Value\_ = 0;

}

else

{

Value\_ = tmp;

}

\*/

if (collected != null) // Крашило тут мед академию

if (collected.Value == null)

{

Value\_ = 0;

}

else

{

Value\_ = (float)collected.Value;

}

else

{

// error

}

#endregion

ChartOneValue DataRowForChart = new ChartOneValue(Name\_, Value\_, Planned\_Value);

return DataRowForChart;

}

public ChartValueArray IndicatorsForAcademyFacultys(int IndicatorID, int AcademyID, int ReportID)

{

KPIWebDataContext kpiWebDataContext = new KPIWebDataContext();

IndicatorsTable Indicator = (from a in kpiWebDataContext.IndicatorsTable

where a.IndicatorsTableID == IndicatorID

select a).FirstOrDefault();

List<SecondLevelSubdivisionTable> FacultyList = (from a in kpiWebDataContext.SecondLevelSubdivisionTable

where a.Active == true && a.FK\_FirstLevelSubdivisionTable == AcademyID

select a).ToList();

ChartValueArray DataForChart = new ChartValueArray("Целевой показатель '" + Indicator.Name + "' в разрезе "+ (from b in kpiWebDataContext.FirstLevelSubdivisionTable where b.FirstLevelSubdivisionTableID == AcademyID select b.Name).FirstOrDefault());

foreach (SecondLevelSubdivisionTable CurrentFavulty in FacultyList)

{

FirstLevelSubdivisionTable Academy = (from a in kpiWebDataContext.FirstLevelSubdivisionTable

where a.FirstLevelSubdivisionTableID == CurrentFavulty.FK\_FirstLevelSubdivisionTable

select a).FirstOrDefault();

DataForChart.ChartValues.Add(GetCalculatedIndicator(1, Indicator, Academy, CurrentFavulty));

}

return DataForChart;

}

protected void Page\_Load(object sender, EventArgs e)

{

Serialization UserSer = (Serialization)Session["UserID"];

if (UserSer == null)

{

Response.Redirect("~/Default.aspx");

}

int userID = UserSer.Id;

KPIWebDataContext kPiDataContext = new KPIWebDataContext();

UsersTable userTable =

(from a in kPiDataContext.UsersTable where a.UsersTableID == userID select a).FirstOrDefault();

if (userTable.AccessLevel != 7)

{

Response.Redirect("~/Default.aspx");

}

String AcademyID = (string)Session["AcademyToDetailed"];

int academy = Convert.ToInt32(AcademyID);

String IndicatorID = (string)Session["IndicatorToDetailed"];

int indicator = Convert.ToInt32(IndicatorID);

ChartItems chartItems = new ChartItems();

ChartValueArray DataForChart = IndicatorsForAcademyFacultys(indicator,academy, 1);

// Формируем GridView

DataTable dataTable = new DataTable();

dataTable.Columns.Add(new DataColumn("IndicatorID", typeof(string)));

dataTable.Columns.Add(new DataColumn("Ratio", typeof(string)));

dataTable.Columns.Add(new DataColumn("IndicatorName", typeof(string)));

dataTable.Columns.Add(new DataColumn("IndicatorValue", typeof(string)));

#region график + чуток gridview

// Форматировать диаграмму

Chart1.BackColor = Color.White;

Chart1.BackSecondaryColor = Color.WhiteSmoke;

Chart1.BackGradientStyle = GradientStyle.DiagonalRight;

Chart1.BorderlineDashStyle = ChartDashStyle.Solid;

Chart1.BorderlineColor = Color.Black;

Chart1.BorderSkin.SkinStyle = BorderSkinStyle.Emboss;

// Форматировать область диаграммы

Chart1.ChartAreas[0].BackColor = Color.White;

// Добавить и форматировать заголовок

Chart1.Titles.Add(DataForChart.chartName);

Chart1.Titles[0].Font = new Font("Utopia", 16);

Chart1.Series.Add(new Series("Default")

{

ChartType = SeriesChartType.StackedBar,

Color = Color.SteelBlue

});

Chart1.ChartAreas[0].AxisX.IntervalAutoMode = IntervalAutoMode.VariableCount;

Chart1.ChartAreas[0].AxisY.IntervalAutoMode = IntervalAutoMode.VariableCount;

int ratio = 1;

var measure = (from ind in kPiDataContext.IndicatorsTable

where ind.IndicatorsTableID == indicator

select ind.Measure).FirstOrDefault().ToString();

foreach (ChartOneValue item in chartItems.SortReverse(DataForChart.ChartValues)) // сортировка для gridview FIFO

{

if (item.value == 0) continue;

DataRow dataRow = dataTable.NewRow();

dataRow["IndicatorID"] =

(from a in kPiDataContext.FirstLevelSubdivisionTable

where a.Name.Equals(item.name)

select a.FirstLevelSubdivisionTableID).FirstOrDefault();

dataRow["Ratio"] = ratio;

dataRow["IndicatorName"] = item.name;

dataRow["IndicatorValue"] = Math.Round(item.value, 3) + " " + measure ;

dataTable.Rows.Add(dataRow);

ratio++;

}

foreach (ChartOneValue item in chartItems.Sort(DataForChart.ChartValues)) // для chart FILO

{

if (item.value == 0) continue;

chartItems.AddChartItem(item.name, item.value);

}

// Привязать источник к диаграмме

Chart1.DataSource = chartItems.GetDataSource();

Chart1.Series[0].XValueMember = "Name";

Chart1.Series[0].YValueMembers = "Value";

Chart1.Series[0].Label = "#VALY" + " " + measure.Substring(0, 3);

Chart1.Series[0].ToolTip = "#VALX #VALY"+ " " + measure;

#endregion

GridView1.DataSource = dataTable;

GridView1.DataBind();

}

protected void Button2\_Click(object sender, EventArgs e)

{

Response.Redirect("~/Rector/RectorMain.aspx");

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Filename: RShowChartFaculty.aspx.designer.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//------------------------------------------------------------------------------

// <автоматически создаваемое>

// Этот код создан программой.

//

// Изменения в этом файле могут привести к неправильной работе и будут потеряны в случае

// повторной генерации кода.

// </автоматически создаваемое>

//------------------------------------------------------------------------------

namespace KPIWeb.Rector {

public partial class RShowChartFaculty {

/// <summary>

/// top\_panel2 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Panel top\_panel2;

/// <summary>

/// GoBackButton элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Button GoBackButton;

/// <summary>

/// GoForwardButton элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Button GoForwardButton;

/// <summary>

/// Button2 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Button Button2;

/// <summary>

/// Button1 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Button Button1;

/// <summary>

/// Button6 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Button Button6;

/// <summary>

/// Chart1 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.DataVisualization.Charting.Chart Chart1;

/// <summary>

/// GridView1 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.GridView GridView1;

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Filename: ViewDocument.aspx.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

using System.Data;

using System.IO;

namespace KPIWeb.Rector

{

public partial class ViewDocument : System.Web.UI.Page

{

protected void Page\_Load(object sender, EventArgs e)

{

Serialization UserSer = (Serialization)Session["UserID"];

if (UserSer == null)

{

Response.Redirect("~/Default.aspx");

}

int userID = UserSer.Id;

KPIWebDataContext kPiDataContext = new KPIWebDataContext();

var login =

(from a in kPiDataContext.UsersTable

where a.UsersTableID == userID

select a.Email).FirstOrDefault();

LogHandler.LogWriter.WriteLog(LogCategory.INFO, "0VD0: Prorector " + login + " pereshel na stranicy normativnih documentov");

UsersTable userTable =

(from a in kPiDataContext.UsersTable where a.UsersTableID == userID select a).FirstOrDefault();

if (!IsPostBack)

{

//KPIWebDataContext kPiDataContext = new KPIWebDataContext();

List<DocumentTable> docs = (from a in kPiDataContext.DocumentTable where a.Active == true select a).ToList();

DataTable dataTable = new DataTable();

dataTable.Columns.Add(new DataColumn("DocumentName", typeof(string)));

dataTable.Columns.Add(new DataColumn("DocumentLink", typeof(string)));

GridView1.DataSource = docs;

GridView1.DataBind();

}

}

protected void DeleteButtonClick(object sender, EventArgs e)

{

Button button = (Button)sender;

string s = button.CommandArgument.ToString();

string n = Server.MapPath(@"~/Rector/docs/"+ s);

if (File.Exists(n))

{

//if (Page.ClientScript.RegisterClientScriptBlock(typeof(Page), "Script", "document.location = 'docs/" + s + "';" ))

// {

Page.ClientScript.RegisterClientScriptBlock(typeof(Page), "Script", "window.open('docs/" + s + "');", true);

//ClientScript.RegisterStartupScript(this.GetType(), "window.open", "window.open('~ViewDocument.aspx')", true);

}

else

{

Page.ClientScript.RegisterClientScriptBlock(typeof(Page), "Script", "alert('Документ не найден');", true);

}

}

protected void Button2\_Click(object sender, EventArgs e)

{

Response.Redirect("~/Rector/RectorMain.aspx");

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Filename: ViewDocument.aspx.designer.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//------------------------------------------------------------------------------

// <auto-generated>

// This code was generated by a tool.

//

// Changes to this file may cause incorrect behavior and will be lost if

// the code is regenerated.

// </auto-generated>

//------------------------------------------------------------------------------

namespace KPIWeb.Rector {

public partial class ViewDocument {

/// <summary>

/// top\_panel2 control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.Panel top\_panel2;

/// <summary>

/// GoBackButton control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.Button GoBackButton;

/// <summary>

/// GoForwardButton control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.Button GoForwardButton;

/// <summary>

/// Button2 control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.Button Button2;

/// <summary>

/// Button5 control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.Button Button5;

/// <summary>

/// GridView1 control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.GridView GridView1;

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Filename: ChooseReport.aspx.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using System;

using System.Collections.Generic;

using System.Configuration;

using System.Data;

using System.Linq;

using System.Web;

using System.Web.DynamicData;

using System.Web.UI;

using System.Web.UI.WebControls;

namespace KPIWeb.Reports

{

public partial class ChooseReport : System.Web.UI.Page

{

protected void Page\_Load(object sender, EventArgs e)

{

Serialization UserSer = (Serialization)Session["UserID"];

if (UserSer == null)

{

Response.Redirect("~/Default.aspx");

}

int userID = UserSer.Id;

ViewState["LocalUserID"] = userID;

KPIWebDataContext kpiWebDataContext = new KPIWebDataContext();

UsersTable userTable =

(from a in kpiWebDataContext.UsersTable where a.UsersTableID == userID select a).FirstOrDefault();

if (userTable.AccessLevel != 0)

{

Response.Redirect("~/Default.aspx");

}

//////////////////////////////////////////////////////////////////////////

if (!Page.IsPostBack)

{

#region get user reports

List<ReportArchiveTable> reportsArchiveTablesTable = (

from a in kpiWebDataContext.UsersTable

join b in kpiWebDataContext.FirstLevelSubdivisionTable

on a.FK\_FirstLevelSubdivisionTable equals b.FirstLevelSubdivisionTableID

join c in kpiWebDataContext.ReportArchiveAndLevelMappingTable

on b.FirstLevelSubdivisionTableID equals c.FK\_FirstLevelSubmisionTableId

join d in kpiWebDataContext.ReportArchiveTable

on c.FK\_ReportArchiveTableId equals d.ReportArchiveTableID

where a.UsersTableID == UserSer.Id

&& a.Active == true

&& b.Active == true

&& c.Active == true

&& d.Active == true

&& d.StartDateTime < DateTime.Now

&& d.EndDateTime > DateTime.Now

select d).ToList();

///тут мы получили список активных отччетов пользователя

/// пользователь привязан к таблице первого подразделения

/// таблица первого подразделения привязана к таблице отчётов(через таблицу связи)

/// на данный момент отчёт можно назначать только первому подразделению!!!

///

#endregion

UsersTable user = (from a in kpiWebDataContext.UsersTable

where a.UsersTableID == userID

select a).FirstOrDefault();

int l\_0 = user.FK\_ZeroLevelSubdivisionTable == null ? 0 : (int)user.FK\_ZeroLevelSubdivisionTable;

int l\_1 = user.FK\_FirstLevelSubdivisionTable == null ? 0 : (int)user.FK\_FirstLevelSubdivisionTable;

int l\_2 = user.FK\_SecondLevelSubdivisionTable == null ? 0 : (int)user.FK\_SecondLevelSubdivisionTable;

int l\_3 = user.FK\_ThirdLevelSubdivisionTable == null ? 0 : (int)user.FK\_ThirdLevelSubdivisionTable;

int l\_4 = user.FK\_FourthLevelSubdivisionTable == null ? 0 : (int)user.FK\_FourthLevelSubdivisionTable;

int l\_5 = user.FK\_FifthLevelSubdivisionTable == null ? 0 : (int)user.FK\_FifthLevelSubdivisionTable;

int userLevel = 5;

userLevel = l\_5 == 0 ? 4 : userLevel;

userLevel = l\_4 == 0 ? 3 : userLevel;

userLevel = l\_3 == 0 ? 2 : userLevel;

userLevel = l\_2 == 0 ? 1 : userLevel;

userLevel = l\_1 == 0 ? 0 : userLevel;

userLevel = l\_0 == 0 ? -1 : userLevel;

DataTable dataTable = new DataTable();

dataTable.Columns.Add(new DataColumn("ReportArchiveID", typeof(string)));

dataTable.Columns.Add(new DataColumn("ReportName", typeof(string)));

dataTable.Columns.Add(new DataColumn("StartDate", typeof(string)));

dataTable.Columns.Add(new DataColumn("EndDate", typeof(string)));

dataTable.Columns.Add(new DataColumn("Status", typeof(string)));

List<int> StatenList = new List<int>();

foreach (ReportArchiveTable ReportRow in reportsArchiveTablesTable)

{

#region

int can\_view =

(from a in kpiWebDataContext.ReportArchiveAndBasicParametrsMappingTable

join b in kpiWebDataContext.BasicParametersTable

on a.FK\_BasicParametrsTable equals b.BasicParametersTableID

join c in kpiWebDataContext.BasicParametrsAndUsersMapping

on b.BasicParametersTableID equals c.FK\_ParametrsTable

join d in kpiWebDataContext.BasicParametrAdditional

on b.BasicParametersTableID equals d.BasicParametrAdditionalID

where

a.FK\_ReportArchiveTable == ReportRow.ReportArchiveTableID //из нужного отчёта

&& c.FK\_UsersTable == userID // свяный с пользователем

//&& d.SubvisionLevel == 3 //нужный уровень заполняющего

&& a.Active == true // запись в таблице связей показателя и отчёта активна

&& c.CanView == true

&& c.Active == true // запись в таблице связей показателя и пользователей активна

&& d.Calculated == false

// этот показатель нужно вводить а не считать

select b).ToList().Count;

if (can\_view>0)

{

}

else

{

continue;

}

DataRow dataRow = dataTable.NewRow();

dataRow["ReportArchiveID"] = ReportRow.ReportArchiveTableID.ToString();

dataRow["ReportName"] = ReportRow.Name;

dataRow["StartDate"] = ReportRow.StartDateTime.ToString().Split(' ')[0];//только дата// время обрезается сплитом

dataRow["EndDate"] = ReportRow.EndDateTime.ToString().Split(' ')[0];

//нужно определить статус данных

//status=0 данных нет

//status=1 данные вернули на доработку

//status=2 данные есть

//status=3 данные отправлены на верификацию

//status=4 данные верифицированы первым первым уровнем(кафедрой)

// возьмем первый попавшийся заполненный показатель

CollectedBasicParametersTable ColTemp =

(from a in kpiWebDataContext.CollectedBasicParametersTable

join b in kpiWebDataContext.UsersTable

on a.FK\_UsersTable equals b.UsersTableID

join c in kpiWebDataContext.BasicParametrAdditional

on a.FK\_BasicParametersTable equals c.BasicParametrAdditionalID

join d in kpiWebDataContext.BasicParametrsAndUsersMapping //

on a.FK\_BasicParametersTable equals d.FK\_ParametrsTable //

where

a.FK\_ReportArchiveTable == ReportRow.ReportArchiveTableID

&& ((b.FK\_ZeroLevelSubdivisionTable == l\_0) || (l\_0 == 0))

&& ((b.FK\_FirstLevelSubdivisionTable == l\_1) || (l\_1 == 0))

&& ((b.FK\_SecondLevelSubdivisionTable == l\_2) || (l\_2 == 0))

&& ((b.FK\_ThirdLevelSubdivisionTable == l\_3) || (l\_3 == 0))

&& ((b.FK\_FourthLevelSubdivisionTable == l\_4) || (l\_4 == 0))

&& ((b.FK\_FifthLevelSubdivisionTable == l\_5) || (l\_5 == 0))

&& ((d.CanConfirm == true) || (d.CanEdit == true)) //

&& (d.FK\_UsersTable == userID) //

&& a.Active == true

&& b.Active == true

&& c.Calculated == false

select a).FirstOrDefault();

string status = "Нет данных";

int Statusn = 0;

if (ColTemp == null)

{

Statusn = 0;

}

else if (ColTemp.Status == null)

{

Statusn = 0;

}

else

{

Statusn = (int) ColTemp.Status;

}

if (Statusn == 4)

{

status = "Данные утверждены";

}

else if (Statusn == 3)

{

status = "Данные ожидают утверждения";

}

else if (Statusn == 2)

{

status = "Данные в процессе заполнения";

}

else if (Statusn == 1)

{

status = "Данные возвращены на доработку";

}

else if (Statusn == 0)

{

status = "Данные в процессе заполнения";

}

else

{

//error

}

StatenList.Add(Statusn);

dataRow["Status"] = status;

dataTable.Rows.Add(dataRow);

#endregion

}

GridView1.DataSource = dataTable;

GridView1.DataBind();

for (int i = 0; i < GridView1.Rows.Count; i++)

{

Button btnEdit = GridView1.Rows[i].FindControl("ButtonEditReport") as Button;

Button btnView = GridView1.Rows[i].FindControl("ButtonViewReport") as Button;

Button btnConfirm = GridView1.Rows[i].FindControl("ButtonConfirmReport") as Button;

//status=0 данных нет

//status=1 данные вернули на доработку

//status=2 данные есть

//status=3 данные отправлены на верификацию

//status=4 данные верифицированы первым первым уровнем(кафедрой)

int ConfButton = 0;

int ViewButton = 0;

int EditButton = 0;

if ((btnEdit != null) && (btnView != null) && (btnConfirm != null))

{

if ((StatenList[i] == 0)||(StatenList[i] == 1)||(StatenList[i] == 2))

{

ConfButton++;

}

else if (StatenList[i] == 3)

{

EditButton++;

}

else

{

ConfButton++;

EditButton++;

}

#region

int ReportArchiveID = Convert.ToInt32(btnEdit.CommandArgument);

if (userLevel == 3)

{

#region

int kaf\_edit =

(from a in kpiWebDataContext.ReportArchiveAndBasicParametrsMappingTable

join b in kpiWebDataContext.BasicParametersTable

on a.FK\_BasicParametrsTable equals b.BasicParametersTableID

join c in kpiWebDataContext.BasicParametrsAndUsersMapping

on b.BasicParametersTableID equals c.FK\_ParametrsTable

join d in kpiWebDataContext.BasicParametrAdditional

on b.BasicParametersTableID equals d.BasicParametrAdditionalID

where

a.FK\_ReportArchiveTable == ReportArchiveID //из нужного отчёта

&& c.FK\_UsersTable == userID // свяный с пользователем

&& d.SubvisionLevel == 3 //нужный уровень заполняющего

&& a.Active == true // запись в таблице связей показателя и отчёта активна

&& c.CanEdit == true

&& c.Active == true // запись в таблице связей показателя и пользователей активна

&& d.Calculated == false

// этот показатель нужно вводить а не считать

select b).ToList().Count;

int kaf\_view =

(from a in kpiWebDataContext.ReportArchiveAndBasicParametrsMappingTable

join b in kpiWebDataContext.BasicParametersTable

on a.FK\_BasicParametrsTable equals b.BasicParametersTableID

join c in kpiWebDataContext.BasicParametrsAndUsersMapping

on b.BasicParametersTableID equals c.FK\_ParametrsTable

join d in kpiWebDataContext.BasicParametrAdditional

on b.BasicParametersTableID equals d.BasicParametrAdditionalID

where

a.FK\_ReportArchiveTable == ReportArchiveID //из нужного отчёта

&& c.FK\_UsersTable == userID // свяный с пользователем

&& d.SubvisionLevel == 3 //нужный уровень заполняющего

&& a.Active == true // запись в таблице связей показателя и отчёта активна

&& c.CanView == true

&& c.Active == true // запись в таблице связей показателя и пользователей активна

&& d.Calculated == false

// этот показатель нужно вводить а не считать

select b).ToList().Count;

int kaf\_conf =

(from a in kpiWebDataContext.ReportArchiveAndBasicParametrsMappingTable

join b in kpiWebDataContext.BasicParametersTable

on a.FK\_BasicParametrsTable equals b.BasicParametersTableID

join c in kpiWebDataContext.BasicParametrsAndUsersMapping

on b.BasicParametersTableID equals c.FK\_ParametrsTable

join d in kpiWebDataContext.BasicParametrAdditional

on b.BasicParametersTableID equals d.BasicParametrAdditionalID

where

a.FK\_ReportArchiveTable == ReportArchiveID //из нужного отчёта

&& c.FK\_UsersTable == userID // свяный с пользователем

&& d.SubvisionLevel == 3 //нужный уровень заполняющего

&& a.Active == true // запись в таблице связей показателя и отчёта активна

&& c.CanConfirm == true

&& c.Active == true // запись в таблице связей показателя и пользователей активна

&& d.Calculated == false

// этот показатель нужно вводить а не считать

select b).ToList().Count;

#endregion

#region

int specEdit =

(from a in kpiWebDataContext.ReportArchiveAndBasicParametrsMappingTable

join b in kpiWebDataContext.BasicParametersTable

on a.FK\_BasicParametrsTable equals b.BasicParametersTableID

join c in kpiWebDataContext.BasicParametrsAndUsersMapping

on b.BasicParametersTableID equals c.FK\_ParametrsTable

join d in kpiWebDataContext.BasicParametrAdditional

on b.BasicParametersTableID equals d.BasicParametrAdditionalID

where a.FK\_ReportArchiveTable == ReportArchiveID //для отчёта

&& d.SubvisionLevel == 4 // для уровня заполняющего

&& d.Calculated == false //только вводимые параметры

&& c.FK\_UsersTable == userID // связаннаые с пользователем

&& a.Active == true

&& c.CanEdit == true

&& c.Active == true

select b).ToList().Count;

int specView =

(from a in kpiWebDataContext.ReportArchiveAndBasicParametrsMappingTable

join b in kpiWebDataContext.BasicParametersTable

on a.FK\_BasicParametrsTable equals b.BasicParametersTableID

join c in kpiWebDataContext.BasicParametrsAndUsersMapping

on b.BasicParametersTableID equals c.FK\_ParametrsTable

join d in kpiWebDataContext.BasicParametrAdditional

on b.BasicParametersTableID equals d.BasicParametrAdditionalID

where a.FK\_ReportArchiveTable == ReportArchiveID //для отчёта

&& d.SubvisionLevel == 4 // для уровня заполняющего

&& d.Calculated == false //только вводимые параметры

&& c.FK\_UsersTable == userID // связаннаые с пользователем

&& a.Active == true

&& c.CanView == true

&& c.Active == true

select b).ToList().Count;

int specConf =

(from a in kpiWebDataContext.ReportArchiveAndBasicParametrsMappingTable

join b in kpiWebDataContext.BasicParametersTable

on a.FK\_BasicParametrsTable equals b.BasicParametersTableID

join c in kpiWebDataContext.BasicParametrsAndUsersMapping

on b.BasicParametersTableID equals c.FK\_ParametrsTable

join d in kpiWebDataContext.BasicParametrAdditional

on b.BasicParametersTableID equals d.BasicParametrAdditionalID

where a.FK\_ReportArchiveTable == ReportArchiveID //для отчёта

&& d.SubvisionLevel == 4 // для уровня заполняющего

&& d.Calculated == false //только вводимые параметры

&& c.FK\_UsersTable == userID // связаннаые с пользователем

&& a.Active == true

&& c.CanConfirm == true

&& c.Active == true

select b).ToList().Count;

#endregion

if (!((kaf\_edit + specEdit)>0))

{

EditButton++;

}

if (!((kaf\_conf + specConf)>0))

{

ConfButton++;

}

if (!((kaf\_view + specView) > 0))

{

ViewButton++;

}

}

else

{

#region

int edit = 0;

int view = 0;

int conf = 0;

edit = (from a in kpiWebDataContext.ReportArchiveAndBasicParametrsMappingTable

join b in kpiWebDataContext.BasicParametersTable

on a.FK\_BasicParametrsTable equals b.BasicParametersTableID

join c in kpiWebDataContext.BasicParametrsAndUsersMapping

on b.BasicParametersTableID equals c.FK\_ParametrsTable

join d in kpiWebDataContext.BasicParametrAdditional

on b.BasicParametersTableID equals d.BasicParametrAdditionalID

where

a.FK\_ReportArchiveTable == ReportArchiveID //из нужного отчёта

&& c.FK\_UsersTable == userID // свяный с пользователем

&& d.SubvisionLevel == userLevel //нужный уровень заполняющего

&& a.Active == true // запись в таблице связей показателя и отчёта активна

&& c.CanEdit == true

&& c.Active == true // запись в таблице связей показателя и пользователей активна

&& d.Calculated == false

select b).ToList().Count;

view = (from a in kpiWebDataContext.ReportArchiveAndBasicParametrsMappingTable

join b in kpiWebDataContext.BasicParametersTable

on a.FK\_BasicParametrsTable equals b.BasicParametersTableID

join c in kpiWebDataContext.BasicParametrsAndUsersMapping

on b.BasicParametersTableID equals c.FK\_ParametrsTable

join d in kpiWebDataContext.BasicParametrAdditional

on b.BasicParametersTableID equals d.BasicParametrAdditionalID

where

a.FK\_ReportArchiveTable == ReportArchiveID //из нужного отчёта

&& c.FK\_UsersTable == userID // свяный с пользователем

&& d.SubvisionLevel == userLevel //нужный уровень заполняющего

&& a.Active == true // запись в таблице связей показателя и отчёта активна

&& c.CanView == true

&& c.Active == true // запись в таблице связей показателя и пользователей активна

&& d.Calculated == false

select b).ToList().Count;

conf = (from a in kpiWebDataContext.ReportArchiveAndBasicParametrsMappingTable

join b in kpiWebDataContext.BasicParametersTable

on a.FK\_BasicParametrsTable equals b.BasicParametersTableID

join c in kpiWebDataContext.BasicParametrsAndUsersMapping

on b.BasicParametersTableID equals c.FK\_ParametrsTable

join d in kpiWebDataContext.BasicParametrAdditional

on b.BasicParametersTableID equals d.BasicParametrAdditionalID

where

a.FK\_ReportArchiveTable == ReportArchiveID //из нужного отчёта

&& c.FK\_UsersTable == userID // свяный с пользователем

&& d.SubvisionLevel == userLevel //нужный уровень заполняющего

&& a.Active == true // запись в таблице связей показателя и отчёта активна

&& c.CanConfirm == true

&& c.Active == true // запись в таблице связей показателя и пользователей активна

&& d.Calculated == false

select b).ToList().Count;

#endregion

if (!(edit > 0))

{

EditButton++;

}

if (!(conf > 0))

{

ConfButton++;

}

if (!(view > 0))

{

ViewButton++;

}

}

#endregion

/////////////////////////////////////////////////////////

btnConfirm.Enabled = ConfButton>0?false:true;

btnEdit.Enabled = EditButton>0?false:true;

btnView.Enabled = ViewButton > 0 ? false : true;

}

}

///вывели все отчёты с параметрами в гридвью

if (GridView1.Rows.Count == 0)

{

var name =

(from tr in kpiWebDataContext.UsersTable

where tr.UsersTableID == UserSer.Id

select tr.FirstLevelSubdivisionTable.Name).FirstOrDefault();

Label1.Visible = true;

Label2.Visible = true;

if (name != null) Label1.Text = "В данный момент для вашего подразделения (" + name + ") отсутствуют активные отчеты. Мы обязательно уведомим Вас о начале новой отчетной кампании.";

else Label1.Text = "В данный момент для вашего подразделения отсутствуют активные отчеты. Мы обязательно уведомим Вас о начале новой отчетной кампании.";

Label2.Text = "С уважением администрация ИАС \"КФУ-Программа развития\"";

}

}

}

protected void ButtonEditClick(object sender, EventArgs e)

{

Button button = (Button)sender;

{

Serialization UserSer = (Serialization)Session["UserID"];

if (UserSer == null)

{

Response.Redirect("~/Default.aspx");

}

int userID = UserSer.Id;

KPIWebDataContext kpiWebDataContext = new KPIWebDataContext();

UsersTable userTable =

(from a in kpiWebDataContext.UsersTable where a.UsersTableID == userID select a).FirstOrDefault();

ThirdLevelParametrs thirdParam = (from a in kpiWebDataContext.ThirdLevelParametrs

where

a.ThirdLevelParametrsID == userTable.FK\_ThirdLevelSubdivisionTable

&& a.Active == true

select a).FirstOrDefault();

Serialization paramSerialization = new Serialization(button.CommandArgument.ToString());

Session["ReportArchiveID"] = paramSerialization; // запомнили в сессии номер отчёта

Serialization modeSer = new Serialization(0,null,null);

Session["mode"] = modeSer;

var login =

(from a in kpiWebDataContext.UsersTable

where a.UsersTableID == (int)ViewState["LocalUserID"]

select a.Email).FirstOrDefault();

LogHandler.LogWriter.WriteLog(LogCategory.INFO, "0CR0: Пользователь " + login + " зашел на страницу редактирования отчета с ID = " + paramSerialization.ReportStr);

if (thirdParam != null)

{

if (thirdParam.CanGraduate)

{

FourthLevelSubdivisionTable fourth =

(from a in kpiWebDataContext.FourthLevelSubdivisionTable

where a.Active == true

&& a.FK\_ThirdLevelSubdivisionTable == thirdParam.ThirdLevelParametrsID

select a).FirstOrDefault();

if (fourth != null)

{

Response.Redirect("~/Reports\_/Parametrs.aspx");

}

else

{

Response.Redirect("~/Reports\_/FillingTheReport.aspx");

}

}

else

{

Response.Redirect("~/Reports\_/FillingTheReport.aspx");

}

}

else

{

Response.Redirect("~/Reports\_/FillingTheReport.aspx");

}

}

}

protected void ButtonViewClick(object sender, EventArgs e)

{

Button button = (Button)sender;

{

Serialization paramSerialization = new Serialization(button.CommandArgument.ToString());

Session["ReportArchiveID"] = paramSerialization; // запомнили в сессии номер отчёта

Serialization modeSer = new Serialization(1, null, null);

Session["mode"] = modeSer;

KPIWebDataContext kpiWebDataContext = new KPIWebDataContext();

var login =

(from a in kpiWebDataContext.UsersTable

where a.UsersTableID == (int)ViewState["LocalUserID"]

select a.Email).FirstOrDefault();

LogHandler.LogWriter.WriteLog(LogCategory.INFO, "0CR1: Пользователь " + login + " зашел на страницу просмотра отчета с ID = " + paramSerialization.ReportStr);

Response.Redirect("~/Reports\_/FillingTheReport.aspx");

}

}

protected void ButtonConfirmClick(object sender, EventArgs e)

{

Button button = (Button)sender;

{

Serialization paramSerialization = new Serialization(button.CommandArgument.ToString());

Session["ReportArchiveID"] = paramSerialization; // запомнили в сессии номер отчёта

Serialization modeSer = new Serialization(2, null, null);

Session["mode"] = modeSer;

KPIWebDataContext kpiWebDataContext = new KPIWebDataContext();

var login =

(from a in kpiWebDataContext.UsersTable

where a.UsersTableID == (int)ViewState["LocalUserID"]

select a.Email).FirstOrDefault();

LogHandler.LogWriter.WriteLog(LogCategory.INFO, "0CR2: Пользователь " + login + " зашел на страницу просмотра и подтверждения отчета с ID = " + paramSerialization.ReportStr);

Response.Redirect("~/Reports\_/FillingTheReport.aspx");

}

}

protected void GridView1\_RowDataBound(object sender, GridViewRowEventArgs e)

{

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Filename: ChooseReport.aspx.designer.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//------------------------------------------------------------------------------

// <auto-generated>

// This code was generated by a tool.

//

// Changes to this file may cause incorrect behavior and will be lost if

// the code is regenerated.

// </auto-generated>

//------------------------------------------------------------------------------

namespace KPIWeb.Reports {

public partial class ChooseReport {

/// <summary>

/// Label1 control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.Label Label1;

/// <summary>

/// Label2 control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.Label Label2;

/// <summary>

/// GridView1 control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.GridView GridView1;

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Filename: EditReport.aspx.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using System;

using System.Collections.Generic;

using System.Configuration;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

using System.Data;

using System.Text;

using System.Threading;

using System.Web.WebPages;

namespace KPIWeb.Reports

{

public partial class EditReport : System.Web.UI.Page

{

UsersTable user;

protected void FillGridVIews(int reportID)

{

KPIWebDataContext kPiDataContext = new KPIWebDataContext();

///////////////////////////////////////////////////////////////////////////////////////////////////////

List<IndicatorsTable> indicatorTable =

(from item in kPiDataContext.IndicatorsTable where item.Active == true select item).OrderBy(c => c.SortID).ToList();

DataTable dataTableIndicator = new DataTable();

dataTableIndicator.Columns.Add(new DataColumn("IndicatorID", typeof(string)));

dataTableIndicator.Columns.Add(new DataColumn("IndicatorName", typeof(string)));

dataTableIndicator.Columns.Add(new DataColumn("IndicatorCheckBox", typeof(bool)));

foreach (IndicatorsTable indicator in indicatorTable)

{

DataRow dataRow = dataTableIndicator.NewRow();

dataRow["IndicatorID"] = indicator.IndicatorsTableID.ToString();

dataRow["IndicatorName"] = indicator.Name;

dataRow["IndicatorCheckBox"] = ((from a in kPiDataContext.ReportArchiveAndIndicatorsMappingTable

where a.Active == true

&& a.FK\_IndicatorsTable == indicator.IndicatorsTableID

&& a.FK\_ReportArchiveTable == reportID

select a).Count() > 0) ? true : false;

dataTableIndicator.Rows.Add(dataRow);

}

IndicatorsTable.DataSource = dataTableIndicator;

IndicatorsTable.DataBind();

////////////////////////////////////////////////////////////////////////////////////////////////////////

List<CalculatedParametrs> calcParams =

(from item in kPiDataContext.CalculatedParametrs where item.Active == true select item).ToList();

DataTable dataTableCalc = new DataTable();

dataTableCalc.Columns.Add(new DataColumn("CalculatedParametrsID", typeof(string)));

dataTableCalc.Columns.Add(new DataColumn("CalculatedParametrsName", typeof(string)));

dataTableCalc.Columns.Add(new DataColumn("CalculatedParametrsCheckBox", typeof(bool)));

foreach (CalculatedParametrs calcParam in calcParams)

{

DataRow dataRow = dataTableCalc.NewRow();

dataRow["CalculatedParametrsID"] = calcParam.CalculatedParametrsID.ToString();

dataRow["CalculatedParametrsName"] = calcParam.Name;

dataRow["CalculatedParametrsCheckBox"] = ((from a in kPiDataContext.ReportArchiveAndCalculatedParametrsMappingTable

where a.Active == true

&& a.FK\_CalculatedParametrsTable == calcParam.CalculatedParametrsID

&& a.FK\_ReportArchiveTable == reportID

select a).Count() > 0) ? true : false;

dataTableCalc.Rows.Add(dataRow);

}

CalculatedParametrsTable.DataSource = dataTableCalc;

CalculatedParametrsTable.DataBind();

////////////////////////////////////////////////////////////////////////////////////////////////////////

List<BasicParametersTable> basicParams =

(from item in kPiDataContext.BasicParametersTable where item.Active == true select item).ToList();

DataTable dataTableBasic = new DataTable();

dataTableBasic.Columns.Add(new DataColumn("BasicParametrsID", typeof(string)));

dataTableBasic.Columns.Add(new DataColumn("BasicParametrsName", typeof(string)));

dataTableBasic.Columns.Add(new DataColumn("BasicParametrsCheckBox", typeof(bool)));

foreach (BasicParametersTable basic in basicParams)

{

DataRow dataRow = dataTableBasic.NewRow();

dataRow["BasicParametrsID"] = basic.BasicParametersTableID.ToString();

dataRow["BasicParametrsName"] = basic.Name;

dataRow["BasicParametrsCheckBox"] = ((from a in kPiDataContext.ReportArchiveAndBasicParametrsMappingTable

where a.Active == true

&& a.FK\_BasicParametrsTable == basic.BasicParametersTableID

&& a.FK\_ReportArchiveTable == reportID

select a).Count() > 0) ? true : false;

dataTableBasic.Rows.Add(dataRow);

}

BasicParametrsTable.DataSource = dataTableBasic;

BasicParametrsTable.DataBind();

///////////////////////////////////////////////////////////////////////////////////////////////////

ViewState["BasicDataTable"] = dataTableBasic;

ViewState["CalculateDataTable"] = dataTableCalc;

ViewState["IndicatorDataTable"] = dataTableIndicator;

/////////////////////////////////////////////////////////////////////

for (int i = 0; i < dataTableBasic.Rows.Count; i++)

{

CheckBox chekBox = (CheckBox)BasicParametrsTable.Rows[i].FindControl("BasicParametrsCheckBox");

if (chekBox.Checked == true)

{

chekBox.Enabled = false;

}

}

for (int i = 0; i < dataTableCalc.Rows.Count; i++)

{

CheckBox chekBox = (CheckBox)CalculatedParametrsTable.Rows[i].FindControl("CalculatedParametrsCheckBox");

if (chekBox.Checked == true)

{

chekBox.Enabled = false;

}

}

for (int i = 0; i < dataTableIndicator.Rows.Count; i++)

{

CheckBox chekBox = (CheckBox)IndicatorsTable.Rows[i].FindControl("IndicatorCheckBox");

if (chekBox.Checked == true)

{

chekBox.Enabled = false;

}

}

}

protected List<int> getCalcByIndicator(List<int> indArr)

{

List<int> calcListTemp = new List<int>();

StringBuilder AllInOne = new StringBuilder();

KPIWebDataContext kpiWebDataContext = new KPIWebDataContext();

foreach (int tmp in indArr)

{

IndicatorsTable indTable = (from a in kpiWebDataContext.IndicatorsTable

where a.IndicatorsTableID == tmp

select a).FirstOrDefault();

AllInOne.Append(indTable.Formula + "\*");

}

string[] abbArr = AllInOne.ToString().Split('/', '^', '+', '-', '(', ')', '\*', ' ', '\n', '\r');

string strArr = "";

foreach (string str in abbArr)

{

if ((str != null) && (str != " ") && (!str.IsEmpty()))

{

if (!str.IsFloat())

{

int tmpp = Convert.ToInt32((from a in kpiWebDataContext.CalculatedParametrs

where a.AbbreviationEN == str

select a.CalculatedParametrsID).FirstOrDefault());

if (tmpp == 0)

{

//ERROR//нужно записать в лог str// это аббревиатура

}

else

{

calcListTemp.Add(tmpp);

}

}

}

}

return calcListTemp;

}

protected List<int> getBasicByCalc(List<int> calcArr)

{

List<int> basicListTemp = new List<int>();

StringBuilder AllInOne = new StringBuilder();

KPIWebDataContext kpiWebDataContext = new KPIWebDataContext();

foreach (int tmp in calcArr)

{

CalculatedParametrs calcTable = (from a in kpiWebDataContext.CalculatedParametrs

where a.CalculatedParametrsID == tmp

select a).FirstOrDefault();

AllInOne.Append(calcTable.Formula + "\*");

}

string[] abbArr = AllInOne.ToString().Split('/', '^', '+', '-', '(', ')', '\*', ' ','\n','\r');

string strArr = "";

foreach (string str in abbArr)

{

if ((str != null) && (str != " ") && (!str.IsEmpty()))

{

if (!str.IsFloat())

{

int tmpp = Convert.ToInt32((from a in kpiWebDataContext.BasicParametersTable

where a.AbbreviationEN == str

select a.BasicParametersTableID).FirstOrDefault());

if (tmpp==0)

{

//ERROR//нужно записать в лог str

}

else

{

basicListTemp.Add(tmpp);

}

}

}

}

return basicListTemp;

}

protected void Page\_Load(object sender, EventArgs e)

{

Serialization UserSer = (Serialization)Session["UserID"];

if (UserSer == null)

{

Response.Redirect("~/Default.aspx");

}

int userID = UserSer.Id;

KPIWebDataContext kPiDataContext = new KPIWebDataContext();

UsersTable userTable =

(from a in kPiDataContext.UsersTable where a.UsersTableID == userID select a).FirstOrDefault();

if ((userTable.AccessLevel != 10)&&(userTable.AccessLevel != 9))

{

Response.Redirect("~/Default.aspx");

}

////////////////////////////////////////////////////////////////////////////

Serialization ReportId = (Serialization)Session["ReportArchiveTableID"];

if (!Page.IsPostBack)

{

if (ReportId.ReportArchiveID != 0)

{

int reportArchiveTableID = ReportId.ReportArchiveID;

KPIWebDataContext KPIWebDataContext = new KPIWebDataContext();

ReportArchiveTable ReportArchiveTable = (from item in KPIWebDataContext.ReportArchiveTable

where item.ReportArchiveTableID == reportArchiveTableID

select item).FirstOrDefault();

if (ReportArchiveTable != null)

{

CheckBoxActive.Checked = ReportArchiveTable.Active;

CheckBoxCalculeted.Checked = ReportArchiveTable.Calculeted;

CheckBoxSent.Checked = ReportArchiveTable.Sent;

CheckBoxRecipientConfirmed.Checked = ReportArchiveTable.RecipientConfirmed;

TextBoxName.Text = ReportArchiveTable.Name;

if (ReportArchiveTable.StartDateTime != null)

CalendarStartDateTime.SelectedDate = (DateTime)ReportArchiveTable.StartDateTime;

if (ReportArchiveTable.EndDateTime != null)

CalendarEndDateTime.SelectedDate = (DateTime)ReportArchiveTable.EndDateTime;

if (ReportArchiveTable.DateToSend != null)

CalendarDateToSend.SelectedDate = (DateTime)ReportArchiveTable.DateToSend;

if (ReportArchiveTable.SentDateTime != null)

CalendarSentDateTime.SelectedDate = (DateTime)ReportArchiveTable.SentDateTime;

if (ReportArchiveTable.RecivedDateTime != null)

CalendarReportRecived.SelectedDate = (DateTime)ReportArchiveTable.RecivedDateTime;

if (ReportArchiveTable.ConfirmEndDay != null)

CalendarConfirmEndDay.SelectedDate = (DateTime)ReportArchiveTable.ConfirmEndDay;

if (ReportArchiveTable.DaysBeforeToCalcForRector != null)

DaysBeforeToCalcForRector.Text = ReportArchiveTable.DaysBeforeToCalcForRector.ToString();

}

////заполнили поля

List<FirstLevelSubdivisionTable> academies =

(from item in KPIWebDataContext.FirstLevelSubdivisionTable

where item.Active==true

select item).ToList();

int i = 0;

foreach (FirstLevelSubdivisionTable academy in academies)

{

CheckBoxList1.Items.Add(academy.Name);

int tmp = (from item in KPIWebDataContext.ReportArchiveAndLevelMappingTable

where item.FK\_FirstLevelSubmisionTableId == academy.FirstLevelSubdivisionTableID

&& item.FK\_ReportArchiveTableId == reportArchiveTableID

select item).Count();

CheckBoxList1.Items[i].Selected = tmp > 0 ? true : false;

CheckBoxList1.Items[i].Value = academy.FirstLevelSubdivisionTableID.ToString();

i++;

}

//////////////////////////////////////////////////////////////////////////////////

FillGridVIews(reportArchiveTableID);

}

else //создаем новый отчёт

{

KPIWebDataContext KPIWebDataContext = new KPIWebDataContext();

ButtonSave.Text="Сохранить новую кампанию";

List<FirstLevelSubdivisionTable> academies =

(from item in KPIWebDataContext.FirstLevelSubdivisionTable

where item.Active == true

select item).ToList();

int i = 0;

foreach (FirstLevelSubdivisionTable academy in academies)

{

CheckBoxList1.Items.Add(academy.Name);

CheckBoxList1.Items[i].Value = academy.FirstLevelSubdivisionTableID.ToString();

i++;

}

FillGridVIews(0);

}

}

}

protected void ButtonSave\_Click(object sender, EventArgs e)

{

if (TextBoxName.Text.Length < 4)

{

Page.ClientScript.RegisterClientScriptBlock(typeof(Page), "Script",

"alert('Введите корректное название отчёта');", true);

}

else if (!(CalendarStartDateTime.SelectedDate > DateTime.MinValue))

{

Page.ClientScript.RegisterClientScriptBlock(typeof(Page), "Script",

"alert('Введите дату начала отчёта');", true);

}

else if (!(CalendarEndDateTime.SelectedDate > DateTime.MinValue))

{

Page.ClientScript.RegisterClientScriptBlock(typeof(Page), "Script",

"alert('Введите дату конца отчёта');", true);

}

else if (CalendarEndDateTime.SelectedDate < CalendarStartDateTime.SelectedDate)

{

Page.ClientScript.RegisterClientScriptBlock(typeof(Page), "Script",

"alert('Неправильно указаны даты');", true);

}

else

{

Serialization ReportId = (Serialization) Session["ReportArchiveTableID"];

KPIWebDataContext kpiWebDataContext = new KPIWebDataContext();

ReportArchiveTable reportArchiveTable = new ReportArchiveTable();

#region //запись в базу нового отчёта или внесение изменений в старый

int reportArchiveTableID = 0;

if (ReportId.ReportArchiveID == 0) // создаем новую запись в БД и узнаем ей айди

{

reportArchiveTable.Active = CheckBoxActive.Checked;

reportArchiveTable.Calculeted = CheckBoxCalculeted.Checked;

reportArchiveTable.Sent = CheckBoxSent.Checked;

reportArchiveTable.RecipientConfirmed = CheckBoxRecipientConfirmed.Checked;

reportArchiveTable.Name = TextBoxName.Text;

if (CalendarStartDateTime.SelectedDate > DateTime.MinValue)

reportArchiveTable.StartDateTime = CalendarStartDateTime.SelectedDate;

if (CalendarEndDateTime.SelectedDate > DateTime.MinValue)

reportArchiveTable.EndDateTime = CalendarEndDateTime.SelectedDate;

if (CalendarDateToSend.SelectedDate > DateTime.MinValue)

reportArchiveTable.DateToSend = CalendarDateToSend.SelectedDate;

if (CalendarSentDateTime.SelectedDate > DateTime.MinValue)

reportArchiveTable.SentDateTime = CalendarSentDateTime.SelectedDate;

if (CalendarReportRecived.SelectedDate > DateTime.MinValue)

reportArchiveTable.RecivedDateTime = CalendarReportRecived.SelectedDate;

if (CalendarConfirmEndDay.SelectedDate > DateTime.MinValue)

reportArchiveTable.ConfirmEndDay = CalendarConfirmEndDay.SelectedDate;

if (DaysBeforeToCalcForRector.Text != "")

reportArchiveTable.DaysBeforeToCalcForRector = Convert.ToInt32(DaysBeforeToCalcForRector.Text);

kpiWebDataContext.ReportArchiveTable.InsertOnSubmit(reportArchiveTable);

kpiWebDataContext.SubmitChanges();

reportArchiveTableID = reportArchiveTable.ReportArchiveTableID;

}

else //если запись в бд уже есть

{

reportArchiveTableID = ReportId.ReportArchiveID;

reportArchiveTable = (from item in kpiWebDataContext.ReportArchiveTable

where item.ReportArchiveTableID == reportArchiveTableID

select item).FirstOrDefault();

}

int rowIndex = 0;

if (reportArchiveTable == null)

reportArchiveTable = new ReportArchiveTable();

reportArchiveTable.Active = CheckBoxActive.Checked;

reportArchiveTable.Calculeted = CheckBoxCalculeted.Checked;

reportArchiveTable.Sent = CheckBoxSent.Checked;

reportArchiveTable.RecipientConfirmed = CheckBoxRecipientConfirmed.Checked;

reportArchiveTable.Name = TextBoxName.Text;

if (CalendarStartDateTime.SelectedDate > DateTime.MinValue)

reportArchiveTable.StartDateTime = CalendarStartDateTime.SelectedDate;

if (CalendarEndDateTime.SelectedDate > DateTime.MinValue)

reportArchiveTable.EndDateTime = CalendarEndDateTime.SelectedDate;

if (CalendarDateToSend.SelectedDate > DateTime.MinValue)

reportArchiveTable.DateToSend = CalendarDateToSend.SelectedDate;

if (CalendarSentDateTime.SelectedDate > DateTime.MinValue)

reportArchiveTable.SentDateTime = CalendarSentDateTime.SelectedDate;

if (CalendarReportRecived.SelectedDate > DateTime.MinValue)

reportArchiveTable.RecivedDateTime = CalendarReportRecived.SelectedDate;

if (CalendarConfirmEndDay.SelectedDate > DateTime.MinValue)

reportArchiveTable.ConfirmEndDay = CalendarConfirmEndDay.SelectedDate;

if (DaysBeforeToCalcForRector.Text != "")

reportArchiveTable.DaysBeforeToCalcForRector = Convert.ToInt32(DaysBeforeToCalcForRector.Text);

#endregion

#region //связь отчёта со структурным подразд 1 уровня создание или изменение

foreach (ListItem checkItem in CheckBoxList1.Items)

{

if (checkItem.Selected)

{

ReportArchiveAndLevelMappingTable repNRole =

(from item in kpiWebDataContext.ReportArchiveAndLevelMappingTable

where item.FK\_FirstLevelSubmisionTableId == Convert.ToInt32(checkItem.Value)

&& item.FK\_ReportArchiveTableId == reportArchiveTableID

select item).FirstOrDefault();

if (repNRole != null)

{

repNRole.Active = true;

}

else

{

repNRole = new ReportArchiveAndLevelMappingTable();

repNRole.Active = true;

repNRole.FK\_FirstLevelSubmisionTableId = Convert.ToInt32(checkItem.Value);

repNRole.FK\_ReportArchiveTableId = reportArchiveTableID;

kpiWebDataContext.ReportArchiveAndLevelMappingTable.InsertOnSubmit(repNRole);

}

}

else

{

ReportArchiveAndLevelMappingTable repNRole =

(from item in kpiWebDataContext.ReportArchiveAndLevelMappingTable

where item.FK\_FirstLevelSubmisionTableId == Convert.ToInt32(checkItem.Value)

&& item.FK\_ReportArchiveTableId == reportArchiveTableID

select item).FirstOrDefault();

if (repNRole != null)

{

repNRole.Active = false; /// // / лучше просто удалить эту запись из БД

}

}

}

kpiWebDataContext.SubmitChanges();

#endregion

////////////////////////////////////////////////////////пора записать данные в таблицы связей

if ((ViewState["BasicDataTable"] != null) && (ViewState["CalculateDataTable"] != null) &&

(ViewState["IndicatorDataTable"] != null))

{

DataTable dt\_basic = (DataTable) ViewState["BasicDataTable"];

DataTable dt\_calculate = (DataTable) ViewState["CalculateDataTable"];

DataTable dt\_indicator = (DataTable) ViewState["IndicatorDataTable"];

////////////////////////////////////////////////////////////////////////////////////////////////////////

for (int i = 0; i < dt\_basic.Rows.Count; i++)

{

CheckBox chekBox = (CheckBox) BasicParametrsTable.Rows[i].FindControl("BasicParametrsCheckBox");

Label label = (Label) BasicParametrsTable.Rows[i].FindControl("BasicParametrsID");

if (chekBox.Checked == true)

{

ReportArchiveAndBasicParametrsMappingTable basicParam =

(from a in kpiWebDataContext.ReportArchiveAndBasicParametrsMappingTable

where a.FK\_BasicParametrsTable == Convert.ToInt32(label.Text)

&& a.FK\_ReportArchiveTable == reportArchiveTableID

select a).FirstOrDefault();

if (basicParam != null)

{

basicParam.Active = true;

}

else

{

basicParam = new ReportArchiveAndBasicParametrsMappingTable();

basicParam.Active = true;

basicParam.FK\_BasicParametrsTable = Convert.ToInt32(label.Text);

basicParam.FK\_ReportArchiveTable = reportArchiveTableID;

kpiWebDataContext.ReportArchiveAndBasicParametrsMappingTable.InsertOnSubmit(basicParam);

}

kpiWebDataContext.SubmitChanges();

}

else

{

ReportArchiveAndBasicParametrsMappingTable basicParam =

(from a in kpiWebDataContext.ReportArchiveAndBasicParametrsMappingTable

where a.FK\_BasicParametrsTable == Convert.ToInt32(label.Text)

&& a.FK\_ReportArchiveTable == reportArchiveTableID

select a).FirstOrDefault();

if (basicParam != null)

{

basicParam.Active = false;

kpiWebDataContext.SubmitChanges();

}

}

}

////////////////////////////////////////////////////////////////////////////////////////////////

List<int> calcId = new List<int>();

////////////////////////////////////////////////////////////////////////////////////////////////////

for (int i = 0; i < dt\_calculate.Rows.Count; i++)

{

CheckBox chekBox =

(CheckBox) CalculatedParametrsTable.Rows[i].FindControl("CalculatedParametrsCheckBox");

Label label = (Label) CalculatedParametrsTable.Rows[i].FindControl("CalculatedParametrsID");

if (chekBox.Checked == true)

{

calcId.Add(Convert.ToInt32(label.Text));

ReportArchiveAndCalculatedParametrsMappingTable calcParam =

(from a in kpiWebDataContext.ReportArchiveAndCalculatedParametrsMappingTable

where a.FK\_CalculatedParametrsTable == Convert.ToInt32(label.Text)

&& a.FK\_ReportArchiveTable == reportArchiveTableID

select a).FirstOrDefault();

if (calcParam != null)

{

calcParam.Active = true;

}

else

{

calcParam = new ReportArchiveAndCalculatedParametrsMappingTable();

calcParam.Active = true;

calcParam.FK\_CalculatedParametrsTable = Convert.ToInt32(label.Text);

calcParam.FK\_ReportArchiveTable = reportArchiveTableID;

kpiWebDataContext.ReportArchiveAndCalculatedParametrsMappingTable.InsertOnSubmit(

calcParam);

}

kpiWebDataContext.SubmitChanges();

}

else

{

ReportArchiveAndCalculatedParametrsMappingTable calcParam =

(from a in kpiWebDataContext.ReportArchiveAndCalculatedParametrsMappingTable

where a.FK\_CalculatedParametrsTable == Convert.ToInt32(label.Text)

&& a.FK\_ReportArchiveTable == reportArchiveTableID

select a).FirstOrDefault();

if (calcParam != null)

{

calcParam.Active = false;

kpiWebDataContext.SubmitChanges();

}

}

}

/////////////////////////////////////////////////////////////////////////////////////////

List<int> indId = new List<int>();

///////////////////////////////////////////////////////////////////////////////////////////////////

for (int i = 0; i < dt\_indicator.Rows.Count; i++)

{

CheckBox chekBox = (CheckBox) IndicatorsTable.Rows[i].FindControl("IndicatorCheckBox");

Label label = (Label) IndicatorsTable.Rows[i].FindControl("IndicatorID");

if (chekBox.Checked == true)

{

indId.Add(Convert.ToInt32(label.Text));

ReportArchiveAndIndicatorsMappingTable indicators =

(from a in kpiWebDataContext.ReportArchiveAndIndicatorsMappingTable

where a.FK\_IndicatorsTable == Convert.ToInt32(label.Text)

&& a.FK\_ReportArchiveTable == reportArchiveTableID

select a).FirstOrDefault();

if (indicators != null)

{

indicators.Active = true;

}

else

{

indicators = new ReportArchiveAndIndicatorsMappingTable();

indicators.Active = true;

indicators.FK\_IndicatorsTable = Convert.ToInt32(label.Text);

indicators.FK\_ReportArchiveTable = reportArchiveTableID;

kpiWebDataContext.ReportArchiveAndIndicatorsMappingTable.InsertOnSubmit(indicators);

}

kpiWebDataContext.SubmitChanges();

}

else

{

ReportArchiveAndIndicatorsMappingTable indicators =

(from a in kpiWebDataContext.ReportArchiveAndIndicatorsMappingTable

where a.FK\_IndicatorsTable == Convert.ToInt32(label.Text)

&& a.FK\_ReportArchiveTable == reportArchiveTableID

select a).FirstOrDefault();

if (indicators != null)

{

indicators.Active = false;

kpiWebDataContext.SubmitChanges();

}

}

}

////////////////////////////////////////////////////////////////////////////////////////////////////

///нужнополучить список айдишников нужных базовых показателей

List<int> CalcID = getCalcByIndicator(indId);

foreach (int tmp in CalcID)

{

calcId.Add(tmp);

}

#region

foreach (int CalcParamID in calcId)

{

ReportArchiveAndCalculatedParametrsMappingTable calcParam =

(from a in kpiWebDataContext.ReportArchiveAndCalculatedParametrsMappingTable

where a.FK\_CalculatedParametrsTable == CalcParamID

&& a.FK\_ReportArchiveTable == reportArchiveTableID

select a).FirstOrDefault();

if (calcParam != null)

{

calcParam.Active = true;

}

else

{

calcParam = new ReportArchiveAndCalculatedParametrsMappingTable();

calcParam.Active = true;

calcParam.FK\_CalculatedParametrsTable = CalcParamID;

calcParam.FK\_ReportArchiveTable = reportArchiveTableID;

kpiWebDataContext.ReportArchiveAndCalculatedParametrsMappingTable.InsertOnSubmit(calcParam);

}

kpiWebDataContext.SubmitChanges();

}

#endregion

List<int> BaseID = getBasicByCalc(calcId);

#region

foreach (int baseID in BaseID)

{

ReportArchiveAndBasicParametrsMappingTable basicParam =

(from a in kpiWebDataContext.ReportArchiveAndBasicParametrsMappingTable

where a.FK\_BasicParametrsTable == baseID

&& a.FK\_ReportArchiveTable == reportArchiveTableID

select a).FirstOrDefault();

if (basicParam != null)

{

basicParam.Active = true;

}

else

{

basicParam = new ReportArchiveAndBasicParametrsMappingTable();

basicParam.Active = true;

basicParam.FK\_BasicParametrsTable = baseID;

basicParam.FK\_ReportArchiveTable = reportArchiveTableID;

kpiWebDataContext.ReportArchiveAndBasicParametrsMappingTable.InsertOnSubmit(basicParam);

}

kpiWebDataContext.SubmitChanges();

}

#endregion

}

Response.Redirect("~/StatisticsDepartment/ReportViewer.aspx");

}

}

protected void CalendarStartDateTime\_SelectionChanged(object sender, EventArgs e)

{

}

protected void BasicParametrsTable\_SelectedIndexChanged(object sender, EventArgs e)

{

}

protected void Button1\_Click(object sender, EventArgs e)

{

Serialization ReportId = (Serialization) Session["ReportArchiveTableID"];

KPIWebDataContext kpiWebDataContext = new KPIWebDataContext();

var deleteBasic = from a in kpiWebDataContext.ReportArchiveAndBasicParametrsMappingTable

where a.FK\_ReportArchiveTable == ReportId.ReportArchiveID select a;

var deleteCalc = from a in kpiWebDataContext.ReportArchiveAndCalculatedParametrsMappingTable

where a.FK\_ReportArchiveTable == ReportId.ReportArchiveID select a;

var deleteIndicator = from a in kpiWebDataContext.ReportArchiveAndIndicatorsMappingTable

where a.FK\_ReportArchiveTable == ReportId.ReportArchiveID select a;

foreach (var delB in deleteBasic)

{

kpiWebDataContext.ReportArchiveAndBasicParametrsMappingTable.DeleteOnSubmit(delB);

}

foreach (var delC in deleteCalc)

{

kpiWebDataContext.ReportArchiveAndCalculatedParametrsMappingTable.DeleteOnSubmit(delC);

}

foreach (var delI in deleteIndicator)

{

kpiWebDataContext.ReportArchiveAndIndicatorsMappingTable.DeleteOnSubmit(delI);

}

kpiWebDataContext.SubmitChanges();

Response.Redirect("~/StatisticsDepartment/ReportViewer.aspx");

}

protected void Button2\_Click(object sender, EventArgs e)

{

for (int i = 0; i < IndicatorsTable.Rows.Count; i++)

{

CheckBox chekBox = (CheckBox)IndicatorsTable.Rows[i].FindControl("IndicatorCheckBox");

chekBox.Checked = true;

}

for (int i = 0; i < CalculatedParametrsTable.Rows.Count; i++)

{

CheckBox chekBox = (CheckBox)CalculatedParametrsTable.Rows[i].FindControl("CalculatedParametrsCheckBox");

chekBox.Checked = true;

}

for (int i = 0; i < BasicParametrsTable.Rows.Count; i++)

{

CheckBox chekBox = (CheckBox)BasicParametrsTable.Rows[i].FindControl("BasicParametrsCheckBox");

chekBox.Checked = true;

}

}

protected void Button3\_Click(object sender, EventArgs e)

{

for (int i = 0; i < CheckBoxList1.Items.Count; i++)

{

CheckBoxList1.Items[i].Selected = true;

}

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Filename: EditReport.aspx.designer.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//------------------------------------------------------------------------------

// <автоматически создаваемое>

// Этот код создан программой.

//

// Изменения в этом файле могут привести к неправильной работе и будут потеряны в случае

// повторной генерации кода.

// </автоматически создаваемое>

//------------------------------------------------------------------------------

namespace KPIWeb.Reports {

public partial class EditReport {

/// <summary>

/// TextBoxName элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.TextBox TextBoxName;

/// <summary>

/// CheckBoxActive элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.CheckBox CheckBoxActive;

/// <summary>

/// CalendarStartDateTime элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Calendar CalendarStartDateTime;

/// <summary>

/// CalendarEndDateTime элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Calendar CalendarEndDateTime;

/// <summary>

/// Label1 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Label Label1;

/// <summary>

/// CalendarConfirmEndDay элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Calendar CalendarConfirmEndDay;

/// <summary>

/// CheckBoxCalculeted элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.CheckBox CheckBoxCalculeted;

/// <summary>

/// CalendarDateToSend элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Calendar CalendarDateToSend;

/// <summary>

/// CheckBoxSent элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.CheckBox CheckBoxSent;

/// <summary>

/// CalendarSentDateTime элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Calendar CalendarSentDateTime;

/// <summary>

/// CheckBoxRecipientConfirmed элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.CheckBox CheckBoxRecipientConfirmed;

/// <summary>

/// CalendarReportRecived элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Calendar CalendarReportRecived;

/// <summary>

/// Label2 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Label Label2;

/// <summary>

/// DaysBeforeToCalcForRector элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.TextBox DaysBeforeToCalcForRector;

/// <summary>

/// Button3 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Button Button3;

/// <summary>

/// CheckBoxList1 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.CheckBoxList CheckBoxList1;

/// <summary>

/// ButtonSave элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Button ButtonSave;

/// <summary>

/// Button1 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Button Button1;

/// <summary>

/// Button2 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Button Button2;

/// <summary>

/// IndicatorsTable элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.GridView IndicatorsTable;

/// <summary>

/// CalculatedParametrsTable элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.GridView CalculatedParametrsTable;

/// <summary>

/// BasicParametrsTable элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.GridView BasicParametrsTable;

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Filename: FillingTheReport.aspx.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.UI.WebControls;

using System.Configuration;

using System.Net;

using System.Data;

using System.Collections.Specialized;

using System.Data.Entity.Core.Common.CommandTrees.ExpressionBuilder;

using System.Data.SqlClient;

using System.Drawing;

using System.IO;

using System.Net.Mail;

using System.Text;

using System.Web.UI.HtmlControls;

using System.Web.WebPages;

using iTextSharp.text;

using iTextSharp.text.pdf;

//using Microsoft.Office.Interop.Excel;

using DataTable = System.Data.DataTable;

using Label = System.Web.UI.WebControls.Label;

using Page = System.Web.UI.Page;

using TextBox = System.Web.UI.WebControls.TextBox;

namespace KPIWeb.Reports

{

public partial class FillingTheReport : System.Web.UI.Page

{

public int col\_ = 0;

public string ExportPDF(GridView GridToExport, int[] WidthArray, string Header, int CurrentPageSize, int ColumnCount, string ReportName, string UserPositionName, string UserStructName)

{

GridToExport.AllowPaging = false;

// GridToExport.DataBind();

BaseFont bf = BaseFont.CreateFont(Environment.GetEnvironmentVariable("windir") + @"\fonts\Arial.TTF",

BaseFont.IDENTITY\_H, true);

iTextSharp.text.pdf.PdfPTable table = new iTextSharp.text.pdf.PdfPTable(ColumnCount + 2);

int[] widths = new int[ColumnCount + 2];

iTextSharp.text.Font font = new iTextSharp.text.Font(bf, 7, iTextSharp.text.Font.NORMAL);

font.Color = new BaseColor(0, 0, 0);

///////

widths[0] = WidthArray[0];

string IdText = Server.HtmlDecode(GridToExport.HeaderRow.Cells[0].Text);

iTextSharp.text.pdf.PdfPCell HeaderIdcell = new iTextSharp.text.pdf.PdfPCell(new Phrase(12, IdText, font));

HeaderIdcell.BackgroundColor = new BaseColor(200, 200, 200);

table.AddCell(HeaderIdcell);

///////

widths[1] = WidthArray[2];

string NameHeaderText = Server.HtmlDecode(GridToExport.HeaderRow.Cells[2].Text);

iTextSharp.text.pdf.PdfPCell NameCell = new iTextSharp.text.pdf.PdfPCell(new Phrase(12, NameHeaderText, font));

NameCell.BackgroundColor = new BaseColor(200, 200, 200);

table.AddCell(NameCell);

for (int x = 0; x < ColumnCount; x++)

{

widths[x + 2] = WidthArray[x + 4];

string cellText = Server.HtmlDecode(GridToExport.HeaderRow.Cells[x + 4].Text);

// iTextSharp.text.Font font = new iTextSharp.text.Font(bf, 10, iTextSharp.text.Font.NORMAL);

font.Color = new BaseColor(0, 0, 0);

iTextSharp.text.pdf.PdfPCell cell = new iTextSharp.text.pdf.PdfPCell(new Phrase(12, cellText, font));

cell.BackgroundColor = new BaseColor(200, 200, 200);

table.AddCell(cell);

}

table.SetWidths(widths);

for (int i = 0; i < GridToExport.Rows.Count; i++)

{

if (GridToExport.Rows[i].RowType == DataControlRowType.DataRow)

{

//iTextSharp.text.Font font = new iTextSharp.text.Font(bf, 10, iTextSharp.text.Font.NORMAL);

//font.Color = new BaseColor(0, 0, 0);

///////////// PARAM ID

iTextSharp.text.pdf.PdfPCell IDcell =

new iTextSharp.text.pdf.PdfPCell(new Phrase(12, Server.HtmlDecode(GridToExport.Rows[i].Cells[0].Text), font));

if (i % 2 == 0)

{

IDcell.BackgroundColor = new BaseColor(230, 230, 230);

}

table.AddCell(IDcell);

/////////////

///////////// PARAM NAME

Label NameTextBox = (Label)GridToExport.Rows[i].FindControl("Name");

iTextSharp.text.pdf.PdfPCell Namecell =

new iTextSharp.text.pdf.PdfPCell(new Phrase(12, Server.HtmlDecode(NameTextBox.Text), font));

if (i % 2 == 0)

{

Namecell.BackgroundColor = new BaseColor(230, 230, 230);

}

table.AddCell(Namecell);

/////////////

for (int j = 0; j < ColumnCount; j++)

{

TextBox textBox = (TextBox)GridToExport.Rows[i].FindControl("Value" + j.ToString());

string cellText = Server.HtmlDecode(textBox.Text);

iTextSharp.text.pdf.PdfPCell cell =

new iTextSharp.text.pdf.PdfPCell(new Phrase(12, cellText, font));

if (i % 2 == 0)

{

cell.BackgroundColor = new BaseColor(230, 230, 230);

}

table.AddCell(cell);

}

}

}

string DocPath = "";

Document pdfDoc = new Document(PageSize.A4, 20f, 20f, 20f, 20f);

using (MemoryStream myMemoryStream = new MemoryStream())

{

PdfWriter.GetInstance(pdfDoc, myMemoryStream);

Chunk c1 = new Chunk(Header);

pdfDoc.Open();

//////////////////////////////////////////

iTextSharp.text.Font TitleFont = new iTextSharp.text.Font(bf, 12, iTextSharp.text.Font.NORMAL);

Paragraph para = new Paragraph(ReportName, TitleFont);//название отчета

para.Alignment = Element.ALIGN\_CENTER;

pdfDoc.Add(para);

para = new Paragraph(UserPositionName, TitleFont);//должность

para.Alignment = Element.ALIGN\_LEFT;

pdfDoc.Add(para);

para = new Paragraph(UserStructName, TitleFont);//кафедра

para.Alignment = Element.ALIGN\_LEFT;

pdfDoc.Add(para);

para = new Paragraph(" ");

pdfDoc.Add(para);

/////////////////////////////////////////

table.WidthPercentage = 100;

pdfDoc.Add(table);

pdfDoc.Close();

byte[] content = myMemoryStream.ToArray();

string DocName = "Document" + DateTime.Now;

DocName = DocName.Replace(":", "");

DocName = DocName.Replace(".", "");

DocName = DocName.Replace(" ", "");

DocName += ".pdf";

DocPath = Server.MapPath("~/PDFArchive/" + DocName);

using (FileStream fs = File.Create(DocPath))

{

fs.Write(content, 0, (int)content.Length);

}

}

return DocPath;

}

#region patterns

protected double pattern1(UsersTable user, int ReportArchiveID, int spectype\_, string basicAbb, string basicAbb2) // по областям знаний

{

KPIWebDataContext kpiWebDataContext = new KPIWebDataContext();

return Convert.ToDouble(

(from a in kpiWebDataContext.CollectedBasicParametersTable

join z in kpiWebDataContext.BasicParametersTable

on a.FK\_BasicParametersTable equals z.BasicParametersTableID

join b in kpiWebDataContext.FourthLevelParametrs

on a.FK\_FourthLevelSubdivisionTable equals b.FourthLevelParametrsID

join c in kpiWebDataContext.ThirdLevelParametrs

on a.FK\_ThirdLevelSubdivisionTable equals c.ThirdLevelParametrsID

join d in kpiWebDataContext.FourthLevelSubdivisionTable

on a.FK\_FourthLevelSubdivisionTable equals d.FourthLevelSubdivisionTableID

join e in kpiWebDataContext.SpecializationTable

on d.FK\_Specialization equals e.SpecializationTableID

where

a.FK\_ZeroLevelSubdivisionTable == user.FK\_ZeroLevelSubdivisionTable

&& a.FK\_FirstLevelSubdivisionTable == user.FK\_FirstLevelSubdivisionTable

&& a.FK\_SecondLevelSubdivisionTable == user.FK\_SecondLevelSubdivisionTable

&& a.FK\_ThirdLevelSubdivisionTable == user.FK\_ThirdLevelSubdivisionTable

&& ((z.AbbreviationEN == basicAbb) || ((basicAbb2 != null) && (z.AbbreviationEN == basicAbb2)))

&& a.FK\_ReportArchiveTable == ReportArchiveID

&& b.SpecType == spectype\_

&& a.Active == true

&& d.Active == true

&& (e.FK\_FieldOfExpertise == 10 || e.FK\_FieldOfExpertise == 11 || e.FK\_FieldOfExpertise == 12)

select a.CollectedValue).Sum());

}

protected double pattern2(UsersTable user, int ReportArchiveID, string basicAbb) // для инстранцев

{

KPIWebDataContext kpiWebDataContext = new KPIWebDataContext();

return Convert.ToDouble(

(from a in kpiWebDataContext.CollectedBasicParametersTable

join z in kpiWebDataContext.BasicParametersTable

on a.FK\_BasicParametersTable equals z.BasicParametersTableID

join b in kpiWebDataContext.FourthLevelParametrs

on a.FK\_FourthLevelSubdivisionTable equals b.FourthLevelParametrsID

where

a.FK\_ZeroLevelSubdivisionTable == user.FK\_ZeroLevelSubdivisionTable

&& a.FK\_FirstLevelSubdivisionTable == user.FK\_FirstLevelSubdivisionTable

&& a.FK\_SecondLevelSubdivisionTable == user.FK\_SecondLevelSubdivisionTable

&& a.FK\_ThirdLevelSubdivisionTable == user.FK\_ThirdLevelSubdivisionTable

&& z.AbbreviationEN == basicAbb

&& a.FK\_ReportArchiveTable == ReportArchiveID

&& b.IsForeignStudentsAccept == true

&& a.Active == true

&& z.Active == true

&& b.Active == true

select a.CollectedValue).Sum());

}

protected double pattern3(UsersTable user, int ReportArchiveID, int SpecType) // Кол-во ООП // считает кол-во прикрепленных специальностьей

{

KPIWebDataContext kpiWebDataContext = new KPIWebDataContext();

return Convert.ToDouble(

(from a in kpiWebDataContext.ThirdLevelSubdivisionTable

join b in kpiWebDataContext.FourthLevelSubdivisionTable

on a.ThirdLevelSubdivisionTableID equals b.FK\_ThirdLevelSubdivisionTable

join c in kpiWebDataContext.FourthLevelParametrs

on b.FourthLevelSubdivisionTableID equals c.FourthLevelParametrsID

join d in kpiWebDataContext.ThirdLevelParametrs

on a.ThirdLevelSubdivisionTableID equals d.ThirdLevelParametrsID

where c.SpecType == SpecType

&& a.ThirdLevelSubdivisionTableID == user.FK\_ThirdLevelSubdivisionTable

&& d.CanGraduate == true

&& a.Active == true

// && z.Active == true

&& b.Active == true

select b).ToList().Count);

}

protected double pattern4(UsersTable user, int ReportArchiveID, int SpecType) // Кол-во ООП с условиями для инвалидов

{

KPIWebDataContext kpiWebDataContext = new KPIWebDataContext();

return Convert.ToDouble(

(from a in kpiWebDataContext.ThirdLevelSubdivisionTable

join b in kpiWebDataContext.FourthLevelSubdivisionTable

on a.ThirdLevelSubdivisionTableID equals b.FK\_ThirdLevelSubdivisionTable

join c in kpiWebDataContext.FourthLevelParametrs

on b.FourthLevelSubdivisionTableID equals c.FourthLevelParametrsID

join d in kpiWebDataContext.ThirdLevelParametrs

on a.ThirdLevelSubdivisionTableID equals d.ThirdLevelParametrsID

where c.SpecType == SpecType

&& a.ThirdLevelSubdivisionTableID == user.FK\_ThirdLevelSubdivisionTable

&& d.CanGraduate == true

&& c.IsInvalidStudentsFacilities == true

&& a.Active == true

&& b.Active == true

select b).ToList().Count);

}

protected double pattern5(UsersTable user, int ReportArchiveID, int SpecType)

{

KPIWebDataContext kpiWebDataContext = new KPIWebDataContext();

return Convert.ToDouble(

(from a in kpiWebDataContext.ThirdLevelSubdivisionTable

join b in kpiWebDataContext.FourthLevelSubdivisionTable

on a.ThirdLevelSubdivisionTableID equals b.FK\_ThirdLevelSubdivisionTable

join c in kpiWebDataContext.FourthLevelParametrs

on b.FourthLevelSubdivisionTableID equals c.FourthLevelParametrsID

join d in kpiWebDataContext.ThirdLevelParametrs

on a.ThirdLevelSubdivisionTableID equals d.ThirdLevelParametrsID

where c.SpecType == SpecType

&& a.ThirdLevelSubdivisionTableID == user.FK\_ThirdLevelSubdivisionTable

&& d.CanGraduate == true

&& c.IsNetworkComunication == true

&& a.Active == true

&& b.Active == true

select b).ToList().Count);

}

protected double pattern6(UsersTable user, int ReportArchiveID, int SpecType)

{

KPIWebDataContext kpiWebDataContext = new KPIWebDataContext();

return Convert.ToDouble(

(from a in kpiWebDataContext.ThirdLevelSubdivisionTable

join b in kpiWebDataContext.FourthLevelSubdivisionTable

on a.ThirdLevelSubdivisionTableID equals b.FK\_ThirdLevelSubdivisionTable

join c in kpiWebDataContext.FourthLevelParametrs

on b.FourthLevelSubdivisionTableID equals c.FourthLevelParametrsID

join d in kpiWebDataContext.ThirdLevelParametrs

on a.ThirdLevelSubdivisionTableID equals d.ThirdLevelParametrsID

where c.SpecType == SpecType

&& a.ThirdLevelSubdivisionTableID == user.FK\_ThirdLevelSubdivisionTable

&& d.CanGraduate == true

&& a.Active == true

&& b.Active == true

&& c.IsModernEducationTechnologies == true

select b).ToList().Count);

}

protected double pattern7(int SpecID, int typeOfCost, int ReportID, FourthLevelSubdivisionTable Fourth, int SpecType) // type 0 очное // 1 очное иностранцы // 2 заочное // 3 вечернее

{

KPIWebDataContext kpiWebDataContext = new KPIWebDataContext();

if (typeOfCost == 0)

{

return Convert.ToDouble((from a in kpiWebDataContext.EducationCostTable

where a.Active == true

&& a.FK\_Specialization == SpecID

select a.CostOfCommercOch).FirstOrDefault()

\*

(from a in kpiWebDataContext.CollectedBasicParametersTable

where a.FK\_ReportArchiveTable == ReportID

&& a.FK\_FourthLevelSubdivisionTable == Fourth.FourthLevelSubdivisionTableID

join c in kpiWebDataContext.BasicParametersTable

on a.FK\_BasicParametersTable equals c.BasicParametersTableID

where

((c.AbbreviationEN == "a\_Och\_M\_Kom" && SpecType == 3)

|| (c.AbbreviationEN == "a\_Och\_B\_Kom" && SpecType == 1)

|| (c.AbbreviationEN == "a\_Och\_S\_Kom" && SpecType == 2)

|| (c.AbbreviationEN == "a\_Och\_A\_Kom" && SpecType == 4))

select a.CollectedValue).Sum());

}

else if (typeOfCost == 1)

{

return Convert.ToDouble((from a in kpiWebDataContext.EducationCostTable

where a.Active == true

&& a.FK\_Specialization == SpecID

select a.CostOfCommercOchIn).FirstOrDefault()

\*

(from a in kpiWebDataContext.CollectedBasicParametersTable

where a.FK\_ReportArchiveTable == ReportID

&& a.FK\_FourthLevelSubdivisionTable == Fourth.FourthLevelSubdivisionTableID

join c in kpiWebDataContext.BasicParametersTable

on a.FK\_BasicParametersTable equals c.BasicParametersTableID

where

((c.AbbreviationEN == "a\_Och\_In\_M" && SpecType == 3)

|| (c.AbbreviationEN == "a\_Och\_In\_B" && SpecType == 1)

|| (c.AbbreviationEN == "a\_Och\_In\_S" && SpecType == 2)

|| (c.AbbreviationEN == "a\_Och\_In\_A" && SpecType == 4))

select a.CollectedValue).Sum());

}

else if (typeOfCost == 2)

{

return Convert.ToDouble((from a in kpiWebDataContext.EducationCostTable

where a.Active == true

&& a.FK\_Specialization == SpecID

select a.CostOfCommercZaoch).FirstOrDefault()

\*

(from a in kpiWebDataContext.CollectedBasicParametersTable

where a.FK\_ReportArchiveTable == ReportID

&& a.FK\_FourthLevelSubdivisionTable == Fourth.FourthLevelSubdivisionTableID

join c in kpiWebDataContext.BasicParametersTable

on a.FK\_BasicParametersTable equals c.BasicParametersTableID

where

((c.AbbreviationEN == "c\_Z\_A\_Kom" && SpecType == 3)

|| (c.AbbreviationEN == "c\_Z\_B\_Kom" && SpecType == 1)

|| (c.AbbreviationEN == "c\_Z\_S\_Kom" && SpecType == 2)

|| (c.AbbreviationEN == "c\_Z\_M\_Kom" && SpecType == 4))

select a.CollectedValue).Sum());

}

else if (typeOfCost == 3)

{

return Convert.ToDouble((from a in kpiWebDataContext.EducationCostTable

where a.Active == true

&& a.FK\_Specialization == SpecID

select a.CostOfCommercEvening).FirstOrDefault()

\*

(from a in kpiWebDataContext.CollectedBasicParametersTable

where a.FK\_ReportArchiveTable == ReportID

&& a.FK\_FourthLevelSubdivisionTable == Fourth.FourthLevelSubdivisionTableID

join c in kpiWebDataContext.BasicParametersTable

on a.FK\_BasicParametersTable equals c.BasicParametersTableID

where

((c.AbbreviationEN == "b\_OchZ\_S\_Kom" && SpecType == 3)

|| (c.AbbreviationEN == "b\_OchZ\_M\_Kom" && SpecType == 1)

|| (c.AbbreviationEN == "b\_OchZ\_A\_Kom" && SpecType == 2)

|| (c.AbbreviationEN == "b\_OchZ\_B\_Kom" && SpecType == 4))

select a.CollectedValue).Sum());

}

return 0;

}

public void patternSwitch(int ReportArchiveID, BasicParametersTable basicParam, FourthLevelSubdivisionTable FourthLevel, int fourthCnt, UsersTable user)

{

KPIWebDataContext kpiWebDataContext = new KPIWebDataContext();

double tmp = 1000000000001;

if (fourthCnt > 0)

{

if (FourthLevel == null)

{

if (basicParam.AbbreviationEN == "a\_Och\_M\_IZO") tmp = pattern1(user, ReportArchiveID, 3, "a\_Och\_M", "a\_Och\_M\_Kom");

if (basicParam.AbbreviationEN == "b\_OchZ\_M\_IZO") tmp = pattern1(user, ReportArchiveID, 3, "b\_OchZ\_M", "b\_OchZ\_M\_Kom");

if (basicParam.AbbreviationEN == "c\_Z\_M\_IZO") tmp = pattern1(user, ReportArchiveID, 3, "c\_Z\_M", "c\_Z\_M\_Kom");

if (basicParam.AbbreviationEN == "d\_E\_M\_IZO") tmp = pattern1(user, ReportArchiveID, 3, "d\_E\_M", "d\_E\_M\_Kom");

if (basicParam.AbbreviationEN == "a\_Och\_M\_NoIn") tmp = pattern2(user, ReportArchiveID, "a\_Och\_M");

if (basicParam.AbbreviationEN == "b\_OchZ\_M\_NoIn") tmp = pattern2(user, ReportArchiveID, "b\_OchZ\_M");

if (basicParam.AbbreviationEN == "c\_Z\_M\_NoIn") tmp = pattern2(user, ReportArchiveID, "c\_Z\_M");

if (basicParam.AbbreviationEN == "d\_E\_M\_NoIn") tmp = pattern2(user, ReportArchiveID, "d\_E\_M");

if (basicParam.AbbreviationEN == "a\_Och\_M\_NoIn\_Kom") tmp = pattern2(user, ReportArchiveID, "a\_Och\_M\_Kom");

if (basicParam.AbbreviationEN == "b\_OchZ\_M\_NoIn\_Kom") tmp = pattern2(user, ReportArchiveID, "b\_OchZ\_M\_Kom");

if (basicParam.AbbreviationEN == "c\_Z\_M\_NoIn\_Kom") tmp = pattern2(user, ReportArchiveID, "c\_Z\_M\_Kom");

if (basicParam.AbbreviationEN == "d\_E\_M\_NoIn\_Kom") tmp = pattern2(user, ReportArchiveID, "d\_E\_M\_Kom");

if (basicParam.AbbreviationEN == "a\_Och\_S\_IZO") tmp = pattern1(user, ReportArchiveID, 2, "a\_Och\_S", "a\_Och\_S\_Kom");

if (basicParam.AbbreviationEN == "b\_OchZ\_S\_IZO") tmp = pattern1(user, ReportArchiveID, 2, "b\_OchZ\_S", "b\_OchZ\_S\_Kom");

if (basicParam.AbbreviationEN == "c\_Z\_S\_IZO") tmp = pattern1(user, ReportArchiveID, 2, "c\_Z\_S", "c\_Z\_S\_Kom");

if (basicParam.AbbreviationEN == "d\_E\_S\_IZO") tmp = pattern1(user, ReportArchiveID, 2, "d\_E\_S", "c\_Z\_S\_Kom");

if (basicParam.AbbreviationEN == "a\_Och\_S\_NoIn") tmp = pattern2(user, ReportArchiveID, "a\_Och\_S");

if (basicParam.AbbreviationEN == "b\_OchZ\_S\_NoIn") tmp = pattern2(user, ReportArchiveID, "b\_OchZ\_S");

if (basicParam.AbbreviationEN == "c\_Z\_S\_NoIn") tmp = pattern2(user, ReportArchiveID, "c\_Z\_S");

if (basicParam.AbbreviationEN == "d\_E\_S\_NoIn") tmp = pattern2(user, ReportArchiveID, "d\_E\_S");

if (basicParam.AbbreviationEN == "a\_Och\_S\_NoIn\_Kom") tmp = pattern2(user, ReportArchiveID, "a\_Och\_S\_Kom");

if (basicParam.AbbreviationEN == "b\_OchZ\_S\_NoIn\_Kom") tmp = pattern2(user, ReportArchiveID, "b\_OchZ\_S\_Kom");

if (basicParam.AbbreviationEN == "c\_Z\_S\_NoIn\_Kom") tmp = pattern2(user, ReportArchiveID, "c\_Z\_S\_Kom");

if (basicParam.AbbreviationEN == "d\_E\_S\_NoIn\_Kom") tmp = pattern2(user, ReportArchiveID, "d\_E\_S\_Kom");

if (basicParam.AbbreviationEN == "a\_Och\_B\_IZO") tmp = pattern1(user, ReportArchiveID, 1, "a\_Och\_B", "a\_Och\_B\_Kom");

if (basicParam.AbbreviationEN == "b\_OchZ\_B\_IZO") tmp = pattern1(user, ReportArchiveID, 1, "b\_OchZ\_B", "b\_OchZ\_B\_Kom");

if (basicParam.AbbreviationEN == "c\_Z\_B\_IZO") tmp = pattern1(user, ReportArchiveID, 1, "c\_Z\_B", "c\_Z\_B\_Kom");

if (basicParam.AbbreviationEN == "d\_E\_B\_IZO") tmp = pattern1(user, ReportArchiveID, 1, "d\_E\_B", "d\_E\_B\_Kom");

if (basicParam.AbbreviationEN == "a\_Och\_B\_NoIn") tmp = pattern2(user, ReportArchiveID, "a\_Och\_B");

if (basicParam.AbbreviationEN == "b\_OchZ\_B\_NoIn") tmp = pattern2(user, ReportArchiveID, "b\_OchZ\_B");

if (basicParam.AbbreviationEN == "c\_Z\_B\_NoIn") tmp = pattern2(user, ReportArchiveID, "c\_Z\_B");

if (basicParam.AbbreviationEN == "d\_E\_B\_NoIn") tmp = pattern2(user, ReportArchiveID, "d\_E\_B");

if (basicParam.AbbreviationEN == "a\_Och\_B\_NoIn\_Kom") tmp = pattern2(user, ReportArchiveID, "a\_Och\_B\_Kom");

if (basicParam.AbbreviationEN == "b\_OchZ\_B\_NoIn\_Kom") tmp = pattern2(user, ReportArchiveID, "b\_OchZ\_B\_Kom");

if (basicParam.AbbreviationEN == "c\_Z\_B\_NoIn\_Kom") tmp = pattern2(user, ReportArchiveID, "c\_Z\_B\_Kom");

if (basicParam.AbbreviationEN == "d\_E\_B\_NoIn\_Kom") tmp = pattern2(user, ReportArchiveID, "d\_E\_B\_Kom");

if (basicParam.AbbreviationEN == "OOP\_M") tmp = pattern3(user, ReportArchiveID, 3);

if (basicParam.AbbreviationEN == "kol\_M\_OP") tmp = pattern4(user, ReportArchiveID, 3);

if (basicParam.AbbreviationEN == "kol\_M\_OP\_SV") tmp = pattern5(user, ReportArchiveID, 3);

if (basicParam.AbbreviationEN == "OOP\_M\_SOT") tmp = pattern6(user, ReportArchiveID, 3);

if (basicParam.AbbreviationEN == "OOP\_S") tmp = pattern3(user, ReportArchiveID, 2);

if (basicParam.AbbreviationEN == "kol\_S\_OP") tmp = pattern4(user, ReportArchiveID, 2);

if (basicParam.AbbreviationEN == "kol\_S\_OP\_SV") tmp = pattern5(user, ReportArchiveID, 2);

if (basicParam.AbbreviationEN == "OOP\_S\_SOT") tmp = pattern6(user, ReportArchiveID, 2);

if (basicParam.AbbreviationEN == "OOP\_B") tmp = pattern3(user, ReportArchiveID, 1);

if (basicParam.AbbreviationEN == "kol\_B\_OP") tmp = pattern4(user, ReportArchiveID, 1);

if (basicParam.AbbreviationEN == "kol\_B\_OP\_SV") tmp = pattern5(user, ReportArchiveID, 1);

if (basicParam.AbbreviationEN == "OOP\_B\_SOT") tmp = pattern6(user, ReportArchiveID, 1);

if (basicParam.AbbreviationEN == "OOP\_A") tmp = pattern3(user, ReportArchiveID, 4);

if (basicParam.AbbreviationEN == "kol\_A\_OP") tmp = pattern4(user, ReportArchiveID, 4);

if (basicParam.AbbreviationEN == "kol\_A\_OP\_SV") tmp = pattern5(user, ReportArchiveID, 4);

if (basicParam.AbbreviationEN == "OOP\_A\_SOT") tmp = pattern6(user, ReportArchiveID, 4);

//новые показатели 13.06.2015

if (basicParam.AbbreviationEN == "a\_Och\_M\_CO\_R") tmp =

pattern1(user, ReportArchiveID, 3, "a\_Och\_M\_C", null);

if (basicParam.AbbreviationEN == "b\_OchZ\_M\_CO\_R") tmp =

pattern1(user, ReportArchiveID, 3, "b\_OchZ\_M\_C", null);

if (basicParam.AbbreviationEN == "c\_Z\_M\_CO\_R") tmp =

pattern1(user, ReportArchiveID, 3, "c\_Z\_M\_C", null);

if (basicParam.AbbreviationEN == "d\_E\_M\_CO\_R") tmp =

pattern1(user, ReportArchiveID, 3, "d\_E\_M\_C", null);

if (basicParam.AbbreviationEN == "a\_Och\_B\_CO\_R") tmp =

pattern1(user, ReportArchiveID, 1, "a\_Och\_B\_C", null);

if (basicParam.AbbreviationEN == "d\_E\_B\_CO\_R") tmp =

pattern1(user, ReportArchiveID, 1, "d\_E\_B\_C", null);

if (basicParam.AbbreviationEN == "c\_Z\_B\_CO\_R") tmp =

pattern1(user, ReportArchiveID, 1, "c\_Z\_B\_C", null);

if (basicParam.AbbreviationEN == "d\_E\_B\_CO\_R") tmp =

pattern1(user, ReportArchiveID, 1, "d\_E\_B\_C", null);

if (basicParam.AbbreviationEN == "a\_Och\_S\_CO\_R") tmp =

pattern1(user, ReportArchiveID, 2, "a\_Och\_S\_C", null);

if (basicParam.AbbreviationEN == "b\_OchZ\_S\_CO\_R") tmp =

pattern1(user, ReportArchiveID, 2, "b\_OchZ\_S\_C", null);

if (basicParam.AbbreviationEN == "c\_Z\_S\_CO\_R") tmp =

pattern1(user, ReportArchiveID, 2, "c\_Z\_S\_C", null);

if (basicParam.AbbreviationEN == "d\_E\_S\_CO\_R") tmp =

pattern1(user, ReportArchiveID, 2, "d\_E\_S\_C", null);

//новые показатели

}

else

{

//новейшие показатели 19.06.2015

//// type 0 очное // 1 очное иностранцы // 2 заочное // 3 вечернее

if (basicParam.AbbreviationEN == "a\_Och\_B\_Kom\_money") tmp =

pattern7(FourthLevel.FK\_Specialization, 0, ReportArchiveID, FourthLevel, 1);

if (basicParam.AbbreviationEN == "a\_OchZ\_B\_Kom\_money") tmp =

pattern7(FourthLevel.FK\_Specialization, 3, ReportArchiveID, FourthLevel, 1);

if (basicParam.AbbreviationEN == "a\_Z\_B\_Kom\_money") tmp =

pattern7(FourthLevel.FK\_Specialization, 2, ReportArchiveID, FourthLevel, 1);

if (basicParam.AbbreviationEN == "a\_IN\_B\_Kom\_money") tmp =

pattern7(FourthLevel.FK\_Specialization, 1, ReportArchiveID, FourthLevel, 1);

if (basicParam.AbbreviationEN == "a\_Och\_S\_Kom\_money") tmp =

pattern7(FourthLevel.FK\_Specialization, 0, ReportArchiveID, FourthLevel, 2);

if (basicParam.AbbreviationEN == "a\_OchZ\_S\_Kom\_money") tmp =

pattern7(FourthLevel.FK\_Specialization, 3, ReportArchiveID, FourthLevel, 2);

if (basicParam.AbbreviationEN == "a\_Z\_S\_Kom\_money") tmp =

pattern7(FourthLevel.FK\_Specialization, 2, ReportArchiveID, FourthLevel, 2);

if (basicParam.AbbreviationEN == "a\_IN\_S\_Kom\_money") tmp =

pattern7(FourthLevel.FK\_Specialization, 1, ReportArchiveID, FourthLevel, 2);

if (basicParam.AbbreviationEN == "a\_Och\_M\_Kom\_money") tmp =

pattern7(FourthLevel.FK\_Specialization, 0, ReportArchiveID, FourthLevel, 3);

if (basicParam.AbbreviationEN == "a\_OchZ\_M\_Kom\_money") tmp =

pattern7(FourthLevel.FK\_Specialization, 3, ReportArchiveID, FourthLevel, 3);

if (basicParam.AbbreviationEN == "a\_Z\_M\_Kom\_money") tmp =

pattern7(FourthLevel.FK\_Specialization, 2, ReportArchiveID, FourthLevel, 3);

if (basicParam.AbbreviationEN == "a\_IN\_M\_Kom\_money") tmp =

pattern7(FourthLevel.FK\_Specialization, 1, ReportArchiveID, FourthLevel, 3);

if (basicParam.AbbreviationEN == "a\_Och\_A\_Kom\_money") tmp =

pattern7(FourthLevel.FK\_Specialization, 0, ReportArchiveID, FourthLevel, 4);

if (basicParam.AbbreviationEN == "a\_OchZ\_A\_Kom\_money") tmp =

pattern7(FourthLevel.FK\_Specialization, 3, ReportArchiveID, FourthLevel, 4);

if (basicParam.AbbreviationEN == "a\_Z\_A\_Kom\_money") tmp =

pattern7(FourthLevel.FK\_Specialization, 2, ReportArchiveID, FourthLevel, 4);

if (basicParam.AbbreviationEN == "a\_IN\_A\_Kom\_money") tmp =

pattern7(FourthLevel.FK\_Specialization, 1, ReportArchiveID, FourthLevel, 4);

}

//новейшие показатели

}

else

{

if (basicParam.AbbreviationEN == "Kol\_Kaf\_R")

tmp = Convert.ToDouble((from a in kpiWebDataContext.ThirdLevelParametrs

where a.ThirdLevelParametrsID == user.FK\_ThirdLevelSubdivisionTable

select a.IsBasic).FirstOrDefault());

}

if (tmp < 1000000000000)

{

CollectedBasicParametersTable collectedBasicTmp = new CollectedBasicParametersTable();

int basicParamLevel = (int)(from a in kpiWebDataContext.BasicParametrAdditional

where a.BasicParametrAdditionalID == basicParam.BasicParametersTableID

select a.SubvisionLevel).FirstOrDefault();

if (basicParamLevel == 3)

{

collectedBasicTmp =

(from a in kpiWebDataContext.CollectedBasicParametersTable

where a.FK\_ZeroLevelSubdivisionTable == user.FK\_ZeroLevelSubdivisionTable

&& a.FK\_FirstLevelSubdivisionTable == user.FK\_FirstLevelSubdivisionTable

&& a.FK\_SecondLevelSubdivisionTable == user.FK\_SecondLevelSubdivisionTable

&& a.FK\_ThirdLevelSubdivisionTable == user.FK\_ThirdLevelSubdivisionTable

&& a.FK\_BasicParametersTable == basicParam.BasicParametersTableID

&& a.FK\_ReportArchiveTable == ReportArchiveID

select a).FirstOrDefault();

}

else

{

collectedBasicTmp =

(from a in kpiWebDataContext.CollectedBasicParametersTable

where a.FK\_ZeroLevelSubdivisionTable == user.FK\_ZeroLevelSubdivisionTable

&& a.FK\_FirstLevelSubdivisionTable == user.FK\_FirstLevelSubdivisionTable

&& a.FK\_SecondLevelSubdivisionTable == user.FK\_SecondLevelSubdivisionTable

&& a.FK\_ThirdLevelSubdivisionTable == user.FK\_ThirdLevelSubdivisionTable

&& a.FK\_FourthLevelSubdivisionTable == FourthLevel.FourthLevelSubdivisionTableID

&& a.FK\_BasicParametersTable == basicParam.BasicParametersTableID

&& a.FK\_ReportArchiveTable == ReportArchiveID

select a).FirstOrDefault();

}

if (collectedBasicTmp == null) // надо создать

{

collectedBasicTmp = new CollectedBasicParametersTable();

collectedBasicTmp.Active = true;

collectedBasicTmp.Status = 0;

collectedBasicTmp.FK\_UsersTable = user.UsersTableID;

collectedBasicTmp.FK\_BasicParametersTable = basicParam.BasicParametersTableID;

collectedBasicTmp.FK\_ReportArchiveTable = ReportArchiveID;

collectedBasicTmp.CollectedValue = tmp;

collectedBasicTmp.UserIP = "0.0.0.0";//Dns.GetHostEntry(Dns.GetHostName()).AddressList.Where(ip => ip.AddressFamily == System.Net.Sockets.AddressFamily.InterNetwork).Select(ip => ip.ToString()).FirstOrDefault() ?? "";

collectedBasicTmp.LastChangeDateTime = DateTime.Now;

collectedBasicTmp.SavedDateTime = DateTime.Now;

collectedBasicTmp.FK\_ZeroLevelSubdivisionTable = user.FK\_ZeroLevelSubdivisionTable;

collectedBasicTmp.FK\_FirstLevelSubdivisionTable = user.FK\_FirstLevelSubdivisionTable;

collectedBasicTmp.FK\_SecondLevelSubdivisionTable = user.FK\_SecondLevelSubdivisionTable;

collectedBasicTmp.FK\_ThirdLevelSubdivisionTable = user.FK\_ThirdLevelSubdivisionTable;

if (basicParamLevel == 4)

{

collectedBasicTmp.FK\_FourthLevelSubdivisionTable = FourthLevel.FourthLevelSubdivisionTableID;

}

kpiWebDataContext.CollectedBasicParametersTable.InsertOnSubmit(collectedBasicTmp);

kpiWebDataContext.SubmitChanges();

}

else

{

collectedBasicTmp.CollectedValue = tmp;

kpiWebDataContext.SubmitChanges();

}

}

}

/\*

protected void ConfCalculate(int ReportArchiveID, UsersTable user)

{

KPIWebDataContext kpiWebDataContext = new KPIWebDataContext();

List<BasicParametersTable> calcBasicParams =

(from a in kpiWebDataContext.ReportArchiveAndBasicParametrsMappingTable

join b in kpiWebDataContext.BasicParametersTable

on a.FK\_BasicParametrsTable equals b.BasicParametersTableID

join c in kpiWebDataContext.BasicParametrsAndUsersMapping

on b.BasicParametersTableID equals c.FK\_ParametrsTable

join d in kpiWebDataContext.BasicParametrAdditional

on b.BasicParametersTableID equals d.BasicParametrAdditionalID

where

a.FK\_ReportArchiveTable == ReportArchiveID //из нужного отчёта

&& c.FK\_UsersTable == user.UsersTableID // свяный с пользователем

&& (d.SubvisionLevel == 3 || d.SubvisionLevel == 4)//нужный уровень заполняющего

&& a.Active == true // запись в таблице связей показателя и отчёта активна

&& c.Active == true // запись в таблице связей показателя и пользователей активна

&& d.Calculated == true // этот показатель нужно считать

select b).ToList();

//узнали показатели кафедры(отчёт,разрешенияПользователя,Уровеньвводяшего,вводящийся показатель)

foreach (BasicParametersTable basicParam in calcBasicParams) //пройдемся по показателям

{

CollectedBasicParametersTable collectedBasicTmp =

(from a in kpiWebDataContext.CollectedBasicParametersTable

where a.FK\_ZeroLevelSubdivisionTable == user.FK\_ZeroLevelSubdivisionTable

&& a.FK\_FirstLevelSubdivisionTable == user.FK\_FirstLevelSubdivisionTable

&& a.FK\_SecondLevelSubdivisionTable == user.FK\_SecondLevelSubdivisionTable

&& a.FK\_ThirdLevelSubdivisionTable == user.FK\_ThirdLevelSubdivisionTable

&& a.FK\_BasicParametersTable == basicParam.BasicParametersTableID

&& a.FK\_ReportArchiveTable == ReportArchiveID

select a).FirstOrDefault();

if (collectedBasicTmp != null) // надо создать

{

collectedBasicTmp.Status = 4;

kpiWebDataContext.SubmitChanges();

}

}

}

\*/

protected void CalcCalculate(int ReportArchiveID, UsersTable user)

{

KPIWebDataContext kpiWebDataContext = new KPIWebDataContext();

List<BasicParametersTable> calcBasicParams =

(from a in kpiWebDataContext.ReportArchiveAndBasicParametrsMappingTable

join b in kpiWebDataContext.BasicParametersTable

on a.FK\_BasicParametrsTable equals b.BasicParametersTableID

join c in kpiWebDataContext.BasicParametrsAndUsersMapping

on b.BasicParametersTableID equals c.FK\_ParametrsTable

join d in kpiWebDataContext.BasicParametrAdditional

on b.BasicParametersTableID equals d.BasicParametrAdditionalID

where

a.FK\_ReportArchiveTable == ReportArchiveID //из нужного отчёта

&& c.FK\_UsersTable == user.UsersTableID // свяный с пользователем

&& (d.SubvisionLevel == 3 || d.SubvisionLevel == 4)//нужный уровень заполняющего

&& a.Active == true // запись в таблице связей показателя и отчёта активна

&& c.Active == true // запись в таблице связей показателя и пользователей активна

&& d.Calculated == true // этот показатель нужно считать

select b).ToList();

#region toDelete

/\*

ThirdLevelParametrs Cangrad = (from a in kpiWebDataContext.ThirdLevelParametrs

where a.CanGraduate == true

&& a.ThirdLevelParametrsID == user.FK\_ThirdLevelSubdivisionTable

select a).FirstOrDefault();

if (Cangrad != null) // кафедра выпускает

{

//определим какого типа специальности есть на данной кафедре

bool AnyB = (from a in kpiWebDataContext.FourthLevelSubdivisionTable

join b in kpiWebDataContext.FourthLevelParametrs

on a.FourthLevelSubdivisionTableID equals b.FourthLevelParametrsID

where

a.FK\_ThirdLevelSubdivisionTable == user.FK\_ThirdLevelSubdivisionTable

&& b.SpecType == 1

select a).ToList().Count() > 0 ? true : false;

bool AnyS = (from a in kpiWebDataContext.FourthLevelSubdivisionTable

join b in kpiWebDataContext.FourthLevelParametrs

on a.FourthLevelSubdivisionTableID equals b.FourthLevelParametrsID

where

a.FK\_ThirdLevelSubdivisionTable == user.FK\_ThirdLevelSubdivisionTable

&& b.SpecType == 2

select a).ToList().Count() > 0 ? true : false;

bool AnyM = (from a in kpiWebDataContext.FourthLevelSubdivisionTable

join b in kpiWebDataContext.FourthLevelParametrs

on a.FourthLevelSubdivisionTableID equals b.FourthLevelParametrsID

where

a.FK\_ThirdLevelSubdivisionTable == user.FK\_ThirdLevelSubdivisionTable

&& b.SpecType == 3

select a).ToList().Count() > 0 ? true : false;

bool AnyA = (from a in kpiWebDataContext.FourthLevelSubdivisionTable

join b in kpiWebDataContext.FourthLevelParametrs

on a.FourthLevelSubdivisionTableID equals b.FourthLevelParametrsID

where

a.FK\_ThirdLevelSubdivisionTable == user.FK\_ThirdLevelSubdivisionTable

&& b.SpecType == 4

select a).ToList().Count() > 0 ? true : false;

//узнали показатели кафедры(отчёт,разрешенияПользователя,Уровеньвводяшего,вводящийся показатель)

\*/

#endregion

List<FourthLevelSubdivisionTable> FourtLevels = new List<FourthLevelSubdivisionTable>();

if (user.FK\_ThirdLevelSubdivisionTable != null)

{

FourtLevels = (from a in kpiWebDataContext.FourthLevelSubdivisionTable

where a.FK\_ThirdLevelSubdivisionTable == user.FK\_ThirdLevelSubdivisionTable

&& a.Active == true

select a).ToList();

}

foreach (BasicParametersTable basicParam in calcBasicParams) //пройдемся по показателям

{

int ii = (int)(from a in kpiWebDataContext.BasicParametrAdditional

where a.BasicParametrAdditionalID == basicParam.BasicParametersTableID

select a.SubvisionLevel).FirstOrDefault();

if (ii == 4)

{

if (FourtLevels.Count() > 0) // есть хоть одна специальность

{

foreach (FourthLevelSubdivisionTable CurrentFourth in FourtLevels)

{

patternSwitch(ReportArchiveID, basicParam, CurrentFourth, FourtLevels.Count(), user); // считаем для каждой специальности

}

}

}

else

{

patternSwitch(ReportArchiveID, basicParam, null, FourtLevels.Count(), user);

}

}

// }

}

#endregion

protected void Page\_Load(object sender, EventArgs e)

{

string script = @"<script>

function ConfirmSubmit() {

var msg = confirm('Режим доступа к данным будет изменен на \'только просмотр\'.Отправить данные на утверждение?');

if (msg == true)

{

document.getElementById('LoadPanel\_').style.visibility = 'visible'

return true;

}

else

{

document.getElementById('LoadPanel\_').style.visibility = 'hidden'

return false;

}

}

</script>";

string script2 = @"<script>

function ConfirmSubmitA() {

var msg = confirm('Режим доступа к данным будет изменен на \'только просмотр\'.Подтвердить достоверность данных и отправить их на обработку?');

if (msg == true)

{

document.getElementById('LoadPanel\_').style.visibility = 'visible'

return true;

}

else

{

document.getElementById('LoadPanel\_').style.visibility = 'hidden'

return false;

}

}

</script>";

string script3 = @"<script>

function ConfirmSubmitOn() {

var msg = confirm('Режим доступа к данным будет изменен на \'только просмотр\'.Вернуть отчёт на доработку?');

if (msg == true)

{

document.getElementById('LoadPanel\_').style.visibility = 'visible'

return true;

}

else

{

document.getElementById('LoadPanel\_').style.visibility = 'hidden'

return false;

}

}

</script>";

string script4 = @"<script>

function ConfirmSubmitOnТ() {

var msg = 'Режим доступа к данным будет изменен на \'только просмотр\'. Отправить отчет на утверждение?';

document.location = '../Default.aspx';

return alert (msg);

}

</script>";

string script5 = @"<script>

function ConfirmSubmitOnTT() {

var msg = 'Режим доступа к данным будет изменен на \'только просмотр\'.Отправить отчет на доработку?';

return confirm(msg);

}

</script>";

Serialization UserSer = (Serialization)Session["UserID"];

if (UserSer == null)

{

Response.Redirect("~/Default.aspx");

}

int userID = UserSer.Id;

KPIWebDataContext kPiDataContext = new KPIWebDataContext();

UsersTable userTable =

(from a in kPiDataContext.UsersTable where a.UsersTableID == userID select a).FirstOrDefault();

ViewState["login"] = (from a in kPiDataContext.UsersTable where a.UsersTableID == userID select a.Email).FirstOrDefault();

if (userTable.AccessLevel != 0)

{

Response.Redirect("~/Default.aspx");

}

Serialization paramSerialization = (Serialization)Session["ReportArchiveID"];

if (paramSerialization == null)

{

Response.Redirect("~/Default.aspx");

}

/////////////////////////////////////////////////////////////////////////

///

if (!Page.IsPostBack)

{

Panel mypanel = (Panel)(Master.FindControl("loading"));

mypanel.Visible = true;

ViewState["IsPostBack"] = true;

Page.ClientScript.RegisterClientScriptBlock(typeof(Page), "Script",

"window.onload = function() {\_\_doPostBack(\"\", \"\");};", true);

//Response.Redirect("~/Reports/FillingTheReport.aspx");

}

else

{

//if (ViewState["IsPostBack"] == null)

if ((bool)ViewState["IsPostBack"] == true)

{

#region

Serialization modeSer = (Serialization)Session["mode"];

if (modeSer == null)

{

Response.Redirect("~/Default.aspx");

}

int mode = modeSer.mode; // 0 заполняем // 1 смотрим // 2 смотрим и утверждаем

////////////////

int UserID = UserSer.Id;

int ReportArchiveID;

ReportArchiveID = Convert.ToInt32(paramSerialization.ReportStr);

KPIWebDataContext kpiWebDataContext = new KPIWebDataContext();

UsersTable user = (from a in kpiWebDataContext.UsersTable

where a.UsersTableID == UserID

select a).FirstOrDefault();

int l\_0 = user.FK\_ZeroLevelSubdivisionTable == null ? 0 : (int)user.FK\_ZeroLevelSubdivisionTable;

int l\_1 = user.FK\_FirstLevelSubdivisionTable == null ? 0 : (int)user.FK\_FirstLevelSubdivisionTable;

int l\_2 = user.FK\_SecondLevelSubdivisionTable == null

? 0

: (int)user.FK\_SecondLevelSubdivisionTable;

int l\_3 = user.FK\_ThirdLevelSubdivisionTable == null ? 0 : (int)user.FK\_ThirdLevelSubdivisionTable;

int l\_4 = user.FK\_FourthLevelSubdivisionTable == null

? 0

: (int)user.FK\_FourthLevelSubdivisionTable;

int l\_5 = user.FK\_FifthLevelSubdivisionTable == null ? 0 : (int)user.FK\_FifthLevelSubdivisionTable;

int userLevel = 5;

userLevel = l\_5 == 0 ? 4 : userLevel;

userLevel = l\_4 == 0 ? 3 : userLevel;

userLevel = l\_3 == 0 ? 2 : userLevel;

userLevel = l\_2 == 0 ? 1 : userLevel;

userLevel = l\_1 == 0 ? 0 : userLevel;

userLevel = l\_0 == 0 ? -1 : userLevel;

////ранги пользователя

/// -1 никто ниоткуда/// 0 с Кфу /// 1 с Академии/// 2 с Факультета/// 3 с кафедры/// 4 с специализация/// 5 с под специализацией,пока нет

#endregion

///узнали все о пользователе

#region

List<string> columnNames = new List<string>(); // сюда сохраняем названия колонок

List<string> basicNames = new List<string>(); // сюда названия параметров для excel

/////создаем дататейбл

DataTable dataTable = new DataTable();

dataTable.Columns.Add(new DataColumn("CurrentReportArchiveID", typeof(string)));

dataTable.Columns.Add(new DataColumn("BasicParametersTableID", typeof(string)));

dataTable.Columns.Add(new DataColumn("CollectedBasicParametersTableID", typeof(string)));

dataTable.Columns.Add(new DataColumn("Name", typeof(string)));

dataTable.Columns.Add(new DataColumn("Comment", typeof(string)));

dataTable.Columns.Add(new DataColumn("CommentEnabled", typeof(string)));

for (int k = 0; k <= 40; k++) //создаем кучу полей

{

dataTable.Columns.Add(new DataColumn("Value" + k.ToString(), typeof(string)));

dataTable.Columns.Add(new DataColumn("CollectId" + k.ToString(), typeof(string)));

dataTable.Columns.Add(new DataColumn("NotNull" + k.ToString(), typeof(string)));

}

#endregion

//создали макет дататейбла

int additionalColumnCount = 0;

List<int> StatusList = new List<int>();

#region

if (userLevel != 3)

{

List<BasicParametersTable> BasicParams =

(from a in kpiWebDataContext.ReportArchiveAndBasicParametrsMappingTable

join b in kpiWebDataContext.BasicParametersTable

on a.FK\_BasicParametrsTable equals b.BasicParametersTableID

join c in kpiWebDataContext.BasicParametrsAndUsersMapping

on b.BasicParametersTableID equals c.FK\_ParametrsTable

join d in kpiWebDataContext.BasicParametrAdditional

on b.BasicParametersTableID equals d.BasicParametrAdditionalID

where a.FK\_ReportArchiveTable == ReportArchiveID //из нужного отчёта

&& c.FK\_UsersTable == UserID // свяный с пользователем

&& d.SubvisionLevel == userLevel //нужный уровень заполняющего

&& a.Active == true // запись в таблице связей показателя и отчёта активна

&& (((c.CanEdit == true) && mode == 0)

|| ((c.CanView == true) && mode == 1)

|| ((c.CanConfirm == true) && mode == 2)) // фильтруем по правам пользователя

&& c.Active == true // запись в таблице связей показателя и пользователей активна

&& d.Calculated == false

// этот показатель нужно вводить а не считать

select b).ToList();

//узнали показатели

foreach (BasicParametersTable basicParam in BasicParams) //пройдемся по показателям

{

DataRow dataRow = dataTable.NewRow();

dataRow["CurrentReportArchiveID"] = ReportArchiveID;

dataRow["BasicParametersTableID"] = basicParam.BasicParametersTableID;

dataRow["Name"] = basicParam.Name;

string comment\_ = (from a in kpiWebDataContext.BasicParametrAdditional

where a.BasicParametrAdditionalID == basicParam.BasicParametersTableID

&& a.Active == true

select a.Comment).FirstOrDefault();

if (comment\_!=null)

{

if (comment\_.Length>3)

{

dataRow["Comment"] = comment\_;

dataRow["CommentEnabled"] = "visible";

}

else

{

dataRow["Comment"] = "nun";

dataRow["CommentEnabled"] = "hidden";

}

}

else

{

dataRow["Comment"] = "nun";

dataRow["CommentEnabled"] = "hidden";

}

basicNames.Add(basicParam.Name);

CollectedBasicParametersTable collectedBasicTmp =

(from a in kpiWebDataContext.CollectedBasicParametersTable

where ((a.FK\_ZeroLevelSubdivisionTable == l\_0) || l\_0 == 0)

&& ((a.FK\_FirstLevelSubdivisionTable == l\_1) || l\_1 == 0)

&& ((a.FK\_SecondLevelSubdivisionTable == l\_2) || l\_2 == 0)

&& ((a.FK\_ThirdLevelSubdivisionTable == l\_3) || l\_3 == 0)

&& a.FK\_BasicParametersTable == basicParam.BasicParametersTableID

&& a.FK\_ReportArchiveTable == ReportArchiveID

select a).FirstOrDefault();

if (collectedBasicTmp == null) // надо создать

{

collectedBasicTmp = new CollectedBasicParametersTable();

collectedBasicTmp.Active = true;

collectedBasicTmp.FK\_UsersTable = UserID;

collectedBasicTmp.FK\_BasicParametersTable = basicParam.BasicParametersTableID;

collectedBasicTmp.FK\_ReportArchiveTable = ReportArchiveID;

collectedBasicTmp.CollectedValue = null;

collectedBasicTmp.UserIP =

Dns.GetHostEntry(Dns.GetHostName())

.AddressList.Where(

ip => ip.AddressFamily == System.Net.Sockets.AddressFamily.InterNetwork)

.Select(ip => ip.ToString())

.FirstOrDefault() ?? "";

collectedBasicTmp.LastChangeDateTime = DateTime.Now;

collectedBasicTmp.SavedDateTime = DateTime.Now;

collectedBasicTmp.FK\_ZeroLevelSubdivisionTable = user.FK\_ZeroLevelSubdivisionTable;

collectedBasicTmp.FK\_FirstLevelSubdivisionTable = user.FK\_FirstLevelSubdivisionTable;

collectedBasicTmp.FK\_SecondLevelSubdivisionTable = user.FK\_SecondLevelSubdivisionTable;

collectedBasicTmp.FK\_ThirdLevelSubdivisionTable = user.FK\_ThirdLevelSubdivisionTable;

collectedBasicTmp.Status = 0;

kpiWebDataContext.CollectedBasicParametersTable.InsertOnSubmit(collectedBasicTmp);

kpiWebDataContext.SubmitChanges();

}

dataRow["Value0"] = collectedBasicTmp.CollectedValue.ToString();

dataRow["CollectId0"] = collectedBasicTmp.CollectedBasicParametersTableID.ToString();

dataRow["NotNull0"] = 1.ToString();

dataTable.Rows.Add(dataRow);

if (collectedBasicTmp.Status != null)

{

StatusList.Add((int)collectedBasicTmp.Status);

}

else

{

StatusList.Add(0);

}

}

additionalColumnCount += 1;

}

/\* columnNames.Add("Кафедра:\r\n" + (from a in kpiWebDataContext.ThirdLevelSubdivisionTable

where a.ThirdLevelSubdivisionTableID == user.FK\_ThirdLevelSubdivisionTable

select a.Name).FirstOrDefault());

\* \*/

#endregion

switch (userLevel) // это штука пока будет работать только для пользователя кафедры

{

case 0: //я КФУ

{

columnNames.Add((from a in kpiWebDataContext.ZeroLevelSubdivisionTable

where a.ZeroLevelSubdivisionTableID == user.FK\_ZeroLevelSubdivisionTable

select a.Name).FirstOrDefault());

break;

}

case 1: //Я Акакдемия

{

//"Академия:\r\n" +

columnNames.Add((from a in kpiWebDataContext.FirstLevelSubdivisionTable

where a.FirstLevelSubdivisionTableID == user.FK\_FirstLevelSubdivisionTable

select a.Name).FirstOrDefault());

break;

}

case 2: //я Факультет

{

//"Факультет:\r\n" +

columnNames.Add((from a in kpiWebDataContext.SecondLevelSubdivisionTable

where a.SecondLevelSubdivisionTableID == user.FK\_SecondLevelSubdivisionTable

select a.Name).FirstOrDefault());

break;

}

case 3: //я кафедра

{

#region

List<BasicParametersTable> KafBasicParams =

(from a in kpiWebDataContext.ReportArchiveAndBasicParametrsMappingTable

join b in kpiWebDataContext.BasicParametersTable

on a.FK\_BasicParametrsTable equals b.BasicParametersTableID

join c in kpiWebDataContext.BasicParametrsAndUsersMapping

on b.BasicParametersTableID equals c.FK\_ParametrsTable

join d in kpiWebDataContext.BasicParametrAdditional

on b.BasicParametersTableID equals d.BasicParametrAdditionalID

where

a.FK\_ReportArchiveTable == ReportArchiveID //из нужного отчёта

&& c.FK\_UsersTable == UserID // свяный с пользователем

&& d.SubvisionLevel == 3 //нужный уровень заполняющего

&& a.Active == true // запись в таблице связей показателя и отчёта активна

&& (((c.CanEdit == true) && mode == 0)

|| ((c.CanView == true) && mode == 1)

|| ((c.CanConfirm == true) && mode == 2))

// фильтруем по правам пользователя

&& c.Active == true

// запись в таблице связей показателя и пользователей активна

&& d.Calculated == false

// этот показатель нужно вводить а не считать

select b).ToList();

//узнали показатели кафедры(отчёт,разрешенияПользователя,Уровеньвводяшего,вводящийся показатель)

foreach (BasicParametersTable basicParam in KafBasicParams) //пройдемся по показателям

{

//если этото параметр и эта кафедра дружат

ThirdLevelParametrs thirdParametrs =

(from a in kpiWebDataContext.ThirdLevelParametrs

where a.ThirdLevelParametrsID == l\_3

select a).FirstOrDefault();

// узнали параметры специальности

BasicParametrAdditional basicParametrs =

(from a in kpiWebDataContext.BasicParametrAdditional

where

a.BasicParametrAdditionalID == basicParam.BasicParametersTableID

select a).FirstOrDefault();

//узнали параметры базового показателя

if ((thirdParametrs.CanGraduate == true) || (basicParametrs.IsGraduating == false))

//фильтруем базовые показатели для невыпускающих кафедр

{

DataRow dataRow = dataTable.NewRow();

dataRow["CurrentReportArchiveID"] = ReportArchiveID;

dataRow["BasicParametersTableID"] = basicParam.BasicParametersTableID;

dataRow["Name"] = basicParam.Name;

string comment\_ = (from a in kpiWebDataContext.BasicParametrAdditional

where a.BasicParametrAdditionalID == basicParam.BasicParametersTableID

&& a.Active == true

select a.Comment).FirstOrDefault();

if (comment\_ != null)

{

if (comment\_.Length > 3)

{

dataRow["Comment"] = comment\_;

dataRow["CommentEnabled"] = "visible";

}

else

{

dataRow["Comment"] = "nun";

dataRow["CommentEnabled"] = "hidden";

}

}

else

{

dataRow["Comment"] = "nun";

dataRow["CommentEnabled"] = "hidden";

}

basicNames.Add(basicParam.Name);

CollectedBasicParametersTable collectedBasicTmp =

(from a in kpiWebDataContext.CollectedBasicParametersTable

where a.FK\_ZeroLevelSubdivisionTable == user.FK\_ZeroLevelSubdivisionTable

&&

a.FK\_FirstLevelSubdivisionTable == user.FK\_FirstLevelSubdivisionTable

&&

a.FK\_SecondLevelSubdivisionTable ==

user.FK\_SecondLevelSubdivisionTable

&&

a.FK\_ThirdLevelSubdivisionTable == user.FK\_ThirdLevelSubdivisionTable

&& a.FK\_BasicParametersTable == basicParam.BasicParametersTableID

&& a.FK\_ReportArchiveTable == ReportArchiveID

select a).FirstOrDefault();

if (collectedBasicTmp == null) // надо создать

{

collectedBasicTmp = new CollectedBasicParametersTable();

collectedBasicTmp.Active = true;

collectedBasicTmp.Status = 0;

collectedBasicTmp.FK\_UsersTable = UserID;

collectedBasicTmp.FK\_BasicParametersTable = basicParam.BasicParametersTableID;

collectedBasicTmp.FK\_ReportArchiveTable = ReportArchiveID;

collectedBasicTmp.CollectedValue = null;

collectedBasicTmp.UserIP =

Dns.GetHostEntry(Dns.GetHostName())

.AddressList.Where(

ip =>

ip.AddressFamily ==

System.Net.Sockets.AddressFamily.InterNetwork)

.Select(ip => ip.ToString())

.FirstOrDefault() ?? "";

collectedBasicTmp.LastChangeDateTime = DateTime.Now;

collectedBasicTmp.SavedDateTime = DateTime.Now;

collectedBasicTmp.FK\_ZeroLevelSubdivisionTable =

user.FK\_ZeroLevelSubdivisionTable;

collectedBasicTmp.FK\_FirstLevelSubdivisionTable =

user.FK\_FirstLevelSubdivisionTable;

collectedBasicTmp.FK\_SecondLevelSubdivisionTable =

user.FK\_SecondLevelSubdivisionTable;

collectedBasicTmp.FK\_ThirdLevelSubdivisionTable =

user.FK\_ThirdLevelSubdivisionTable;

kpiWebDataContext.CollectedBasicParametersTable.InsertOnSubmit(collectedBasicTmp);

kpiWebDataContext.SubmitChanges();

}

dataRow["Value0"] = collectedBasicTmp.CollectedValue.ToString();

dataRow["CollectId0"] = collectedBasicTmp.CollectedBasicParametersTableID.ToString();

dataRow["NotNull0"] = 1.ToString();

dataTable.Rows.Add(dataRow);

if (collectedBasicTmp.Status != null)

{

StatusList.Add((int)collectedBasicTmp.Status);

}

else

{

StatusList.Add(0);

}

}

}

columnNames.Add((from a in kpiWebDataContext.ThirdLevelSubdivisionTable

where a.ThirdLevelSubdivisionTableID == user.FK\_ThirdLevelSubdivisionTable

select a.Name).FirstOrDefault());

#endregion

//Кафедра готова

additionalColumnCount += 1;

#region

if ((from zz in kpiWebDataContext.ThirdLevelParametrs

where zz.ThirdLevelParametrsID == l\_3

select zz.CanGraduate).FirstOrDefault() == true)

// кафедра выпускающая значит специальности есть

{

List<BasicParametersTable> SpecBasicParams =

(from a in kpiWebDataContext.ReportArchiveAndBasicParametrsMappingTable

join b in kpiWebDataContext.BasicParametersTable

on a.FK\_BasicParametrsTable equals b.BasicParametersTableID

join c in kpiWebDataContext.BasicParametrsAndUsersMapping

on b.BasicParametersTableID equals c.FK\_ParametrsTable

join d in kpiWebDataContext.BasicParametrAdditional

on b.BasicParametersTableID equals d.BasicParametrAdditionalID

where a.FK\_ReportArchiveTable == ReportArchiveID //для отчёта

&& d.SubvisionLevel == 4 // для уровня заполняющего

&& d.Calculated == false //только вводимые параметры

&& c.FK\_UsersTable == UserID // связаннаые с пользователем

&& a.Active == true

&& (((c.CanEdit == true) && mode == 0)

|| ((c.CanView == true) && mode == 1)

|| ((c.CanConfirm == true) && mode == 2))

// фильтруем по правам пользователя

&& c.Active == true

select b).ToList();

//Получили показатели разрешенные пользователю в данном отчёте

List<FourthLevelSubdivisionTable> Specialzations =

(from a in kpiWebDataContext.FourthLevelSubdivisionTable

where a.FK\_ThirdLevelSubdivisionTable == l\_3

&& a.Active == true

select a).ToList();

//Получили список специальностей для кафедры под пользователем

foreach (FourthLevelSubdivisionTable spec in Specialzations)

{

/\*

columnNames.Add("Направление подготовки\r" +

(from a in kpiWebDataContext.SpecializationTable

where a.SpecializationTableID == spec.FK\_Specialization

select a.Name).FirstOrDefault().ToString() +" : "+

(from a in kpiWebDataContext.SpecializationTable

where a.SpecializationTableID == spec.FK\_Specialization

select a.SpecializationNumber).FirstOrDefault().ToString());

\*/

string CurrentColumnName = "<div style=\"transform:rotate(90deg);\">" + (from a in kpiWebDataContext.SpecializationTable

where a.SpecializationTableID == spec.FK\_Specialization

select a.SpecializationNumber).FirstOrDefault().ToString() + "</div>";

columnNames.Add(CurrentColumnName);

//запомнили название специальности // оно нам пригодится)

}

foreach (BasicParametersTable specBasicParam in SpecBasicParams)

{

int i = additionalColumnCount;

DataRow dataRow = dataTable.NewRow();

BasicParametrAdditional basicParametrs =

(from a in kpiWebDataContext.BasicParametrAdditional

where

a.BasicParametrAdditionalID == specBasicParam.BasicParametersTableID

select a).FirstOrDefault();

//узнали параметры базового показателя

int j = 0;

//если хоть одной специальности базовый показатель нужен то мы его выведем

foreach (FourthLevelSubdivisionTable spec in Specialzations)

{

FourthLevelParametrs fourthParametrs =

(from a in kpiWebDataContext.FourthLevelParametrs

where a.FourthLevelParametrsID == spec.FourthLevelSubdivisionTableID

select a).FirstOrDefault();

// узнали параметры специальности

//если этото параметр и эта специальность дружат

if (((fourthParametrs.IsForeignStudentsAccept == true) ||

(basicParametrs.ForForeignStudents == false)) //это для иностранцев

&&

((fourthParametrs.SpecType == basicParametrs.SpecType) ||

(basicParametrs.SpecType == 0)))

// это для деления на магистров аспирантов итд

{

j++; //потом проверка и следовательно БП нуно выводить

CollectedBasicParametersTable collectedBasicTmp =

(from a in kpiWebDataContext.CollectedBasicParametersTable

where

a.FK\_BasicParametersTable ==

specBasicParam.BasicParametersTableID

&& a.FK\_ReportArchiveTable == ReportArchiveID

&&

(a.FK\_ZeroLevelSubdivisionTable ==

user.FK\_ZeroLevelSubdivisionTable)

&&

(a.FK\_FirstLevelSubdivisionTable ==

user.FK\_FirstLevelSubdivisionTable)

&&

(a.FK\_SecondLevelSubdivisionTable ==

user.FK\_SecondLevelSubdivisionTable)

&&

(a.FK\_ThirdLevelSubdivisionTable ==

user.FK\_ThirdLevelSubdivisionTable)

&&

(a.FK\_FourthLevelSubdivisionTable ==

spec.FourthLevelSubdivisionTableID)

select a).FirstOrDefault();

if (collectedBasicTmp == null)

{

collectedBasicTmp = new CollectedBasicParametersTable();

collectedBasicTmp.Active = true;

collectedBasicTmp.Status = 0;

collectedBasicTmp.FK\_UsersTable = UserID;

collectedBasicTmp.FK\_BasicParametersTable =

specBasicParam.BasicParametersTableID;

collectedBasicTmp.FK\_ReportArchiveTable = ReportArchiveID;

collectedBasicTmp.CollectedValue = null;

collectedBasicTmp.UserIP =

Dns.GetHostEntry(Dns.GetHostName())

.AddressList.Where(

ip =>

ip.AddressFamily ==

System.Net.Sockets.AddressFamily.InterNetwork)

.Select(ip => ip.ToString())

.FirstOrDefault() ?? "";

collectedBasicTmp.LastChangeDateTime = DateTime.Now;

collectedBasicTmp.SavedDateTime = DateTime.Now;

collectedBasicTmp.FK\_ZeroLevelSubdivisionTable =

user.FK\_ZeroLevelSubdivisionTable;

collectedBasicTmp.FK\_FirstLevelSubdivisionTable =

user.FK\_FirstLevelSubdivisionTable;

collectedBasicTmp.FK\_SecondLevelSubdivisionTable =

user.FK\_SecondLevelSubdivisionTable;

collectedBasicTmp.FK\_ThirdLevelSubdivisionTable =

spec.FK\_ThirdLevelSubdivisionTable;

collectedBasicTmp.FK\_FourthLevelSubdivisionTable =

spec.FourthLevelSubdivisionTableID;

kpiWebDataContext.CollectedBasicParametersTable.InsertOnSubmit(

collectedBasicTmp);

kpiWebDataContext.SubmitChanges();

}

dataRow["Value" + i] = collectedBasicTmp.CollectedValue.ToString();

dataRow["CollectId" + i] =

collectedBasicTmp.CollectedBasicParametersTableID.ToString();

dataRow["NotNull" + i] = 1.ToString();

if (collectedBasicTmp.Status != null)

{

StatusList.Add((int)collectedBasicTmp.Status);

}

else

{

StatusList.Add(0);

}

}

i++;

}

if (j > 0)

{

basicNames.Add(specBasicParam.Name);

dataRow["Name"] = specBasicParam.Name;

dataRow["CurrentReportArchiveID"] = ReportArchiveID;

dataRow["BasicParametersTableID"] = specBasicParam.BasicParametersTableID;

string comment\_ = (from a in kpiWebDataContext.BasicParametrAdditional

where a.BasicParametrAdditionalID == specBasicParam.BasicParametersTableID

&& a.Active == true

select a.Comment).FirstOrDefault();

if (comment\_ != null)

{

if (comment\_.Length > 3)

{

dataRow["Comment"] = comment\_;

dataRow["CommentEnabled"] = "visible";

}

else

{

dataRow["Comment"] = "nun";

dataRow["CommentEnabled"] = "hidden";

}

}

else

{

dataRow["Comment"] = "nun";

dataRow["CommentEnabled"] = "hidden";

}

dataTable.Rows.Add(dataRow);

}

///////////////////////закинули все в дататейбл

}

additionalColumnCount += Specialzations.Count;

}

#endregion

// специальности готовы

break;

}

#region

case 4: //пока рано//у нас нет ничего глубже специальности

{

break;

}

case 5: //выводить нечего

{

break;

}

default: // если уровня вообще нет или он неправильно задан//хорошо бы ошибку вывести

{

break;

}

#endregion

//неиспользуемая часть свича

}

ViewState["CollectedBasicParametersTable"] = dataTable;

ViewState["CurrentReportArchiveID"] = ReportArchiveID;

ViewState["ValueColumnCnt"] = additionalColumnCount;

ViewState["ColumnName"] = columnNames;

ViewState["basicNames"] = basicNames;

int tmpStatCount = 0;

foreach (int tmpStat in StatusList)

{

if (tmpStat == 2)

{

tmpStatCount++;

}

}

//определение дней

DateTime endDate = (DateTime)(from a in kpiWebDataContext.ReportArchiveTable

where a.ReportArchiveTableID == ReportArchiveID

select a.EndDateTime).FirstOrDefault();

if (endDate == null)

{

endDate = DateTime.Now.AddDays(2);

}

DateTime startDate = DateTime.Now;

int dateCount = 0;

while (startDate < endDate)

{

startDate = startDate.AddDays(1);

dateCount++;

}

// определение дней

if (mode == 0)

{

GoBackButton.Visible = true;

GoBackButton.Text = "Вернуться без сохранения";

ButtonSave.Visible = true;

ButtonSave.Text = "Сохранить внесенные данные";

UpnDownButton.Visible = true;

UpnDownButton.Text = "Отправить отчёт на утверждение";

TextBox1.Visible = false;

if (StatusList[0] == 1)

{

Label1.Text = "Данные возвращены на доработку. Проверьте корректность введенных данных";

}

else

{

if (tmpStatCount == StatusList.Count())

{

Label1.Text = "Все показатели заполнены. Необходимо отправить отчёт на утверждение";

Label3.Text = "Все показатели заполнены. Необходимо отправить отчёт на утверждение";

UpnDownButton.Enabled = true;

}

else

{

Label1.Text = "Заполнено " + tmpStatCount + " показателей из " + StatusList.Count();

Label3.Text = "Заполнено " + tmpStatCount + " показателей из " + StatusList.Count();

UpnDownButton.Enabled = false;

}

}

ViewState["AllCnt"] = StatusList.Count();

Label2.Text = "Осталось " + dateCount + " дней до закрытия отчёта";

Page.ClientScript.RegisterClientScriptBlock(this.GetType(),

"Confirm", script);

UpnDownButton.Attributes.Add("OnClick", "return ConfirmSubmit();");

Page.ClientScript.RegisterClientScriptBlock(typeof(Page), "Script",

"window.onbeforeunload = function() { return 'Date will be lost: are you sure?'; };", true);

ButtonSave.OnClientClick = "window.onbeforeunload = null; document.getElementById('LoadPanel\_').style.visibility = 'visible'; ";

Button1.OnClientClick = "window.onbeforeunload = null; document.getElementById('LoadPanel\_').style.visibility = 'visible'; ";

UpnDownButton.OnClientClick = "window.onbeforeunload = null; document.getElementById('LoadPanel\_').style.visibility = 'visible'; ";

GoBackButton.OnClientClick="window.onbeforeunload = null; document.getElementById('LoadPanel\_').style.visibility = 'visible'; ";

}

else if

(mode == 1)

{

GoBackButton.Visible = true;

GoBackButton.Text = "Вернуться в меню";

ButtonSave.Visible = false;

UpnDownButton.Visible = false;

TextBox1.Visible = false;

if ((StatusList[0] == 0) || (StatusList[0] == 2))

{

Label1.Text = "Заполнено " + tmpStatCount + " показателей из " + StatusList.Count();

Label3.Text = "Заполнено " + tmpStatCount + " показателей из " + StatusList.Count();

}

else if (StatusList[0] == 1)

{

Label1.Text = "Данные возвращены на доработку";

Label2.Text = "Данные возвращены на доработку";

}

else if (StatusList[0] == 3)

{

Label1.Text = "Данные отправлены на утверждение";

Label2.Text = "Данные отправлены на утверждение";

}

else if (StatusList[0] == 4)

{

Label1.Text = "Данные утверждены";

Label2.Text = "Данные утверждены";

}

Label2.Text = "Осталось " + dateCount + " дней до закрытия отчёта";

//Label2.Visible = false;

// OnClientClick="javascript:return confirm('Do you really want to \ndelete the item?');"

}

else if (mode == 2)

{

GoBackButton.Visible = true;

GoBackButton.Text = "Вернуться в меню без утверждения";

ButtonSave.Visible = true;

ButtonSave.Text = "Утвердить данные";

UpnDownButton.Visible = true;

UpnDownButton.Text = "Вернуть отчёт на доработку";

TextBox1.Visible = true;

Label1.Text = "Утверждение данных";

Label2.Text = "Осталось " + dateCount + " дней до закрытия отчёта";

Page.ClientScript.RegisterClientScriptBlock(this.GetType(),

"Confirm", script2);

Page.ClientScript.RegisterClientScriptBlock(this.GetType(),

"ConfirmOn", script3);

ButtonSave.Attributes.Add("OnClick", "return ConfirmSubmitA();");

UpnDownButton.Attributes.Add("OnClick", "return ConfirmSubmitOn();");

}

GridviewCollectedBasicParameters.DataSource = dataTable;

for (int j = 0; j < additionalColumnCount; j++)

{

GridviewCollectedBasicParameters.Columns[j + 4].Visible = true;

GridviewCollectedBasicParameters.Columns[j + 4].HeaderText = columnNames[j];

}

GridviewCollectedBasicParameters.DataBind();

}

Page.ClientScript.RegisterClientScriptBlock(typeof(Page), "Script",

"window.onload = null", true);

Panel mypanel = (Panel)(Master.FindControl("loading"));

ViewState["IsPostBack"] = false;

mypanel.Visible = false;

}

}

protected void ButtonSave\_Click(object sender, EventArgs e) //сохранение данных и пожтверждение данных

{

Serialization UserSer = (Serialization)Session["UserID"];

if (UserSer == null)

{

Response.Redirect("~/Default.aspx");

}

Serialization paramSerialization = (Serialization)Session["ReportArchiveID"];

if (paramSerialization == null)

{

Response.Redirect("~/Default.aspx");

}

if (ViewState["CollectedBasicParametersTable"] != null && ViewState["CurrentReportArchiveID"] != null)

{

KPIWebDataContext KPIWebDataContext = new KPIWebDataContext();

Dictionary<int, double> tempDictionary = new Dictionary<int, double>();

int UserID = UserSer.Id;

int ReportArchiveID;

ReportArchiveID = Convert.ToInt32(paramSerialization.ReportStr);

UsersTable user = (from a in KPIWebDataContext.UsersTable

where a.UsersTableID == UserSer.Id

select a).FirstOrDefault();

DataTable collectedBasicParametersTable = (DataTable)ViewState["CollectedBasicParametersTable"];

int columnCnt = (int)ViewState["ValueColumnCnt"];

Serialization modeSer = (Serialization)Session["mode"];

if (modeSer == null)

{

Response.Redirect("~/Default.aspx");

}

int mode = modeSer.mode;

if (mode == 0) //сохранение данных

{

#region save data

//int allCnt=0;

int notNullCnt = 0;

if (collectedBasicParametersTable.Rows.Count > 0)

{

int rowIndex = 0;

for (int i = 1; i <= collectedBasicParametersTable.Rows.Count; i++) //в каждой строчке

{

/// //сохраним вложенные данные

for (int k = 0; k < columnCnt; k++) // пройдемся по каждой колонке

{

TextBox textBox = (TextBox)GridviewCollectedBasicParameters.Rows[rowIndex].FindControl("Value" + k.ToString());

Label label = (Label)GridviewCollectedBasicParameters.Rows[rowIndex].FindControl("CollectId" + k.ToString());

if (textBox != null && label != null)

{

//allCnt++;

double collectedValue = double.NaN;

if (textBox.Text.IsFloat())

{

notNullCnt++;

collectedValue = Convert.ToInt32(textBox.Text);

}

int collectedBasicParametersTableID = -1;

if (int.TryParse(label.Text, out collectedBasicParametersTableID) &&

collectedBasicParametersTableID > -1)

tempDictionary.Add(collectedBasicParametersTableID, collectedValue);

}

}

rowIndex++;

}

}

if (tempDictionary.Count > 0)

{

//Список ранее введенных пользователем данных для данной кампании (отчёта)

List<CollectedBasicParametersTable> сollectedBasicParametersTable =

(from collectedBasicParameters in KPIWebDataContext.CollectedBasicParametersTable

where (from item in tempDictionary select item.Key).ToList()

.Contains((int)collectedBasicParameters.CollectedBasicParametersTableID)

select collectedBasicParameters).ToList();

string localIP = Dns.GetHostEntry(Dns.GetHostName()).AddressList.Where(ip => ip.AddressFamily == System.Net.Sockets.AddressFamily.InterNetwork).Select(ip => ip.ToString()).FirstOrDefault() ?? "";

foreach (var сollectedBasicParameter in сollectedBasicParametersTable)

{

double tmpD = (from item in tempDictionary

where item.Key == сollectedBasicParameter.CollectedBasicParametersTableID

select item.Value).FirstOrDefault();

if (double.IsNaN(tmpD))

{

сollectedBasicParameter.CollectedValue = null;

}

else

{

сollectedBasicParameter.CollectedValue = tmpD;

}

сollectedBasicParameter.LastChangeDateTime = DateTime.Now;

сollectedBasicParameter.UserIP = localIP;

сollectedBasicParameter.Status = сollectedBasicParameter.CollectedValue == null ? 0 : 2;

}

KPIWebDataContext.SubmitChanges();

}

//надо рассчитать рассчетные

CalcCalculate(ReportArchiveID, user);

int AllCnt = (int)ViewState["AllCnt"];

if (AllCnt == notNullCnt)

{

LogHandler.LogWriter.WriteLog(LogCategory.INFO, "0RT0: User " + (string)ViewState["login"] + " save data in report ID = " + paramSerialization.ReportStr + "All indicators are filled" + " from ip: "+Dns.GetHostEntry(Dns.GetHostName()).AddressList.Where(ip => ip.AddressFamily == System.Net.Sockets.AddressFamily.InterNetwork).Select(ip => ip.ToString()).FirstOrDefault());

ClientScript.RegisterStartupScript(this.GetType(), "alert", "alert('Все показатели заполнены. Необходимо отправить отчёт на утверждение');" +

"document.location = '../Reports\_/FillingTheReport.aspx';", true);

}

else

{

LogHandler.LogWriter.WriteLog(LogCategory.INFO, "0RT1: User " + (string)ViewState["login"] + " save data in report ID = " + paramSerialization.ReportStr + "Filled " + notNullCnt + " indicators from " + AllCnt + " Ip: "+Dns.GetHostEntry(Dns.GetHostName()).AddressList.Where(ip => ip.AddressFamily == System.Net.Sockets.AddressFamily.InterNetwork).Select(ip => ip.ToString()).FirstOrDefault());

ClientScript.RegisterStartupScript(this.GetType(), "alert", "alert('Данные сохранены на сервере. Заполнено " + notNullCnt + " показателей из " + AllCnt + ", для отправки отчёта необходимо заполнить еще " + (AllCnt - notNullCnt) + " показателя.');" +

"document.location = '../Reports\_/FillingTheReport.aspx';", true);

}

#endregion

}

else if (mode == 1) // просмотр

{

Response.Redirect("~/Reports\_/ChooseReport.aspx");

}

else if (mode == 2) // подтверждение

{

#region confirm all

if (GridviewCollectedBasicParameters.Rows.Count > 0)

{

List<CollectedBasicParametersTable> CollectedToChange = (from a in KPIWebDataContext.CollectedBasicParametersTable

where a.FK\_ThirdLevelSubdivisionTable == user.FK\_ThirdLevelSubdivisionTable

&& a.FK\_ReportArchiveTable == ReportArchiveID

&& a.Active == true

select a).ToList();

foreach (CollectedBasicParametersTable CollectedBasic in CollectedToChange)

{

CollectedBasic.Status = 4;

}

KPIWebDataContext.SubmitChanges();

/\*

for (int k = 0; k < columnCnt; k++) // пройдемся по каждой колонке

{

for (int i = 0; i < GridviewCollectedBasicParameters.Rows.Count; i++)

{

Label label =

(Label)

GridviewCollectedBasicParameters.Rows[i].FindControl("CollectId" + k.ToString());

if (label != null)

{

if (label.Text == "")

{

//error

}

else

{

CollectedBasicParametersTable tmpColTable =

(from a in KPIWebDataContext.CollectedBasicParametersTable

where a.CollectedBasicParametersTableID == Convert.ToInt32(label.Text)

select a).FirstOrDefault();

if (tmpColTable != null)

{

if (tmpColTable.Status == 3)

{

tmpColTable.Status = 4;

KPIWebDataContext.SubmitChanges();

}

else

{

LogHandler.LogWriter.WriteLog(LogCategory.ERROR, "0FRE9: Пользователь " + (string)ViewState["login"] + " сгенерировал ошибку 9 в отчете с ID = ");

}

}

else

{

LogHandler.LogWriter.WriteLog(LogCategory.ERROR, "0FRE10:Пользователь " + (string)ViewState["login"] + " сгенерировал ошибку 10 в отчете с ID = ");

}

}

}

}

}\*/

LogHandler.LogWriter.WriteLog(LogCategory.INFO, "0RT3: User " + (string)ViewState["login"] + " confirm data in report ID = " + paramSerialization.ReportStr + " from ip: "+Dns.GetHostEntry(Dns.GetHostName()).AddressList.Where(ip => ip.AddressFamily == System.Net.Sockets.AddressFamily.InterNetwork).Select(ip => ip.ToString()).FirstOrDefault());

ClientScript.RegisterStartupScript(this.GetType(), "alert", "alert('Вы утвердили данные всех базовых показателей. Отчёт отправлен и доступен только в режиме \"Просмотр\".');" +

"document.location = '../Default.aspx';", true);

#region

UsersTable UserToSend = (from a in KPIWebDataContext.UsersTable

where a.UsersTableID == UserID

select a).FirstOrDefault();

ReportArchiveTable CurrentReport = (from a in KPIWebDataContext.ReportArchiveTable

where a.ReportArchiveTableID == Convert.ToInt32(paramSerialization.ReportStr)

select a).FirstOrDefault();

if (UserToSend == null )

{

}

else

{

string reportName = "-";

if (CurrentReport != null)

{

reportName = CurrentReport.Name;

}

string pdfPath = CreatePdf();

EmailTemplate EmailParams = (from a in KPIWebDataContext.EmailTemplate

where a.Name == "DataConfirmed"

&& a.Active == true

select a).FirstOrDefault();

if (EmailParams != null)

{

Action.MassMailing(UserToSend.Email, EmailParams.EmailTitle,

EmailParams.EmailContent.Replace("#SiteName#", ConfigurationManager.AppSettings.Get("SiteName")).Replace("#ReportName#", reportName), pdfPath);

BasicParametersTable BasicConnectedToUser = (from a in KPIWebDataContext.BasicParametersTable

join b in KPIWebDataContext.BasicParametrsAndUsersMapping

on a.BasicParametersTableID equals b.FK\_ParametrsTable

where b.FK\_UsersTable == UserToSend.UsersTableID

&& b.CanEdit == true

&& b.Active == true

&& a.Active == true

select a).FirstOrDefault();

UsersTable UserToSend2 = (from a in KPIWebDataContext.UsersTable

join b in KPIWebDataContext.BasicParametrsAndUsersMapping

on a .UsersTableID equals b.FK\_UsersTable

where b.FK\_ParametrsTable == BasicConnectedToUser.BasicParametersTableID

&& b.CanConfirm == true

&& a.Active == true

&& b.Active == true

&& (( a.FK\_FirstLevelSubdivisionTable == UserToSend.FK\_FirstLevelSubdivisionTable )||UserToSend.FK\_FirstLevelSubdivisionTable==null)

&& (( a.FK\_SecondLevelSubdivisionTable == UserToSend.FK\_SecondLevelSubdivisionTable)||UserToSend.FK\_SecondLevelSubdivisionTable==null)

&& (( a.FK\_ThirdLevelSubdivisionTable == UserToSend.FK\_ThirdLevelSubdivisionTable)||UserToSend.FK\_ThirdLevelSubdivisionTable==null)

&& (( a.FK\_FourthLevelSubdivisionTable == UserToSend.FK\_FourthLevelSubdivisionTable)||UserToSend.FK\_FourthLevelSubdivisionTable==null)

&& (( a.FK\_FifthLevelSubdivisionTable == UserToSend.FK\_FifthLevelSubdivisionTable)||UserToSend.FK\_FifthLevelSubdivisionTable==null)

select a).FirstOrDefault();

if (UserToSend2.Email != UserToSend.Email) // если одно мыло на двоих то не отправляем 2 письмо

Action.MassMailing(UserToSend2.Email, EmailParams.EmailTitle,

EmailParams.EmailContent.Replace("#SiteName#", ConfigurationManager.AppSettings.Get("SiteName")).Replace("#ReportName#", reportName), pdfPath);

}

}

#endregion

}

else

{

//error

LogHandler.LogWriter.WriteLog(LogCategory.ERROR, "0RTE1: Пользователь " + (string)ViewState["login"] + " сгенерировал ошибку 1 в отчете с ID = " + paramSerialization.ReportStr);

ClientScript.RegisterStartupScript(this.GetType(), "alert", "alert('Ошибка'); document.location = '../Default.aspx'; ", true);

}

#endregion

}

else

{

LogHandler.LogWriter.WriteLog(LogCategory.ERROR, "0RTE2 User " + (string)ViewState["login"] + " generate an error 2 in report c ID = " + paramSerialization.ReportStr);

//error

}

}

}

protected void GridviewCollectedBasicParameters\_RowDataBound(object sender, GridViewRowEventArgs e)

{

Color color;

Color confirmedColor = System.Drawing.Color.LimeGreen;

Color disableColor = System.Drawing.Color.LightGray;

if (col\_ == 0)

{

col\_ = 1;

color = System.Drawing.Color.FloralWhite;

}

else

{

col\_ = 0;

color = System.Drawing.Color.GhostWhite;

}

int rowIndex = 0;

e.Row.BackColor = color;

Serialization modeSer = (Serialization)Session["mode"];

if (modeSer == null)

{

Response.Redirect("~/Default.aspx");

}

int mode = modeSer.mode;

int columnCnt = (int)ViewState["ValueColumnCnt"];

KPIWebDataContext kpiWebDataContext = new KPIWebDataContext();

for (int i = 1; i <= columnCnt; i++)

{

{

var lblMinutes = e.Row.FindControl("Value" + rowIndex) as TextBox;

var NotNullLbl = e.Row.FindControl("NotNull" + rowIndex) as Label;

if (NotNullLbl != null)

{

if (NotNullLbl.Text.Count() == 0)

{

lblMinutes.Visible = false;

if (e.Row.RowType == DataControlRowType.DataRow)

{

DataControlFieldCell d = lblMinutes.Parent as DataControlFieldCell;

// lblMinutes.Attributes.Add("OnChange", "textChanged()");

d.BackColor = disableColor;

//d.CssClass = "DisableClass";

}

}

else

{

Label lbl = e.Row.FindControl("CollectId" + rowIndex) as Label;

RangeValidator Validator = e.Row.FindControl("Validate" + rowIndex) as RangeValidator;

int Status = Convert.ToInt32((from a in kpiWebDataContext.CollectedBasicParametersTable

where a.CollectedBasicParametersTableID == Convert.ToInt32(lbl.Text)

select a.Status).FirstOrDefault());

if (mode == 0) // редактировать

{

#region edit

int type = Convert.ToInt32((from a in kpiWebDataContext.CollectedBasicParametersTable

join b in kpiWebDataContext.BasicParametrAdditional

on a.FK\_BasicParametersTable equals b.BasicParametrAdditionalID

where a.CollectedBasicParametersTableID == Convert.ToInt32(lbl.Text)

select b.DataType).FirstOrDefault());

if (Status == 4) // данные подтверждены первым уровнем

{

lblMinutes.ReadOnly = true;

DataControlFieldCell d = lblMinutes.Parent as DataControlFieldCell;

d.BackColor = confirmedColor;

lblMinutes.BackColor = confirmedColor;

if (Validator != null)

{

Validator.Enabled = false;

}

}

else

{

DataControlFieldCell d = lblMinutes.Parent as DataControlFieldCell;

d.BackColor = color;

lblMinutes.BackColor = color;

#region validator choose

if (Validator != null)

{

if (type == 0)

{

Validator.MinimumValue = "0";

Validator.MaximumValue = "1";

Validator.Type = ValidationDataType.Integer;

Validator.Text = "Только 0 или 1";

}

if (type == 1)

{

Validator.MinimumValue = "0";

Validator.MaximumValue = "1000000";

Validator.Type = ValidationDataType.Integer;

Validator.Text = "Только целочисленное значение";

}

if (type == 2)

{

Validator.MinimumValue = "0";

Validator.MaximumValue = "1000000000000";

Validator.Type = ValidationDataType.Double;

Validator.Text = "Только цифры и запятая";

}

}

#endregion

}

#endregion

}

else if (mode == 1) //смотреть

{

#region view

lblMinutes.ReadOnly = true;

lblMinutes.BackColor = color;

/\*

Color tmpColor = Color.Red;

if (Status == 0) // не должны сюда попадать

{

tmpColor = Color.Blue;

}

else if (Status == 1) // вернули на доработку

{

tmpColor = Color.Orange;

}

else if (Status == 2) // данные внесены

{

tmpColor = Color.Yellow;

}

else if (Status == 3)// данные отпралены на верификацию

{

tmpColor = Color.GreenYellow;

}

else if (Status == 4) // данные верифицированы

{

tmpColor = Color.Green;

}

DataControlFieldCell d = lblMinutes.Parent as DataControlFieldCell;

d.BackColor = tmpColor;

lblMinutes.BackColor = tmpColor;

\*/

if (Validator != null)

{

Validator.Enabled = false;

}

#endregion

}

else if (mode == 2) // утверждать

{

#region confirm

lblMinutes.ReadOnly = true;

Validator.Enabled = false;

lblMinutes.BackColor = color;

/\* // страница подтверждения с чексбоксами

var chBox = e.Row.FindControl("Checked" + rowIndex) as CheckBox;

chBox.Visible = true;

lblMinutes.ReadOnly = true;

Validator.MinimumValue = "0";

Validator.MaximumValue = "10000000";

Validator.Type = ValidationDataType.Double;

Validator.Text = "Невозможно утвердить";

if (Status == 4) // данные подтверждены

{

DataControlFieldCell d = lblMinutes.Parent as DataControlFieldCell;

d.BackColor = confirmedColor;

lblMinutes.BackColor = confirmedColor;

chBox.Checked = true;

chBox.Enabled = false;

chBox.BackColor = confirmedColor;

}

else

{

//lblMinutes.ReadOnly = false;

DataControlFieldCell d = lblMinutes.Parent as DataControlFieldCell;

d.BackColor = color;

lblMinutes.BackColor = color;

chBox.Checked = false;

chBox.Enabled = true;

chBox.BackColor = color;

}\*/

#endregion

}

}

}

rowIndex++;

}

}

}

protected void GridviewCollectedBasicParameters\_SelectedIndexChanged(object sender, EventArgs e)

{

}

protected void GridviewCollectedBasicParameters\_SelectedIndexChanging(object sender, GridViewSelectEventArgs e)

{

}

protected void GridviewCollectedBasicParameters\_PageIndexChanging(object sender, GridViewPageEventArgs e)

{

}

public string CreatePdf()

{

int[] Widhts = new int[40];

for (int i = 0; i < 40; i++)

Widhts[i] = 0;

Widhts[0] = 2;

Widhts[2] = 10;

int colcnt = (int)ViewState["ValueColumnCnt"];

for (int i = 0; i < colcnt; i++)

{

Widhts[i + 4] = 2;

}

Serialization UserSer = (Serialization)Session["UserID"];

if (UserSer == null)

{

Response.Redirect("~/Default.aspx");

}

int userID = UserSer.Id;

KPIWebDataContext kPiDataContext = new KPIWebDataContext();

UsersTable userTable =

(from a in kPiDataContext.UsersTable where a.UsersTableID == userID select a).FirstOrDefault();

Serialization paramSerialization = (Serialization)Session["ReportArchiveID"];

if (paramSerialization == null)

{

Response.Redirect("~/Default.aspx");

}

ReportArchiveTable CurrentReport = (from a in kPiDataContext.ReportArchiveTable

where a.ReportArchiveTableID == Convert.ToInt32(paramSerialization.ReportStr) select a).FirstOrDefault();

string filePath = ExportPDF(GridviewCollectedBasicParameters, Widhts, " ", 3, colcnt, "Название отчета: " + CurrentReport.Name, "Ваш email адрес: " + userTable.Email, StructLastName(userTable.UsersTableID));

return filePath;

}

public string StructLastName(int UserID)

{

KPIWebDataContext kPiDataContext = new KPIWebDataContext();

UsersTable userTable =

(from a in kPiDataContext.UsersTable where a.UsersTableID == UserID select a).FirstOrDefault();

int deepness =5;

if (userTable.FK\_FifthLevelSubdivisionTable == null)

{

deepness = 4;

}

if (userTable.FK\_FourthLevelSubdivisionTable == null)

{

deepness = 3;

}

if (userTable.FK\_ThirdLevelSubdivisionTable == null)

{

deepness = 2;

}

if (userTable.FK\_SecondLevelSubdivisionTable == null)

{

deepness = 1;

}

if (userTable.FK\_FirstLevelSubdivisionTable == null)

{

deepness = 0;

}

switch(deepness)

{

case 0:

{

return (from a in kPiDataContext.ZeroLevelSubdivisionTable

where

a.ZeroLevelSubdivisionTableID == userTable.FK\_ZeroLevelSubdivisionTable

select a.Name).FirstOrDefault();

break;

}

case 1:

{

return (from a in kPiDataContext.FirstLevelSubdivisionTable

where

a.FirstLevelSubdivisionTableID == userTable.FK\_FirstLevelSubdivisionTable

select a.Name).FirstOrDefault();

break;

}

case 2:

{

return (from a in kPiDataContext.SecondLevelSubdivisionTable

where

a.SecondLevelSubdivisionTableID == userTable.FK\_SecondLevelSubdivisionTable

select a.Name).FirstOrDefault();

break;

}

case 3:

{

return (from a in kPiDataContext.ThirdLevelSubdivisionTable

where

a.ThirdLevelSubdivisionTableID == userTable.FK\_ThirdLevelSubdivisionTable

select a.Name).FirstOrDefault();

break;

}

case 4:

{

return (from a in kPiDataContext.FourthLevelSubdivisionTable

where

a.FourthLevelSubdivisionTableID == userTable.FK\_FourthLevelSubdivisionTable

select a.Name).FirstOrDefault();

break;

}

default:

{

return "";

break;

}

}

}

protected void Button1\_Click(object sender, EventArgs e) // экспорт в excel

{

/\*string pdfFile = CreatePdf();

Response.Write("<script>");

Response.Write("window.open('" + pdfFile + "','\_blank')");

Response.Write("</script>");\*/

Response.ContentType = "Application/pdf";

Response.TransmitFile(CreatePdf());

Response.End();

}

protected void Button2\_Click(object sender, EventArgs e) // вернуться в меню

{

Response.Redirect("~/Default.aspx");

}

protected void Button3\_Click(object sender, EventArgs e) // отправка на доработку и возвращение с доработки

{

KPIWebDataContext kPiDataContext = new KPIWebDataContext();

Serialization UserSer = (Serialization)Session["UserID"];

if (UserSer == null)

{

Response.Redirect("~/Default.aspx");

}

int userID = UserSer.Id;

Serialization modeSer = (Serialization)Session["mode"];

if (modeSer == null)

{

Response.Redirect("~/Default.aspx");

}

int mode = modeSer.mode;

int columnCnt = (int)ViewState["ValueColumnCnt"];

if (columnCnt != null)

{

if (GridviewCollectedBasicParameters.Rows.Count > 0)

{

if ((mode == 0) || (mode == 2))

{

if (mode == 0) //отправляем данные на подтверждение

{

bool wasReturned = false; // данные сейчас на доработке?

#region send to confirm

for (int k = 0; k < columnCnt; k++) // пройдемся по каждой колонке

{

for (int i = 0; i < GridviewCollectedBasicParameters.Rows.Count; i++)

{

Label label =

(Label)

GridviewCollectedBasicParameters.Rows[i].FindControl("CollectId" +

k.ToString());

if (label != null)

{

if (label.Text == "")

{

}

else

{

CollectedBasicParametersTable tmpColTable =

(from a in kPiDataContext.CollectedBasicParametersTable

where

a.CollectedBasicParametersTableID == Convert.ToInt32(label.Text)

select a).FirstOrDefault();

if (tmpColTable != null)

{

if ((tmpColTable.Status == 2) || (tmpColTable.Status == 1))

{

if ((tmpColTable.Status == 1))

{

wasReturned = true;

}

tmpColTable.Status = 3;

kPiDataContext.SubmitChanges();

}

else

{

LogHandler.LogWriter.WriteLog(LogCategory.ERROR, "0FRE3: Пользователь " + (string)ViewState["login"] + " сгенерировал ошибку 3 в отчете с ID = ");

}

}

else

{

LogHandler.LogWriter.WriteLog(LogCategory.ERROR, "0FRE4: Пользователь " + (string)ViewState["login"] + " сгенерировал ошибку 4 в отчете с ID = ");

}

}

}

}

}

ClientScript.RegisterStartupScript(this.GetType(), "alert", "alert('Отчёт отправлен на утверждение');" +

"document.location = '../Default.aspx';", true);

#region

BasicParametersTable BasicConnectedToUser = (from a in kPiDataContext.BasicParametersTable

join b in kPiDataContext.BasicParametrsAndUsersMapping

on a.BasicParametersTableID equals b.FK\_ParametrsTable

where b.FK\_UsersTable == userID

&& b.CanEdit == true

&& b.Active == true

&& a.Active == true

select a).FirstOrDefault();

UsersTable CurrentUser = (from a in kPiDataContext.UsersTable

where a.UsersTableID == userID

select a).FirstOrDefault();

UsersTable UserToSend = (from a in kPiDataContext.UsersTable

join b in kPiDataContext.BasicParametrsAndUsersMapping

on a .UsersTableID equals b.FK\_UsersTable

where b.FK\_ParametrsTable == BasicConnectedToUser.BasicParametersTableID

&& b.CanConfirm == true

&& a.Active == true

&& b.Active == true

&& (( a.FK\_FirstLevelSubdivisionTable == CurrentUser.FK\_FirstLevelSubdivisionTable )||CurrentUser.FK\_FirstLevelSubdivisionTable==null)

&& (( a.FK\_SecondLevelSubdivisionTable == CurrentUser.FK\_SecondLevelSubdivisionTable)||CurrentUser.FK\_SecondLevelSubdivisionTable==null)

&& (( a.FK\_ThirdLevelSubdivisionTable == CurrentUser.FK\_ThirdLevelSubdivisionTable)||CurrentUser.FK\_ThirdLevelSubdivisionTable==null)

&& (( a.FK\_FourthLevelSubdivisionTable == CurrentUser.FK\_FourthLevelSubdivisionTable)||CurrentUser.FK\_FourthLevelSubdivisionTable==null)

&& (( a.FK\_FifthLevelSubdivisionTable == CurrentUser.FK\_FifthLevelSubdivisionTable)||CurrentUser.FK\_FifthLevelSubdivisionTable==null)

select a).FirstOrDefault();

if (UserToSend == null)

{

}

else

{

EmailTemplate EmailParams;

if (wasReturned)

{

EmailParams = (from a in kPiDataContext.EmailTemplate

where a.Name == "DataSendToConfirmAfterRemake"

&& a.Active == true

select a).FirstOrDefault();

}

else

{

EmailParams = (from a in kPiDataContext.EmailTemplate

where a.Name == "DataSendToConfirm"

&& a.Active == true

select a).FirstOrDefault();

}

if (EmailParams != null)

Action.MassMailing(UserToSend.Email, EmailParams.EmailTitle,

EmailParams.EmailContent.Replace("#SiteName#", ConfigurationManager.AppSettings.Get("SiteName")), null);

}

#endregion

#endregion

}

else // (mode == 2) // данные обратно на доработку

{

#region send back to correct

for (int k = 0; k < columnCnt; k++) //

{

for (int i = 0; i < GridviewCollectedBasicParameters.Rows.Count; i++)

{

Label label =

(Label)

GridviewCollectedBasicParameters.Rows[i].FindControl("CollectId" +

k.ToString());

if (label != null)

{

if (label.Text == "")

{

}

else

{

CollectedBasicParametersTable tmpColTable =

(from a in kPiDataContext.CollectedBasicParametersTable

where

a.CollectedBasicParametersTableID == Convert.ToInt32(label.Text)

select a).FirstOrDefault();

if (tmpColTable != null)

{

if (tmpColTable.Status == 3)

{

tmpColTable.Status = 1;

kPiDataContext.SubmitChanges();

}

else

{

LogHandler.LogWriter.WriteLog(LogCategory.ERROR, "0FRE5: Пользователь " + (string)ViewState["login"] + " сгенерировал ошибку 5 в отчете с ID = ");

}

}

else

{

LogHandler.LogWriter.WriteLog(LogCategory.ERROR, "0FRE6: Пользователь " + (string)ViewState["login"] + " сгенерировал ошибку 6 в отчете с ID = ");

}

}

}

}

}

ClientScript.RegisterStartupScript(this.GetType(), "alert", "alert('Отчёт отправлен на доработку');" +

"document.location = '../Default.aspx';", true);

#region

BasicParametersTable BasicConnectedToUser = (from a in kPiDataContext.BasicParametersTable

join b in kPiDataContext.BasicParametrsAndUsersMapping

on a.BasicParametersTableID equals b.FK\_ParametrsTable

where b.FK\_UsersTable == userID

&& b.CanConfirm == true

&& b.Active == true

&& a.Active == true

select a).FirstOrDefault();

UsersTable CurrentUser = (from a in kPiDataContext.UsersTable

where a.UsersTableID == userID

select a).FirstOrDefault();

UsersTable UserToSend = (from a in kPiDataContext.UsersTable

join b in kPiDataContext.BasicParametrsAndUsersMapping

on a.UsersTableID equals b.FK\_UsersTable

where b.FK\_ParametrsTable == BasicConnectedToUser.BasicParametersTableID

&& b.CanEdit == true

&& a.Active == true

&& b.Active == true

&& ((a.FK\_FirstLevelSubdivisionTable == CurrentUser.FK\_FirstLevelSubdivisionTable) || CurrentUser.FK\_FirstLevelSubdivisionTable == null)

&& ((a.FK\_SecondLevelSubdivisionTable == CurrentUser.FK\_SecondLevelSubdivisionTable) || CurrentUser.FK\_SecondLevelSubdivisionTable == null)

&& ((a.FK\_ThirdLevelSubdivisionTable == CurrentUser.FK\_ThirdLevelSubdivisionTable) || CurrentUser.FK\_ThirdLevelSubdivisionTable == null)

&& ((a.FK\_FourthLevelSubdivisionTable == CurrentUser.FK\_FourthLevelSubdivisionTable) || CurrentUser.FK\_FourthLevelSubdivisionTable == null)

&& ((a.FK\_FifthLevelSubdivisionTable == CurrentUser.FK\_FifthLevelSubdivisionTable) || CurrentUser.FK\_FifthLevelSubdivisionTable == null)

select a).FirstOrDefault();

if (UserToSend == null)

{

}

else

{

EmailTemplate EmailParams = (from a in kPiDataContext.EmailTemplate

where a.Name == "DataSendToRemake"

&& a.Active == true

select a).FirstOrDefault();

if (EmailParams != null)

Action.MassMailing(UserToSend.Email, EmailParams.EmailTitle,

EmailParams.EmailContent.Replace("#SiteName#", ConfigurationManager.AppSettings.Get("SiteName")).Replace("#Comment#", TextBox1.Text), null);

}

#endregion

#endregion

}

}

}

else

{

LogHandler.LogWriter.WriteLog(LogCategory.ERROR, "0FRE7: Пользователь " + (string)ViewState["login"] + " сгенерировал ошибку 7 в отчете с ID = ");

}

}

else

{

LogHandler.LogWriter.WriteLog(LogCategory.ERROR, "Пользователь " + (string)ViewState["login"] + " сгенерировал ошибку 8 в отчете с ID = ");

}

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Filename: FillingTheReport.aspx.designer.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//------------------------------------------------------------------------------

// <auto-generated>

// This code was generated by a tool.

//

// Changes to this file may cause incorrect behavior and will be lost if

// the code is regenerated.

// </auto-generated>

//------------------------------------------------------------------------------

namespace KPIWeb.Reports {

public partial class FillingTheReport {

/// <summary>

/// top\_panel2 control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.Panel top\_panel2;

/// <summary>

/// Label2 control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.Label Label2;

/// <summary>

/// Label3 control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.Label Label3;

/// <summary>

/// Label1 control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.Label Label1;

/// <summary>

/// GridviewCollectedBasicParameters control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.GridView GridviewCollectedBasicParameters;

/// <summary>

/// Summary control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.ValidationSummary Summary;

/// <summary>

/// ButtonSave control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.Button ButtonSave;

/// <summary>

/// Button1 control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.Button Button1;

/// <summary>

/// GoBackButton control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.Button GoBackButton;

/// <summary>

/// UpnDownButton control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.Button UpnDownButton;

/// <summary>

/// TextBox1 control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.TextBox TextBox1;

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Filename: GenerateReport.aspx.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

using System.Data;

namespace KPIWeb.Reports

{

public partial class GenerateReport : System.Web.UI.Page

{

UsersTable user;

protected void Page\_Load(object sender, EventArgs e)

{

Response.Redirect("~/Default.aspx");

Serialization UserSer = (Serialization)Session["UserID"];

if (UserSer == null)

{

Response.Redirect("~/Account/Login.aspx");

}

int userID = UserSer.Id;

KPIWebDataContext kPiDataContext = new KPIWebDataContext();

UsersTable userTable =

(from a in kPiDataContext.UsersTable where a.UsersTableID == userID select a).FirstOrDefault();

if (userTable.AccessLevel != 9)

{

Response.Redirect("~/Default.aspx");

}

if (!Page.IsPostBack)

{

KPIWebDataContext KPIWebDataContext = new KPIWebDataContext();

UsersTable user = (from usersTables in KPIWebDataContext.UsersTable

where usersTables.UsersTableID == UserSer.Id

select usersTables).FirstOrDefault();

Serialization ReportId = (Serialization)Session["ReportArchiveTableID"];

if (ReportId != null) ////////////////////////////////////////

{

int reportArchiveTableID = ReportId.ReportArchiveID;

List<IndicatorsTable> indicatorsTable = (from item in KPIWebDataContext.IndicatorsTable

select item).OrderBy(c => c.SortID).ToList();

DataTable dataTable = new DataTable();

dataTable.Columns.Add(new DataColumn("IndicatorsTableID", typeof(string)));

dataTable.Columns.Add(new DataColumn("Name", typeof(string)));

dataTable.Columns.Add(new DataColumn("IndicatorsValue", typeof(string)));

foreach (var item in indicatorsTable)

{

DataRow dataRow = dataTable.NewRow();

dataRow["IndicatorsTableID"] = item.IndicatorsTableID;

dataRow["Name"] = item.Name;

dataRow["IndicatorsValue"] = string.Format("{0:N2}"+item.Measure, CalculateIndicator.Calculate(item.IndicatorsTableID, reportArchiveTableID));

dataTable.Rows.Add(dataRow);

}

GridviewReport.DataSource = dataTable;

GridviewReport.DataBind();

}

}

}

protected void Button1\_Click(object sender, EventArgs e)

{

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Filename: GenerateReport.aspx.designer.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//------------------------------------------------------------------------------

// <автоматически создаваемое>

// Этот код создан программой.

//

// Изменения в этом файле могут привести к неправильной работе и будут потеряны в случае

// повторной генерации кода.

// </автоматически создаваемое>

//------------------------------------------------------------------------------

namespace KPIWeb.Reports {

public partial class GenerateReport {

/// <summary>

/// GridviewReport элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.GridView GridviewReport;

/// <summary>

/// Button1 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Button Button1;

/// <summary>

/// Button2 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Button Button2;

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Filename: GridViewExportUtil.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using System;

using System.Data;

using System.Configuration;

using System.IO;

using System.Web;

using System.Web.Security;

using System.Web.UI;

using System.Web.UI.WebControls;

using System.Web.UI.WebControls.WebParts;

using System.Web.UI.HtmlControls;/// <summary>

///

/// </summary>

///

namespace KPIWeb.Reports

{

public class GridViewExportUtil

{

/// <summary>

///

/// </summary>

/// <param name=”fileName”></param>

/// <param name=”gv”></param>

public static void Export(string fileName, GridView gv)

{

HttpContext.Current.Response.Clear();

HttpContext.Current.Response.AddHeader("content-disposition", string.Format("attachment; filename={0}", fileName));

HttpContext.Current.Response.ContentType = "application/ms-excel";using (StringWriter sw = new StringWriter())

{

using (HtmlTextWriter htw = new HtmlTextWriter(sw))

{

// Create a form to contain the grid

Table table = new Table();table.GridLines = gv.GridLines;

// add the header row to the table

if (gv.HeaderRow != null)

{

GridViewExportUtil.PrepareControlForExport(gv.HeaderRow);

table.Rows.Add(gv.HeaderRow);

}

// add each of the data rows to the table

foreach (GridViewRow row in gv.Rows)

{

GridViewExportUtil.PrepareControlForExport(row);

table.Rows.Add(row);

}

// add the footer row to the table

if (gv.FooterRow != null)

{

GridViewExportUtil.PrepareControlForExport(gv.FooterRow);

table.Rows.Add(gv.FooterRow);

}

// render the table into the htmlwriter

table.RenderControl(htw);

// render the htmlwriter into the response

HttpContext.Current.Response.Write(sw.ToString());

HttpContext.Current.Response.End();

}

}

}

/// <summary>

/// Replace any of the contained controls with literals

/// </summary>

/// <param name=”control”></param>

private static void PrepareControlForExport(Control control)

{

for (int i = 0; i < control.Controls.Count; i++)

{

Control current = control.Controls[i];

if (current is LinkButton)

{

control.Controls.Remove(current);

control.Controls.AddAt(i, new LiteralControl((current as LinkButton).Text));

}

else if (current is ImageButton)

{

control.Controls.Remove(current);

control.Controls.AddAt(i, new LiteralControl((current as ImageButton).AlternateText));

}

else if (current is HyperLink)

{

control.Controls.Remove(current);

control.Controls.AddAt(i, new LiteralControl((current as HyperLink).Text));

}

else if (current is DropDownList)

{

control.Controls.Remove(current);

control.Controls.AddAt(i, new LiteralControl((current as DropDownList).SelectedItem.Text));

}

else if (current is CheckBox)

{

control.Controls.Remove(current);

control.Controls.AddAt(i, new LiteralControl((current as CheckBox).Checked ? "True" : "False"));

}

if (current.HasControls())

{

GridViewExportUtil.PrepareControlForExport(current);

}

}

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Filename: Parametrs.aspx.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using System;

using System.Collections.Generic;

using System.Data;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

namespace KPIWeb.Reports

{

public partial class Parametrs : System.Web.UI.Page

{

protected void Page\_Load(object sender, EventArgs e)

{

Serialization UserSer = (Serialization)Session["UserID"];

if (UserSer == null)

{

Response.Redirect("~/Default.aspx");

}

int userID = UserSer.Id;

KPIWebDataContext kPiDataContext = new KPIWebDataContext();

UsersTable userTable =

(from a in kPiDataContext.UsersTable where a.UsersTableID == userID select a).FirstOrDefault();

ViewState["UserTable"] = userTable.FK\_ThirdLevelSubdivisionTable;

if (userTable.AccessLevel != 0)

{

Response.Redirect("~/Default.aspx");

}

////////////////////////////////////////////////////

if (!Page.IsPostBack)

{

int UserID = UserSer.Id;

List<SpecializationTable> specializationTableData = (from a in kPiDataContext.SpecializationTable

join b in kPiDataContext.FourthLevelSubdivisionTable

on a.SpecializationTableID equals b.FK\_Specialization

where b.FK\_ThirdLevelSubdivisionTable == userTable.FK\_ThirdLevelSubdivisionTable

&& b.Active == true

select a).ToList();

DataTable dataTable = new DataTable();

dataTable.Columns.Add(new DataColumn("SpecializationID", typeof (string)));

dataTable.Columns.Add(new DataColumn("SpecializationName", typeof (string)));

dataTable.Columns.Add(new DataColumn("FourthlvlId", typeof (int)));

dataTable.Columns.Add(new DataColumn("SpecNumber", typeof (string)));

dataTable.Columns.Add(new DataColumn("Param1Label", typeof (string)));

dataTable.Columns.Add(new DataColumn("Param1CheckBox", typeof (string)));

dataTable.Columns.Add(new DataColumn("Param2Label", typeof (string)));

dataTable.Columns.Add(new DataColumn("Param2CheckBox", typeof (string)));

dataTable.Columns.Add(new DataColumn("Param3Label", typeof (string)));

dataTable.Columns.Add(new DataColumn("Param3CheckBox", typeof (string)));

dataTable.Columns.Add(new DataColumn("Param4Label", typeof (string)));

dataTable.Columns.Add(new DataColumn("Param4CheckBox", typeof (string)));

dataTable.Columns.Add(new DataColumn("Param5Label", typeof (string)));

dataTable.Columns.Add(new DataColumn("Param5CheckBox", typeof (string)));

foreach (SpecializationTable spec in specializationTableData)

{

DataRow dataRow = dataTable.NewRow();

dataRow["SpecializationID"] = spec.SpecializationTableID;

dataRow["SpecializationName"] = spec.Name;

dataRow["FourthlvlId"] = (from a in kPiDataContext.FourthLevelSubdivisionTable

where

a.FK\_ThirdLevelSubdivisionTable == userTable.FK\_ThirdLevelSubdivisionTable &&

a.FK\_Specialization == spec.SpecializationTableID

select a.FourthLevelSubdivisionTableID).FirstOrDefault();

dataRow["SpecNumber"] = spec.SpecializationNumber;

dataTable.Rows.Add(dataRow);

}

ViewState["GridviewSpec"] = dataTable;

GridView1.DataSource = dataTable;

GridView1.DataBind();

}

}

protected void GridView1\_RowDataBound(object sender, GridViewRowEventArgs e)

{

// Проставляем галочки в GridView

for (int i = 0; i <= GridView1.Rows.Count; i++)

{

var fourthlvlId = e.Row.FindControl("FourthlvlId") as Label;

var checkBoxparamIsModern = e.Row.FindControl("IsModern") as CheckBox;

var checkBoxparamIsNetwork = e.Row.FindControl("IsNetwork") as CheckBox;

var checkBoxparamIsInvalid = e.Row.FindControl("IsInvalid") as CheckBox;

var checkBoxparamIsForeign = e.Row.FindControl("IsForeign") as CheckBox;

KPIWebDataContext kpiWebDataContext = new KPIWebDataContext();

FourthLevelParametrs fourthLevelParametrs = new FourthLevelParametrs();

if (fourthlvlId != null)

fourthLevelParametrs = (from a in kpiWebDataContext.FourthLevelParametrs where a.FourthLevelParametrsID == Convert.ToInt32(fourthlvlId.Text) select a).FirstOrDefault();

if (fourthLevelParametrs != null)

{

if (checkBoxparamIsModern != null)

checkBoxparamIsModern.Checked = fourthLevelParametrs.IsModernEducationTechnologies.Value;

if (checkBoxparamIsNetwork != null)

checkBoxparamIsNetwork.Checked = fourthLevelParametrs.IsNetworkComunication.Value;

if (checkBoxparamIsInvalid != null)

checkBoxparamIsInvalid.Checked = fourthLevelParametrs.IsInvalidStudentsFacilities.Value;

if (checkBoxparamIsForeign != null)

checkBoxparamIsForeign.Checked = fourthLevelParametrs.IsForeignStudentsAccept.Value;

}

}

}

protected void Button1\_Click(object sender, EventArgs e)

{

int rowIndex = 0;

if (ViewState["GridviewSpec"] != null)

{

DataTable dataTable = (DataTable)ViewState["GridviewSpec"];

if (dataTable.Rows.Count > 0)

{

for (int i = 1; i <= dataTable.Rows.Count; i++)

{

CheckBox checkBoxparamIsModern = (CheckBox)GridView1.Rows[rowIndex].FindControl("IsModern");

CheckBox checkBoxparamIsNetwork = (CheckBox)GridView1.Rows[rowIndex].FindControl("IsNetwork");

CheckBox checkBoxparamIsInvalid = (CheckBox)GridView1.Rows[rowIndex].FindControl("IsInvalid");

CheckBox checkBoxparamIsForeign = (CheckBox)GridView1.Rows[rowIndex].FindControl("IsForeign");

Label labelId = (Label)GridView1.Rows[rowIndex].FindControl("FourthlvlId");

rowIndex++;

var ss = labelId.Text;

using (KPIWebDataContext kpiWebDataContext = new KPIWebDataContext())

{

FourthLevelParametrs fourthLevelParametrsTables = (from a in kpiWebDataContext.FourthLevelParametrs where a.FourthLevelParametrsID == Convert.ToInt32(labelId.Text) select a).FirstOrDefault();

ThirdLevelParametrs thirdLevelParametrs = (from a in kpiWebDataContext.ThirdLevelParametrs where a.ThirdLevelParametrsID == (int)ViewState["UserTable"] select a).FirstOrDefault();

if (fourthLevelParametrsTables != null)

{

fourthLevelParametrsTables.IsModernEducationTechnologies = checkBoxparamIsModern.Checked;

fourthLevelParametrsTables.IsNetworkComunication = checkBoxparamIsNetwork.Checked;

fourthLevelParametrsTables.IsInvalidStudentsFacilities = checkBoxparamIsInvalid.Checked;

fourthLevelParametrsTables.IsForeignStudentsAccept = checkBoxparamIsForeign.Checked;

}

kpiWebDataContext.SubmitChanges();

}

}

}

}

Response.Redirect("~/Reports\_/FillingTheReport.aspx");

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Filename: Parametrs.aspx.designer.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//------------------------------------------------------------------------------

// <auto-generated>

// This code was generated by a tool.

//

// Changes to this file may cause incorrect behavior and will be lost if

// the code is regenerated.

// </auto-generated>

//------------------------------------------------------------------------------

namespace KPIWeb.Reports {

public partial class Parametrs {

/// <summary>

/// GridView1 control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.GridView GridView1;

/// <summary>

/// Button1 control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.Button Button1;

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Filename: SpecializationParametrs.aspx.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using System;

using System.Collections.Generic;

using System.Configuration;

using System.Data;

using System.Data.SqlTypes;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

namespace KPIWeb.Reports

{

public partial class SpecializationParametrs : System.Web.UI.Page

{

protected void Page\_Load(object sender, EventArgs e)

{

Serialization UserSer = (Serialization)Session["UserID"];

if (UserSer == null)

{

Response.Redirect("~/Default.aspx");

}

int userID = UserSer.Id;

KPIWebDataContext kPiDataContext = new KPIWebDataContext();

UsersTable userTable =

(from a in kPiDataContext.UsersTable where a.UsersTableID == userID select a).FirstOrDefault();

ViewState["UserTable"] = userTable.FK\_ThirdLevelSubdivisionTable;

if (userTable.AccessLevel != 0)

{

Response.Redirect("~/Default.aspx");

}

if (!Page.IsPostBack)

{

int UserID = UserSer.Id;

List<SpecializationTable> specializationTableData = (from a in kPiDataContext.SpecializationTable

join b in kPiDataContext.FourthLevelSubdivisionTable

on a.SpecializationTableID equals b.FK\_Specialization

where b.FK\_ThirdLevelSubdivisionTable == userTable.FK\_ThirdLevelSubdivisionTable

&& b.Active == true

select a).ToList();

CheckBox1.Checked = (from a in kPiDataContext.ThirdLevelParametrs where a.ThirdLevelParametrsID == userTable.FK\_ThirdLevelSubdivisionTable select a.CanGraduate).FirstOrDefault();

DataTable dataTable = new DataTable();

dataTable.Columns.Add(new DataColumn("SpecializationID", typeof(string)));

dataTable.Columns.Add(new DataColumn("SpecializationName", typeof(string)));

dataTable.Columns.Add(new DataColumn("FourthlvlId", typeof(int)));

dataTable.Columns.Add(new DataColumn("SpecNumber", typeof(string)));

dataTable.Columns.Add(new DataColumn("Param1Label", typeof(string)));

dataTable.Columns.Add(new DataColumn("Param1CheckBox", typeof(string)));

dataTable.Columns.Add(new DataColumn("Param2Label", typeof(string)));

dataTable.Columns.Add(new DataColumn("Param2CheckBox", typeof(string)));

dataTable.Columns.Add(new DataColumn("Param3Label", typeof(string)));

dataTable.Columns.Add(new DataColumn("Param3CheckBox", typeof(string)));

dataTable.Columns.Add(new DataColumn("Param4Label", typeof(string)));

dataTable.Columns.Add(new DataColumn("Param4CheckBox", typeof(string)));

dataTable.Columns.Add(new DataColumn("Param5Label", typeof(string)));

dataTable.Columns.Add(new DataColumn("Param5CheckBox", typeof(string)));

dataTable.Columns.Add(new DataColumn("DeleteSpecializationLabel", typeof(string)));

dataTable.Columns.Add(new DataColumn("DeleteSpecializationButton", typeof(string)));

foreach (SpecializationTable spec in specializationTableData)

{

DataRow dataRow = dataTable.NewRow();

dataRow["SpecializationID"] = spec.SpecializationTableID ;

dataRow["SpecializationName"] = spec.Name;

dataRow["FourthlvlId"] = (from a in kPiDataContext.FourthLevelSubdivisionTable

where

a.FK\_ThirdLevelSubdivisionTable == userTable.FK\_ThirdLevelSubdivisionTable &&

a.FK\_Specialization == spec.SpecializationTableID

select a.FourthLevelSubdivisionTableID).FirstOrDefault();

dataRow["SpecNumber"] = spec.SpecializationNumber;

dataTable.Rows.Add(dataRow);

}

ViewState["GridviewSpec"] = dataTable;

GridView1.DataSource = dataTable;

GridView1.DataBind();

}

if (CheckBox1.Checked)

{

GridView1.Visible = true;

Label3.Visible = true;

Label1.Visible = true;

GridView2.Visible = true;

Button2.Visible = true;

TextBox1.Visible = true;

}

else

{

GridView1.Visible = false;

Label3.Visible = false;

Label1.Visible = false;

GridView2.Visible = false;

Button2.Visible = false;

TextBox1.Visible = false;

}

}

protected void DeleteSpecializationButtonClick(object sender, EventArgs e)

{

Button button = (Button)sender;

{

using (KPIWebDataContext kPiDataContext = new KPIWebDataContext())

{

var check =

(from a in kPiDataContext.FourthLevelSubdivisionTable where

a.FourthLevelSubdivisionTableID == Convert.ToInt32(button.CommandArgument) select a)

.FirstOrDefault();

check.Active = false;

kPiDataContext.SubmitChanges();

}

Response.Redirect("~/Reports\_/SpecializationParametrs.aspx");

}

}

protected void AddSpecializationButtonClick(object sender, EventArgs e)

{

Button button = (Button)sender;

{

KPIWebDataContext kPiDataContext = new KPIWebDataContext();

var par = button.CommandArgument.ToString();

var check =

(from a in kPiDataContext.SpecializationTable where a.SpecializationNumber == par select a)

.FirstOrDefault(); // выбираем специальности по её коду

/\* List<SpecializationTable> specializationTableData = (from a in kPiDataContext.SpecializationTable

join b in kPiDataContext.FourthLevelSubdivisionTable

on a.SpecializationTableID equals b.FK\_Specialization

where b.FK\_ThirdLevelSubdivisionTable == (int)ViewState["UserTable"]

select a).ToList();

if (check != null)

{

foreach (SpecializationTable spec in specializationTableData)

{

if (spec.SpecializationNumber.Equals(par)) isHere = true;

}

}\*/

using (KPIWebDataContext kpiWebDataContext = new KPIWebDataContext()) // проверяем есть/нет записываем в базу

{

FourthLevelSubdivisionTable forthlvlsudtab = (from a in kpiWebDataContext.FourthLevelSubdivisionTable

where a.FK\_Specialization == check.SpecializationTableID && a.FK\_ThirdLevelSubdivisionTable == (int)ViewState["UserTable"]

select a).FirstOrDefault();

if (forthlvlsudtab != null)

{

forthlvlsudtab.Active = true;

}

else

{

forthlvlsudtab = new FourthLevelSubdivisionTable();

forthlvlsudtab.FK\_Specialization = check.SpecializationTableID;

forthlvlsudtab.Active = true;

forthlvlsudtab.Name = check.Name;

forthlvlsudtab.FK\_ThirdLevelSubdivisionTable = (int)ViewState["UserTable"];

kpiWebDataContext.FourthLevelSubdivisionTable.InsertOnSubmit(forthlvlsudtab);

}

kpiWebDataContext.SubmitChanges();

// Добавляем запись в таблицу параметров для этой специальности

FourthLevelParametrs fourthLevelParametrs = (from a in kpiWebDataContext.FourthLevelParametrs

where a.FourthLevelParametrsID == forthlvlsudtab.FourthLevelSubdivisionTableID

select a).FirstOrDefault();

if (fourthLevelParametrs == null)

{

fourthLevelParametrs = new FourthLevelParametrs();

fourthLevelParametrs.Active = true;

fourthLevelParametrs.IsModernEducationTechnologies = false;

fourthLevelParametrs.IsNetworkComunication = false;

fourthLevelParametrs.IsInvalidStudentsFacilities = false;

fourthLevelParametrs.IsForeignStudentsAccept = false;

fourthLevelParametrs.FourthLevelParametrsID = forthlvlsudtab.FourthLevelSubdivisionTableID;

var code = (from f4 in kpiWebDataContext.FourthLevelSubdivisionTable // получаем код специальности

join spec in kpiWebDataContext.SpecializationTable

on f4.FK\_Specialization equals spec.SpecializationTableID

where f4.FourthLevelSubdivisionTableID == forthlvlsudtab.FourthLevelSubdivisionTableID

select spec.SpecializationNumber).FirstOrDefault();

fourthLevelParametrs.SpecType = Action.Encode(code);

kpiWebDataContext.FourthLevelParametrs.InsertOnSubmit(fourthLevelParametrs);

}

kpiWebDataContext.SubmitChanges();

}

Response.Redirect("~/Reports\_/SpecializationParametrs.aspx");

}

}

protected void Button1\_Click(object sender, EventArgs e)

{

int rowIndex = 0;

if (ViewState["GridviewSpec"] != null)

{

DataTable dataTable = (DataTable) ViewState["GridviewSpec"];

if (dataTable.Rows.Count > 0)

{

for (int i = 1; i <= dataTable.Rows.Count; i++)

{

CheckBox checkBoxparamIsModern = (CheckBox) GridView1.Rows[rowIndex].FindControl("IsModern");

CheckBox checkBoxparamIsNetwork = (CheckBox)GridView1.Rows[rowIndex].FindControl("IsNetwork");

CheckBox checkBoxparamIsInvalid = (CheckBox)GridView1.Rows[rowIndex].FindControl("IsInvalid");

CheckBox checkBoxparamIsForeign = (CheckBox)GridView1.Rows[rowIndex].FindControl("IsForeign");

Label labelId = (Label) GridView1.Rows[rowIndex].FindControl("FourthlvlId");

rowIndex++;

var ss = labelId.Text;

using (KPIWebDataContext kpiWebDataContext = new KPIWebDataContext())

{

FourthLevelParametrs fourthLevelParametrsTables = (from a in kpiWebDataContext.FourthLevelParametrs where a.FourthLevelParametrsID == Convert.ToInt32(labelId.Text) select a).FirstOrDefault();

ThirdLevelParametrs thirdLevelParametrs = (from a in kpiWebDataContext.ThirdLevelParametrs where a.ThirdLevelParametrsID == (int)ViewState["UserTable"] select a).FirstOrDefault();

if (fourthLevelParametrsTables != null)

{

fourthLevelParametrsTables.IsModernEducationTechnologies =checkBoxparamIsModern.Checked;

fourthLevelParametrsTables.IsNetworkComunication = checkBoxparamIsNetwork.Checked;

fourthLevelParametrsTables.IsInvalidStudentsFacilities = checkBoxparamIsInvalid.Checked;

fourthLevelParametrsTables.IsForeignStudentsAccept = checkBoxparamIsForeign.Checked;

}

if (thirdLevelParametrs != null)

thirdLevelParametrs.CanGraduate = CheckBox1.Checked;

kpiWebDataContext.SubmitChanges();

}

}

}

}

Response.Redirect("~/Reports\_/ChooseReport.aspx");

}

protected void Button2\_Click(object sender, EventArgs e)

{

KPIWebDataContext kPiDataContext = new KPIWebDataContext();

List<SpecializationTable> specializationTableData = (from a in kPiDataContext.SpecializationTable

where a.Name.Contains(TextBox1.Text)

|| a.SpecializationNumber.Contains(TextBox1.Text)

select a).ToList();

DataTable dataTable = new DataTable();

dataTable.Columns.Add(new DataColumn("SpecializationID", typeof(string)));

dataTable.Columns.Add(new DataColumn("SpecializationName", typeof(string)));

dataTable.Columns.Add(new DataColumn("SpecializationNumber", typeof(string)));

dataTable.Columns.Add(new DataColumn("AddSpecializationLabel", typeof(string)));

dataTable.Columns.Add(new DataColumn("AddSpecializationButton", typeof(string)));

foreach (SpecializationTable spec in specializationTableData)

{

DataRow dataRow = dataTable.NewRow();

dataRow["SpecializationID"] = spec.SpecializationTableID;

dataRow["SpecializationName"] = spec.Name;

dataRow["SpecializationNumber"] = spec.SpecializationNumber;

dataTable.Rows.Add(dataRow);

}

GridView2.DataSource = dataTable;

GridView2.DataBind();

}

protected void GridView1\_RowDataBound(object sender, GridViewRowEventArgs e)

{

// Проставляем галочки в GridView

for (int i = 0; i <= GridView1.Rows.Count; i++)

{

var fourthlvlId = e.Row.FindControl("FourthlvlId") as Label;

var checkBoxparamIsModern = e.Row.FindControl("IsModern") as CheckBox;

var checkBoxparamIsNetwork = e.Row.FindControl("IsNetwork") as CheckBox;

var checkBoxparamIsInvalid = e.Row.FindControl("IsInvalid") as CheckBox;

var checkBoxparamIsForeign = e.Row.FindControl("IsForeign") as CheckBox;

KPIWebDataContext kpiWebDataContext = new KPIWebDataContext();

FourthLevelParametrs fourthLevelParametrs = new FourthLevelParametrs();

if (fourthlvlId != null)

fourthLevelParametrs = (from a in kpiWebDataContext.FourthLevelParametrs where a.FourthLevelParametrsID == Convert.ToInt32(fourthlvlId.Text) select a).FirstOrDefault();

if (fourthLevelParametrs != null)

{

if (checkBoxparamIsModern != null)

checkBoxparamIsModern.Checked = fourthLevelParametrs.IsModernEducationTechnologies.Value;

if (checkBoxparamIsNetwork != null)

checkBoxparamIsNetwork.Checked = fourthLevelParametrs.IsNetworkComunication.Value;

if (checkBoxparamIsInvalid != null)

checkBoxparamIsInvalid.Checked = fourthLevelParametrs.IsInvalidStudentsFacilities.Value;

if (checkBoxparamIsForeign != null)

checkBoxparamIsForeign.Checked = fourthLevelParametrs.IsForeignStudentsAccept.Value;

}

}

}

protected void CheckBox1\_CheckedChanged(object sender, EventArgs e)

{

if (CheckBox1.Checked)

{

GridView1.Visible = true;

Label3.Visible = true;

Label1.Visible = true;

GridView2.Visible = true;

Button2.Visible = true;

TextBox1.Visible = true;

}

else

{

GridView1.Visible = false;

Label3.Visible = false;

Label1.Visible = false;

GridView2.Visible = false;

Button2.Visible = false;

TextBox1.Visible = false;

}

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Filename: SpecializationParametrs.aspx.designer.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//------------------------------------------------------------------------------

// <автоматически создаваемое>

// Этот код создан программой.

//

// Изменения в этом файле могут привести к неправильной работе и будут потеряны в случае

// повторной генерации кода.

// </автоматически создаваемое>

//------------------------------------------------------------------------------

namespace KPIWeb.Reports {

public partial class SpecializationParametrs {

/// <summary>

/// CheckBox1 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.CheckBox CheckBox1;

/// <summary>

/// Label1 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Label Label1;

/// <summary>

/// GridView1 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.GridView GridView1;

/// <summary>

/// Button1 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Button Button1;

/// <summary>

/// Label3 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Label Label3;

/// <summary>

/// TextBox1 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.TextBox TextBox1;

/// <summary>

/// Button2 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Button Button2;

/// <summary>

/// GridView2 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.GridView GridView2;

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Filename: AddSpecialization.aspx.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using System;

using System.Collections.Generic;

using System.Configuration;

using System.Data;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

namespace KPIWeb.StatisticsDepartment

{

public partial class AddSpecialization : System.Web.UI.Page

{

protected void Page\_Load(object sender, EventArgs e)

{

Serialization UserSer = (Serialization)Session["UserID"];

if (UserSer == null)

{

Response.Redirect("~/Default.aspx");

}

int userID = UserSer.Id;

KPIWebDataContext kPiDataContext = new KPIWebDataContext();

UsersTable userTable =

(from a in kPiDataContext.UsersTable where a.UsersTableID == userID select a).FirstOrDefault();

if ((userTable.AccessLevel != 10) && (userTable.AccessLevel != 9))

{

Response.Redirect("~/Default.aspx");

}

if (!IsPostBack)

{

////записали роли в дроп даун

List<FirstLevelSubdivisionTable> First\_stageList =

(from item in kPiDataContext.FirstLevelSubdivisionTable

select item).OrderBy(mc => mc.Name).ToList();

var dictionary = new Dictionary<int, string>();

dictionary.Add(0, "Выберите значение");

foreach (var item in First\_stageList)

dictionary.Add(item.FirstLevelSubdivisionTableID, item.Name);

DropDownList1.DataTextField = "Value";

DropDownList1.DataValueField = "Key";

DropDownList1.DataSource = dictionary;

DropDownList1.DataBind();

/// записали академии в дроп даун

}

if (ViewState["Selected"] != null)

if (!IsPostBack && (bool)ViewState["Selected"] == true)

{

List<SpecializationTable> specializationTableData = (from a in kPiDataContext.SpecializationTable

join b in kPiDataContext.FourthLevelSubdivisionTable

on a.SpecializationTableID equals b.FK\_Specialization

where b.FK\_ThirdLevelSubdivisionTable == Convert.ToInt32(DropDownList3.SelectedItem.Value)

&& b.Active == true

select a).ToList();

CheckBox1.Checked = (from a in kPiDataContext.ThirdLevelParametrs where a.ThirdLevelParametrsID == Convert.ToInt32(DropDownList3.SelectedItem.Value) select a.CanGraduate).FirstOrDefault();

DataTable dataTable = new DataTable();

dataTable.Columns.Add(new DataColumn("SpecializationID", typeof(string)));

dataTable.Columns.Add(new DataColumn("SpecializationName", typeof(string)));

dataTable.Columns.Add(new DataColumn("FourthlvlId", typeof(int)));

dataTable.Columns.Add(new DataColumn("SpecNumber", typeof(string)));

dataTable.Columns.Add(new DataColumn("DeleteSpecializationLabel", typeof(string)));

dataTable.Columns.Add(new DataColumn("DeleteSpecializationButton", typeof(string)));

foreach (SpecializationTable spec in specializationTableData)

{

DataRow dataRow = dataTable.NewRow();

dataRow["SpecializationID"] = spec.SpecializationTableID ;

dataRow["SpecializationName"] = spec.Name;

dataRow["FourthlvlId"] = (from a in kPiDataContext.FourthLevelSubdivisionTable

where

a.FK\_ThirdLevelSubdivisionTable == Convert.ToInt32(DropDownList3.SelectedItem.Value) &&

a.FK\_Specialization == spec.SpecializationTableID && a.Active==true

select a.FourthLevelSubdivisionTableID).FirstOrDefault();

dataRow["SpecNumber"] = spec.SpecializationNumber;

dataTable.Rows.Add(dataRow);

}

ViewState["GridviewSpec"] = dataTable;

GridView1.DataSource = dataTable;

GridView1.DataBind();

kPiDataContext.SubmitChanges();

}

}

protected void DropDownList1\_SelectedIndexChanged(object sender, EventArgs e)

{

DropDownList2.Items.Clear();

DropDownList3.Items.Clear();

KPIWebDataContext kPiDataContext = new KPIWebDataContext();

int SelectedValue = -1;

if (int.TryParse(DropDownList1.SelectedValue, out SelectedValue) && SelectedValue != -1)

{

List<SecondLevelSubdivisionTable> second\_stageList =

(from item in kPiDataContext.SecondLevelSubdivisionTable

where item.FK\_FirstLevelSubdivisionTable == SelectedValue && item.Active == true

select item).OrderBy(mc => mc.SecondLevelSubdivisionTableID).ToList();

if (second\_stageList != null && second\_stageList.Count() > 0)

{

var dictionary = new Dictionary<int, string>();

dictionary.Add(-1, "Выберите значение");

foreach (var item in second\_stageList)

dictionary.Add(item.SecondLevelSubdivisionTableID, item.Name);

DropDownList2.DataTextField = "Value";

DropDownList2.DataValueField = "Key";

DropDownList2.DataSource = dictionary;

DropDownList2.DataBind();

}

else

{

Page.ClientScript.RegisterClientScriptBlock(typeof(Page), "Script", "alert('Произошла ошибка.');", true);

}

}

else

{

Page.ClientScript.RegisterClientScriptBlock(typeof(Page), "Script", "alert('Произошла ошибка.');", true);

}

}

protected void DropDownList2\_SelectedIndexChanged(object sender, EventArgs e)

{

DropDownList3.Items.Clear();

KPIWebDataContext kPiDataContext = new KPIWebDataContext();

int SelectedValue = -1;

if (int.TryParse(DropDownList2.SelectedValue, out SelectedValue) && SelectedValue != -1)

{

List<ThirdLevelSubdivisionTable> third\_stage = (from item in kPiDataContext.ThirdLevelSubdivisionTable

where item.FK\_SecondLevelSubdivisionTable == SelectedValue

&& item.Active == true

select item).OrderBy(mc => mc.ThirdLevelSubdivisionTableID).ToList();

if (third\_stage != null && third\_stage.Count() > 0)

{

var dictionary = new Dictionary<int, string>();

dictionary.Add(-1, "Выберите значение");

foreach (var item in third\_stage)

dictionary.Add(item.ThirdLevelSubdivisionTableID, item.Name);

DropDownList3.DataTextField = "Value";

DropDownList3.DataValueField = "Key";

DropDownList3.DataSource = dictionary;

DropDownList3.DataBind();

}

else

{

Page.ClientScript.RegisterClientScriptBlock(typeof(Page), "Script", "alert('Произошла ошибка.');", true);

}

}

else

{

Page.ClientScript.RegisterClientScriptBlock(typeof(Page), "Script", "alert('Произошла ошибка.');", true);

}

}

protected void CheckBox1\_CheckedChanged(object sender, EventArgs e)

{

}

protected void Button1\_Click(object sender, EventArgs e)

{

KPIWebDataContext kpiWeb = new KPIWebDataContext();

var CafParam = (from a in kpiWeb.ThirdLevelParametrs

where a.ThirdLevelParametrsID == Convert.ToInt32(DropDownList3.SelectedItem.Value) && a.Active == true

select a).FirstOrDefault();

CafParam.CanGraduate = CheckBox1.Checked;

CafParam.IsBasic = CheckBox2.Checked;

kpiWeb.SubmitChanges();

Page.ClientScript.RegisterClientScriptBlock(typeof(Page), "Script", "alert('Данные успешно сохранены!');", true);

}

protected void Button2\_Click(object sender, EventArgs e)

{

KPIWebDataContext kPiDataContext = new KPIWebDataContext();

List<SpecializationTable> specializationTableData = (from a in kPiDataContext.SpecializationTable

where a.Name.Contains(TextBox1.Text)

|| a.SpecializationNumber.Contains(TextBox1.Text) && a.Active == true

select a).ToList();

DataTable dataTable = new DataTable();

dataTable.Columns.Add(new DataColumn("SpecializationID", typeof(string)));

dataTable.Columns.Add(new DataColumn("SpecializationName", typeof(string)));

dataTable.Columns.Add(new DataColumn("SpecializationNumber", typeof(string)));

dataTable.Columns.Add(new DataColumn("AddSpecializationLabel", typeof(string)));

dataTable.Columns.Add(new DataColumn("AddSpecializationButton", typeof(string)));

foreach (SpecializationTable spec in specializationTableData)

{

DataRow dataRow = dataTable.NewRow();

dataRow["SpecializationID"] = spec.SpecializationTableID;

dataRow["SpecializationName"] = spec.Name;

dataRow["SpecializationNumber"] = spec.SpecializationNumber;

dataTable.Rows.Add(dataRow);

}

GridView2.DataSource = dataTable;

GridView2.DataBind();

}

protected void AddSpecializationButtonClick(object sender, EventArgs e)

{

Button button = (Button)sender;

{

KPIWebDataContext kPiDataContext = new KPIWebDataContext();

var par = button.CommandArgument.ToString();

var check =

(from a in kPiDataContext.SpecializationTable where a.SpecializationNumber == par select a)

.FirstOrDefault(); // выбираем специальности по её коду

using (KPIWebDataContext kpiWebDataContext = new KPIWebDataContext()) // проверяем есть/нет записываем в базу

{

FourthLevelSubdivisionTable forthlvlsudtab = (from a in kpiWebDataContext.FourthLevelSubdivisionTable

where a.FK\_Specialization == check.SpecializationTableID && a.FK\_ThirdLevelSubdivisionTable == Convert.ToInt32(DropDownList3.SelectedItem.Value)

&& a.Active == true

select a).FirstOrDefault();

if (forthlvlsudtab != null)

{

forthlvlsudtab.Active = true;

}

else

{

forthlvlsudtab = new FourthLevelSubdivisionTable();

forthlvlsudtab.FK\_Specialization = check.SpecializationTableID;

forthlvlsudtab.Active = true;

forthlvlsudtab.Name = check.Name;

forthlvlsudtab.FK\_ThirdLevelSubdivisionTable = Convert.ToInt32(DropDownList3.SelectedItem.Value);

kpiWebDataContext.FourthLevelSubdivisionTable.InsertOnSubmit(forthlvlsudtab);

}

kpiWebDataContext.SubmitChanges();

// Добавляем запись в таблицу параметров для этой специальности

FourthLevelParametrs fourthLevelParametrs = (from a in kpiWebDataContext.FourthLevelParametrs

where a.FourthLevelParametrsID == forthlvlsudtab.FourthLevelSubdivisionTableID

&& a.Active == true

select a).FirstOrDefault();

if (fourthLevelParametrs == null)

{

fourthLevelParametrs = new FourthLevelParametrs();

fourthLevelParametrs.Active = true;

fourthLevelParametrs.IsModernEducationTechnologies = false;

fourthLevelParametrs.IsNetworkComunication = false;

fourthLevelParametrs.IsInvalidStudentsFacilities = false;

fourthLevelParametrs.IsForeignStudentsAccept = false;

fourthLevelParametrs.FourthLevelParametrsID = forthlvlsudtab.FourthLevelSubdivisionTableID;

var code = (from f4 in kpiWebDataContext.FourthLevelSubdivisionTable // получаем код специальности

join spec in kpiWebDataContext.SpecializationTable

on f4.FK\_Specialization equals spec.SpecializationTableID

where f4.FourthLevelSubdivisionTableID == forthlvlsudtab.FourthLevelSubdivisionTableID && f4.Active == true

select spec.SpecializationNumber).FirstOrDefault();

fourthLevelParametrs.SpecType = Action.Encode(code);

kpiWebDataContext.FourthLevelParametrs.InsertOnSubmit(fourthLevelParametrs);

}

kpiWebDataContext.SubmitChanges();

FillGridView(kPiDataContext);

}

//Response.Redirect("~/StatisticsDepartment/AddSpecialization.aspx");

}

}

private void FillGridView(KPIWebDataContext kPiDataContext)

{

List<SpecializationTable> specializationTableData = (from a in kPiDataContext.SpecializationTable

join b in kPiDataContext.FourthLevelSubdivisionTable

on a.SpecializationTableID equals b.FK\_Specialization

where b.FK\_ThirdLevelSubdivisionTable == Convert.ToInt32(DropDownList3.SelectedItem.Value)

&& b.Active == true

select a).ToList();

CheckBox1.Checked = (from a in kPiDataContext.ThirdLevelParametrs

where a.ThirdLevelParametrsID == Convert.ToInt32(DropDownList3.SelectedItem.Value) && a.Active == true

select a.CanGraduate).FirstOrDefault();

DataTable dataTable = new DataTable();

dataTable.Columns.Add(new DataColumn("SpecializationID", typeof (string)));

dataTable.Columns.Add(new DataColumn("SpecializationName", typeof (string)));

dataTable.Columns.Add(new DataColumn("FourthlvlId", typeof (int)));

dataTable.Columns.Add(new DataColumn("SpecNumber", typeof (string)));

dataTable.Columns.Add(new DataColumn("DeleteSpecializationLabel", typeof (string)));

dataTable.Columns.Add(new DataColumn("DeleteSpecializationButton", typeof (string)));

foreach (SpecializationTable spec in specializationTableData)

{

DataRow dataRow = dataTable.NewRow();

dataRow["SpecializationID"] = spec.SpecializationTableID;

dataRow["SpecializationName"] = spec.Name;

dataRow["FourthlvlId"] = (from a in kPiDataContext.FourthLevelSubdivisionTable

where

a.FK\_ThirdLevelSubdivisionTable == Convert.ToInt32(DropDownList3.SelectedItem.Value) &&

a.FK\_Specialization == spec.SpecializationTableID

&& a.Active == true

select a.FourthLevelSubdivisionTableID).FirstOrDefault();

dataRow["SpecNumber"] = spec.SpecializationNumber;

dataTable.Rows.Add(dataRow);

}

ViewState["GridviewSpec"] = dataTable;

GridView1.DataSource = dataTable;

GridView1.DataBind();

KPIWebDataContext kpiWeb = new KPIWebDataContext();

var Isgraduate = (from a in kpiWeb.ThirdLevelParametrs

where a.ThirdLevelParametrsID == Convert.ToInt32(DropDownList3.SelectedItem.Value) && a.Active == true

select a).FirstOrDefault();

if ((from a in kpiWeb.FourthLevelSubdivisionTable

where

a.FK\_ThirdLevelSubdivisionTable == Convert.ToInt32(DropDownList3.SelectedItem.Value) &&

a.Active == true

select a).Count() > 0 && Isgraduate != null ) // в общем если на кафедре специальностей больше чем 0, то она является выпускающей.

Isgraduate.CanGraduate = true;

else if (Isgraduate != null)

Isgraduate.CanGraduate = false;

kpiWeb.SubmitChanges();

CheckBox1.Checked = (from a in kPiDataContext.ThirdLevelParametrs

where a.ThirdLevelParametrsID == Convert.ToInt32(DropDownList3.SelectedItem.Value)

&& a.Active == true

select a.CanGraduate).FirstOrDefault();

CheckBox2.Checked = (from t in kPiDataContext.ThirdLevelParametrs

where t.ThirdLevelParametrsID == Convert.ToInt32(DropDownList3.SelectedItem.Value)

&& t.Active == true

select t.IsBasic).FirstOrDefault().GetValueOrDefault();

}

protected void DropDownList3\_SelectedIndexChanged(object sender, EventArgs e)

{

if (DropDownList3.SelectedItem != null) ViewState["Selected"] = true;

else ViewState["Selected"] = false;

KPIWebDataContext kPiDataContext = new KPIWebDataContext();

FillGridView(kPiDataContext);

}

protected void DeleteSpecializationButtonClick(object sender, EventArgs e)

{

Button button = (Button)sender;

{

using (KPIWebDataContext kPiDataContext = new KPIWebDataContext())

{

var check =

(from a in kPiDataContext.FourthLevelSubdivisionTable

where

a.FourthLevelSubdivisionTableID == Convert.ToInt32(button.CommandArgument) && a.Active == true

select a)

.FirstOrDefault();

check.Active = false;

kPiDataContext.SubmitChanges();

FillGridView(kPiDataContext);

}

//Response.Redirect("~/StatisticsDepartment/AddSpecialization.aspx");

}

}

protected void CheckBox2\_CheckedChanged(object sender, EventArgs e)

{

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Filename: AddSpecialization.aspx.designer.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//------------------------------------------------------------------------------

// <автоматически создаваемое>

// Этот код создан программой.

//

// Изменения в этом файле могут привести к неправильной работе и будут потеряны в случае

// повторной генерации кода.

// </автоматически создаваемое>

//------------------------------------------------------------------------------

namespace KPIWeb.StatisticsDepartment {

public partial class AddSpecialization {

/// <summary>

/// Label1 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Label Label1;

/// <summary>

/// DropDownList1 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.DropDownList DropDownList1;

/// <summary>

/// Label2 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Label Label2;

/// <summary>

/// DropDownList2 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.DropDownList DropDownList2;

/// <summary>

/// Label3 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Label Label3;

/// <summary>

/// DropDownList3 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.DropDownList DropDownList3;

/// <summary>

/// CheckBox1 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.CheckBox CheckBox1;

/// <summary>

/// CheckBox2 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.CheckBox CheckBox2;

/// <summary>

/// Label4 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Label Label4;

/// <summary>

/// GridView1 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.GridView GridView1;

/// <summary>

/// Button1 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Button Button1;

/// <summary>

/// Label5 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Label Label5;

/// <summary>

/// TextBox1 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.TextBox TextBox1;

/// <summary>

/// Button2 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Button Button2;

/// <summary>

/// GridView2 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.GridView GridView2;

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Filename: BasicParametrs.aspx.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using System;

using System.Collections.Generic;

using System.Configuration;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

using System.Web.WebPages;

using Microsoft.Ajax.Utilities;

namespace KPIWeb.StatisticsDepartment

{

public partial class BasicParametrs : System.Web.UI.Page

{

protected void Page\_Load(object sender, EventArgs e)

{

Serialization UserSer = (Serialization)Session["UserID"];

if (UserSer == null)

{

Response.Redirect("~/Default.aspx");

}

int userID = UserSer.Id;

KPIWebDataContext kPiDataContext = new KPIWebDataContext();

UsersTable userTable =

(from a in kPiDataContext.UsersTable where a.UsersTableID == userID select a).FirstOrDefault();

if ((userTable.AccessLevel != 10)&&(userTable.AccessLevel != 9))

{

Response.Redirect("~/Default.aspx");

}

}

protected void Button1\_Click(object sender, EventArgs e)

{

KPIWebDataContext kPiDataContext = new KPIWebDataContext();

List<BasicParametersTable> basicTable = (from item in kPiDataContext.BasicParametersTable select item).ToList();

string tmpStr = "0";

string str;

int idTmp;

foreach (BasicParametersTable tableRow in basicTable)

{

idTmp = tableRow.BasicParametersTableID;

str = tableRow.AbbreviationEN;

if ((str != null) && (str != " ") && (!str.IsEmpty()))

{

if (!str.IsFloat())

{

int a = (from basic in kPiDataContext.BasicParametersTable

where

basic.AbbreviationEN == str

select basic).Count();

if (a > 1)

{

tmpStr += "\r\n" + "ID=" + idTmp.ToString() + " " + a.ToString() +

" включений аббревиатуры " + str;

}

else if (a < 1)

{

tmpStr += "\r\n" + "ID=" + idTmp.ToString() + " " + str +

" такой аббревиатеры не существует, это странно";

}

}

else

{

tmpStr += "\r\n" + "ID=" + idTmp.ToString() + " Аббревиатура не может быть числом " + str;

}

}

else

{

tmpStr += "\r\n" + "ID=" + idTmp.ToString() + " Поле аббревиатуры должно быть заполнено " + str;

}

}

if (tmpStr != "0")

{

TextBox1.Text = tmpStr;

}

else

{

TextBox1.Text = "Базовые показатели введены корректно";

}

}

protected void Menu1\_MenuItemClick(object sender, MenuEventArgs e)

{

MultiView1.ActiveViewIndex = Int32.Parse(e.Item.Value);

}

protected void Button2\_Click(object sender, EventArgs e)

{

if (TextBox8.Text == "")

{ Page.ClientScript.RegisterClientScriptBlock(typeof(Page), "Script", "alert(' Введите ID базового показателя ');", true); }

else

{

KPIWebDataContext kPiDataContext = new KPIWebDataContext();

BasicParametersTable basicParametr = (from item in kPiDataContext.BasicParametersTable where item.BasicParametersTableID == Convert.ToInt32(TextBox8.Text) select item).FirstOrDefault();

if (basicParametr != null)

{

TextBox2.Text = basicParametr.BasicParametersTableID.ToString();

if (basicParametr.Active)

{

CheckBox1.Checked = true;

}

else

{

CheckBox1.Checked = false;

}

TextBox4.Text = basicParametr.Name;

TextBox5.Text = basicParametr.AbbreviationEN;

TextBox6.Text = basicParametr.AbbreviationRU;

TextBox7.Text = basicParametr.Measure;

}

else

{

Page.ClientScript.RegisterClientScriptBlock(typeof(Page), "Script", "alert('Базового показателя с таким ID не существует');", true);

}

}

}

protected void Button3\_Click(object sender, EventArgs e)

{

if (TextBox8.Text!="")

{

KPIWebDataContext kPiDataContext = new KPIWebDataContext();

BasicParametersTable basicParametr = (from item in kPiDataContext.BasicParametersTable

where item.BasicParametersTableID == Convert.ToInt32(TextBox8.Text)

select item).FirstOrDefault();

if (CheckBox1.Checked == true)

{

basicParametr.Active = true;

}

else

{

basicParametr.Active = false;

}

basicParametr.Name = TextBox4.Text;

basicParametr.AbbreviationEN = TextBox5.Text;

basicParametr.AbbreviationRU = TextBox6.Text;

basicParametr.Measure = TextBox7.Text;

kPiDataContext.SubmitChanges();

}

else

{

Page.ClientScript.RegisterClientScriptBlock(typeof(Page), "Script", "alert('Нужно загрузить базовый показатель');", true);

}

}

protected void Button4\_Click(object sender, EventArgs e)

{

KPIWebDataContext kPiDataContext = new KPIWebDataContext();

string tmpStr;

tmpStr = TextBox9.Text;

string[] tmpStrArr = tmpStr.Split('\r');

int i = 0;

foreach (string tmpStrf in tmpStrArr)

{

string tmp = tmpStrf.Replace("\n", "");

if (((tmp.Split('#').Length - 1) != 5) && (tmp != ""))

{

Page.ClientScript.RegisterClientScriptBlock(typeof(Page), "Script", "alert('Ошибка в строке"+i.ToString()+"');", true);

i = 0;

break;

}

i++;

}

if (i>0)

{

foreach (string tmpStrf in tmpStrArr)

{

if (tmpStrf.Length > 10)

{

string tmp = tmpStrf.Replace("\n", "");

BasicParametersTable basicParametr = new BasicParametersTable();

string[] strArrf = tmp.Split('#');

strArrf[0] = strArrf[0].TrimEnd();

strArrf[0] = strArrf[0].TrimStart();

strArrf[1] = strArrf[1].TrimEnd();

strArrf[1] = strArrf[1].TrimStart();

strArrf[2] = strArrf[2].TrimEnd();

strArrf[2] = strArrf[2].TrimStart();

strArrf[3] = strArrf[3].TrimEnd();

strArrf[3] = strArrf[3].TrimStart();

strArrf[4] = strArrf[4].TrimEnd();

strArrf[4] = strArrf[4].TrimStart();

strArrf[5] = strArrf[5].TrimEnd();

strArrf[5] = strArrf[5].TrimStart();

basicParametr.Active = true;

basicParametr.Name = strArrf[0];

basicParametr.AbbreviationEN = strArrf[1];

basicParametr.AbbreviationRU = strArrf[2];

basicParametr.Measure = strArrf[3];

// basicParametr.SubvisionLevel = Convert.ToInt32(strArrf[4]);

// basicParametr.ForeignStudents = Convert.ToInt32(strArrf[5]);

kPiDataContext.BasicParametersTable.InsertOnSubmit(basicParametr);

}

}

}

kPiDataContext.SubmitChanges();

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Filename: BasicParametrs.aspx.designer.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//------------------------------------------------------------------------------

// <автоматически создаваемое>

// Этот код создан программой.

//

// Изменения в этом файле могут привести к неправильной работе и будут потеряны в случае

// повторной генерации кода.

// </автоматически создаваемое>

//------------------------------------------------------------------------------

namespace KPIWeb.StatisticsDepartment {

public partial class BasicParametrs {

/// <summary>

/// Menu1 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Menu Menu1;

/// <summary>

/// MultiView1 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.MultiView MultiView1;

/// <summary>

/// Tab1 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.View Tab1;

/// <summary>

/// Button1 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Button Button1;

/// <summary>

/// TextBox1 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.TextBox TextBox1;

/// <summary>

/// Tab2 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.View Tab2;

/// <summary>

/// TextBox8 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.TextBox TextBox8;

/// <summary>

/// Button2 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Button Button2;

/// <summary>

/// TextBox2 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.TextBox TextBox2;

/// <summary>

/// CheckBox1 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.CheckBox CheckBox1;

/// <summary>

/// TextBox4 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.TextBox TextBox4;

/// <summary>

/// TextBox5 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.TextBox TextBox5;

/// <summary>

/// TextBox6 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.TextBox TextBox6;

/// <summary>

/// TextBox7 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.TextBox TextBox7;

/// <summary>

/// Button3 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Button Button3;

/// <summary>

/// Tab3 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.View Tab3;

/// <summary>

/// TextBox9 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.TextBox TextBox9;

/// <summary>

/// Button4 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Button Button4;

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Filename: Confirm.aspx.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using System;

using System.Collections.Generic;

using System.Data;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

namespace KPIWeb.StatisticsDepartment

{

public partial class Confirm : System.Web.UI.Page

{

protected void Page\_Load(object sender, EventArgs e)

{

KPIWebDataContext kPiDataContext = new KPIWebDataContext();

Serialization UserSer = (Serialization)Session["UserID"];

if (UserSer == null)

{

Response.Redirect("~/Default.aspx");

}

int userID = UserSer.Id;

UsersTable userTable =

(from a in kPiDataContext.UsersTable where a.UsersTableID == userID select a).FirstOrDefault();

if ((userTable.AccessLevel != 10) && (userTable.AccessLevel != 9))

{

Response.Redirect("~/Default.aspx");

}

if (!IsPostBack)

{

List<ReportArchiveTable> report =

(from item in kPiDataContext.ReportArchiveTable

where item.Active == true

select item).OrderBy(mc => mc.Name).ToList();

var dictionary = new Dictionary<int, string>();

dictionary.Add(0, "Выберите значение");

foreach (var item in report)

dictionary.Add(item.ReportArchiveTableID, item.Name);

DropDownList1.DataTextField = "Value";

DropDownList1.DataValueField = "Key";

DropDownList1.DataSource = dictionary;

DropDownList1.DataBind();

List<UsersTable> users =

(from item in kPiDataContext.UsersTable

where item.Active == true

&& item.AccessLevel == 5

select item).OrderBy(mc => mc.Email).ToList();

//var dictionary = new Dictionary<int, string>();

dictionary.Clear();

dictionary.Add(0, "Выберите значение");

foreach (var item in users)

dictionary.Add(item.UsersTableID, item.Email);

DropDownList2.DataTextField = "Value";

DropDownList2.DataValueField = "Key";

DropDownList2.DataSource = dictionary;

DropDownList2.DataBind();

}

List<ConfirmationHistory> confirms =

(from a in kPiDataContext.ConfirmationHistory

where

((a.FK\_ReportTable == Convert.ToInt32(DropDownList1.Items[DropDownList1.SelectedIndex].Value))

|| (Convert.ToInt32(DropDownList1.Items[DropDownList1.SelectedIndex].Value)==0))

&&

((a.FK\_UsersTable == Convert.ToInt32(DropDownList2.Items[DropDownList2.SelectedIndex].Value))

|| (Convert.ToInt32(DropDownList2.Items[DropDownList2.SelectedIndex].Value)==0))

select a).ToList();

DataTable dataTable = new DataTable();

dataTable.Columns.Add(new DataColumn("Name", typeof(string)));

dataTable.Columns.Add(new DataColumn("Comment", typeof(string)));

dataTable.Columns.Add(new DataColumn("Date", typeof(string)));

dataTable.Columns.Add(new DataColumn("ParamName", typeof(string)));

dataTable.Columns.Add(new DataColumn("ConfirmUser", typeof(string)));

foreach (ConfirmationHistory ConfHist in confirms)

{

DataRow dataRow = dataTable.NewRow();

dataRow["Name"] = ConfHist.Name;

dataRow["Comment"] = ConfHist.Comment;

// dataRow["Name"] =

dataRow["Date"] = ConfHist.Date.ToString();

if (ConfHist.FK\_BasicParamTable !=null)

{

dataRow["ParamName"] = (from a in kPiDataContext.BasicParametersTable where a.BasicParametersTableID == ConfHist.FK\_BasicParamTable select a.Name).FirstOrDefault();

}

else if (ConfHist.FK\_CalculatedParamTable!=null)

{

dataRow["ParamName"] = (from a in kPiDataContext.CalculatedParametrs where a.CalculatedParametrsID == ConfHist.FK\_CalculatedParamTable select a.Name).FirstOrDefault();

}

else if (ConfHist.FK\_IndicatorsTable !=null)

{

dataRow["ParamName"] = (from a in kPiDataContext.IndicatorsTable where a.IndicatorsTableID == ConfHist.FK\_IndicatorsTable select a.Name).FirstOrDefault();

}

else

{

continue;

}

if (ConfHist.FK\_UsersTable != null)

{

dataRow["ConfirmUser"] = (from a in kPiDataContext.UsersTable where a.UsersTableID == ConfHist.FK\_UsersTable select a.Email).FirstOrDefault();

}

else

{

continue;

}

dataTable.Rows.Add(dataRow);

}

GridView1.DataSource = dataTable;

GridView1.DataBind();

}

protected void GridView1\_SelectedIndexChanged(object sender, EventArgs e)

{

}

protected void DropDownList1\_SelectedIndexChanged(object sender, EventArgs e)

{

}

protected void DropDownList2\_SelectedIndexChanged(object sender, EventArgs e)

{

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Filename: Confirm.aspx.designer.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//------------------------------------------------------------------------------

// <auto-generated>

// This code was generated by a tool.

//

// Changes to this file may cause incorrect behavior and will be lost if

// the code is regenerated.

// </auto-generated>

//------------------------------------------------------------------------------

namespace KPIWeb.StatisticsDepartment {

public partial class Confirm {

/// <summary>

/// Label1 control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.Label Label1;

/// <summary>

/// DropDownList1 control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.DropDownList DropDownList1;

/// <summary>

/// Label2 control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.Label Label2;

/// <summary>

/// DropDownList2 control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.DropDownList DropDownList2;

/// <summary>

/// GridView1 control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.GridView GridView1;

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Filename: Document.aspx.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

using System.Data;

namespace KPIWeb.StatisticsDepartment

{

public partial class Document : System.Web.UI.Page

{

protected void Page\_Load(object sender, EventArgs e)

{

KPIWebDataContext kPiDataContext = new KPIWebDataContext();

Serialization UserSer = (Serialization)Session["UserID"];

if (UserSer == null)

{

Response.Redirect("~/Default.aspx");

}

int userID = UserSer.Id;

UsersTable userTable =

(from a in kPiDataContext.UsersTable where a.UsersTableID == userID select a).FirstOrDefault();

if ((userTable.AccessLevel != 10) && (userTable.AccessLevel != 9))

{

Response.Redirect("~/Default.aspx");

}

if (!IsPostBack)

{

List<DocumentTable> docs = (from a in kPiDataContext.DocumentTable where a.Active == true select a).ToList();

DataTable dataTable = new DataTable();

dataTable.Columns.Add(new DataColumn("DocumentName", typeof(string)));

dataTable.Columns.Add(new DataColumn("DocumentLink", typeof(string)));

GridView1.DataSource = docs;

GridView1.DataBind();

}

}

protected void Button1\_Click(object sender, EventArgs e)

{

if (TextBox1.Text != "")

{

KPIWebDataContext kPiDataContext = new KPIWebDataContext();

DocumentTable doc = new DocumentTable();

doc.DocumentName = TextBox2.Text;

doc.DocumentLink = TextBox1.Text;

doc.Active = true;

kPiDataContext.DocumentTable.InsertOnSubmit(doc);

kPiDataContext.SubmitChanges();

string savepath = Server.MapPath("//docs//");

string fileName = TextBox1.Text;

if (FileUpload1.HasFile)

{

FileUpload1.PostedFile.SaveAs(Server.MapPath("//Rector/docs//" + fileName));

}

else

{

Page.ClientScript.RegisterClientScriptBlock(typeof(Page), "Script", "alert('Документ не прикреплен');" + "document.location = 'Document.aspx';", true);

}

Page.ClientScript.RegisterClientScriptBlock(typeof(Page), "Script", "alert('Документ сохранен');" + "document.location = 'Document.aspx';", true);

}

else

{

Page.ClientScript.RegisterClientScriptBlock(typeof(Page), "Script", "alert('Введите точное имя файла');", true);

}

}

protected void DeleteButtonClick(object sender, EventArgs e)

{

Button button = (Button)sender;

{

using (KPIWebDataContext kPiDataContext = new KPIWebDataContext())

{

var check =

(from a in kPiDataContext.DocumentTable

where

a.DocumentID == Convert.ToInt32(button.CommandArgument)

select a)

.FirstOrDefault();

check.Active = false;

kPiDataContext.SubmitChanges();

Response.Redirect("~/StatisticsDepartment/Document.aspx");

}

}

}

protected void Button2\_Click(object sender, EventArgs e)

{

Response.Redirect("~/Rector/ViewDocument.aspx");

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Filename: Document.aspx.designer.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//------------------------------------------------------------------------------

// <автоматически создаваемое>

// Этот код создан программой.

//

// Изменения в этом файле могут привести к неправильной работе и будут потеряны в случае

// повторной генерации кода.

// </автоматически создаваемое>

//------------------------------------------------------------------------------

namespace KPIWeb.StatisticsDepartment {

public partial class Document {

/// <summary>

/// Button2 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Button Button2;

/// <summary>

/// GridView1 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.GridView GridView1;

/// <summary>

/// Label1 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Label Label1;

/// <summary>

/// TextBox1 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.TextBox TextBox1;

/// <summary>

/// Label2 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Label Label2;

/// <summary>

/// TextBox2 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.TextBox TextBox2;

/// <summary>

/// FileUpload1 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.FileUpload FileUpload1;

/// <summary>

/// Button1 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Button Button1;

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Filename: FastStructure.aspx.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

namespace KPIWeb.StatisticsDepartment

{

public partial class FastStructure : System.Web.UI.Page

{

public class MyObject

{

public int Id;

public int ParentId;

public string Name;

//public string UrlAddr;

//public int Active;

}

private void BindTree(IEnumerable<MyObject> list, TreeNode parentNode)

{

var nodes = list.Where(x => parentNode == null ? x.ParentId == 0 : x.ParentId == int.Parse(parentNode.Value));

foreach (var node in nodes)

{

TreeNode newNode = new TreeNode(node.Name, node.Id.ToString());

/\* if (node.Active == 1)

{

//newNode.NavigateUrl = node.UrlAddr;

}

else\*/

newNode.SelectAction = TreeNodeSelectAction.None;

if (parentNode == null)

{

TreeView1.Nodes.Add(newNode);

}

else

{

parentNode.ChildNodes.Add(newNode);

}

BindTree(list, newNode);

}

}

protected void Page\_Load(object sender, EventArgs e)

{

KPIWebDataContext kPiDataContext = new KPIWebDataContext();

Serialization UserSer = (Serialization)Session["UserID"];

if (UserSer == null)

{

Response.Redirect("~/Default.aspx");

}

int userID = UserSer.Id;

UsersTable userTable =

(from a in kPiDataContext.UsersTable where a.UsersTableID == userID select a).FirstOrDefault();

if ((userTable.AccessLevel != 10) && (userTable.AccessLevel != 9))

{

Response.Redirect("~/Default.aspx");

}

List<MyObject> list = new List<MyObject>();

int allZeroLevel = 0;

int insertZeroLevel = 0;

int confirmZeroLevel = 0;

int i = 1;

int UsrCnt = 0;

List<UsersTable> Users;

string tmp;

string tmp2;

#region get zero leve list

List<ZeroLevelSubdivisionTable> zeroLevelList = (from a in kPiDataContext.ZeroLevelSubdivisionTable

where a.Active == true

select a).ToList();

#endregion

foreach (ZeroLevelSubdivisionTable zeroLevelItem in zeroLevelList)//по каждому университету

{

#region get first level list

List<FirstLevelSubdivisionTable> firstLevelList = (from b in kPiDataContext.FirstLevelSubdivisionTable

/\* join c in kPiDataContext.ReportArchiveAndLevelMappingTable

on b.FirstLevelSubdivisionTableID equals c.FK\_FirstLevelSubmisionTableId\*/

where b.FK\_ZeroLevelSubvisionTable == zeroLevelItem.ZeroLevelSubdivisionTableID

&& b.Active == true

// && c.Active == true

select b).ToList();

#endregion

//TextBox1.Text+="\_\_" + zeroLevelItem.Name +"\n";

i++;

int par0 = i;

UsrCnt = 0;

Users = (from a in kPiDataContext.UsersTable

where a.Active == true

&& a.FK\_ZeroLevelSubdivisionTable == zeroLevelItem.ZeroLevelSubdivisionTableID

&& a.FK\_FirstLevelSubdivisionTable == null

select a).ToList();

tmp = "";

foreach (UsersTable curuser in Users )

{

if (curuser.Confirmed==true)

{

tmp += "<font style=\"color:#00FF00;\">" + curuser.Email + "</font> ";

}

else

{

tmp += "<font style=\"color:#FF0000;\">" + curuser.Email + "</font> ";

}

}

UsrCnt = (from a in kPiDataContext.UsersTable

where a.Active == true

&& a.FK\_ZeroLevelSubdivisionTable == zeroLevelItem.ZeroLevelSubdivisionTableID

select a).Count();

tmp2 = "";

if (UsrCnt>0)

{

tmp2 = "<font style=\"color:#0000FF;\">" + zeroLevelItem.Name + "</font> ";

}

else

{

tmp2 = zeroLevelItem.Name;

}

list.Add(new MyObject() { Id = i, ParentId = 0, Name = tmp2 + " ( " + UsrCnt.ToString() + ") " + tmp });

foreach (FirstLevelSubdivisionTable firstLevelItem in firstLevelList)//по каждой академии

{

#region get second level list

List<SecondLevelSubdivisionTable> secondLevelList =

(from d in kPiDataContext.SecondLevelSubdivisionTable

where d.FK\_FirstLevelSubdivisionTable == firstLevelItem.FirstLevelSubdivisionTableID

&& d.Active == true

select d).ToList();

#endregion

// TextBox1.Text +="\_\_\_\_" + firstLevelItem.Name+"\n";

i++;

int par1 = i;

UsrCnt = 0;

Users = (from a in kPiDataContext.UsersTable

where a.Active == true

&& a.FK\_FirstLevelSubdivisionTable == firstLevelItem.FirstLevelSubdivisionTableID

&& a.FK\_SecondLevelSubdivisionTable == null

select a).ToList();

tmp = "";

foreach (UsersTable curuser in Users)

{

if (curuser.Confirmed == true)

{

tmp += "<font style=\"color:#00FF00;\">" + curuser.Email + "</font> ";

}

else

{

tmp += "<font style=\"color:#FF0000;\">" + curuser.Email + "</font> ";

}

}

UsrCnt = (from a in kPiDataContext.UsersTable

where a.Active == true

&& a.FK\_FirstLevelSubdivisionTable == firstLevelItem.FirstLevelSubdivisionTableID

select a).Count();

tmp2 = "";

if (UsrCnt > 0)

{

tmp2 = "<font style=\"color:#0000FF;\">" + firstLevelItem.Name + "</font> ";

}

else

{

tmp2 = firstLevelItem.Name;

}

list.Add(new MyObject() { Id = i, ParentId = par0, Name = tmp2 + " (" + UsrCnt.ToString() + ") " + tmp });

foreach (SecondLevelSubdivisionTable secondLevelItem in secondLevelList)//по каждому факультету

{

// TextBox1.Text += "\_\_\_\_\_\_" + secondLevelItem.Name + "\n";

#region get third level list

List<ThirdLevelSubdivisionTable> thirdLevelList =

(from f in kPiDataContext.ThirdLevelSubdivisionTable

where f.FK\_SecondLevelSubdivisionTable == secondLevelItem.SecondLevelSubdivisionTableID

&& f.Active == true

select f).ToList();

#endregion

i++;

int par2 = i;

UsrCnt = 0;

Users = (from a in kPiDataContext.UsersTable

where a.Active == true

&& a.FK\_SecondLevelSubdivisionTable == secondLevelItem.SecondLevelSubdivisionTableID

&& a.FK\_ThirdLevelSubdivisionTable == null

select a).ToList();

tmp = "";

foreach (UsersTable curuser in Users)

{

if (curuser.Confirmed == true)

{

tmp += "<font style=\"color:#00FF00;\">" + curuser.Email + "</font> ";

}

else

{

tmp += "<font style=\"color:#FF0000;\">" + curuser.Email + "</font> ";

}

}

UsrCnt = (from a in kPiDataContext.UsersTable

where a.Active == true

&& a.FK\_SecondLevelSubdivisionTable == secondLevelItem.SecondLevelSubdivisionTableID

select a).Count();

tmp2 = "";

if (UsrCnt > 0)

{

tmp2 = "<font style=\"color:#0000FF;\">" + secondLevelItem.Name + "</font> ";

}

else

{

tmp2 = secondLevelItem.Name;

}

list.Add(new MyObject() { Id = i, ParentId = par1, Name = tmp2 + " (" + UsrCnt.ToString() + ") " + tmp });

foreach (ThirdLevelSubdivisionTable thirdLevelItem in thirdLevelList)//по кафедре

{

//TextBox1.Text += "\_\_\_\_\_\_\_\_" + thirdLevelItem.Name + "\n";

#region get fourth level list

/\*

List<FourthLevelSubdivisionTable> fourthLevelList = (from g in kPiDataContext.FourthLevelSubdivisionTable

where

g.FK\_ThirdLevelSubdivisionTable ==

thirdLevelItem.ThirdLevelSubdivisionTableID

&& g.Active == true

select g).ToList();

\*/

#endregion

i++;

int par3 = i;

UsrCnt = 0;

Users = (from a in kPiDataContext.UsersTable

where a.Active == true

&& a.FK\_ThirdLevelSubdivisionTable == thirdLevelItem.ThirdLevelSubdivisionTableID

&& a.FK\_FourthLevelSubdivisionTable == null

select a).ToList();

tmp = "";

foreach (UsersTable curuser in Users)

{

if (curuser.Confirmed == true)

{

tmp += "<font style=\"color:#00FF00;\">" + curuser.Email + "</font> ";

}

else

{

tmp += "<font style=\"color:#FF0000;\">" + curuser.Email + "</font> ";

}

}

UsrCnt = (from a in kPiDataContext.UsersTable

where a.Active == true

&& a.FK\_ThirdLevelSubdivisionTable == thirdLevelItem.ThirdLevelSubdivisionTableID

select a).Count();

tmp2 = "";

if (UsrCnt > 0)

{

tmp2 = "<font style=\"color:#0000FF;\">" + thirdLevelItem.Name + "</font> ";

}

else

{

tmp2 = thirdLevelItem.Name;

}

list.Add(new MyObject() { Id = i, ParentId = par2, Name = tmp2 + " (" + UsrCnt.ToString() + ") " + tmp });

/\*

foreach (FourthLevelSubdivisionTable fourthLevelItem in fourthLevelList)//по специальности

{

TextBox1.Text += "\_\_\_\_\_\_\_\_\_\_" + fourthLevelItem.Name + "\n";

}

\*/

}

}

}

}

BindTree(list, null);

TreeView1.CollapseAll();

//TreeView1.ExpandAll();

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Filename: FastStructure.aspx.designer.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//------------------------------------------------------------------------------

// <auto-generated>

// This code was generated by a tool.

//

// Changes to this file may cause incorrect behavior and will be lost if

// the code is regenerated.

// </auto-generated>

//------------------------------------------------------------------------------

namespace KPIWeb.StatisticsDepartment {

public partial class FastStructure {

/// <summary>

/// TextBox1 control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.TextBox TextBox1;

/// <summary>

/// TreeView1 control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.TreeView TreeView1;

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Filename: Indicators.aspx.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using System;

using System.Collections.Generic;

using System.Configuration;

using System.Drawing;

using System.Linq;

using System.Web;

using System.Web.UI.WebControls;

//using Microsoft.Office.Interop.Excel;

using Page = System.Web.UI.Page;

namespace KPIWeb.StatisticsDepartment

{

public partial class Indicators : System.Web.UI.Page

{

protected void Page\_Load(object sender, EventArgs e)

{

Serialization UserSer = (Serialization)Session["UserID"];

if (UserSer == null)

{

Response.Redirect("~/Default.aspx");

}

int userID = UserSer.Id;

KPIWebDataContext kPiDataContext = new KPIWebDataContext();

UsersTable userTable =

(from a in kPiDataContext.UsersTable where a.UsersTableID == userID select a).FirstOrDefault();

if ((userTable.AccessLevel != 10)&&(userTable.AccessLevel != 9))

{

Response.Redirect("~/Default.aspx");

}

////////////////////////////////////////////////////////////////

if (!Page.IsPostBack)

{

}

}

protected bool abbExists(string abb,int id)

{

KPIWebDataContext kPiDataContext = new KPIWebDataContext();

// int indabbcnt = from a in kPiDataContext.IndicatorsTable

// where a.

int calcabbcnt = (from a in kPiDataContext.CalculatedParametrs

where a.AbbreviationEN == abb

&& a.CalculatedParametrsID!=id

select a).ToList().Count();

int basicabbcnt = (from a in kPiDataContext.BasicParametersTable

where a.AbbreviationEN == abb

&& a.BasicParametersTableID!=id

select a).ToList().Count();

if ((calcabbcnt + basicabbcnt) > 0)

{

return true;

}

else

{

return false;

}

}

protected void DropDownList1\_SelectedIndexChanged(object sender, EventArgs e)

{

Label5.Text = "0";

IndicatorName.Text = "";

IndicatorFormula.Text = "";

IndicatorMeasure.Text = "";

TextBoxAbb.Text = "";

LabelAbbError.Visible = false;

int SelectedValue = -1;

if (int.TryParse(DropDownList1.SelectedValue, out SelectedValue) && SelectedValue != -1)

{

if (SelectedValue > 0)

{

KPIWebDataContext kPiDataContext =

new KPIWebDataContext();

if ((int)ViewState["state"] == 0) ///// целевой показатель

{

IndicatorsTable indicator = (from item in kPiDataContext.IndicatorsTable

where item.IndicatorsTableID == SelectedValue

select item).FirstOrDefault();

if (indicator.Active == true) CheckBox1.Checked = true;

else CheckBox1.Checked = false;

Label5.Text = indicator.IndicatorsTableID.ToString();

IndicatorName.Text = indicator.Name;

IndicatorFormula.Text = indicator.Formula;

IndicatorMeasure.Text = indicator.Measure;

}

else if ((int)ViewState["state"] == 1)///рассчетные показатели

{

CalculatedParametrs calcParams = (from item in kPiDataContext.CalculatedParametrs

where item.CalculatedParametrsID == SelectedValue

select item).FirstOrDefault();

if (calcParams.Active == true) CheckBox1.Checked = true;

else CheckBox1.Checked = false;

Label5.Text = calcParams.CalculatedParametrsID.ToString();

IndicatorName.Text = calcParams.Name;

TextBoxAbb.Text = calcParams.AbbreviationEN;

IndicatorFormula.Text = calcParams.Formula;

IndicatorMeasure.Text = calcParams.Measure;

}

}

}

else

{

//error

}

}

protected void Button3\_Click(object sender, EventArgs e)

{

KPIWebDataContext kPiDataContext = new KPIWebDataContext();

int SelectedValue = -1;

if (int.TryParse(Label5.Text, out SelectedValue) && SelectedValue != -1)

{

if (SelectedValue > 0)

{

if ((int)ViewState["state"] == 0) ///// целевой показатель

{

IndicatorsTable indicators = (from item in kPiDataContext.IndicatorsTable

where item.IndicatorsTableID == SelectedValue

select item).FirstOrDefault();

if (CheckBox1.Checked) indicators.Active = true;

else indicators.Active = false;

indicators.Name = IndicatorName.Text;

indicators.Formula = IndicatorFormula.Text;

indicators.Measure = IndicatorMeasure.Text;

Page.ClientScript.RegisterClientScriptBlock(typeof(Page), "Script", "alert('Целевой показатель изменен');", true);

}

else if ((int)ViewState["state"] == 1)///рассчетные показатели

{

if (abbExists(TextBoxAbb.Text,

Convert.ToInt32(DropDownList1.Items[DropDownList1.SelectedIndex].Value)))

{

Page.ClientScript.RegisterClientScriptBlock(typeof (Page), "Script",

"alert('Аббревиатура не является уникальной, измените её');", true);

}

else

{

CalculatedParametrs calcParams = (from item in kPiDataContext.CalculatedParametrs

where item.CalculatedParametrsID == SelectedValue

select item).FirstOrDefault();

if (CheckBox1.Checked) calcParams.Active = true;

else calcParams.Active = false;

calcParams.Name = IndicatorName.Text;

calcParams.Formula = IndicatorFormula.Text;

calcParams.AbbreviationEN = TextBoxAbb.Text;

calcParams.Measure = IndicatorMeasure.Text;

Page.ClientScript.RegisterClientScriptBlock(typeof (Page), "Script",

"alert('Расчётный показатель изменен');", true);

}

}

}

else

{

if ((int)ViewState["state"] == 0) ///// целевой показатель

{

IndicatorsTable indicators = new IndicatorsTable();

if (CheckBox1.Checked) indicators.Active = true;

else indicators.Active = false;

indicators.Name = IndicatorName.Text;

indicators.Formula = IndicatorFormula.Text;

indicators.Measure = IndicatorMeasure.Text;

kPiDataContext.IndicatorsTable.InsertOnSubmit(indicators);

Page.ClientScript.RegisterClientScriptBlock(typeof(Page), "Script", "alert('Целевой показатель создан');", true);

}

else if ((int)ViewState["state"] == 1)///рассчетные показатели

{

if (abbExists(TextBoxAbb.Text,Convert.ToInt32(DropDownList1.Items[DropDownList1.SelectedIndex].Value)))

{

Page.ClientScript.RegisterClientScriptBlock(typeof(Page), "Script", "alert('Аббревиатура не является уникальной, измените её');", true);

}

else

{

CalculatedParametrs calcParams = new CalculatedParametrs();

if (CheckBox1.Checked) calcParams.Active = true;

else calcParams.Active = false;

calcParams.Name = IndicatorName.Text;

calcParams.Formula = IndicatorFormula.Text;

calcParams.Measure = IndicatorMeasure.Text;

calcParams.AbbreviationEN = TextBoxAbb.Text;

kPiDataContext.CalculatedParametrs.InsertOnSubmit(calcParams);

Page.ClientScript.RegisterClientScriptBlock(typeof(Page), "Script", "alert('Расчётный параметр создан');", true);

}

}

}

kPiDataContext.SubmitChanges();

}

} //Добавление в БД // внутри разделение на целевой показатель и расчётные

protected void Button2\_Click(object sender, EventArgs e)

{

}///Рассчет

protected void Button4\_Click(object sender, EventArgs e)

{

KPIWebDataContext kPiDataContext = new KPIWebDataContext();

List<BasicParametersTable> basicParamsList = (from a in kPiDataContext.BasicParametersTable

where a.Name.Contains(SearchBox.Text) || a.AbbreviationRU.Contains(SearchBox.Text) || a.AbbreviationEN.Contains(SearchBox.Text)

select a).ToList();

GridView1.DataSource = basicParamsList;

GridView1.DataBind();

}////ПОИСК

protected void Button5\_Click(object sender, EventArgs e)

{

if (TextBox2.Text == "") TextBox2.Text = 0.ToString();

if (TextBox3.Text == "") TextBox3.Text = 0.ToString();

if (TextBox4.Text == "") TextBox4.Text = 0.ToString();

if (TextBox5.Text == "") TextBox5.Text = 0.ToString();

if (TextBox6.Text == "") TextBox6.Text = 0.ToString();

if (TextBox7.Text == "") TextBox7.Text = 0.ToString();

TextBox1.Text = CalculateAbb.CalculateForLevel(1, IndicatorFormula0.Text, 2007, 0,

Convert.ToInt32(TextBox2.Text), Convert.ToInt32(TextBox3.Text), Convert.ToInt32(TextBox4.Text),

Convert.ToInt32(TextBox5.Text), Convert.ToInt32(TextBox6.Text), Convert.ToInt32(TextBox7.Text),0).ToString();

}//Рассчет с подразделениями

protected void Menu1\_MenuItemClick(object sender, MenuEventArgs e)

{

int viewidx = Int32.Parse(e.Item.Value);

switch (viewidx)

{

case 0://работа с целевой показатель

{

if (ViewState["state"]!=null)

if ((int)ViewState["state"] != 0)

{

CheckBox1.Checked = false;

Label5.Text = "0";

IndicatorName.Text = "";

IndicatorFormula.Text = "";

//TextBox8.Text = "";

IndicatorMeasure.Text = "";

}

ViewState["state"] = 0;

addtitle.Text = "Форма редактирования целевого показателя";

Label1.Text = "Название целевого показателя";

Button3.Text = "Сохранить изменения целевого показателя";

TextBoxAbb.Visible = false;

LabelAbb.Visible = false;

LabelAbbError.Visible = false;

// DropDownList1.DataSource=""

KPIWebDataContext kPiDataContext =

new KPIWebDataContext();

List<IndicatorsTable> indicatorList = (from item in kPiDataContext.IndicatorsTable select item).ToList();

var dictionary = new Dictionary<int, string>();

dictionary.Add(0, "Добавить новый целевой показатель");

foreach (IndicatorsTable item in indicatorList)

dictionary.Add(item.IndicatorsTableID, item.Name);

DropDownList1.DataTextField = "Value";

DropDownList1.DataValueField = "Key";

DropDownList1.DataSource = dictionary;

DropDownList1.DataBind();

MultiView1.ActiveViewIndex = 0;

break;

}

case 1://работа с расчётными показателями

{

if (ViewState["state"] != null)

if ((int)ViewState["state"] != 1)

{

CheckBox1.Checked = false;

Label5.Text = "0";

IndicatorName.Text = "";

IndicatorFormula.Text = "";

TextBoxAbb.Text = "";

IndicatorMeasure.Text = "";

}

ViewState["state"] = 1;

addtitle.Text = "Форма редактирования расчётного показателя";

Label1.Text = "Название расчётного показателя";

Button3.Text = "Сохранить изменения расчётного показателя";

TextBoxAbb.Visible = true;

LabelAbb.Visible = true;

LabelAbbError.Visible = false;

KPIWebDataContext kPiDataContext =

new KPIWebDataContext();

List<CalculatedParametrs> calcParamsTable = (from item in kPiDataContext.CalculatedParametrs select item).ToList();

var dictionary = new Dictionary<int, string>();

dictionary.Add(0, "Добавить новый расчётный показатель");

foreach (CalculatedParametrs item in calcParamsTable)

dictionary.Add(item.CalculatedParametrsID, item.Name);

DropDownList1.DataTextField = "Value";

DropDownList1.DataValueField = "Key";

DropDownList1.DataSource = dictionary;

DropDownList1.DataBind();

MultiView1.ActiveViewIndex = 0;

break;

}

case 2://проверка формул

{

MultiView1.ActiveViewIndex = 1;

break;

}

case 3://поисх аббревиатур

{

MultiView1.ActiveViewIndex = 2;

break;

}

default://невозможно)

{

break;

}

}

}

protected void TextBoxAbb\_TextChanged(object sender, EventArgs e)

{

LabelAbbError.Visible = true;

if (!abbExists(TextBoxAbb.Text,Convert.ToInt32(DropDownList1.Items[DropDownList1.SelectedIndex].Value)))

{

LabelAbbError.ForeColor = Color.Green;

LabelAbbError.Text = "Аббревиатура уникальна.";

LabelAbbError.Visible = true;

}

else

{

LabelAbbError.ForeColor = Color.Red;

LabelAbbError.Text = "Аббревиатура не является уникальной.";

LabelAbbError.Visible = true;

}

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Filename: Indicators.aspx.designer.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//------------------------------------------------------------------------------

// <автоматически создаваемое>

// Этот код создан программой.

//

// Изменения в этом файле могут привести к неправильной работе и будут потеряны в случае

// повторной генерации кода.

// </автоматически создаваемое>

//------------------------------------------------------------------------------

namespace KPIWeb.StatisticsDepartment {

public partial class Indicators {

/// <summary>

/// Menu1 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Menu Menu1;

/// <summary>

/// MultiView1 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.MultiView MultiView1;

/// <summary>

/// Tab1 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.View Tab1;

/// <summary>

/// addtitle элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Label addtitle;

/// <summary>

/// DropDownList1 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.DropDownList DropDownList1;

/// <summary>

/// Label5 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Label Label5;

/// <summary>

/// Label1 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Label Label1;

/// <summary>

/// IndicatorName элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.TextBox IndicatorName;

/// <summary>

/// Label3 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Label Label3;

/// <summary>

/// IndicatorMeasure элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.TextBox IndicatorMeasure;

/// <summary>

/// CheckBox1 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.CheckBox CheckBox1;

/// <summary>

/// LabelAbb элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Label LabelAbb;

/// <summary>

/// TextBoxAbb элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.TextBox TextBoxAbb;

/// <summary>

/// LabelAbbError элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Label LabelAbbError;

/// <summary>

/// Label2 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Label Label2;

/// <summary>

/// IndicatorFormula элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.TextBox IndicatorFormula;

/// <summary>

/// Button3 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Button Button3;

/// <summary>

/// Tab2 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.View Tab2;

/// <summary>

/// IndicatorFormula0 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.TextBox IndicatorFormula0;

/// <summary>

/// Button2 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Button Button2;

/// <summary>

/// Button5 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Button Button5;

/// <summary>

/// TextBox2 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.TextBox TextBox2;

/// <summary>

/// TextBox3 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.TextBox TextBox3;

/// <summary>

/// TextBox4 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.TextBox TextBox4;

/// <summary>

/// TextBox5 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.TextBox TextBox5;

/// <summary>

/// TextBox6 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.TextBox TextBox6;

/// <summary>

/// TextBox7 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.TextBox TextBox7;

/// <summary>

/// Label8 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Label Label8;

/// <summary>

/// TextBox1 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.TextBox TextBox1;

/// <summary>

/// Tab3 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.View Tab3;

/// <summary>

/// Label7 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Label Label7;

/// <summary>

/// SearchBox элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.TextBox SearchBox;

/// <summary>

/// Button4 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Button Button4;

/// <summary>

/// GridView1 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.GridView GridView1;

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Filename: Manual.aspx.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using System;

using System.Collections.Generic;

using System.Data;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

namespace KPIWeb.StatisticsDepartment

{

public partial class Manual : System.Web.UI.Page

{

protected void Page\_Load(object sender, EventArgs e)

{

KPIWebDataContext kPiDataContext = new KPIWebDataContext();

Serialization UserSer = (Serialization)Session["UserID"];

if (UserSer == null)

{

Response.Redirect("~/Default.aspx");

}

int userID = UserSer.Id;

UsersTable userTable =

(from a in kPiDataContext.UsersTable where a.UsersTableID == userID select a).FirstOrDefault();

if ((userTable.AccessLevel != 10) && (userTable.AccessLevel != 9))

{

Response.Redirect("~/Default.aspx");

}

if (!IsPostBack)

{

List<ManualTable> manuals = (from a in kPiDataContext.ManualTable where a.Active == true select a).ToList();

DataTable dataTable = new DataTable();

dataTable.Columns.Add(new DataColumn("ManualName", typeof(string)));

dataTable.Columns.Add(new DataColumn("ManualLink", typeof(string)));

GridView1.DataSource = manuals;

GridView1.DataBind();

}

}

protected void Button1\_Click(object sender, EventArgs e)

{

if (TextBox1.Text != "")

{

KPIWebDataContext kPiDataContext = new KPIWebDataContext();

ManualTable man = new ManualTable();

man.ManualName = TextBox2.Text;

man.ManualLink = TextBox1.Text;

man.Active = true;

kPiDataContext.ManualTable.InsertOnSubmit(man);

kPiDataContext.SubmitChanges();

string savepath = Server.MapPath("//manuals//");

string fileName = TextBox1.Text;

if (FileUpload1.HasFile)

{

FileUpload1.PostedFile.SaveAs(Server.MapPath("//manuals//" + fileName));

}

else

{

Page.ClientScript.RegisterClientScriptBlock(typeof(Page), "Script", "alert('Документ не прикреплен');" + "document.location = 'Manual.aspx';", true);

}

Page.ClientScript.RegisterClientScriptBlock(typeof(Page), "Script", "alert('Документ сохранен');" + "document.location = 'Manual.aspx';", true);

}

else

{

Page.ClientScript.RegisterClientScriptBlock(typeof(Page), "Script", "alert('Введите точное имя файла');", true);

}

}

protected void DeleteButtonClick(object sender, EventArgs e)

{

Button button = (Button)sender;

{

using (KPIWebDataContext kPiDataContext = new KPIWebDataContext())

{

var check =

(from a in kPiDataContext.ManualTable

where

a.ManualID == Convert.ToInt32(button.CommandArgument)

select a)

.FirstOrDefault();

check.Active = false;

kPiDataContext.SubmitChanges();

Response.Redirect("~/StatisticsDepartment/Manual.aspx");

}

}

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Filename: Manual.aspx.designer.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//------------------------------------------------------------------------------

// <автоматически создаваемое>

// Этот код создан программой.

//

// Изменения в этом файле могут привести к неправильной работе и будут потеряны в случае

// повторной генерации кода.

// </автоматически создаваемое>

//------------------------------------------------------------------------------

namespace KPIWeb.StatisticsDepartment {

public partial class Manual {

/// <summary>

/// GridView1 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.GridView GridView1;

/// <summary>

/// Label1 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Label Label1;

/// <summary>

/// TextBox1 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.TextBox TextBox1;

/// <summary>

/// Label2 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Label Label2;

/// <summary>

/// TextBox2 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.TextBox TextBox2;

/// <summary>

/// FileUpload1 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.FileUpload FileUpload1;

/// <summary>

/// Button1 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Button Button1;

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Filename: MonitoringMain.aspx.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

namespace KPIWeb.StatisticsDepartment

{

public partial class MonitoringMain : System.Web.UI.Page

{

protected void Page\_Load(object sender, EventArgs e)

{

Serialization UserSer = (Serialization)Session["UserID"];

if (UserSer == null)

{

Response.Redirect("~/Default.aspx");

}

int userID = UserSer.Id;

KPIWebDataContext kPiDataContext = new KPIWebDataContext();

UsersTable userTable =

(from a in kPiDataContext.UsersTable where a.UsersTableID == userID select a).FirstOrDefault();

if (userTable.AccessLevel != 9)//только мониторинга отдел

{

Response.Redirect("~/Default.aspx");

}

}

protected void Button5\_Click(object sender, EventArgs e)

{

Response.Redirect("~/StatisticsDepartment/ReportViewer.aspx");

}

protected void Button4\_Click(object sender, EventArgs e)

{

Response.Redirect("~/StatisticsDepartment/BasicParametrs.aspx");

}

protected void Button3\_Click(object sender, EventArgs e)

{

Response.Redirect("~/StatisticsDepartment/Indicators.aspx");

}

protected void Button2\_Click(object sender, EventArgs e)

{

Response.Redirect("~/AutomationDepartment/EditUser.aspx");

}

protected void Button6\_Click(object sender, EventArgs e)

{

Response.Redirect("~/StatisticsDepartment/AddSpecialization.aspx");

}

protected void Button7\_Click(object sender, EventArgs e)

{

Response.Redirect("~/StatisticsDepartment/PlannedIndicator.aspx");

}

protected void Button8\_Click(object sender, EventArgs e)

{

Response.Redirect("~/StatisticsDepartment/Document.aspx");

}

protected void Button1\_Click(object sender, EventArgs e)

{

Response.Redirect("~/StatisticsDepartment/Manual.aspx");

}

protected void Button9\_Click(object sender, EventArgs e)

{

Response.Redirect("~/AutomationDepartment/AddLevel.aspx");

}

protected void Button10\_Click(object sender, EventArgs e)

{

Response.Redirect("~/StatisticsDepartment/Confirm.aspx");

}

protected void Button22\_Click(object sender, EventArgs e)

{

Response.Redirect("~/StatisticsDepartment/FastStructure.aspx");

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Filename: MonitoringMain.aspx.designer.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//------------------------------------------------------------------------------

// <auto-generated>

// This code was generated by a tool.

//

// Changes to this file may cause incorrect behavior and will be lost if

// the code is regenerated.

// </auto-generated>

//------------------------------------------------------------------------------

namespace KPIWeb.StatisticsDepartment {

public partial class MonitoringMain {

/// <summary>

/// Button10 control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.Button Button10;

/// <summary>

/// Button1 control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.Button Button1;

/// <summary>

/// Button2 control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.Button Button2;

/// <summary>

/// Button3 control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.Button Button3;

/// <summary>

/// Button7 control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.Button Button7;

/// <summary>

/// Button4 control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.Button Button4;

/// <summary>

/// Button5 control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.Button Button5;

/// <summary>

/// Button6 control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.Button Button6;

/// <summary>

/// Button8 control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.Button Button8;

/// <summary>

/// Button22 control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.Button Button22;

/// <summary>

/// Button9 control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.Button Button9;

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Filename: PlannedIndicator.aspx.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using System;

using System.Collections.Generic;

using System.Configuration;

using System.Drawing;

using System.Linq;

using System.Web;

using System.Web.UI.WebControls;

//using Microsoft.Office.Interop.Excel;

using Page = System.Web.UI.Page;

namespace KPIWeb

{

public partial class PlannedIndicator : System.Web.UI.Page

{

protected void Page\_Load(object sender, EventArgs e)

{

KPIWebDataContext kPiDataContext = new KPIWebDataContext();

Serialization UserSer = (Serialization)Session["UserID"];

if (UserSer == null)

{

Response.Redirect("~/Default.aspx");

}

int userID = UserSer.Id;

UsersTable userTable =

(from a in kPiDataContext.UsersTable where a.UsersTableID == userID select a).FirstOrDefault();

if ((userTable.AccessLevel != 10) && (userTable.AccessLevel != 9))

{

Response.Redirect("~/Default.aspx");

}

if (!(Page.IsPostBack))

{

List<IndicatorsTable> indicatorList = (from item in kPiDataContext.IndicatorsTable

where item.Active == true

select item).ToList();

var dictionary = new Dictionary<int, string>();

dictionary.Add(0, "Выберите целевой показатель");

foreach (IndicatorsTable item in indicatorList)

dictionary.Add(item.IndicatorsTableID, item.Name);

DropDownList1.DataTextField = "Value";

DropDownList1.DataValueField = "Key";

DropDownList1.DataSource = dictionary;

DropDownList1.DataBind();

}

}

protected void Button1\_Click(object sender, EventArgs e)

{

KPIWebDataContext kPiDataContext = new KPIWebDataContext();

List<PlannedIndicator> plannedList = (from item in kPiDataContext.PlannedIndicator where item.Active == true

select item ).ToList();

GridView1.DataSource = plannedList;

GridView1.DataBind();

}

protected void Button2\_Click(object sender, EventArgs e)

{

KPIWebDataContext kPiDataContext = new KPIWebDataContext();

PlannedIndicator indicators = new PlannedIndicator();

if (CheckBox1.Checked) indicators.Active = true;

else indicators.Active = false;

indicators.FK\_IndicatorsTable = Convert.ToInt32(DropDownList1.Items[DropDownList1.SelectedIndex].Value);

indicators.Value = Convert.ToDouble(IndicatorMeasure.Text);

indicators.Date = Calendar1.SelectedDate;

kPiDataContext.PlannedIndicator.InsertOnSubmit(indicators);

kPiDataContext.SubmitChanges();

Page.ClientScript.RegisterClientScriptBlock(typeof(Page), "Script", "alert('Целевого показателя создан');", true);

}

protected void DropDownList1\_SelectedIndexChanged(object sender, EventArgs e)

{

}

protected void GridView1\_SelectedIndexChanged(object sender, EventArgs e)

{

}

protected void DeleteButtonClick (object sender, EventArgs e)

{

Button button = (Button)sender;

{

using (KPIWebDataContext kPiDataContext = new KPIWebDataContext())

{

PlannedIndicator value = (from a in kPiDataContext.PlannedIndicator

where a.PlanedIndicatorID == Convert.ToInt32(button.CommandArgument)

select a).FirstOrDefault();

value.Active = false;

kPiDataContext.SubmitChanges();

}

Response.Redirect("~/PlannedIndicator.aspx");

}

Page.ClientScript.RegisterClientScriptBlock(typeof(Page), "Script",

"alert('Значение удалено');", true);

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Filename: PlannedIndicator.aspx.designer.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//------------------------------------------------------------------------------

// <автоматически создаваемое>

// Этот код создан программой.

//

// Изменения в этом файле могут привести к неправильной работе и будут потеряны в случае

// повторной генерации кода.

// </автоматически создаваемое>

//------------------------------------------------------------------------------

namespace KPIWeb {

public partial class PlannedIndicator {

/// <summary>

/// Button1 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Button Button1;

/// <summary>

/// GridView1 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.GridView GridView1;

/// <summary>

/// addtitle элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Label addtitle;

/// <summary>

/// Label1 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Label Label1;

/// <summary>

/// DropDownList1 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.DropDownList DropDownList1;

/// <summary>

/// Label3 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Label Label3;

/// <summary>

/// IndicatorMeasure элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.TextBox IndicatorMeasure;

/// <summary>

/// CheckBox1 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.CheckBox CheckBox1;

/// <summary>

/// Calendar1 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Calendar Calendar1;

/// <summary>

/// Button2 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Button Button2;

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Filename: ReportCreate.aspx.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using System;

using System.Collections.Generic;

using System.Configuration;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

namespace KPIWeb.StatisticsDepartment

{

public partial class ReportCreate : System.Web.UI.Page

{

protected void Page\_Load(object sender, EventArgs e)

{

Serialization UserSer = (Serialization)Session["UserID"];

if (UserSer == null)

{

Response.Redirect("~/Account/Login.aspx");

}

int userID = UserSer.Id;

KPIWebDataContext kPiDataContext = new KPIWebDataContext(ConfigurationManager.AppSettings.Get("ConnectionString"));

UsersTable userTable =

(from a in kPiDataContext.UsersTable where a.UsersTableID == userID select a).FirstOrDefault();

if (userTable.AccessLevel != 10)

{

Response.Redirect("~/Account/Login.aspx");

}

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Filename: ReportCreate.aspx.designer.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//------------------------------------------------------------------------------

// <автоматически создаваемое>

// Этот код создан программой.

//

// Изменения в этом файле могут привести к неправильной работе и будут потеряны в случае

// повторной генерации кода.

// </автоматически создаваемое>

//------------------------------------------------------------------------------

namespace KPIWeb.StatisticsDepartment {

public partial class ReportCreate {

/// <summary>

/// form1 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.HtmlControls.HtmlForm form1;

/// <summary>

/// Label1 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Label Label1;

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Filename: ReportFilling.aspx.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using System;

using System.Collections.Generic;

using System.Linq;

using System.Data;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

namespace KPIWeb.StatisticsDepartment

{

public partial class ReportFilling : System.Web.UI.Page

{

protected void Page\_Load(object sender, EventArgs e)

{

KPIWebDataContext kPiDataContext = new KPIWebDataContext();

Serialization UserSer = (Serialization)Session["UserID"];

if (UserSer == null)

{

Response.Redirect("~/Default.aspx");

}

int userID = UserSer.Id;

UsersTable userTable =

(from a in kPiDataContext.UsersTable where a.UsersTableID == userID select a).FirstOrDefault();

if ((userTable.AccessLevel != 10) && (userTable.AccessLevel != 9))

{

Response.Redirect("~/Default.aspx");

}

Serialization paramSerialization = (Serialization)Session["ReportArchiveID"];

if (paramSerialization == null)

{

Response.Redirect("~/Default.aspx");

}

int ReportArchiveID;

ReportArchiveID = Convert.ToInt32(paramSerialization.ReportStr);

DataTable DataTableStatus = new DataTable();

DataTableStatus.Columns.Add(new DataColumn("LV\_1", typeof(string)));

DataTableStatus.Columns.Add(new DataColumn("LV\_2", typeof(string)));

DataTableStatus.Columns.Add(new DataColumn("LV\_3", typeof(string)));

DataTableStatus.Columns.Add(new DataColumn("Status", typeof(string)));

DataTableStatus.Columns.Add(new DataColumn("EmailEdit", typeof(string)));

DataTableStatus.Columns.Add(new DataColumn("EmailConfirm", typeof(string)));

List<UsersTable> Users = (from a in kPiDataContext.UsersTable

join b in kPiDataContext.ReportArchiveAndLevelMappingTable

on a.FK\_FirstLevelSubdivisionTable equals b.FK\_FirstLevelSubmisionTableId

where b.FK\_ReportArchiveTableId == ReportArchiveID

&& b.Active == true

&& a.AccessLevel == 0

&& a.Active == true

select a).ToList();

int ii = 0;

foreach (UsersTable currentUser in Users)

{

BasicParametrsAndUsersMapping UserBasicRight = (from a in kPiDataContext.BasicParametrsAndUsersMapping

where a.FK\_UsersTable == currentUser.UsersTableID

&& a.Active == true

select a).FirstOrDefault();

if (UserBasicRight != null) // к пользователю прикреплены базовые показатели

{

CollectedBasicParametersTable CurrentUserFirstCollected = (from a in kPiDataContext.CollectedBasicParametersTable

where

// a.FK\_UsersTable == currentUser.UsersTableID

// &&

a.Active == true

&& a.FK\_ReportArchiveTable == ReportArchiveID

&& ((a.FK\_FirstLevelSubdivisionTable == currentUser.FK\_FirstLevelSubdivisionTable) || currentUser.FK\_FirstLevelSubdivisionTable == null)

&& ((a.FK\_SecondLevelSubdivisionTable == currentUser.FK\_SecondLevelSubdivisionTable) || currentUser.FK\_SecondLevelSubdivisionTable == null)

&& ((a.FK\_ThirdLevelSubdivisionTable == currentUser.FK\_ThirdLevelSubdivisionTable) || currentUser.FK\_ThirdLevelSubdivisionTable == null)

&& ((a.FK\_FourthLevelSubdivisionTable == currentUser.FK\_FourthLevelSubdivisionTable) || currentUser.FK\_FourthLevelSubdivisionTable == null)

&& ((a.FK\_FifthLevelSubdivisionTable == currentUser.FK\_FifthLevelSubdivisionTable) || currentUser.FK\_FifthLevelSubdivisionTable == null)

select a).FirstOrDefault();

string status = "Нет данных";

int Statusn = 0;

if (CurrentUserFirstCollected == null)

{

status = "Заполнение не начато!";

}

else if (CurrentUserFirstCollected.Status == null)

{

Statusn = 0;

}

else

{

Statusn = (int)CurrentUserFirstCollected.Status;

}

if (Statusn == 4)

{

continue;

status = "Данные утверждены";

}

else if (Statusn == 3)

{

status = "Данные ожидают утверждения";

}

else if (Statusn == 2)

{

status = "Данные в процессе заполнения";

}

else if (Statusn == 1)

{

status = "Данные возвращены на доработку";

}

else if (Statusn == 0)

{

status = "Данные в процессе заполнения";

}

else

{

//error

}

if (UserBasicRight.CanEdit == true)

{

DataRow dataRow = DataTableStatus.NewRow();

dataRow["LV\_1"] = (from a in kPiDataContext.FirstLevelSubdivisionTable

where a.FirstLevelSubdivisionTableID == currentUser.FK\_FirstLevelSubdivisionTable

select a.Name).FirstOrDefault();

dataRow["LV\_2"] = (from a in kPiDataContext.SecondLevelSubdivisionTable

where a.SecondLevelSubdivisionTableID == currentUser.FK\_SecondLevelSubdivisionTable

select a.Name).FirstOrDefault();

dataRow["LV\_3"] = (from a in kPiDataContext.ThirdLevelSubdivisionTable

where a.ThirdLevelSubdivisionTableID == currentUser.FK\_ThirdLevelSubdivisionTable

select a.Name).FirstOrDefault();

dataRow["Status"] = status;

dataRow["EmailEdit"] = currentUser.Email;

BasicParametersTable BasicConnectedToUser = (from a in kPiDataContext.BasicParametersTable

join b in kPiDataContext.BasicParametrsAndUsersMapping

on a.BasicParametersTableID equals b.FK\_ParametrsTable

where b.FK\_UsersTable == currentUser.UsersTableID

&& b.CanEdit == true

&& b.Active == true

&& a.Active == true

select a).FirstOrDefault();

UsersTable ConfirmUserEmail = (from a in kPiDataContext.UsersTable

join b in kPiDataContext.BasicParametrsAndUsersMapping

on a.UsersTableID equals b.FK\_UsersTable

where b.FK\_ParametrsTable == BasicConnectedToUser.BasicParametersTableID

&& b.CanConfirm == true

&& a.Active == true

&& b.Active == true

&& ((a.FK\_FirstLevelSubdivisionTable == currentUser.FK\_FirstLevelSubdivisionTable) || currentUser.FK\_FirstLevelSubdivisionTable == null)

&& ((a.FK\_SecondLevelSubdivisionTable == currentUser.FK\_SecondLevelSubdivisionTable) || currentUser.FK\_SecondLevelSubdivisionTable == null)

&& ((a.FK\_ThirdLevelSubdivisionTable == currentUser.FK\_ThirdLevelSubdivisionTable) || currentUser.FK\_ThirdLevelSubdivisionTable == null)

&& ((a.FK\_FourthLevelSubdivisionTable == currentUser.FK\_FourthLevelSubdivisionTable) || currentUser.FK\_FourthLevelSubdivisionTable == null)

&& ((a.FK\_FifthLevelSubdivisionTable == currentUser.FK\_FifthLevelSubdivisionTable) || currentUser.FK\_FifthLevelSubdivisionTable == null)

select a).FirstOrDefault();

if (ConfirmUserEmail != null)

{

dataRow["EmailConfirm"] = ConfirmUserEmail.Email;

}

else

{

dataRow["EmailConfirm"] = "Ошибка: Отсутствует утверждающий пользователь!";

}

ii++;

DataTableStatus.Rows.Add(dataRow);

}

else if (UserBasicRight.CanConfirm == true)

{

BasicParametersTable BasicConnectedToUser = (from a in kPiDataContext.BasicParametersTable

join b in kPiDataContext.BasicParametrsAndUsersMapping

on a.BasicParametersTableID equals b.FK\_ParametrsTable

where b.FK\_UsersTable == currentUser.UsersTableID

&& b.CanConfirm == true

&& b.Active == true

&& a.Active == true

select a).FirstOrDefault();

UsersTable ConfirmUserEmail = (from a in kPiDataContext.UsersTable

join b in kPiDataContext.BasicParametrsAndUsersMapping

on a.UsersTableID equals b.FK\_UsersTable

where b.FK\_ParametrsTable == BasicConnectedToUser.BasicParametersTableID

&& b.CanEdit == true

&& a.Active == true

&& b.Active == true

&& ((a.FK\_FirstLevelSubdivisionTable == currentUser.FK\_FirstLevelSubdivisionTable) || currentUser.FK\_FirstLevelSubdivisionTable == null)

&& ((a.FK\_SecondLevelSubdivisionTable == currentUser.FK\_SecondLevelSubdivisionTable) || currentUser.FK\_SecondLevelSubdivisionTable == null)

&& ((a.FK\_ThirdLevelSubdivisionTable == currentUser.FK\_ThirdLevelSubdivisionTable) || currentUser.FK\_ThirdLevelSubdivisionTable == null)

&& ((a.FK\_FourthLevelSubdivisionTable == currentUser.FK\_FourthLevelSubdivisionTable) || currentUser.FK\_FourthLevelSubdivisionTable == null)

&& ((a.FK\_FifthLevelSubdivisionTable == currentUser.FK\_FifthLevelSubdivisionTable) || currentUser.FK\_FifthLevelSubdivisionTable == null)

select a).FirstOrDefault();

ii++;

if (ConfirmUserEmail == null)

{

DataRow dataRow = DataTableStatus.NewRow();

dataRow["LV\_1"] = (from a in kPiDataContext.FirstLevelSubdivisionTable

where a.FirstLevelSubdivisionTableID == currentUser.FK\_FirstLevelSubdivisionTable

select a.Name).FirstOrDefault();

dataRow["LV\_2"] = (from a in kPiDataContext.SecondLevelSubdivisionTable

where a.SecondLevelSubdivisionTableID == currentUser.FK\_SecondLevelSubdivisionTable

select a.Name).FirstOrDefault();

dataRow["LV\_3"] = (from a in kPiDataContext.ThirdLevelSubdivisionTable

where a.ThirdLevelSubdivisionTableID == currentUser.FK\_ThirdLevelSubdivisionTable

select a.Name).FirstOrDefault();

dataRow["Status"] = status;

dataRow["EmailConfirm"] = currentUser.Email;

dataRow["EmailEdit"] = "Ошибка: Отсутствует вносяший данные пользователь!";

DataTableStatus.Rows.Add(dataRow);

ii++;

}

}

}

}

GridWhoOws.DataSource = DataTableStatus;

GridWhoOws.DataBind();

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Filename: ReportFilling.aspx.designer.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//------------------------------------------------------------------------------

// <auto-generated>

// This code was generated by a tool.

//

// Changes to this file may cause incorrect behavior and will be lost if

// the code is regenerated.

// </auto-generated>

//------------------------------------------------------------------------------

namespace KPIWeb.StatisticsDepartment {

public partial class ReportFilling {

/// <summary>

/// GridWhoOws control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.GridView GridWhoOws;

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Filename: ReportViewer.aspx.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using System;

using System.Collections.Generic;

using System.Configuration;

using System.Data;

using System.Linq;

using System.Text;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

namespace KPIWeb.StatisticsDepartment

{

public partial class ReportViewer : System.Web.UI.Page

{

UsersTable user;

protected void Page\_Load(object sender, EventArgs e)

{

Serialization UserSer = (Serialization)Session["UserID"];

if (UserSer == null)

{

Response.Redirect("~/Default.aspx");

}

int userID = UserSer.Id;

KPIWebDataContext kPiDataContext = new KPIWebDataContext();

UsersTable userTable =

(from a in kPiDataContext.UsersTable where a.UsersTableID == userID select a).FirstOrDefault();

ViewState["User"] = userTable.Email;

if ((userTable.AccessLevel != 10)&&(userTable.AccessLevel != 9))

{

Response.Redirect("~/Default.aspx");

}

if (!Page.IsPostBack)

{

List<ReportArchiveTable> ReportArchiveTable\_ = (from item in kPiDataContext.ReportArchiveTable

where item.Active == true

select item).ToList();

DataTable dataTable = new DataTable();

dataTable.Columns.Add(new DataColumn("ReportArchiveTableID", typeof(string)));

dataTable.Columns.Add(new DataColumn("Active", typeof(string)));

dataTable.Columns.Add(new DataColumn("Calculeted", typeof(string)));

dataTable.Columns.Add(new DataColumn("Sent", typeof(string)));

dataTable.Columns.Add(new DataColumn("SentDateTime", typeof(string)));

dataTable.Columns.Add(new DataColumn("RecipientConfirmed", typeof(string)));

dataTable.Columns.Add(new DataColumn("Name", typeof(string)));

dataTable.Columns.Add(new DataColumn("StartDateTime", typeof(string)));

dataTable.Columns.Add(new DataColumn("EndDateTime", typeof(string)));

dataTable.Columns.Add(new DataColumn("DateToSend", typeof(string)));

foreach (ReportArchiveTable reportTable in ReportArchiveTable\_)

{

DataRow dataRow = dataTable.NewRow();

dataRow["ReportArchiveTableID"] = reportTable.ReportArchiveTableID.ToString();

dataRow["Active"] = reportTable.Active ? "Да" : "Нет";

dataRow["Calculeted"] = reportTable.Calculeted ? "Да" : "Нет";

dataRow["Sent"] = reportTable.Sent ? "Да" : "Нет";

dataRow["SentDateTime"] = reportTable.SentDateTime.ToString().Split(' ')[0];

dataRow["RecipientConfirmed"] = reportTable.RecipientConfirmed ? "Да" : "Нет";

dataRow["Name"] = reportTable.Name;

dataRow["StartDateTime"] = reportTable.StartDateTime.ToString().Split(' ')[0];

dataRow["EndDateTime"] = reportTable.EndDateTime.ToString().Split(' ')[0];

dataRow["DateToSend"] = reportTable.DateToSend.ToString().Split(' ')[0];

dataTable.Rows.Add(dataRow);

}

//GridviewActiveCampaign.DataSource = ReportArchiveTable;

GridviewActiveCampaign.DataSource = dataTable;

GridviewActiveCampaign.DataBind();

}

int rowIndex = 0;

if (GridviewActiveCampaign.Rows.Count > 0)

{

for (int i = 1; i <= GridviewActiveCampaign.Rows.Count; i++)

{

Label LabelID = (Label)GridviewActiveCampaign.Rows[rowIndex].FindControl("LabelReportArchiveTableID");

Label LabelDate1 = (Label)GridviewActiveCampaign.Rows[rowIndex].FindControl("LabelDate1");

Label LabelDate2 = (Label)GridviewActiveCampaign.Rows[rowIndex].FindControl("LabelDate2");

int id = Convert.ToInt32(LabelID.Text);

EmailSendHistory esh0 = (from a in kPiDataContext.EmailSendHistory where a.FK\_ReportsArchiveTable == id && a.Count == 0 select a).FirstOrDefault(); ;

EmailSendHistory esh1 = (from a in kPiDataContext.EmailSendHistory where a.FK\_ReportsArchiveTable == id && a.Count == 1 select a).FirstOrDefault(); ;

if ( esh0 != null )

{

var date =

(from a in kPiDataContext.EmailSendHistory

where a.FK\_ReportsArchiveTable == id && a.Count == 0

select a.Date).FirstOrDefault();

LabelDate1.Text = date + " " + (from a in kPiDataContext.EmailSendHistory

where a.FK\_ReportsArchiveTable == id && a.Count == 0

select a.Value).FirstOrDefault();

}

else

{

esh0 = new EmailSendHistory();

esh0.Active = true;

esh0.FK\_ReportsArchiveTable = id;

esh0.Value = "0";

esh0.Count = 0;

kPiDataContext.EmailSendHistory.InsertOnSubmit(esh0);

}

if (esh1 != null)

{

var date =

(from a in kPiDataContext.EmailSendHistory

where a.FK\_ReportsArchiveTable == id && a.Count == 1

select a.Date).FirstOrDefault();

LabelDate2.Text = date + " " + (from a in kPiDataContext.EmailSendHistory

where a.FK\_ReportsArchiveTable == id && a.Count == 1

select a.Value).FirstOrDefault(); ;

}

else

{

esh1 = new EmailSendHistory();

esh1.Active = true;

esh1.FK\_ReportsArchiveTable = id;

esh1.Value = "0";

esh1.Count = 1;

kPiDataContext.EmailSendHistory.InsertOnSubmit(esh1);

}

rowIndex++;

}

}

kPiDataContext.SubmitChanges();

}

protected void ButtonEditReport\_Click(object sender, EventArgs e)

{

Button button = (Button)sender;

int reportArchiveTableID = 0;

if (int.TryParse(button.CommandArgument, out reportArchiveTableID) && reportArchiveTableID > 0)

{

Serialization ReportID = new Serialization((int)reportArchiveTableID, null);

Session["ReportArchiveTableID"] = ReportID;

LogHandler.LogWriter.WriteLog(LogCategory.INFO, "0RW2: User " + ViewState["User"] + @" moved to page /Reports\_/EditReport.aspx");

Response.Redirect("~/Reports\_/EditReport.aspx");

}

}

protected void ButtonEditReport\_Click\_2(object sender, EventArgs e)

{

Button button = (Button)sender;

int reportArchiveTableID = 0;

if (int.TryParse(button.CommandArgument, out reportArchiveTableID) && reportArchiveTableID > 0)

{

Serialization ReportID = new Serialization((int)reportArchiveTableID, null);

Session["ReportArchiveTableID"] = ReportID;

Response.Redirect("~/Reports\_/GenerateReport.aspx");

}

}

protected void GenerateReport\_Click(object sender, EventArgs e)

{

}

protected void GridviewActiveCampaign\_SelectedIndexChanged(object sender, EventArgs e)

{

}

protected void Button1\_Click(object sender, EventArgs e)

{

Serialization ReportID = new Serialization(0, null);

Session["ReportArchiveTableID"] = ReportID;

Response.Redirect("~/Reports\_/EditReport.aspx");

}

protected void Button2\_Click(object sender, EventArgs e)

{

Response.Redirect("~/StatisticsDepartment/Indicators.aspx");

}

protected void ButtonViewReportClick(object sender, EventArgs e)

{

Button button = (Button)sender;

{

Serialization paramSerialization = new Serialization(button.CommandArgument.ToString());

Session["ReportArchiveID"] = paramSerialization;

Serialization modeSer = new Serialization(1, null, null);

Session["mode"] = modeSer;

Serialization level = new Serialization(0, 0, 0, 0, 0, 0);

Session["level"] = level;

Response.Redirect("~/StatisticsDepartment/FastStructure.aspx");

}

}

protected void ButtonViewStruct(object sender, EventArgs e)

{

Button button = (Button)sender;

{

Serialization paramSerialization = new Serialization(button.CommandArgument.ToString());

Session["ReportArchiveID"] = paramSerialization;

Response.Redirect("~/StatisticsDepartment/ReportFilling.aspx");

}

}

protected void ButtonMailSending\_Click(object sender, EventArgs e)

{

if (!CheckBox1.Checked)

{

LogHandler.LogWriter.WriteLog(LogCategory.INFO, "0RW1: MassMailing was started by: " + ViewState["User"]);

int errors = 0;

Button button = (Button) sender;

{

KPIWebDataContext kPiDataContext = new KPIWebDataContext();

var emailListTo = (from a in kPiDataContext.ReportArchiveAndLevelMappingTable

where a.FK\_ReportArchiveTableId == Convert.ToInt32(button.CommandArgument) && a.Active

join b in kPiDataContext.UsersTable

on a.FK\_FirstLevelSubmisionTableId equals b.FK\_FirstLevelSubdivisionTable

where b.Active

select b.Email).ToList();

EmailTemplate EmailParams = (from a in kPiDataContext.EmailTemplate

where a.Name == "NewCampaign"

&& a.Active == true

select a).FirstOrDefault();

string RepStartDate = ((from a in kPiDataContext.ReportArchiveTable

where a.ReportArchiveTableID == Convert.ToInt32(button.CommandArgument)

select a.StartDateTime).FirstOrDefault()).ToString().Split()[0];

string RepEndDate = (from a in kPiDataContext.ReportArchiveTable

where a.ReportArchiveTableID == Convert.ToInt32(button.CommandArgument)

select a.EndDateTime).FirstOrDefault().ToString().Split()[0];

string RepName = (from a in kPiDataContext.ReportArchiveTable

where a.ReportArchiveTableID == Convert.ToInt32(button.CommandArgument)

select a.Name).FirstOrDefault();

foreach (var email in emailListTo)

{

if (EmailParams != null)

Action.MassMailing(email, EmailParams.EmailTitle,

EmailParams.EmailContent.Replace("#SiteName#",

ConfigurationManager.AppSettings.Get("SiteName"))

.Replace("#StartDate#", RepStartDate)

.Replace("#EndDate#", RepEndDate)

.Replace("#ReportName#", RepName)

, null);

/\*

errors = Action.MassMailing(email,"Новая отчетная кампания \"" +

(from a in kPiDataContext.ReportArchiveTable where a.ReportArchiveTableID == Convert.ToInt32(button.CommandArgument) select a.Name).FirstOrDefault() + "\"",

"Здравствуйте, " + email.ToString().Substring(0, email.ToString().LastIndexOf('@')) + ". Информируем Вас о начале новой отчетной кампании \"" +

(from a in kPiDataContext.ReportArchiveTable where a.ReportArchiveTableID == Convert.ToInt32(button.CommandArgument) select a.Name).FirstOrDefault()

+ "\", которая пройдет в период с " +

((from a in kPiDataContext.ReportArchiveTable where a.ReportArchiveTableID == Convert.ToInt32(button.CommandArgument) select a.StartDateTime).FirstOrDefault()).ToString().Split()[0] +

" по " + (from a in kPiDataContext.ReportArchiveTable where a.ReportArchiveTableID == Convert.ToInt32(button.CommandArgument) select a.EndDateTime).FirstOrDefault().ToString().Split()[0] +

". Для авторизации в системе перейдите по ссылке: " + ConfigurationManager.AppSettings.Get("SiteName")

, null);

\*/

}

EmailSendHistory esh0 = (from a in kPiDataContext.EmailSendHistory

where a.FK\_ReportsArchiveTable == Convert.ToInt32(button.CommandArgument) && a.Count == 0

select a).FirstOrDefault();

;

//EmailSendHistory esh1 = (from a in kPiDataContext.EmailSendHistories where a.FK\_ReportsArchiveTable == Convert.ToInt32(button.CommandArgument) && a.Count == 1 select a).FirstOrDefault(); ;

esh0.Date = DateTime.Now;

esh0.Value = " [ " + (emailListTo.Count - errors).ToString() + "/" + emailListTo.Count.ToString() +

" ]";

kPiDataContext.SubmitChanges();

var date = (from a in kPiDataContext.EmailSendHistory

where a.FK\_ReportsArchiveTable == Convert.ToInt32(button.CommandArgument) && a.Count == 0

select a.Date).FirstOrDefault();

int rowIndex = 0;

if (GridviewActiveCampaign.Rows.Count > 0)

{

for (int i = 1; i <= GridviewActiveCampaign.Rows.Count; i++)

{

Label LabelDate1 = (Label) GridviewActiveCampaign.Rows[rowIndex].FindControl("LabelDate1");

LabelDate1.Text = date + " " + " [ " + (emailListTo.Count - errors).ToString() + "/" +

emailListTo.Count.ToString() + " ]";

rowIndex++;

}

}

}

Page\_Load(null, null);

}

else

{

DisplayAlert("Снимите предохранитель!");

}

}

private void DisplayAlert(string message)

{

ClientScript.RegisterStartupScript(

this.GetType(),

Guid.NewGuid().ToString(),

string.Format("alert('{0}');window.location.href = 'ReportViewer.aspx'",

message.Replace("'", @"\'").Replace("\n", "\\n").Replace("\r", "\\r")),

true);

}

protected void ButtonMailSending2\_Click(object sender, EventArgs e)

{

if (!CheckBox1.Checked)

{

LogHandler.LogWriter.WriteLog(LogCategory.INFO, "0RW0: MassMailing was started by: " + ViewState["User"]);

Button button = (Button) sender;

{

KPIWebDataContext kPiDataContext = new KPIWebDataContext();

int errors = 0;

var emailListToDebt = (from a in kPiDataContext.ReportArchiveAndLevelMappingTable

where a.FK\_ReportArchiveTableId == Convert.ToInt32(button.CommandArgument) && a.Active

join b in kPiDataContext.UsersTable on a.FK\_FirstLevelSubmisionTableId equals

b.FK\_FirstLevelSubdivisionTable

where b.Active

join c in kPiDataContext.CollectedBasicParametersTable on b.UsersTableID equals c.FK\_UsersTable

where (c.Status == 0 || c.Status == null) && c.Active

select b.Email).ToList();

var uniqueMails = emailListToDebt.Distinct().ToList();

EmailTemplate EmailParams = (from a in kPiDataContext.EmailTemplate

where a.Name == "CampaignReminder"

&& a.Active == true

select a).FirstOrDefault();

string RepStartDate = ((from a in kPiDataContext.ReportArchiveTable

where a.ReportArchiveTableID == Convert.ToInt32(button.CommandArgument)

select a.StartDateTime).FirstOrDefault()).ToString().Split()[0];

string RepEndDate = (from a in kPiDataContext.ReportArchiveTable

where a.ReportArchiveTableID == Convert.ToInt32(button.CommandArgument)

select a.EndDateTime).FirstOrDefault().ToString().Split()[0];

string RepName = (from a in kPiDataContext.ReportArchiveTable

where a.ReportArchiveTableID == Convert.ToInt32(button.CommandArgument)

select a.Name).FirstOrDefault();

foreach (var email in uniqueMails)

{

if (EmailParams != null)

Action.MassMailing(email, EmailParams.EmailTitle,

EmailParams.EmailContent.Replace("#SiteName#",

ConfigurationManager.AppSettings.Get("SiteName"))

.Replace("#StartDate#", RepStartDate)

.Replace("#EndDate#", RepEndDate)

.Replace("#ReportName#", RepName)

, null);

/\*

errors = Action.MassMailing(email, "Заполните данные в отчетной кампании \"" +

(from a in kPiDataContext.ReportArchiveTable where a.ReportArchiveTableID == Convert.ToInt32(button.CommandArgument) select a.Name).FirstOrDefault() + "\"",

"Здравствуйте, " + email.ToString().Substring(0, email.ToString().LastIndexOf('@')) + ". Напоминаем вам о том, что вы являетесь участником отчетной кампании \"" +

(from a in kPiDataContext.ReportArchiveTable where a.ReportArchiveTableID == Convert.ToInt32(button.CommandArgument) select a.Name).FirstOrDefault()

+ "\", которая проходит в период с " +

((from a in kPiDataContext.ReportArchiveTable where a.ReportArchiveTableID == Convert.ToInt32(button.CommandArgument) select a.StartDateTime).FirstOrDefault()).ToString().Split()[0] +

" по " + (from a in kPiDataContext.ReportArchiveTable where a.ReportArchiveTableID == Convert.ToInt32(button.CommandArgument) select a.EndDateTime).FirstOrDefault().ToString().Split()[0] +

". На данный момент вы не отправили на утверждение прикрепленные за вами данные. Для авторизации в системе перейдите по ссылке: " + ConfigurationManager.AppSettings.Get("SiteName")

, null);

\*/

}

//EmailSendHistory esh0 = (from a in kPiDataContext.EmailSendHistories where a.FK\_ReportsArchiveTable == Convert.ToInt32(button.CommandArgument) && a.Count == 0 select a).FirstOrDefault(); ;

EmailSendHistory esh1 = (from a in kPiDataContext.EmailSendHistory

where a.FK\_ReportsArchiveTable == Convert.ToInt32(button.CommandArgument) && a.Count == 1

select a).FirstOrDefault();

;

esh1.Date = DateTime.Now;

esh1.Value = " [ " + (uniqueMails.Count - errors).ToString() + "/" + uniqueMails.Count.ToString() +

" ]";

kPiDataContext.SubmitChanges();

var date = (from a in kPiDataContext.EmailSendHistory

where a.FK\_ReportsArchiveTable == Convert.ToInt32(button.CommandArgument) && a.Count == 1

select a.Date).FirstOrDefault();

int rowIndex = 0;

if (GridviewActiveCampaign.Rows.Count > 0)

{

for (int i = 1; i <= GridviewActiveCampaign.Rows.Count; i++)

{

Label LabelDate2 = (Label) GridviewActiveCampaign.Rows[rowIndex].FindControl("LabelDate2");

LabelDate2.Text = date + " " + " [ " + (uniqueMails.Count - errors).ToString() + "/" +

uniqueMails.Count.ToString() + " ]";

rowIndex++;

}

}

}

Page\_Load(null, null);

}

else

{

DisplayAlert("Снимите предохранитель!");

}

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Filename: ReportViewer.aspx.designer.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//------------------------------------------------------------------------------

// <auto-generated>

// This code was generated by a tool.

//

// Changes to this file may cause incorrect behavior and will be lost if

// the code is regenerated.

// </auto-generated>

//------------------------------------------------------------------------------

namespace KPIWeb.StatisticsDepartment {

public partial class ReportViewer {

/// <summary>

/// Button1 control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.Button Button1;

/// <summary>

/// CheckBox1 control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.CheckBox CheckBox1;

/// <summary>

/// GridviewActiveCampaign control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.GridView GridviewActiveCampaign;

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Filename: StastisticsHomePage.aspx.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using System;

using System.Collections.Generic;

using System.Configuration;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

namespace KPIWeb.StatisticsDepartment

{

public partial class StastisticsHomePage : System.Web.UI.Page

{

protected void Page\_Load(object sender, EventArgs e)

{

Serialization UserSer = (Serialization)Session["UserID"];

if (UserSer == null)

{

Response.Redirect("~/Account/Login.aspx");

}

int userID = UserSer.Id;

KPIWebDataContext kPiDataContext = new KPIWebDataContext();

UsersTable userTable =

(from a in kPiDataContext.UsersTable where a.UsersTableID == userID select a).FirstOrDefault();

if (userTable.AccessLevel != 10)

{

Response.Redirect("~/Account/Login.aspx");

}

}

protected void Button1\_Click(object sender, EventArgs e)

{

Response.Redirect("~/StatisticsDepartment/ReportViewer.aspx");

}

protected void Button2\_Click(object sender, EventArgs e)

{

Serialization ReportID = new Serialization(0, null);

Session["ReportArchiveTableID"] = ReportID;

Response.Redirect("~/Reports\_/EditReport.aspx");

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Filename: StastisticsHomePage.aspx.designer.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//------------------------------------------------------------------------------

// <автоматически создаваемое>

// Этот код создан программой.

//

// Изменения в этом файле могут привести к неправильной работе и будут потеряны в случае

// повторной генерации кода.

// </автоматически создаваемое>

//------------------------------------------------------------------------------

namespace KPIWeb.StatisticsDepartment {

public partial class StastisticsHomePage {

/// <summary>

/// Button1 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Button Button1;

/// <summary>

/// Button2 элемент управления.

/// </summary>

/// <remarks>

/// Автоматически создаваемое поле.

/// Для изменения переместите объявление поля из файла конструктора в файл кода программной части.

/// </remarks>

protected global::System.Web.UI.WebControls.Button Button2;

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Filename: Role.cs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

namespace KPIWeb

{

public enum Role

{

Robot = 0,

Superadmin = 1,

Automation = 2,

Statistics = 3

}

}