using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Web;

using System.Web.Mvc;

using UIShell.AdminiseShellPlugin.ViewModels;

namespace UIShell.AdminiseShellPlugin.Controllers

{

public class AccountController : Controller

{

public ActionResult Login()

{

ViewBag.Title = "登录";

return View();

}

[HttpPost]

public ActionResult Login(Account account)

{

var node = BundleActivator.PageFlowServiceTracker.DefaultOrFirstService.GetPageNode("LayoutPage");

string bundleLocation = BundleActivator.Bundle.Location.

Replace(Path.DirectorySeparatorChar, Path.AltDirectorySeparatorChar);

string baseDirectory = AppDomain.CurrentDomain.BaseDirectory.

Replace(Path.DirectorySeparatorChar, Path.AltDirectorySeparatorChar);

string bundleRelativePath = string.Empty;

if (bundleLocation.StartsWith(baseDirectory))

{

bundleRelativePath = bundleLocation.Remove(0, baseDirectory.Length);

}

string pattern = "~/{0}/{1}";

if (node != null)

{

account.SelectedNavigationType = "0";

if (account.SelectedNavigationType == "0")

{

node.Value = string.Format(pattern, bundleRelativePath, "Views/Shared/\_Layout.cshtml");

}

else if (account.SelectedNavigationType == "1")

{

node.Value = string.Format(pattern, bundleRelativePath, "Views/Shared/\_Layout\_Right.cshtml");

}

else

{

node.Value = string.Format(pattern, bundleRelativePath, "Views/Shared/\_Layout\_Top.cshtml");

}

}

var nodes = BundleActivator.NavigationModel.NavigationNodes;

if(nodes.Count > 0)

{

var firstNode = nodes.First();

if(firstNode.HasSubMenu())

{

firstNode = firstNode.Children.First();

}

return Redirect(firstNode.Value);

}

else

{

ViewBag.Title = "登录";

return View();

}

}

public ActionResult LogOut()

{

if (BundleActivator.PermissionServiceTracker.IsServiceAvailable)

{

BundleActivator.PermissionServiceTracker.DefaultOrFirstService.Logout();

}

return RedirectToAction("Login");

}

}

}

using BasicInfomationPlugin.DataAccessors;

using BasicInfomationPlugin.Models;

using BasicInfomationPlugin.ViewModels;

using LinqToExcel;

using LinqToExcel.Query;

using Newtonsoft.Json;

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Web;

using System.Web.Mvc;

namespace BasicInfomationPlugin.Controllers

{

public class HeatStationController : JsonBaseController

{

public static HeatStationDataAccessor heatingStationDB = new HeatStationDataAccessor();

public ActionResult Query(QueryCondition queryCondition,int? currentPageNum, int? pageSize)

{

var heatStationList = heatingStationDB.GetHeatingStationPage(queryCondition, currentPageNum, pageSize);

var viewModel = new HeatStationViewModel

{

Pagination = heatStationList,

QueryCondition = queryCondition

};

return View(viewModel);

}

public ActionResult AddItem()

{

DropDownListDataAccessor dropDownDataAccessor = new DropDownListDataAccessor();

var subCompanyDropDownList = dropDownDataAccessor.getSubCompanyDropDownList(0);

HeatStationModel item = new HeatStationModel();

if (TempData["model"] != null)

{

item = TempData["model"] as HeatStationModel;

}

return View(item);

}

public ActionResult EditItem(decimal id)

{

HeatStationModel item = heatingStationDB.GetItem(id);

return View(item);

}

public ActionResult DelItem(decimal id)

{

heatingStationDB.DelItem(id);

return RedirectToAction("Query");

}

public ActionResult SaveItem(HeatStationModel item)

{

if (!ModelState.IsValid)

{

if (item.ID == 0)

{

item.TipMsg = "保存失败";

return View("AddItem", item);

}

else

{

return View("EditItem", item);

}

}

else

{

if (item.ID == 0)

{

heatingStationDB.SaveItem(item);

setViewBag();

item.ID = 0;

item.Name = string.Empty;

item.Description = string.Empty;

item.TipMsg = "保存成功";

TempData["model"] = item;

return RedirectToAction("AddItem");

}

else

{

heatingStationDB.SaveItem(item);

return RedirectToAction("Query");

}

}

}

public JsonResult CheckHeatingStationName(string name,decimal id,decimal subCompanyID)

{

Dictionary<string,object> limitCondition = new Dictionary<string,object>();

limitCondition.Add("Name",name);

limitCondition.Add("ID",id);

limitCondition.Add("SubCompanyID",subCompanyID);

bool exists = false;

exists = heatingStationDB.IsExists(limitCondition);

return Json(!exists, JsonRequestBehavior.AllowGet);

}

public JsonResult CheckIsUsed(decimal id)

{

bool isUsed = false;

isUsed = heatingStationDB.IsUsed(id);

return new JsonResult { Data = isUsed, JsonRequestBehavior = JsonRequestBehavior.AllowGet };

}

public void Export()

{

var excel = heatingStationDB.ExportData();

DataImpExpExcel dataToExcel = new DataImpExpExcel();

HttpContext context = System.Web.HttpContext.Current;

new DataImpExpExcel().Export(System.Web.HttpContext.Current, excel, "../../");

}

public void Import(string uploadType,HttpPostedFileBase fileName)

{

HttpPostedFileBase file = fileName;

string type = uploadType;

string json = string.Empty;

string JsonError = string.Empty;

Response.ContentType = "text/html";

if (file != null)

{

string targetFilePath= DataImpExpExcel.UploadFileAndGetTargetPath(System.Web.HttpContext.Current, file);

try

{

JsonError += heatingStationDB.ImportData(targetFilePath, type);//导入

}

catch (Exception ex)

{

JsonError += "\r\n" + "导入失败：" + ex.Message;

}

}

else

{

JsonError += "\r\n" + "导入失败：文件为空";

}

if (!string.IsNullOrEmpty(JsonError))

{

json = JsonConvert.SerializeObject(new { success = false, msg = JsonError });

}

else

{

json = JsonConvert.SerializeObject(new { success = true, msg = "导入成功" });

}

Response.Write(json);

Response.End();

}

}

}

using BasicInfomationPlugin.DataAccessors;

using BasicInfomationPlugin.Models;

using BasicInfomationPlugin.ViewModels;

using Newtonsoft.Json;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.Mvc;

namespace BasicInfomationPlugin.Controllers

{

public class CommunityController : JsonBaseController

{

private static CommunityDataAccessor dbAccessor = new CommunityDataAccessor();

private static DropDownListDataAccessor dropDownDataAccessor = new DropDownListDataAccessor();

public ActionResult Query(CommunityViewModel model,int? currentPageNum, int? pageSize)

{

var heatSID = Convert.ToDecimal(model.heatSID);

var communityList = dbAccessor.GetCommunityPagedData(heatSID, currentPageNum, pageSize);

setViewBag();

model.Pagination = communityList;

return View(model);

}

public ActionResult AddItem(CommunityViewModel model)

{

setViewBag();

CommunityModel item = new CommunityModel();

if (TempData["model"] != null)

{

item = TempData["model"] as CommunityModel;

}

else

{

item.HeatingStationId = Convert.ToDecimal(model.heatSID);

}

return View(item);

}

public ActionResult EditItem(decimal id)

{

setViewBag();

CommunityModel item = dbAccessor.GetItem(id);

return View(item);

}

public ActionResult DelItem(decimal id)

{

dbAccessor.DelItem(id);

return RedirectToAction("Query");

}

public ActionResult AddToSaveItem(CommunityModel item)

{

if (!ModelState.IsValid)

{

setViewBag();

item.TipMsg = "保存失败";

return View("AddItem", item);

}

else

{

dbAccessor.AddToSaveItem(item);

setViewBag();

item.id = 0;

item.name = string.Empty;

item.description = string.Empty;

item.TipMsg = "保存成功";

TempData["model"] = item;

return RedirectToAction("AddItem");

}

}

public ActionResult EditToSaveItem(CommunityModel item)

{

if (!ModelState.IsValid)

{

setViewBag();

return View("EditItem", item);

}

else

{

dbAccessor.EditToSaveItem(item);

return RedirectToAction("Query");

}

}

public void Export(CommunityViewModel model)

{

var excel = dbAccessor.ExportData(model.heatSID);

DataImpExpExcel dataToExcel = new DataImpExpExcel();

HttpContext context = System.Web.HttpContext.Current;

new DataImpExpExcel().Export(System.Web.HttpContext.Current, excel, "../../");

}

public void Import(string uploadType,HttpPostedFileBase fileName)

{

HttpPostedFileBase file = fileName;

string type = uploadType;

string json = string.Empty;

string JsonError = string.Empty;

Response.ContentType = "text/html";

if (file != null)

{

string targetFilePath = DataImpExpExcel.UploadFileAndGetTargetPath(System.Web.HttpContext.Current, file);

try

{

JsonError += dbAccessor.ImportData(targetFilePath, type);//导入

}

catch (Exception ex)

{

JsonError += "\r\n" + "导入失败：" + ex.Message;

}

}

else

{

JsonError += "\r\n" + "导入失败：文件为空";

}

if (!string.IsNullOrEmpty(JsonError))

{

json = JsonConvert.SerializeObject(new { success = false, msg = JsonError });

}

else

{

json = JsonConvert.SerializeObject(new { success = true, msg = "导入成功" });

}

Response.Write(json);

Response.End();

}

public JsonResult CheckCommunityName(string name,decimal id,decimal heatingStationId)

{

Dictionary<string, object> limitCondition = new Dictionary<string, object>();

limitCondition.Add("Name", name);

limitCondition.Add("ID", id);

limitCondition.Add("HeatingStationID", heatingStationId);

bool exists = false;

exists = dbAccessor.IsExists(limitCondition);

return Json(!exists, JsonRequestBehavior.AllowGet);

}

public JsonResult CheckIsUsed(decimal id)

{

bool isUsed = false;

isUsed = dbAccessor.IsUsed(id);

return new JsonResult { Data = isUsed, JsonRequestBehavior = JsonRequestBehavior.AllowGet };

}

}

}

using BasicInfomationPlugin.DataAccessors;

using BasicInfomationPlugin.Models;

using BasicInfomationPlugin.ViewModels;

using Newtonsoft.Json;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.Mvc;

namespace BasicInfomationPlugin.Controllers

{

public class UnitController : JsonBaseController

{

private static UnitDataAccessor dbAccessor = new UnitDataAccessor();

private static DropDownListDataAccessor dropDownDataAccessor = new DropDownListDataAccessor();

public ActionResult Query(UnitViewModel model,int? currentPageNum, int? pageSize)

{

var hid = Convert.ToDecimal(model.heatSID);

var cid = Convert.ToDecimal(model.comID);

var bid = Convert.ToDecimal(model.buildID);

ViewBag.HeatingStationDropDownList = dropDownDataAccessor.getHeatStationDropDownList();

var UnitList = dbAccessor.GetUnitDataPage(hid, cid, bid, currentPageNum, pageSize);

model.Pagination = UnitList;

model.CommunityList = new List<SelectListItem>();

model.BuildingList = new List<SelectListItem>(); ;

if (hid != 0)

{

model.CommunityList = dropDownDataAccessor.getCommunityDropDownList(hid);

model.BuildingList = dropDownDataAccessor.getBuildingDropDownList(cid);

}

return View(model);

}

public ActionResult AddItem(UnitViewModel model)

{

ViewBag.subCompanyDDList = dropDownDataAccessor.getSubCompanyDropDownList(0);

UnitModel item = new UnitModel();

if (TempData["model"] != null)

{

item = TempData["model"] as UnitModel;

}

else

{

item.HeatingStationId = Convert.ToDecimal(model.heatSID);

item.communityId = Convert.ToDecimal(model.comID);

item.buildingId = Convert.ToDecimal(model.buildID);

}

return View(item);

}

public ActionResult EditItem(decimal id)

{

ViewBag.subCompanyDDList = dropDownDataAccessor.getSubCompanyDropDownList(0);

UnitModel item = dbAccessor.GetItem(id);

return View(item);

}

public ActionResult DelItem(decimal id)

{

dbAccessor.DelItem(id);

return RedirectToAction("Query");

}

public ActionResult AddToSaveItem(UnitModel item)

{

if (!ModelState.IsValid)

{

ViewBag.subCompanyDDList = dropDownDataAccessor.getSubCompanyDropDownList(0);

item.TipMsg = "保存失败";

return View("AddItem", item);

}

else

{

dbAccessor.AddToSaveItem(item);

setViewBag();

item.id = 0;

item.name = string.Empty;

item.description = string.Empty;

item.TipMsg = "保存成功";

TempData["model"] = item;

return RedirectToAction("AddItem");

}

}

public ActionResult EditToSaveItem(UnitModel item)

{

if (!ModelState.IsValid)

{

ViewBag.subCompanyDDList = dropDownDataAccessor.getSubCompanyDropDownList(0);

return View("EditItem", item);

}

else

{

dbAccessor.EditToSaveItem(item);

return RedirectToAction("Query");

}

}

public void Export(UnitViewModel model)

{

var excel = dbAccessor.ExportData(model.heatSID,model.comID,model.buildID);

DataImpExpExcel dataToExcel = new DataImpExpExcel();

HttpContext context = System.Web.HttpContext.Current;

new DataImpExpExcel().Export(System.Web.HttpContext.Current, excel, "../../");

}

public void Import(string uploadType,HttpPostedFileBase fileName)

{

HttpPostedFileBase file = fileName;

string type = uploadType;

string json = string.Empty;

string JsonError = string.Empty;

Response.ContentType = "text/html";

if (file != null)

{

string targetFilePath = DataImpExpExcel.UploadFileAndGetTargetPath(System.Web.HttpContext.Current, file);

try

{

JsonError += dbAccessor.ImportData(targetFilePath, type);//导入

}

catch (Exception ex)

{

JsonError += "\r\n" + "导入失败：" + ex.Message;

}

}

else

{

JsonError += "\r\n" + "导入失败：文件为空";

}

if (!string.IsNullOrEmpty(JsonError))

{

json = JsonConvert.SerializeObject(new { success = false, msg = JsonError });

}

else

{

json = JsonConvert.SerializeObject(new { success = true, msg = "导入成功" });

}

Response.Write(json);

Response.End();

}

public JsonResult CheckUnitName(string name,decimal id,decimal buildingID)

{

Dictionary<string, object> limitCondition = new Dictionary<string, object>();

limitCondition.Add("Name", name);

limitCondition.Add("ID", id);

limitCondition.Add("BuildingID", buildingID);

bool exists = false;

exists = dbAccessor.IsExists(limitCondition);

return Json(!exists, JsonRequestBehavior.AllowGet);

}

public JsonResult CheckIsUsed(decimal id)

{

bool isUsed = false;

isUsed = dbAccessor.IsUsed(id);

return new JsonResult { Data = isUsed, JsonRequestBehavior = JsonRequestBehavior.AllowGet };

}

}

}

using BasicInfomationPlugin.DataAccessors;

using BasicInfomationPlugin.Models;

using BasicInfomationPlugin.ViewModels;

using Newtonsoft.Json;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.Mvc;

namespace BasicInfomationPlugin.Controllers

{

public class BuildingController : JsonBaseController

{

private static BuildingDataAccessor dbAccessor = new BuildingDataAccessor();

private static DropDownListDataAccessor dropDownDataAccessor = new DropDownListDataAccessor();

public ActionResult Query(BuildingViewModel model, int? currentPageNum, int? pageSize)

{

var hid = Convert.ToDecimal(model.heatSID);

var cid = Convert.ToDecimal(model.comID);

var BuildingList = dbAccessor.GetBuildingDataPage(hid, cid, currentPageNum, pageSize);

setViewBag();

model.CommunityList = new List<SelectListItem>();

if (hid != 0)

{

model.CommunityList = dropDownDataAccessor.getCommunityDropDownList(hid);

}

model.Pagination = BuildingList;

return View(model);

}

public ActionResult AddItem(BuildingViewModel model)

{

ViewBag.subCompanyDDList = dropDownDataAccessor.getSubCompanyDropDownList(0);

BuildingModel item = new BuildingModel();

if (TempData["model"] != null)

{

item = TempData["model"] as BuildingModel;

}

else

{

item.HeatingStationId =Convert.ToDecimal(model.heatSID);

item.communityId = Convert.ToDecimal(model.comID);

}

return View(item);

}

public ActionResult EditItem(decimal id)

{

ViewBag.subCompanyDDList = dropDownDataAccessor.getSubCompanyDropDownList(0);

BuildingModel item = dbAccessor.GetItem(id);

return View(item);

}

public ActionResult DelItem(decimal id)

{

dbAccessor.DelItem(id);

return RedirectToAction("Query");

}

public ActionResult AddToSaveItem(BuildingModel item)

{

if (!ModelState.IsValid)

{

ViewBag.subCompanyDDList = dropDownDataAccessor.getSubCompanyDropDownList(0);

item.TipMsg = "保存失败";

return View("AddItem", item);

}

else

{

dbAccessor.AddToSaveItem(item);

setViewBag();

item.id = 0;

item.name = string.Empty;

item.description = string.Empty;

item.TipMsg = "保存成功";

TempData["model"] = item;

return RedirectToAction("AddItem");

}

}

public ActionResult EditToSaveItem(BuildingModel item)

{

if (!ModelState.IsValid)

{

ViewBag.subCompanyDDList = dropDownDataAccessor.getSubCompanyDropDownList(0);

return View("EditItem", item);

}

else

{

dbAccessor.EditToSaveItem(item);

return RedirectToAction("Query");

}

}

public void Export(BuildingViewModel model)

{

var excel = dbAccessor.ExportData(model);

DataImpExpExcel dataToExcel = new DataImpExpExcel();

HttpContext context = System.Web.HttpContext.Current;

new DataImpExpExcel().Export(System.Web.HttpContext.Current, excel, "../../");

}

public void Import(string uploadType,HttpPostedFileBase fileName)

{

HttpPostedFileBase file = fileName;

string type = uploadType;

string json = string.Empty;

string JsonError = string.Empty;

Response.ContentType = "text/html";

if (file != null)

{

string targetFilePath = DataImpExpExcel.UploadFileAndGetTargetPath(System.Web.HttpContext.Current, file);

try

{

JsonError += dbAccessor.ImportData(targetFilePath, type);//导入

}

catch (Exception ex)

{

JsonError += "\r\n" + "导入失败：" + ex.Message;

}

}

else

{

JsonError += "\r\n" + "导入失败：文件为空";

}

if (!string.IsNullOrEmpty(JsonError))

{

json = JsonConvert.SerializeObject(new { success = false, msg = JsonError });

}

else

{

json = JsonConvert.SerializeObject(new { success = true, msg = "导入成功" });

}

Response.Write(json);

Response.End();

}

public JsonResult CheckBuildingName(string name,decimal id,decimal? communityID)

{

Dictionary<string, object> limitCondition = new Dictionary<string, object>();

limitCondition.Add("Name", name);

limitCondition.Add("ID", id);

limitCondition.Add("CommunityID", communityID??0);

bool exists = false;

exists = dbAccessor.IsExists(limitCondition);

return Json(!exists, JsonRequestBehavior.AllowGet);

}

public JsonResult CheckIsUsed(decimal id)

{

bool isUsed = false;

isUsed = dbAccessor.IsUsed(id);

return new JsonResult { Data = isUsed, JsonRequestBehavior = JsonRequestBehavior.AllowGet };

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.Mvc;

namespace BasicInfomationPlugin.Controllers

{

using BasicInfomationPlugin.DataAccessors;

using BasicInfomationPlugin.Models;

using BasicInfomationPlugin.ViewModels;

public class ServerController : MasterController

{

private ServerDataAccessor \_serverDb;

public ServerController()

{

\_serverDb = new ServerDataAccessor();

}

#region Index() 默认加载方法...

public ActionResult Index(int? currentPageNum, int? pageSize)

{

var dataPagination = \_serverDb.GetDataPagination(currentPageNum, pageSize);

var model = new ServerListViewModel

{

Pagination = dataPagination,

DeleteFailed = TempData["DeleteFailed"] == null ? false : true

};

return View(model);

}

#endregion

#region 增删改方法 Add() Edit() Delete() GetModel() SaveItem()

public ActionResult Add()

{

return View("Edit", GetModel(EditPara.Add,0));

}

public ActionResult Edit(decimal id)

{

return View("Edit", GetModel(EditPara.Modi,id));

}

private ServerModel GetModel(EditPara editPara,decimal id = 0)

{

var model = new ServerModel();

if (editPara == EditPara.Modi)

{

model = \_serverDb.GetById(id);

}

else

{

if (TempData["Model"] != null)

{

var tempModel = TempData["Model"] as ServerModel;

model.TipMsg = "保存成功！";

}

}

model.EditParaType = editPara;

return model;

}

public ActionResult SaveItem(ServerModel model)

{

if (!ModelState.IsValid)

{

return View("Edit", GetModel(model.EditParaType,model.ID));

}

else

{

\_serverDb.Update(model);

if (model.EditParaType == EditPara.Modi)

return RedirectToAction("Index");

else

{

TempData["Model"] = model;

return RedirectToAction("Add");

}

}

}

public ActionResult Delete(decimal id)

{

int modelcount = \_serverDb.Delete(id);

if (modelcount == -10)

{

TempData["DeleteFailed"] = "true";

}

return base.ReturnOperationUrl();

}

#endregion

public JsonResult CheckServerNameExists(string Name,int ID)

{

bool exists = false;

exists = \_serverDb.IsExists(Name, ID);

return Json(!exists, JsonRequestBehavior.AllowGet);

}

#region 已注释 /\*校验服务器IP地址是否存在\*/

}

#endregion

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.Mvc;

namespace BasicInfomationPlugin.Controllers

{

using BasicInfomationPlugin.DataAccessors;

using BasicInfomationPlugin.Models;

using BasicInfomationPlugin.ViewModels;

public class MeterHardwareTypeController : MasterController

{

private MeterHardwareTypeDataAccessor \_meterHeardwareTypedb;

public MeterHardwareTypeController()

{

\_meterHeardwareTypedb = new MeterHardwareTypeDataAccessor();

}

public ActionResult Index(int? currentPageNum, int? pageSize)

{

var dataPagination = \_meterHeardwareTypedb.GetDataPagination(currentPageNum, pageSize);

var dbLitViewModel = new MeterHardwareTypeListViewModel

{

Pagination = dataPagination,

DeleteFailed = TempData["DeleteFailed"] == null ? false : true

};

return View(dbLitViewModel);

}

public ActionResult Add()

{

return View("Edit", GetModel(EditPara.Add,0));

}

public ActionResult Edit(decimal id)

{

return View("Edit", GetModel(EditPara.Modi, id));

}

private MeterHardwareTypeModel GetModel(EditPara editPara,decimal id)

{

var model = new MeterHardwareTypeModel();

if (editPara==EditPara.Modi)

{

model = \_meterHeardwareTypedb.GetById(id);

}

else

{

if (TempData["Model"] != null)

{

var tempModel = TempData["Model"] as MeterHardwareTypeModel;

model.TipMsg = "保存成功！";

}

}

model.EditParaType = editPara;

return model;

}

public ActionResult SaveItem(MeterHardwareTypeModel model)

{

if (!ModelState.IsValid)

{

return View("Edit", GetModel(model.EditParaType, model.ID));

}

else

{

\_meterHeardwareTypedb.Update(model);

if (model.EditParaType==EditPara.Modi){

return RedirectToAction("Index");

}

else

{

TempData["Model"] = model;

return RedirectToAction("Add");

}

}

}

public ActionResult Delete(decimal id)

{

int modelcount = \_meterHeardwareTypedb.Delete(id);

if (modelcount == -10)

{

TempData["DeleteFailed"] = "true";

}

return base.ReturnOperationUrl();

}

#region 验证属性值是否存在...

public JsonResult CheckNameIsExist(string name, decimal id)

{

bool isExist = false;

isExist = \_meterHeardwareTypedb.NameIsExist(name,id);

return Json(!isExist, JsonRequestBehavior.AllowGet);

}

public JsonResult CheckIdIsExist(decimal id, string editParaType)

{

bool isExist = false;

isExist = \_meterHeardwareTypedb.IdIsExist(id, editParaType);

return Json(!isExist, JsonRequestBehavior.AllowGet);

}

#endregion

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.Mvc;

namespace BasicInfomationPlugin.Controllers

{

using OracleEFModel;

using BasicInfomationPlugin.DataAccessors;

using BasicInfomationPlugin.Models;

using BasicInfomationPlugin.ViewModels;

using Newtonsoft.Json;

public class MasterStationController : MasterController

{

private MasterStationDataAccessor \_masterStationDb;

private CascadeDataAccessor \_cascadeDb;

public MasterStationController()

{

\_masterStationDb = new MasterStationDataAccessor();

\_cascadeDb = new CascadeDataAccessor();

}

public ActionResult Index(int? currentPageNum, int? pageSize)

{

var dataPagination = \_masterStationDb.GetDataPagination(currentPageNum, pageSize);

var dbLitViewModel = new MasterStationListViewModel

{

Pagination = dataPagination,

DeleteFailed = TempData["DeleteFailed"] == null ? false : true

};

return View(dbLitViewModel);

}

#region 增删改方法 Add() Edit() Delete() GetModel() SaveItem()

public ActionResult Add()

{

return View("Edit", GetModel(EditPara.Add, 0));

}

public ActionResult Edit(decimal id)

{

return View(GetModel(EditPara.Modi, id));

}

private MasterStationModel GetModel(EditPara editPara, decimal id)

{

var model = new MasterStationModel();

if (editPara == EditPara.Modi)

{

model = \_masterStationDb.GetById(id);

}

else

{

if (TempData["Model"] != null)

{

var tempModel = TempData["Model"] as MasterStationModel;

model.ServerID = tempModel.ServerID;

model.NeedZigbeeAddress = tempModel.NeedZigbeeAddress;

model.TipMsg = "保存成功！";

}

}

model.EditParaType = editPara;

model.ServerList = \_cascadeDb.GetSelectList(model.ServerID, "Server");

model.NeedZigbeeAddressList = \_cascadeDb.GetEnumList(model.NeedZigbeeAddress, "NeedZigbeeAddress");

return model;

}

public ActionResult SaveItem(MasterStationModel model)

{

if (!ModelState.IsValid)

{

return View("Edit", GetModel(model.EditParaType, model.ID));

}

else

{

\_masterStationDb.Update(model);

if (model.EditParaType == EditPara.Modi)

{

return RedirectToAction("Index");

}

else

{

TempData["Model"] = model;

return RedirectToAction("Add");

}

}

}

public ActionResult Delete(decimal id)

{

int modelcount = \_masterStationDb.Delete(id);

if (modelcount == 0)

{

TempData["DeleteFailed"] = "true";

}

return base.ReturnOperationUrl();

}

#endregion

#region 检测字段行数据是否存在

public JsonResult CheckNameIsExist(string name, decimal id, decimal serverID)

{

bool isExist = false;

isExist = \_masterStationDb.CheckNameIsExist(name, id, serverID);

return Json(!isExist, JsonRequestBehavior.AllowGet);

}

public JsonResult CheckIpAddressIsExist(string ipAddress, decimal id, decimal serverID)

{

bool isExist = false;

isExist = \_masterStationDb.CheckIpAddressIsExist(ipAddress, id, serverID);

return Json(!isExist, JsonRequestBehavior.AllowGet);

}

public JsonResult CheckIpPortIsExist(string ipPortNo, decimal id, decimal serverID, string ipAddress)

{

bool isExist = false;

isExist = \_masterStationDb.CheckIpPortIsExist(ipPortNo, id, serverID, ipAddress);

return Json(!isExist, JsonRequestBehavior.AllowGet);

}

public JsonResult CheckHardwareAddressIsExist(string hardwareAddress, decimal id, decimal serverID)

{

bool isExist = false;

isExist = \_masterStationDb.CheckHardwareAddressIsExist(hardwareAddress, id, serverID);

return Json(!isExist, JsonRequestBehavior.AllowGet);

}

#endregion

#region 导入导出数据

public void Export()

{

var excel = \_masterStationDb.ExportData();

DataImpExpExcel dataToExcel = new DataImpExpExcel();

HttpContext context = System.Web.HttpContext.Current;

new DataImpExpExcel().Export(System.Web.HttpContext.Current, excel, "../../");

}

public void Import(string uploadType, HttpPostedFileBase fileName)

{

HttpPostedFileBase file = fileName;

string type = uploadType;

string json = string.Empty;

string JsonError = string.Empty;

Response.ContentType = "text/html";

if (file != null)

{

string targetFilePath = DataImpExpExcel.UploadFileAndGetTargetPath(System.Web.HttpContext.Current, file);

try

{

JsonError += \_masterStationDb.ImportData(targetFilePath, type);//导入

}

catch (Exception ex)

{

JsonError += "\r\n" + "导入失败：" + ex.Message;

}

}

else

{

JsonError += "\r\n" + "导入失败：文件为空";

}

if (!string.IsNullOrEmpty(JsonError))

{

json = JsonConvert.SerializeObject(new

{

success = false,

msg = JsonError.Replace("\r\n", "<br />")

});

}

else

{

json = JsonConvert.SerializeObject(new { success = true, msg = "导入成功" });

}

Response.Write(json);

Response.End();

}

#endregion

}

}

using BasicInfomationPlugin.Models;

using BasicInfomationPlugin.Models.ExcelModel;

using BasicInfomationPlugin.ViewModels;

using LinqToExcel;

using OracleEFModel;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using UIShell.OSGi.Utility;

using UIShell.PartialViewShellPlugin.ViewModels;

namespace BasicInfomationPlugin.DataAccessors

{

public class BuildingDataAccessor

{

private const string ERROR\_IN\_EXCEL = "导入失败：在文档第 {0} 行 {1} 的值【{2}】在导入的Excel中不符合 {3} 原则，请确认后重新导入";

private const string ERROR\_IN\_DATASOURCE = "导入失败：在文档第 {0} 行 {1} 的值【{2}】在数据库中不符合 {3} 原则，请确认后重新导入";

public Pagination<BuildingModel> GetBuildingDataPage(decimal heatSID, decimal comID, int? currentPageNum, int? pageSize)

{

using (Entities entities = new Entities())

{

try

{

var queryResult = (from i in entities.Building

join t in entities.Community on i.CommunityID equals t.ID

join j in entities.HeatingStation on t.HeatingStationID equals j.ID

join k in entities.SubCompany on j.SubCompanyID equals k.ID

where k.ID == j.SubCompanyID

&& heatSID == 0 ? 1 == 1 : t.HeatingStationID == heatSID

&& comID == 0 ? 1 == 1 : i.CommunityID == comID

orderby i.ID descending

select new BuildingModel

{

id = i.ID,

name = i.Name,

description = i.Description,

isInsulating = i.ISINSULATING == 0 ? false : true,

isInsulatingStr = i.ISINSULATING == 1 ? "是" : "否",

subCompanyId = j.SubCompanyID,

subCompanyName = k.Name,

HeatingStationId = t.HeatingStationID,

heatingStationName = j.Name,

communityId = i.CommunityID,

communityName = t.Name

});

Pagination<BuildingModel> pagination = new Pagination<BuildingModel>(currentPageNum, pageSize);

pagination.DataCount = queryResult.Count();

pagination.DataList = queryResult.Skip((pagination.CurrentPageNum - 1) \* pagination.PageSize).Take(pagination.PageSize).ToList<BuildingModel>();

return pagination;

}

catch (Exception e)

{

throw new Exception(e.StackTrace);

}

}

}

public BuildingModel GetItem(decimal id)

{

using (Entities entities = new Entities())

{

try

{

var result = (from i in entities.Building

join t in entities.Community on i.CommunityID equals t.ID

join j in entities.HeatingStation on t.HeatingStationID equals j.ID

join k in entities.SubCompany on j.SubCompanyID equals k.ID

where i.ID == id

select new BuildingModel()

{

id = i.ID,

name = i.Name,

description = i.Description,

isInsulating = i.ISINSULATING == 0 ? false : true,

isInsulatingStr = i.ISINSULATING == 1 ? "是" : "否",

subCompanyId = j.SubCompanyID,

HeatingStationId = t.HeatingStationID,

subCompanyName = k.Name,

heatingStationName = j.Name,

communityId = i.CommunityID,

communityName = t.Name

}).FirstOrDefault();

return result;

}

catch (Exception e)

{

throw new Exception(e.StackTrace);

}

}

}

public void DelItem(decimal id)

{

using (Entities entities = new Entities())

{

try

{

var entity = entities.Set<Building>().Find(id);

entities.Set<Building>().Remove(entity);

entities.SaveChanges();

}

catch (Exception e)

{

throw new Exception(e.StackTrace);

}

}

}

public void EditToSaveItem(BuildingModel item)

{

using (Entities entities = new Entities())

{

try

{

var sc = entities.Set<Building>().Find(item.id);

sc.Name = item.name;

sc.Description = item.description;

sc.CommunityID = item.communityId;

sc.ISINSULATING = item.isInsulating ? 1 : 0;

entities.SaveChanges();

}

catch (Exception e)

{

throw new Exception(e.StackTrace);

}

}

}

public void AddToSaveItem(BuildingModel item)

{

using (Entities entities = new Entities())

{

try

{

Building sc = new Building();

sc.Name = item.name;

sc.Description = item.description;

sc.CommunityID = item.communityId;

sc.ISINSULATING = item.isInsulating ? 1 : 0;

entities.Set<Building>().Add(sc);

entities.SaveChanges();

}

catch (Exception e)

{

throw new Exception(e.StackTrace);

}

}

}

internal string ImportData(string targetFilePath, string type)

{

string jsonError = "";

try

{

var excel = new ExcelQueryFactory(targetFilePath);

var defaultTable = from c in excel.Worksheet(0)

select c;

int rowNum = 0;

var items = defaultTable.ToList();

var checkExcelSource = new List<BuildingExcelModel>();

foreach (var item in items)

{

rowNum++;

var model = new BuildingExcelModel

{

RowNum = rowNum,

BuildingName = item[0].Value.ToString().Trim(),

Description = item[1].Value.ToString().Trim(),

IsInsulation = item[2].Value.ToString().Trim(),

SubCompanyName = item[3].Value.ToString().Trim(),

HeatingStationName = item[4].Value.ToString().Trim(),

CommunityName = item[5].Value.ToString().Trim()

};

checkExcelSource.Add(model);

}

FileLogUtility.Inform(string.Format("上传的数据包含{0}条记录。", items.Count));

using (Entities entities = new Entities())

{

foreach (var item in checkExcelSource)

{

string errorStr = string.Empty;

if ("ADD".Equals(type.ToUpper()))

{

#region ===添加部分

var model = new Building();

jsonError += GetCheckData(ref model, item, checkExcelSource);

entities.Building.Add(model);

#endregion

}

}

if (string.IsNullOrEmpty(jsonError))

{

entities.SaveChanges(System.Data.Objects.SaveOptions.None);

}

}

}

catch (Exception ex)

{

if (!(ex is InvalidOperationException))

jsonError += "<br>" + "导入失败：" + ex.Message;

else

jsonError += "<br>" + "导入过程出现异常：" + ex.Message;

}

return jsonError;

}

private string GetCheckData(ref Building buiding,BuildingExcelModel item, List<BuildingExcelModel> checkSource)

{

string jsonError = string.Empty;

using (Entities entities = new Entities())

{

decimal rowID = item.RowNum;

string name = item.BuildingName; //名称

string description = item.Description;//描述

string isinulation = item.IsInsulation; //是否有外墙保温

string subCompany = item.SubCompanyName;//分公司

string heatStation = item.HeatingStationName;//换热站

string community = item.CommunityName;//小区

var queryExcel = from i in checkSource

where i.CommunityName.Equals(community) && i.BuildingName.Equals(name)

select i;

if (queryExcel.Count() > 1)

{

jsonError += "<br/>" + string.Format(ERROR\_IN\_EXCEL, rowID,"楼栋名称", name,"同一小区楼栋名称唯一性");

}

var queryData = from i in entities.Building

where i.Name.Equals(name) && i.Community.Name.Equals(community)

select i;

if (queryData.Count() > 0)

{

jsonError += "<br>" + string.Format(ERROR\_IN\_DATASOURCE, rowID, "楼栋名称", name, "同一小区楼栋名称唯一性");

}

var queryCommunity = from i in entities.Community

where i.Name.Equals(community)

&& i.HeatingStation.Name.Equals(heatStation)

&& i.HeatingStation.SubCompany.Name.Equals(subCompany)

select i;

Community tmpCommunity = queryCommunity.FirstOrDefault();

if (tmpCommunity == null)

{

jsonError += "<br>" + string.Format(ERROR\_IN\_DATASOURCE, rowID,

"分公司~换热站~小区名称",

subCompany +"~"+heatStation+"~"+ community,

"分公司~换热站~小区名称必须已录入数据库");

}

if (!isinulation.Equals("是") && !isinulation.Equals("否"))

{

jsonError += "<br>" + string.Format(ERROR\_IN\_DATASOURCE, rowID, "是否外墙保温", name, "是否外墙保温只能填写是或者否 ");

}

if (string.IsNullOrEmpty(jsonError))

{

buiding.CommunityID = tmpCommunity.ID;

buiding.Description = description;

buiding.ISINSULATING = isinulation.Equals("是") ? 1 : 0;

buiding.Name = name;

}

return jsonError;

}

}

internal List<object> ExportData(BuildingViewModel model)

{

decimal tmpHeatSId = model.heatSID ?? 0;

decimal tmpComId = model.comID ?? 0;

using (Entities entities = new Entities())

{

var query = (from i in entities.Building

select i);

if (tmpHeatSId == 0)

return query.Select(i => new

{

名称 = i.Name,

描述 = i.Description,

是否有外墙保温 = i.ISINSULATING == 1 ? "是" : "否",

分公司 = i.Community.HeatingStation.SubCompany.Name,

换热站 = i.Community.HeatingStation.Name,

小区 = i.Community.Name

}).ToList<object>();

else

{

if (tmpComId == 0)

{

return query.Where(m => m.Community.HeatingStationID == tmpHeatSId).Select(i => new

{

名称 = i.Name,

描述 = i.Description,

是否有外墙保温 = i.ISINSULATING == 1 ? "是" : "否",

分公司 = i.Community.HeatingStation.SubCompany.Name,

换热站 = i.Community.HeatingStation.Name,

小区 = i.Community.Name

}).ToList<object>();

}

else

return query.Where(m => m.Community.HeatingStationID == tmpHeatSId && m.CommunityID == tmpComId).Select(i => new

{

名称 = i.Name,

描述 = i.Description,

是否有外墙保温 = i.ISINSULATING == 1 ? "是" : "否",

分公司 = i.Community.HeatingStation.SubCompany.Name,

换热站 = i.Community.HeatingStation.Name,

小区 = i.Community.Name

}).ToList<object>();

}

}

}

public bool IsExists(Dictionary<string, object> limitCondition)

{

string name = Convert.ToString(limitCondition["Name"]);

decimal id = Convert.ToDecimal(limitCondition["ID"]);

decimal communityID = Convert.ToDecimal(limitCondition["CommunityID"]);

bool isExists = false;

if (id > 0)//修改验证是否存在重复名称存在返回true

{

using (Entities entities = new Entities())

{

var modelCount = entities.Building.Count(i => i.Name == name && i.CommunityID == communityID);

var num = entities.Building.Count(i => i.Name == name && i.ID == id);

if (modelCount > 0 && num != 1)

{

isExists = true;

}

}

}

else {

using (Entities entities = new Entities())

{

var modelNum = entities.Building.Count(i => i.Name == name && i.CommunityID == communityID);

if (modelNum > 0)

isExists = true;

}

}

return isExists;

}

public bool IsUsed(decimal id)

{

using (Entities entities = new Entities())

{

try

{

bool isUsed = false;

var resultNum = entities.Unit.Count(i => i.BuildingID == id);

if (resultNum > 0)

{

isUsed = true;

}

return isUsed;

}

catch (Exception e)

{

throw new Exception(e.StackTrace);

}

}

}

}

}

using BasicInfomationPlugin.Models;

using BasicInfomationPlugin.Models.ExcelModel;

using LinqToExcel;

using OracleEFModel;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using UIShell.OSGi.Utility;

using UIShell.PartialViewShellPlugin.ViewModels;

namespace BasicInfomationPlugin.DataAccessors

{

public class CommunityDataAccessor

{

private const string ERROR\_IN\_EXCEL = "导入失败：在文档第 {0} 行 {1} 的值【{2}】在导入的Excel中不符合 {3} 原则，请确认后重新导入";

private const string ERROR\_IN\_DATASOURCE = "导入失败：在文档第 {0} 行 {1} 的值【{2}】在数据库中不符合 {3} 原则，请确认后重新导入";

public CommunityDataAccessor()

{

}

public Pagination<CommunityModel> GetCommunityPagedData(decimal heatSID, int? currentPageNum, int? pageSize)

{

using (Entities entities = new Entities())

{

try

{

var QueryResult = (from i in entities.Community

join j in entities.HeatingStation on i.HeatingStationID equals j.ID

join k in entities.SubCompany on j.SubCompanyID equals k.ID

where k.ID==j.SubCompanyID && heatSID == 0 ? 1 == 1 : i.HeatingStationID == heatSID

orderby i.ID descending

select new CommunityModel

{

id = i.ID,

name = i.Name,

description = i.Description,

subCompanyId = j.SubCompanyID,

subCompanyName = k.Name,

HeatingStationId = i.HeatingStationID,

heatingStationName = j.Name

});

Pagination<CommunityModel> pagination = new Pagination<CommunityModel>(currentPageNum, pageSize);

pagination.DataCount = QueryResult.Count();

pagination.DataList = QueryResult.Skip((pagination.CurrentPageNum - 1) \* pagination.PageSize).Take(pagination.PageSize).ToList<CommunityModel>() ;

return pagination;

}

catch (Exception e)

{

throw new Exception(e.StackTrace);

}

}

}

public CommunityModel GetItem(decimal id)

{

using (Entities entities = new Entities())

{

try

{

CommunityModel item = (from i in entities.Community

join j in entities.HeatingStation on i.HeatingStationID equals j.ID

join k in entities.SubCompany on j.SubCompanyID equals k.ID

where i.ID == id

select new CommunityModel()

}

}

}

public void EditToSaveItem(CommunityModel item)

{

using (Entities entities = new Entities())

{

try

{

var sc = entities.Set<Community>().Find(item.id);

sc.Name = item.name;

sc.Description = item.description;

sc.HeatingStationID = item.HeatingStationId;

entities.SaveChanges();

}

catch (Exception e)

{

throw new Exception(e.StackTrace);

}

}

}

public void AddToSaveItem(CommunityModel item)

{

using (Entities entities = new Entities())

{

try

{

Community sc = new Community();

sc.Name = item.name;

sc.Description = item.description;

sc.HeatingStationID = item.HeatingStationId;

sc.GISID = 1;

entities.Set<Community>().Add(sc);

entities.SaveChanges();

}

catch (Exception e)

{

throw new Exception(e.StackTrace);

}

}

}

internal string ImportData(string targetFilePath, string type)

{

string jsonError = "";

try

{

var excel = new ExcelQueryFactory(targetFilePath);

var defaultTable = from c in excel.Worksheet(0)

select c;

int rowNum = 1;

var items = defaultTable.ToList();

var checkExcelSource = new List<CommunityExcelModel>();

foreach (var item in items)

{

rowNum++;

var model = new CommunityExcelModel

{

RowNum = rowNum,

CommunityName = item[0].Value.ToString().Trim(),

Description = item[1].Value.ToString().Trim(),

SubCompanyName = item[2].Value.ToString().Trim(),

HeatingStationName = item[3].Value.ToString().Trim()

};

checkExcelSource.Add(model);

}

FileLogUtility.Inform(string.Format("上传的数据包含{0}条记录。", items.Count));

using (Entities entities = new Entities())

{

foreach (var item in checkExcelSource)

{

string errorStr = string.Empty;

if ("ADD".Equals(type.ToUpper()))

{

#region ===添加部分

Community model = new Community();

jsonError += GetCheckData(ref model,item,checkExcelSource);

entities.Set<Community>().Add(model);

#endregion

}

}

if (string.IsNullOrEmpty(jsonError))

{

entities.SaveChanges(System.Data.Objects.SaveOptions.None);

}

}

}

catch (Exception ex)

{

if (!(ex is InvalidOperationException))

jsonError += "<br>" + "导入失败：" + ex.Message;

else

jsonError += "<br>" + "导入过程出现异常：" + ex.Message;

}

return jsonError;

}

private string GetCheckData(ref Community community,CommunityExcelModel item,List<CommunityExcelModel> checkSource)

{

string jsonError = string.Empty;

using (Entities entities = new Entities())

{

decimal rowID = item.RowNum;

string name = item.CommunityName; //名称

string description = item.Description;//描述

string subCompany = item.SubCompanyName; //分公司

string heatStation = item.HeatingStationName;//换热站

var queryExcel = from i in checkSource

where i.CommunityName.Equals(name)

&& i.HeatingStationName.Equals(heatStation)

&& i.SubCompanyName.Equals(subCompany)

select i;

if (queryExcel.Count() > 1)

{

jsonError += "<br/>" + string.Format(ERROR\_IN\_EXCEL, rowID, "小区名称", name, "在同一个换热站下小区名称唯一性");

}

var queryData = from i in entities.Community

where i.Name.Equals(name)

&& i.HeatingStation.Name.Equals(heatStation)

&& i.HeatingStation.SubCompany.Name.Equals(subCompany)

select i;

if (queryData.Count() > 0)

{

jsonError += "<br/>" + string.Format(ERROR\_IN\_DATASOURCE, rowID, "小区名称", name, "在同一个换热站下小区名称唯一性");

}

var queryHeatStation = (from i in entities.HeatingStation

where i.Name.Equals(heatStation)

&& i.SubCompany.Name.Equals(subCompany)

select i);

HeatingStation tmpHeatStation = queryHeatStation.First();

if (tmpHeatStation == null)

{

jsonError += "<br>" + string.Format(ERROR\_IN\_DATASOURCE, rowID, "分公司名称~换热站名称", subCompany +"~"+ heatStation, "分公司和换热站在数据库中必须已存在");

}

if (string.IsNullOrEmpty(jsonError))

{

community.Name = name;

community.Description = description;

community.HeatingStationID = tmpHeatStation.ID;

community.GISID = 1;

}

return jsonError;

}

}

internal List<object> ExportData(decimal? heatSID)

{

decimal tmpHeatSId = heatSID??0;

using (Entities entities = new Entities())

{

var query = (from i in entities.Community

select i);

if (tmpHeatSId != 0)

return query.Where(m => m.HeatingStationID == tmpHeatSId).Select(i => new

{

名称 = i.Name,

描述 = i.Description,

分公司 = i.HeatingStation.SubCompany.Name,

换热站 = i.HeatingStation.Name

}).ToList<object>();

else

return query.Select(i => new

{

名称 = i.Name,

描述 = i.Description,

分公司 = i.HeatingStation.SubCompany.Name,

换热站 = i.HeatingStation.Name

}).ToList<object>();

}

}

public bool IsExists(Dictionary<string, object> limitCondition)

{

string name = Convert.ToString(limitCondition["Name"]);

decimal id = Convert.ToDecimal(limitCondition["ID"]);

decimal heatingStationID = Convert.ToDecimal(limitCondition["HeatingStationID"]);

bool isExists = false;

if (id > 0)//修改验证是否存在重复名称存在返回true

{

using (Entities entities = new Entities())

{

var modelCount = entities.Community.Count(i => i.Name == name && i.HeatingStationID == heatingStationID);

var num = entities.Community.Count(i => i.Name == name && i.ID == id);

if (modelCount > 0 && num != 1)

{

isExists = true;

}

}

}

else //添加验证是否存在重复名称存在返回true

{

using (Entities entities = new Entities())

{

var modelNum = entities.Community.Count(i => i.Name == name

&& i.HeatingStationID == heatingStationID);

if (modelNum > 0)

isExists = true;

}

}

return isExists;

}

public bool IsUsed(decimal id)

{

using (Entities entities = new Entities())

{

try

{

bool isUsed = false;

var buildingCount = entities.Building.Count(i => i.CommunityID == id);

var EquipmentCount = entities.Equipment.Count(i => i.CommunityID == id);

if (buildingCount > 0 || EquipmentCount > 0)

{

isUsed = true;

}

return isUsed;

}

catch (Exception e)

{

throw new Exception(e.StackTrace);

}

}

}

}

}

using BasicInfomationPlugin.Models;

using BasicInfomationPlugin.Models.ExcelModel;

using LinqToExcel;

using OracleEFModel;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using UIShell.OSGi.Utility;

using UIShell.PartialViewShellPlugin.ViewModels;

namespace BasicInfomationPlugin.DataAccessors

{

public class UnitDataAccessor

{

private const string ERROR\_IN\_EXCEL = "导入失败：在文档第 {0} 行 {1} 的值【{2}】在导入的Excel中不符合 {3} 原则，请确认后重新导入";

private const string ERROR\_IN\_DATASOURCE = "导入失败：在文档第 {0} 行 {1} 的值【{2}】在数据库中不符合 {3} 原则，请确认后重新导入";

public Pagination<UnitModel> GetUnitDataPage(decimal heatSID, decimal comID, decimal buildID,

int? currentPageNum, int? pageSize)

{

using (Entities entities = new Entities())

{

try

{

var queryResult = (from i in entities.Unit

join s in entities.Building on i.BuildingID equals s.ID

join t in entities.Community on s.CommunityID equals t.ID

join j in entities.HeatingStation on t.HeatingStationID equals j.ID

join k in entities.SubCompany on j.SubCompanyID equals k.ID

where k.ID == j.SubCompanyID

&& heatSID == 0 ? 1 == 1 : t.HeatingStationID == heatSID

&& comID == 0 ? 1 == 1 : s.CommunityID == comID

&& buildID == 0 ? 1 == 1 : i.BuildingID == buildID

orderby i.ID descending

select new UnitModel()

{

id = i.ID,

name = i.Name,

description = i.Description,

subCompanyId = j.SubCompanyID,

subCompanyName = k.Name,

HeatingStationId = t.HeatingStationID,

heatingStationName = j.Name,

communityId = s.CommunityID,

communityName = t.Name,

buildingId = i.BuildingID,

buildingName = s.Name

});

Pagination<UnitModel> pagination = new Pagination<UnitModel>(currentPageNum, pageSize);

pagination.DataCount = queryResult.Count();

pagination.DataList = queryResult.Skip((pagination.CurrentPageNum - 1) \* pagination.PageSize).Take(pagination.PageSize).ToList<UnitModel>();

return pagination;

}

catch (Exception e)

{

throw new Exception(e.StackTrace);

}

}

}

public UnitModel GetItem(decimal id)

{

using (Entities entities = new Entities())

{

try

{

var result = (from i in entities.Unit

join s in entities.Building on i.BuildingID equals s.ID

join t in entities.Community on s.CommunityID equals t.ID

join j in entities.HeatingStation on t.HeatingStationID equals j.ID

join k in entities.SubCompany on j.SubCompanyID equals k.ID

where i.ID == id

select new UnitModel()

{

id = i.ID,

name = i.Name,

description = i.Description,

subCompanyId = j.SubCompanyID,

HeatingStationId = t.HeatingStationID,

subCompanyName = k.Name,

heatingStationName = j.Name,

communityId = s.CommunityID,

communityName = t.Name,

buildingId = i.BuildingID,

buildingName = s.Name

}).FirstOrDefault();

return result;

}

catch (Exception e)

{

throw new Exception(e.StackTrace);

}

}

}

public void DelItem(decimal id)

{

using (Entities entities = new Entities())

{

try

{

var entity = entities.Set<Unit>().Find(id);

entities.Set<Unit>().Remove(entity);

entities.SaveChanges();

}

catch (Exception e)

{

throw new Exception(e.StackTrace);

}

}

}

public void EditToSaveItem(UnitModel item)

{

using (Entities entities = new Entities())

{

try

{

var sc = entities.Set<Unit>().Find(item.id);

sc.Name = item.name;

sc.Description = item.description;

sc.BuildingID = item.buildingId;

entities.SaveChanges();

}

catch (Exception e)

{

throw new Exception(e.StackTrace);

}

}

}

public void AddToSaveItem(UnitModel item)

{

using (Entities entities = new Entities())

{

try

{

Unit sc = new Unit();

sc.Name = item.name;

sc.Description = item.description;

sc.BuildingID = item.buildingId;

entities.Set<Unit>().Add(sc);

entities.SaveChanges();

}

catch (Exception e)

{

throw new Exception(e.StackTrace);

}

}

}

internal string ImportData(string targetFilePath, string type)

{

string jsonError = "";

try

{

var excel = new ExcelQueryFactory(targetFilePath);

var defaultTable = from c in excel.Worksheet(0)

select c;

int rowNum = 1;

var items = defaultTable.ToList();

var checkExcelSource = new List<UnitExcelModel>();

foreach (var item in items)

{

rowNum++;

var model = new UnitExcelModel

{

RowNum = rowNum,

UnitName = item[0].Value.ToString().Trim(),

Description = item[1].Value.ToString().Trim(),

SubCompanyName = item[2].Value.ToString().Trim(),

HeatingStationName = item[3].Value.ToString().Trim(),

CommunityName = item[4].Value.ToString().Trim(),

BuildingName = item[5].Value.ToString().Trim()

};

checkExcelSource.Add(model);

}

FileLogUtility.Inform(string.Format("上传的数据包含{0}条记录。", items.Count));

using (Entities entities = new Entities())

{

foreach (var item in checkExcelSource)

{

string errorStr = string.Empty;

if ("ADD".Equals(type.ToUpper()))

{

#region ===添加部分

Unit model = new Unit();

jsonError += GetCheckData(ref model,item,checkExcelSource);

entities.Unit.Add(model);

#endregion

}

}

if (string.IsNullOrEmpty(jsonError))

{

entities.SaveChanges(System.Data.Objects.SaveOptions.None);

}

}

}

catch (Exception ex)

{

if (!(ex is InvalidOperationException))

jsonError += "<br>" + "导入失败：" + ex.Message;

else

jsonError += "<br>" + "导入过程出现异常：" + ex.Message;

}

return jsonError;

}

private string GetCheckData(ref Unit unit ,UnitExcelModel item, List<UnitExcelModel> checkSource)

{

string jsonError = string.Empty;

using (Entities entities = new Entities())

{

decimal rowID = item.RowNum;

string name = item.UnitName; //名称

string description = item.Description;//描述

string subCompany = item.SubCompanyName;//分公司

string heatStation = item.HeatingStationName;//换热站

string community = item.CommunityName;//小区

string building = item.BuildingName;//楼宇

var queryExcel = from i in checkSource

where i.UnitName.Equals(name)

&& i.BuildingName.Equals(building)

&& i.CommunityName.Equals(community)

&& i.HeatingStationName.Equals(heatStation)

&& i.SubCompanyName.Equals(subCompany)

select i;

if (queryExcel.Count() > 1)

{

jsonError += "<br/>" + string.Format(ERROR\_IN\_EXCEL, rowID, "单元名称", name, "在同一个楼宇下单元名称不能重复");

}

var queryData = from i in entities.Unit

where i.Name.Equals(name)

&& i.Building.Community.HeatingStation.SubCompany.Name.Equals(subCompany)

&& i.Building.Community.HeatingStation.Name.Equals(heatStation)

&& i.Building.Community.Name.Equals(community)

&& i.Building.Name.Equals(building)

select i;

if (queryData.Count() > 0)

{

jsonError += "<br/>" + string.Format(ERROR\_IN\_DATASOURCE, rowID, "单元名称", name, "在同一个楼宇下单元名称不能重复");

}

var queryBuild = from i in entities.Building

where i.Community.HeatingStation.SubCompany.Name.Equals(subCompany)

&& i.Community.HeatingStation.Name.Equals(heatStation)

&& i.Community.Name.Equals(community)

&& i.Name.Equals(building)

select i;

Building tmpBuilding = queryBuild.First();

if (tmpBuilding == null)

{

jsonError += "<br/>" + string.Format(ERROR\_IN\_DATASOURCE, rowID,

"分公司~换热站~小区~楼栋名称",

subCompany+"~"+heatStation+"~"+community+"~"+building,

"分公司~换热站~小区~楼栋名称在数据库中必须存在");

}

if (string.IsNullOrEmpty(jsonError))

{

unit.BuildingID = tmpBuilding.ID;

unit.Name = name;

unit.Description = description;

}

return jsonError;

}

}

internal List<object> ExportData(decimal? heatSID, decimal? comID,decimal? buildID)

{

decimal tmpHeatSId = heatSID ?? 0;

decimal tmpComId = comID ?? 0;

decimal tmpBuildId = buildID ?? 0;

using (Entities entities = new Entities())

{

var query = (from i in entities.Unit

select i);

if (tmpHeatSId == 0)

return query.Select(i => new

{

名称 = i.Name,

描述 = i.Description,

分公司 = i.Building.Community.HeatingStation.SubCompany.Name,

换热站 = i.Building.Community.HeatingStation.Name,

小区 = i.Building.Community.Name,

楼宇=i.Building.Name

}).ToList<object>();

else

{

if (tmpComId == 0)

{

return query.Where(m => m.Building.Community.HeatingStationID == tmpHeatSId).Select(i => new

{

名称 = i.Name,

描述 = i.Description,

分公司 = i.Building.Community.HeatingStation.SubCompany.Name,

换热站 = i.Building.Community.HeatingStation.Name,

小区 = i.Building.Community.Name,

楼宇 = i.Building.Name

}).ToList<object>();

}

else

{

if (tmpBuildId == 0)

{

return query.Where(m => m.Building.Community.HeatingStationID == tmpHeatSId && m.Building.CommunityID == tmpComId).Select(i => new

{

名称 = i.Name,

描述 = i.Description,

分公司 = i.Building.Community.HeatingStation.SubCompany.Name,

换热站 = i.Building.Community.HeatingStation.Name,

小区 = i.Building.Community.Name,

楼宇 = i.Building.Name

}).ToList<object>();

}

else

{

return query.Where(m => m.Building.Community.HeatingStationID == tmpHeatSId && m.Building.CommunityID == tmpComId&&m.BuildingID==tmpBuildId).Select(i => new

{

名称 = i.Name,

描述 = i.Description,

分公司 = i.Building.Community.HeatingStation.SubCompany.Name,

换热站 = i.Building.Community.HeatingStation.Name,

小区 = i.Building.Community.Name,

楼宇 = i.Building.Name

}).ToList<object>();

}

}

}

}

}

public bool IsExists(Dictionary<string, object> limitCondition)

{

string name = Convert.ToString(limitCondition["Name"]);

decimal id = Convert.ToDecimal(limitCondition["ID"]);

decimal buildingID = Convert.ToDecimal(limitCondition["BuildingID"]);

bool isExists = false;

if (id > 0)//修改验证是否存在重复名称存在返回true

{

using (Entities entities = new Entities())

{

var modelCount = entities.Unit.Count(i => i.Name == name && i.BuildingID == buildingID);

var num = entities.Unit.Count(i => i.Name == name && i.ID == id);

if (modelCount > 0 && num != 1)

{

isExists = true;

}

}

}

else //添加验证是否存在重复名称存在返回true

{

using (Entities entities = new Entities())

{

var modelNum = entities.Unit.Count(i => i.Name == name && i.BuildingID == buildingID);

if (modelNum > 0)

isExists = true;

}

}

return isExists;

}

public bool IsUsed(decimal id)

{

using (Entities entities = new Entities())

{

try

{

bool isUsed = false;

var resultNum = entities.User.Count(i => i.UnitID == id);

if (resultNum > 0)

{

isUsed = true;

}

return isUsed;

}

catch (Exception e)

{

throw new Exception(e.StackTrace);

}

}

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

namespace BasicInfomationPlugin.DataAccessors

{

using OracleEFModel;

using UIShell.PartialViewShellPlugin.ViewModels;

using BasicInfomationPlugin.Models;

public class ServerDataAccessor

{

public ServerDataAccessor() {}

public Pagination<ServerModel> GetDataPagination(int? currentPageNum,int? pageSize)

{

try

{

using (Entities entities = new Entities())

{

var queryResult = entities.Server.OrderByDescending(o => o.ID);

var dataCount = queryResult.Count();

var pagination = new Pagination<ServerModel>(currentPageNum, pageSize, dataCount);

pagination.DataList = queryResult

.Skip((pagination.CurrentPageNum - 1) \* pagination.PageSize)

.Take(pagination.PageSize).AsEnumerable()

.Select(i => new ServerModel

{

ID = i.ID,

Name = i.Name,

IPAddress = i.IPAddress,

Description = i.Description

}).ToList();

return pagination;

}

}

catch (Exception e)

{

throw new Exception(e.StackTrace);

}

}

public ServerModel GetById(decimal id)

{

using (Entities entities = new Entities())

{

var model = entities.Server.FirstOrDefault(f => f.ID == id);

return new ServerModel

{

ID = model.ID,

Name = model.Name,

IPAddress = model.IPAddress,

Description = model.Description

};

}

}

public bool IsExists(string condRule, decimal id)

{

bool isExists = false;

if (id > 0)//修改服务器名称时验证是否存在重复名称存在返回true

{

using (Entities entities = new Entities())

{

var modelList = (from i in entities.Server where i.Name == condRule select i).ToList();

int num = modelList.Where(t => t.Name == condRule && t.ID == id).Count();

if (modelList.Count() > 0 && num != 1)

{

isExists = true;

}

}

}

else //添加服务器名称时验证是否存在重复名称存在返回true

{

using (Entities entities = new Entities())

{

int modelNum = (from i in entities.Server where i.Name == condRule select i).Count();

if (modelNum > 0)

isExists = true;

}

}

return isExists;

}

public int Update(ServerModel model)

{

int result = 0;

using (Entities entities = new Entities())

{

if (model.EditParaType == EditPara.Add)

{

Server xt = new Server();

xt.Name = model.Name;

xt.IPAddress = model.IPAddress;

xt.Description = model.Description;

entities.Set<Server>().Add(xt);

entities.SaveChanges();

result = 1;

}

else

{

var xt = entities.Set<Server>().Find(model.ID);

xt.Name = model.Name;

xt.IPAddress = model.IPAddress;

xt.Description = model.Description;

entities.SaveChanges();

result = 1;

}

}

return result;

}

public int Delete(decimal id)

{

int result = 0;

int masterCount = MasterStationDataAccessor.FindMasterStationByServerID(id);

if (masterCount == 0)

{

using (Entities entities = new Entities())

{

Server model = entities.Set<Server>().Find(id);

entities.Set<Server>().Remove(model);

entities.SaveChanges();

result = 1;

}

}

else

{

result = -10;

}

return result;

}

}

}

using HomePagePlugin.DataAccessores;

using HomePagePlugin.Models;

using HomePagePlugin.ViewModels;

using Newtonsoft.Json;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Web;

using System.Web.Mvc;

namespace HomePagePlugin.Controllers

{

public class HomeController : Controller

{

IndexAccessor accessor = new IndexAccessor();

public ActionResult Index(IndexViewModel homePageViewModel)

{

homePageViewModel = accessor.getHeatMeterCount();

if (homePageViewModel.MeterTotalCount != 0)

{

homePageViewModel.MeterSuccessRate = Math.Round(Convert.ToDecimal(homePageViewModel.MeterSuccessCount / homePageViewModel.MeterTotalCount \* 100), 2);

homePageViewModel.MeterFailRate = Math.Round(Convert.ToDecimal(homePageViewModel.MeterFailCount / homePageViewModel.MeterTotalCount \* 100), 2);

}

else

{

homePageViewModel.MeterSuccessRate = homePageViewModel.MeterFailRate = 0;

}

if (homePageViewModel.VavleTotalCount != 0)

{

homePageViewModel.VavleSuccessRate = Math.Round(Convert.ToDecimal(homePageViewModel.VavleSuccessCount / homePageViewModel.VavleTotalCount \* 100), 2);

homePageViewModel.VavleFailRate = Math.Round(Convert.ToDecimal(homePageViewModel.VavleFailCount / homePageViewModel.VavleTotalCount \* 100), 2);

}

else

{

homePageViewModel.VavleSuccessRate = homePageViewModel.VavleFailRate = 0;

}

if (homePageViewModel.CurrentPageNum == 0)

{

homePageViewModel.CurrentPageNum = 1;

}

if (homePageViewModel.PageSize == 0)

{

homePageViewModel.PageSize = IndexViewModel.DefaultPageSize;

}

int curPageNum = homePageViewModel.CurrentPageNum, count, pageCount;

homePageViewModel.tableDataList = accessor.GetPage(homePageViewModel.HeatingStationID,homePageViewModel.CommunityID, homePageViewModel.PageSize, ref curPageNum, out count, out pageCount);

homePageViewModel.DataCount = count;

homePageViewModel.PageCount = pageCount;

return View(homePageViewModel);

}

public string GetTableDateByCommunityID(decimal HeatingStationID,decimal CommunityID,int currentPage)

{

int count, pageCount;

var list = accessor.GetPage(HeatingStationID,CommunityID, 5, ref currentPage, out count, out pageCount);

var JosnObj = new JsonObject

{

TableList = list,

CurrentPageNum = currentPage,

PageCount = pageCount,

Count = count

};

return JsonConvert.SerializeObject(JosnObj);

}

public string RateTrendAnalyse(int days)

{

return accessor.getDignonalbatDayData(days);

}

public string getCommDignonalbatDayData(int days, decimal communityId, decimal HeatStationId)

{

return accessor.getComDignonalbatDayData(days, communityId,HeatStationId);

}

public string getHeatStationSelectList()

{

return accessor.getHeatstationSelectList();

}

public string getCommunitySelectList(decimal heatstationId)

{

return accessor.getCommunitySelectList(heatstationId);

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.Mvc;

using HistoryQueryPlugin.DataAccessors;

using System.Text;

using HistoryQueryPlugin.ViewModels;

using HistoryQueryPlugin.Models;

using UIShell.CommServerQueueServiceContract;

using System.IO;

using ThirdParties.ExportToExcel;

using HistoryQueryPlugin.Models;

using HistoryQueryPlugin.Helper;

namespace HistoryQueryPlugin.Controllers

{

public class HeatingMeterController:Controller

{

private HeatingMeterDataAccessor dataaccessors ;

private DropDownListDataAccessor ddlDataAccessor;

private UserDataAccessor userAccessor = new UserDataAccessor();

#region///抄表所需部分

private static int type;

private static int ticket;

public static decimal meterReadCount;

public static decimal meterReadTotal;

#endregion

public HeatingMeterController()

{

dataaccessors = new HeatingMeterDataAccessor();

ddlDataAccessor = new DropDownListDataAccessor();

}

public ActionResult HeatingMeter(HeatingMeterViewModel model, int? currentPageNum, int? pageSize)

{

model.HeatingStationList = dataaccessors.getHeatStationList();

if (!currentPageNum.HasValue)

{

currentPageNum = 1;

}

if (!pageSize.HasValue)

{

pageSize = HeatingMeterViewModel.DefaultPageSize;

}

int pageNum = currentPageNum.Value;

model.UserPagination = new UIShell.PartialViewShellPlugin.ViewModels.Pagination<UserModel> {

DataList = new List<UserModel>(),

DataCount = 0

};

model.pagination = new UIShell.PartialViewShellPlugin.ViewModels.Pagination<MeterDataModel> {

DataCount = 0,

DataList = new List<MeterDataModel>()

};

int SuccessCount = 0; int TotalCount = 0; int CommunitionFailCount = 0; int FailCount = 0; int EquipmentCount = 0;

if (model.Flag == 0)

{

model.pagination = dataaccessors.GetPage(model, pageSize.Value, ref pageNum, out TotalCount, out SuccessCount, out FailCount, out CommunitionFailCount, out EquipmentCount);

model.TotalCount = TotalCount;

model.SuccessCount = SuccessCount;

model.FailCount = FailCount;

model.CommunicationFailCount = CommunitionFailCount;

model.EquipmentCount = EquipmentCount;

model.Area = dataaccessors.GetTollArea(model);

if (string.IsNullOrEmpty(model.DanWei))

{

model.DanWei = "KWH";

}

}

else if (model.Flag == 1)

{

model.UserPagination= userAccessor.GetPage(model, pageSize.Value, ref pageNum, out TotalCount,out SuccessCount,out FailCount,out CommunitionFailCount);

model.VavleTotalCount = TotalCount;

model.VavleSuccessCount = SuccessCount;

model.VavleFailCount = FailCount;

model.VavleCommunicationCount = CommunitionFailCount;

model.Area = userAccessor.GetTollArea(model);

if (string.IsNullOrEmpty(model.DanWei))

{

model.DanWei = "KWH";

}

}

model.CurrentPageNum = pageNum;

return View(model);

}

#region ===== 数据导出

public void ExportToExcel(HeatingMeterViewModel model)

{

HttpContext context = System.Web.HttpContext.Current;

string timeTicks = DateTime.Now.Ticks.ToString();

string directory = Path.Combine(context.Server.MapPath("./"), "download");

Directory.CreateDirectory(directory);

string absolutedPath = Path.Combine(directory, timeTicks + ".xls");

List<Object> excel = dataaccessors.GetObjectListToExport(model);

CreateExcelFile.CreateExcelDocument(excel, ref absolutedPath);

if (absolutedPath.Contains("zip"))

{

context.Response.AddHeader("Content-Disposition", "attachment;filename=" + System.Web.HttpUtility.UrlEncode("数据列表.zip", System.Text.Encoding.UTF8));

context.Response.ContentType = "applicationnd.x-zip-compressed";

}

else

{

context.Response.AddHeader("Content-Disposition", "attachment;filename=" + System.Web.HttpUtility.UrlEncode("数据列表.xls", System.Text.Encoding.UTF8));

context.Response.ContentType = "applicationnd.ms-excel";

}

context.Response.WriteFile("../../HistoryQueryPlugin/HeatingMeter/download/" + Path.GetFileName(absolutedPath));

}

public void ExportDateToExcel(HeatingMeterViewModel model)

{

HttpContext context = System.Web.HttpContext.Current;

string timeTicks = DateTime.Now.Ticks.ToString();

string directory = Path.Combine(context.Server.MapPath("./"), "download");

Directory.CreateDirectory(directory);

string absolutedPath = Path.Combine(directory, timeTicks + ".xls");

List<Object> excel = userAccessor.GetObjectListToExport(model);

CreateExcelFile.CreateExcelDocument(excel, ref absolutedPath);

if (absolutedPath.Contains("zip"))

{

context.Response.AddHeader("Content-Disposition", "attachment;filename=" + System.Web.HttpUtility.UrlEncode("数据列表.zip", System.Text.Encoding.UTF8));

context.Response.ContentType = "applicationnd.x-zip-compressed";

}

else

{

context.Response.AddHeader("Content-Disposition", "attachment;filename=" + System.Web.HttpUtility.UrlEncode("数据列表.xls", System.Text.Encoding.UTF8));

context.Response.ContentType = "applicationnd.ms-excel";

}

context.Response.WriteFile("../../HistoryQueryPlugin/HeatingMeter/download/" + Path.GetFileName(absolutedPath));

}

#endregion

#region ==== 级联

public string getCommunitySelectList(decimal heatstationId)

{

return dataaccessors.getCommunitySelectList(heatstationId);

}

public string getBuildingAndMeterInfo(decimal communityId)

{

return dataaccessors.getBuildingAndMeterInfo(communityId);

}

public string getBuildingList(decimal communityId)

{

return dataaccessors.getBuildingInfo(communityId);

}

public string getUnitSelectList(decimal buildingId)

{

return dataaccessors.getUnitListInfo(buildingId);

}

public string getUserSelectList(decimal unitId)

{

return dataaccessors.getUserSelectListInfo(unitId);

}

#endregion

public string GetBaseHeatingMeterID(decimal concentrator)

{

string getUrl = default(string);

string returnID = dataaccessors.GetBaseHeatingMeterID(concentrator);

if (!string.IsNullOrEmpty(returnID))

{

getUrl = Url.Action("Index", "ChangeMeter", new { id = returnID });

}

return getUrl;

}

public string GetHeatSeasonMaintainInfo()

{

string count = dataaccessors.GetHeatSeasonInfo();

if (count == "")

{

return "fail";

}

else

{

return count;

}

}

#region

public JsonResult ForCopyHistoryDate(string timeStamp, decimal MeterID)

{

JsonResult jsonResult = new JsonResult();

var accessor = new ForCopyDataAccessor();

var ResultList = accessor.GetCopyDateParamByMeterID(MeterID);

meterReadCount = 0;

type = Msg.CreateType();

ticket = Msg.CreatTicket();

ProgressManager.AddMsg(new Msg

{

Ticket = ticket,

MsgType = type,

MsgParam = meterReadCount,

CompleteRate = 0,

Description = "正在获取处理消息，请稍候..."

});

foreach (var item in ResultList)

{

var ServerID = Convert.ToInt32(item.ServerID);

var MasterStationID = Convert.ToInt32(item.MasterStationID);

var EquipmentID = Convert.ToInt32(item.EquipmentID);

BundleActivator.ObjectQueueService.Publish<WEBReadHistorialMetersByConcentratorMessage>(new WEBReadHistorialMetersByConcentratorMessage() { Id = ticket, TimeStamp = DateTime.Parse(timeStamp), IsTmp = false, ServerId = ServerID, MasterStationId = MasterStationID, ConcentratorId = EquipmentID });

meterReadCount++;

}

List<SelectListItem> optList = new List<SelectListItem>();

jsonResult = new JsonResult { Data = optList, JsonRequestBehavior = JsonRequestBehavior.AllowGet };

return jsonResult;

}

public JsonResult ForCopyHistoryDateByButton(HeatingMeterViewModel model)

{

JsonResult jsonResult = new JsonResult();

var accessor = new ForCopyDataAccessor();

var ResultList = accessor.GetCopyDateByModel(model);

meterReadCount = 0;

type = Msg.CreateType();

ticket = Msg.CreatTicket();

ProgressManager.AddMsg(new Msg

{

Ticket = ticket,

MsgType = type,

MsgParam = meterReadCount,

CompleteRate = 0,

Description = "正在获取处理消息，请稍候..."

});

foreach (var item in ResultList)

{

var ServerID = Convert.ToInt32(item.ServerID);

var MasterStationID = Convert.ToInt32(item.MasterStationID);

var EquipmentID = Convert.ToInt32(item.EquipmentID);

BundleActivator.ObjectQueueService.Publish<WEBReadHistorialMetersByConcentratorMessage>(new WEBReadHistorialMetersByConcentratorMessage() { Id = ticket, TimeStamp = DateTime.Parse(model.CopyDateTime), IsTmp = false, ServerId = ServerID, MasterStationId = MasterStationID, ConcentratorId = EquipmentID });

meterReadCount++;

}

List<SelectListItem> optList = new List<SelectListItem>();

jsonResult = new JsonResult { Data = optList, JsonRequestBehavior = JsonRequestBehavior.AllowGet };

return jsonResult;

}

public JsonResult StatisticsSchedule(decimal MeterID)

{

var accessor = new ForCopyDataAccessor();

JsonResult jsonResult = new JsonResult();

var parameterList = accessor.GetCopyDateParamByMeterID(MeterID);

meterReadTotal = parameterList.Count;

Msg msg = ProgressManager.GetMsg(ticket);

decimal cnt = meterReadTotal;

if (msg != null)

cnt = int.Parse(msg.MsgParam.ToString());

ForCopyDataScheduleModel meterModel = new ForCopyDataScheduleModel()

{

MeterCount = meterReadTotal - cnt,

MeterTotal = meterReadTotal,

};

jsonResult = new JsonResult { Data = meterModel, JsonRequestBehavior = JsonRequestBehavior.AllowGet };

return jsonResult;

}

public JsonResult StatisticsScheduleClick(HeatingMeterViewModel model)

{

var accessor = new ForCopyDataAccessor();

JsonResult jsonResult = new JsonResult();

var parameterList = accessor.GetCopyDateByModel(model);

meterReadTotal = parameterList.Count;

Msg msg = ProgressManager.GetMsg(ticket);

decimal cnt = meterReadTotal;

if (msg != null)

cnt = int.Parse(msg.MsgParam.ToString());

ForCopyDataScheduleModel meterModel = new ForCopyDataScheduleModel()

{

MeterCount = meterReadTotal - cnt,

MeterTotal = meterReadTotal,

};

jsonResult = new JsonResult { Data = meterModel, JsonRequestBehavior = JsonRequestBehavior.AllowGet };

return jsonResult;

}

public JsonResult GetCopyDataTime(decimal HeatingStationID, decimal CommunityID, decimal MeterID)

{

var accessor = new ForCopyDataAccessor();

List<SelectListItem> list = accessor.GetCopyForDateTime(HeatingStationID, CommunityID, MeterID);

JsonResult jsonResult = new JsonResult { Data = list, JsonRequestBehavior = JsonRequestBehavior.AllowGet };

return jsonResult;

}

#endregion

}

}

using HistoryQueryPlugin.DataAccessors;

using HistoryQueryPlugin.Helper;

using HistoryQueryPlugin.Models;

using HistoryQueryPlugin.ViewModels;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.Mvc;

using UIShell.CommServerQueueServiceContract;

namespace HistoryQueryPlugin.Controllers

{

public class MeterChangeController : Controller

{

private static MeterChangeDataAccessor accssor = new MeterChangeDataAccessor();

private static int ticket;

private static decimal ChangeMeterID;

public ActionResult Index(decimal MeterID)

{

var item= accssor.GetHeatingMeterInfoByMeterID(MeterID);

var HeatingStationList = new List<SelectListItem>();

HeatingStationList.Add(new SelectListItem{Text = item.HeatingStationName,Value=item.HeatingStationID.ToString()});

var CommunityList = new List<SelectListItem>();

CommunityList.Add(new SelectListItem{Text = item.CommunityName,Value = item.CommunityID.ToString()});

var EquipmentList = new List<SelectListItem>();

EquipmentList.Add(new SelectListItem{Text = item.EquipmentNO.ToString(),Value = item.EquipmentID.ToString()});

var model = new MeterChangeViewModel

{

HeatingMeterID = item.HeatingMeterID,

MeterInfoList = accssor.GetHeatingMeterInfoList(),

HeatingMeterSerialNO = item.MeterNo,

HeatingMeterHardAddress = item.MeterHardAddress,

HeatingMeterTunnelNO = item.MeterChannelNo,

HeatingStationID = item.HeatingStationID,

HeatingStationList = HeatingStationList,

CommunityID = item.CommunityID,

CommunityList = CommunityList,

EquipmentID = item.EquipmentID,

EquipmentList = EquipmentList,

InstallTime = item.InstallTime.Date.ToString("yyyy-MM-dd"),

InstallPosition = item.InstallPosition,

UserNO = item.UserNo,

CalPoint = item.Point,

ExecutionSituationType = item.Situation.Substring(0,2),

SituationTypeList = GetSituationTypeList(),

HeatingMeterNo = item.Situation.Substring(2,2),

HeatingMeterNOList = GetHeatingMeterNoList(item.Situation.Substring(0,2))

};

return View("Index",model);

}

private List<SelectListItem> GetSituationTypeList()

{

return new List<SelectListItem>(){

new SelectListItem(){Text="一楼一表",Value="00",Selected=true},

new SelectListItem(){Text="商户",Value="01",Selected=false},

new SelectListItem(){Text="高低层",Value="02",Selected=false},

new SelectListItem(){Text="多楼一表",Value="03",Selected=false}};

}

private List<SelectListItem> GetHeatingMeterNoList(string SituationType)

{

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

if (SituationType == "01")

{

list.Add(new SelectListItem { Text = "总表", Value = "01" });

list.Add(new SelectListItem { Text = "商户表", Value = "02" });

}

if (SituationType == "02")

{

list.Add(new SelectListItem { Text = "高层", Value = "01" });

list.Add(new SelectListItem { Text = "低层", Value = "02" });

}

return list;

}

public JsonResult MeterChangeInfo(MeterChangeViewModel model)

{

ChangeMeterID = accssor.InsertMeterChangeInfo(model);

accssor.UpdateBaseHeatingMeterInfo(model);

return IssuedMeterAddr(model.EquipmentID);

}

public string GetMeterNo(MeterChangeViewModel model)

{

return accssor.GetHeatingMeterNo(model);

}

public string GetMeterHardAddress(MeterChangeViewModel model)

{

return accssor.GetHeatingMeterHardAddress(model);

}

public JsonResult IssuedMeterAddr(decimal eqid)

{

bool isSuccess = true;

string msgStr = "";

var Param = accssor.GetChangeMeterParam(eqid);

ticket = Msg.CreatTicket();

var type = Msg.CreateType();

if (Param == null)

{

isSuccess = false;

msgStr = "未找到您需要管理的集中器！";

}

else

{

try

{

ProgressManager.AddMsg(new Msg

{

Ticket = ticket,

MsgType = type,

MsgParam=1,

CompleteRate = 0,

Description = "正在获取处理消息，请稍候..."

});

BundleActivator.ObjectQueueService.Publish<WEBSingleTransmitMeterAddressesMessage>(new WEBSingleTransmitMeterAddressesMessage()

{

Id = ticket,

Type=1,

ServerId = Convert.ToInt32(Param.ServerID),

MasterStationId = Convert.ToInt32(Param.MasterStationID),

ConcentratorId = Convert.ToInt32(eqid)

});

isSuccess = true;

msgStr = "已完成发送命令，请稍候...";

}

catch

{

isSuccess = false;

msgStr = "下发失败，服务端未响应！~";

}

}

return new JsonResult

{

Data = new

{

success = isSuccess,

msg = msgStr,

token = ticket

},

JsonRequestBehavior = JsonRequestBehavior.AllowGet

};

}

public JsonResult GetIssuedMeterAddrStatus(decimal tryCount)

{

var jsonResult = new JsonResult();

Msg message = ProgressManager.GetMsg(ticket);

if (message != null)

{

jsonResult.Data = new

{

token = ticket,

completeRate = message.CompleteRate,

description = message.Description

};

if (message.CompleteRate == 100 || tryCount == -1)

{

ProgressManager.RemoveMsg(ticket);

}

}

else

{//未获取到消息

jsonResult.Data = new

{

token = ticket,

completeRate = -1,

description = "下发表地址失败，请重新下发！"

};

}

return new JsonResult

{

Data = jsonResult.Data,

JsonRequestBehavior = JsonRequestBehavior.AllowGet

};

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using UIShell.OSGi;

using UIShell.PageFlowService;

using UIShell.DynamicEnumBuilderService;

using UIShell.CommServerQueueServiceContract;

using ServerResultInterface.DataAccessors;

using HistoryQueryPlugin.DataAccessors;

namespace HistoryQueryPlugin

{

public class BundleActivator : IBundleActivator

{

public static IBundle Bundle { get; private set; }

public static string LayoutPage { get; private set; }

public static IBundle CommonBundle { get; private set; }//树形菜单接口

public static IBundle AnalyseDateBundle { get; private set; }

public static IDynamicEnumService DynamicEnumService { get; private set; }

public static IObjectQueueService ObjectQueueService { get; private set; }

public void Start(IBundleContext context)

{

ObjectQueueService = context.GetFirstOrDefaultService<IObjectQueueService>();

Bundle = context.Bundle;

CommonBundle = context.GetBundleBySymbolicName("CommonPlugin");//树形菜单

AnalyseDateBundle = context.GetBundleBySymbolicName("AnalyseHistoryDatePlugin");

var pageFlowService = context.GetFirstOrDefaultService<IPageFlowService>();

LayoutPage = pageFlowService.GetPageNode("LayoutPage").Value;

DynamicEnumService = context.GetFirstOrDefaultService<IDynamicEnumService>();

context.AddService<IHistoryDateCopyServerResult>(new ForCopyDateImpl());

}

public void Stop(IBundleContext context)

{

}

}

}

using System;

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

using System.Linq;

using System.Web;

namespace HistoryQueryPlugin

{

public class EnumExt

{

public enum DataStatus

{

[Display(Name = "全部")]

All = 0,

[Display(Name = "成功")]

Success = 1,

[Display(Name = "失败")]

Failed = 2,

[Display(Name = "通讯失败")]

CommunicationFailed = 3

}

}

}

using AlarmInfoPlugin.DataAccessors;

using AlarmInfoPlugin.Models;

using AlarmInfoPlugin.ViewModels;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.Mvc;

namespace AlarmInfoPlugin.Controllers

{

public class ValveAlarmController : Controller

{

public ValveAlarmDataAccessor alarmDB = new ValveAlarmDataAccessor();

public ActionResult Query(QueryCondition queryCondition)

{

List<ValveAlarmSummaryModel> dataList = alarmDB.GetAlarmDataList(queryCondition);

总数

var userSummary = alarmDB.GetUserSummary();

var valveSummary = userSummary;

var softwareAlarmSummary = dataList.Sum(i => i.SoftwareAlarmSummary);

var hardwareAlarmSummary = dataList.Sum(i => i.HardwareAlarmSummary);

var communicationAlarmSummary = dataList.Sum(i => i.CommunicationAlarmSummary);

var recentTime = DateTime.Now;

if (dataList.Count > 0)

{

recentTime = dataList.FirstOrDefault().RecentTime ?? DateTime.Now;

}

var normalSummary = valveSummary - softwareAlarmSummary - hardwareAlarmSummary - communicationAlarmSummary;

var normalPer = valveSummary == 0 ? 0 : Math.Round((normalSummary / valveSummary) \* 100, 2);

var communicationAlarmPer = valveSummary == 0 ? 0 : Math.Round((communicationAlarmSummary / valveSummary) \* 100, 2);

var hardwareAlarmPer = valveSummary == 0 ? 0 : Math.Round((hardwareAlarmSummary / valveSummary) \* 100, 2);

var softwareAlarmPer = valveSummary == 0 ? 0 : Math.Round((softwareAlarmSummary / valveSummary) \* 100, 2);

ValveAlarmViewModel viewModel = new ValveAlarmViewModel()

{

QueryCondition = queryCondition,

AlarmDataList = dataList,

UserSummary = userSummary,

ValveSummary = valveSummary,

NormalSummary = normalSummary,

SoftwareAlarmSummary = softwareAlarmSummary,

HardwareAlarmSummary = hardwareAlarmSummary,

CommunicationAlarmSummary = communicationAlarmSummary,

NormalPercentage = normalPer,

CommunicationAlarmPercentage = communicationAlarmPer,

HardwareAlarmPercentage = hardwareAlarmPer,

SoftwareAlarmPercentage = softwareAlarmPer,

RecentTime = recentTime

};

return View(viewModel);

}

public JsonResult GetCascadeJson(decimal idPrev, decimal idSel, string flag)

{

CascadeDataAccessor cascadeDB = new CascadeDataAccessor();

JsonResult jsonResult = new JsonResult();

List<SelectListItem> optList = cascadeDB.GetCascadeList(idPrev, idSel, flag);

jsonResult = new JsonResult { Data = optList, JsonRequestBehavior = JsonRequestBehavior.AllowGet };

return jsonResult;

}

public JsonResult GetCommunityAlarmChart(decimal communityID, decimal timeSection)

{

QueryCondition queryCondition = new QueryCondition()

{

CommunityID = communityID,

TimeSection = timeSection

};

List<ValveAlarmSummaryModel> chartData = new List<ValveAlarmSummaryModel>();

chartData = alarmDB.GetCommunityAlarmChartDataList(queryCondition);

return new JsonResult { Data = chartData, JsonRequestBehavior = JsonRequestBehavior.AllowGet };

}

#region 刷新小区报警信息列表

public ActionResult GetAlarmCommunityTable(decimal? heatingStationID,decimal? communityID, int? currentPageNum, int? pageSize)

{

QueryCondition query = new QueryCondition();

query.HeatingStationID = Convert.ToDecimal(heatingStationID);

query.CommunityID = Convert.ToDecimal(communityID);

var pagination = alarmDB.GetDataPagination(query, currentPageNum, pageSize);

ValveAlarmTableViewModel viewModel = new ValveAlarmTableViewModel()

{

Pagination = pagination

};

return View(viewModel);

}

#endregion

}

}