MVC CRUD

Controller

* List

public ActionResult GetPageSettingDetail()

{

try

{

string pid = "PageSettingId";

List<IEnumerable<dynamic>> listData = new List<IEnumerable<dynamic>>();

var data = \_ClientServiceRepository.GetList("usp\_Admin\_GetPageSettingList");

var dataList = data.ToList();

data = \_CommonRepository.EncryptDataId((IEnumerable<dynamic>)dataList, pid);

var list = JsonConvert.SerializeObject(data,

Formatting.None,

new JsonSerializerSettings()

{

ReferenceLoopHandling = ReferenceLoopHandling.Ignore

});

return Content(list, "application/json");

}

catch (Exception ex)

{

}

return Content("", "application/json");

}

* Create

public ActionResult CreatePageSetting(FormCollection collection)

{

string message = "";

TempData.Remove("Message");

try

{

string formXml = PageSettingXml.SetPageSettingXml(collection);

\_message = \_ClientServiceRepository.AddDetails("usp\_Admin\_PageSettingAdd", formXml, UserDetails.GetUserId());

TempData["Message"] = \_message.MessageText;

TempData["MessageId"] = \_message.MessageStatusId;

}

catch (SqlException ex)

{

message = ex.Message;

\_message = new Message { MessageStatusId = MessageDetail.ErrorMessage, MessageText = message };

}

var list = JsonConvert.SerializeObject(\_message, Formatting.None, new JsonSerializerSettings() { ReferenceLoopHandling = ReferenceLoopHandling.Ignore });

return Content(list, "application/json");

}

* Edit

public ActionResult EditPageSetting(string id, FormCollection collection)

{

string message = "";

TempData.Remove("Message");

try

{

int PrimaryId = Convert.ToInt32(\_CommonRepository.DecryptDataId(id));

if (PrimaryId > 0)

{

string formXml = PageSettingXml.SetPageSettingXml(collection);

\_message = \_ClientServiceRepository.UpdateDetails("usp\_Admin\_PageSettingUpdate", formXml, PrimaryId,UserDetails.GetUserId());

TempData["Message"] = \_message.MessageText;

TempData["MessageId"] = \_message.MessageStatusId;

}

// return RedirectToAction("Index");

}

catch (SqlException ex)

{

message = ex.Message;

\_message = new Message { MessageStatusId = MessageDetail.ErrorMessage, MessageText = message };

}

var list = JsonConvert.SerializeObject(\_message, Formatting.None, new JsonSerializerSettings() { ReferenceLoopHandling = ReferenceLoopHandling.Ignore });

return Content(list, "application/json");

}

* Select By Id

public ActionResult SelectByPrimary(string id)

{

try

{

int PrimaryId = Convert.ToInt32(\_CommonRepository.DecryptDataId(id));

if (PrimaryId > 0)

{

var data = \_ClientServiceRepository.SelectByPrimary("usp\_Admin\_PageSettingSBP", PrimaryId);

var list = JsonConvert.SerializeObject(data, Formatting.None, new JsonSerializerSettings() { ReferenceLoopHandling = ReferenceLoopHandling.Ignore });

return Content(list, "application/json");

}

}

catch (Exception ex)

{

}

return Content("", "application/json");

}

* DELETE

public ActionResult DeletePageSetting(string id)

{

string message = "";

TempData.Remove("Message");

try

{

int PrimaryId = Convert.ToInt32(\_CommonRepository.DecryptDataId(id));

if (PrimaryId > 0)

{

\_message = \_ClientServiceRepository.DeleteDetails("usp\_Admin\_PageSettingDelete", PrimaryId,UserDetails.GetUserId());

TempData["Message"] = \_message.MessageText;

TempData["MessageId"] = \_message.MessageStatusId;

}

}

catch (SqlException ex)

{

message = ex.Message;

\_message = new Message { MessageStatusId = MessageDetail.ErrorMessage, MessageText = message };

}

var list = JsonConvert.SerializeObject(\_message, Formatting.None, new JsonSerializerSettings() { ReferenceLoopHandling = ReferenceLoopHandling.Ignore });

return Content(list, "application/json");

}

* Requirement

public static class PageSettingXml

{

public static string SetPageSettingXml(FormCollection form)

{

string xmlString = string.Empty;

StringBuilder strBuilder = new StringBuilder();

strBuilder.Append("<formXml>");

strBuilder.Append("<MerchantID>" + ToReplaceSpecialChar.ReplaceSpecialChar(form["MerchantID"]) + "</MerchantID>");

strBuilder.Append("</formXml>");

xmlString = strBuilder.ToString();

return xmlString;

}

}

public IEnumerable<dynamic> EncryptDataId(IEnumerable<dynamic> dataList, string PId)

{

string strValue = string.Empty;

foreach (var item in dataList)

{

var itemDetails = ((IDictionary<string, object>)item);

strValue = itemDetails[PId].ToString();

itemDetails[PId] = Encrypt(strValue);

}

return dataList;

}

public string DecryptDataId(string DataId)

{

string strDataId = "0";

try

{

strDataId = Decrypt(DataId);

}

catch (System.Security.Cryptography.CryptographicException ex)

{

strDataId = "-1";

}

catch (Exception ex)

{

}

return strDataId;

}

public static class MessageDetail

{

public static int SuccessMessage = 1;

public static int ErrorMessage = 2;

}

* Repository

public Message AddDetails(string ProcName, string formXml,string UserId)

{

var dynParam = new DynamicParameters();

dynParam.Add("formXml", formXml);

dynParam.Add("UserId", UserId);

dynParam.Add("Message", dbType: DbType.String, direction: ParameterDirection.Output, size: 250);

dynParam.Add("MessageId", dbType: DbType.String, direction: ParameterDirection.Output, size: 250);

Connection.Execute(ProcName, dynParam, commandType: CommandType.StoredProcedure);

\_Message.MessageText = dynParam.Get<string>("Message");

string MessageId = dynParam.Get<string>("MessageId");

\_Message.MessageStatusId = Convert.ToInt16(MessageId);

return \_Message;

}

* UPDATE

public Message UpdateDetails(string ProcName, string formXml,int PrimaryId,string UserId)

{

var dynParam = new DynamicParameters();

dynParam.Add("formXml", formXml);

dynParam.Add("PrimaryId", PrimaryId);

dynParam.Add("UserId", UserId);

dynParam.Add("Message", dbType: DbType.String, direction: ParameterDirection.Output, size: 250);

dynParam.Add("MessageId", dbType: DbType.String, direction: ParameterDirection.Output, size: 250);

Connection.Execute(ProcName, dynParam, commandType: CommandType.StoredProcedure);

\_Message.MessageText= dynParam.Get<string>("Message");

string MessageId = dynParam.Get<string>("MessageId");

\_Message.MessageStatusId = Convert.ToInt16(MessageId);

return \_Message;

}

* DELETE

public Message DeleteDetails(string ProcName, int PrimaryId,string UserId)

{

var dynParam = new DynamicParameters();

dynParam.Add("PrimaryId", PrimaryId);

dynParam.Add("UserId", UserId);

dynParam.Add("Message", dbType: DbType.String, direction: ParameterDirection.Output, size: 250);

dynParam.Add("MessageId", dbType: DbType.String, direction: ParameterDirection.Output, size: 250);

Connection.Execute(ProcName, dynParam, commandType: CommandType.StoredProcedure);

\_Message.MessageText = dynParam.Get<string>("Message");

string MessageId = dynParam.Get<string>("MessageId");

\_Message.MessageStatusId = Convert.ToInt16(MessageId);

return \_Message;

}

* LIST

public IEnumerable<dynamic> GetList(string ProcName)

{

return Connection.Query(ProcName, commandType: CommandType.StoredProcedure);

}