Usercontroller

using ClassLibrary.WebModels;

using EntitiesLibrary.Data;

using Newtonsoft.Json;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.Mvc;

using System.Web.Script.Serialization;

using System.Web.Security;

namespace AirbncleanWeb.Controllers

{

public class UserController : UserBaseController

{

/// <summary>

///

/// </summary>

/// <returns></returns>

[HttpGet]

public ActionResult home()

{

return View();

}

/// <summary>

///

/// </summary>

/// <returns></returns>

[HttpGet]

public ActionResult logout()

{

Session.Clear();

Session.RemoveAll();

Session.Abandon();

FormsAuthentication.SignOut();

return RedirectToAction("home", "Account", new { area = "" });

}

/// <summary>

///

/// </summary>

/// <returns></returns>

[HttpGet]

public ActionResult postjob()

{

return View();

}

/// <summary>

///

/// </summary>

/// <param name="id"></param>

/// <returns></returns>

[HttpGet]

public object getPropertyByID(string id)

{

bool status = false;

string message = "Failed";

long propertyId = Convert.ToInt64(id);

try

{

var row = \_entities.tUserProperties.Where(z => z.Id == propertyId).ToList().Select(z => new UserProperty(z)).FirstOrDefault();

if (row != null)

{

status = true;

message = "Success";

var jsonData = new JavaScriptSerializer().Serialize(row);

return Json(new { Status = status, Message = message, Data = jsonData }, JsonRequestBehavior.AllowGet);

}

else

{

message = "No data found";

return Json(new { Status = status, Message = message }, JsonRequestBehavior.AllowGet);

}

}

catch (Exception ex)

{

return new { Status = status, Message = ex.Message };

}

}

/// <summary>

///

/// </summary>

/// <param name="model"></param>

/// <returns></returns>

[HttpPost]

public object postjob(PostJobModel model)

{

bool status = false;

string message = "Failed";

try

{

Tuple<bool, string> t = \_repository.postjob(model);

status = t.Item1;

message = t.Item2;

if (status)

{

return Json(new { Status = status, Message = message }, JsonRequestBehavior.AllowGet);

}

else

return Json(new { Status = status, Message = message }, JsonRequestBehavior.AllowGet);

}

catch (Exception ex)

{

return new { Status = status, Message = ex.Message };

}

}

/// <summary>

///

/// </summary>

/// <returns></returns>

[HttpGet]

public ActionResult addproperty()

{

return View();

}

/// <summary>

///

/// </summary>

/// <param name="model"></param>

/// <returns></returns>

[HttpPost]

public object addproperty(AddUserPropertyModel model)

{

bool status = false;

string message = "Failed";

try

{

RootObject1 bsObj = JsonConvert.DeserializeObject<RootObject1>(model.PropertyImagesString);

model.PropertyImagesList = bsObj.PropertyImagesList;

Tuple<bool, string, string> t = \_repository.addUserProperty(model);

status = t.Item1;

message = t.Item2;

if (status)

{

return Json(new { Status = status, Message = message }, JsonRequestBehavior.AllowGet);

}

else

return Json(new { Status = status, Message = message }, JsonRequestBehavior.AllowGet);

}

catch (Exception ex)

{

return new { Status = status, Message = ex.Message };

}

}

/// <summary>

///

/// </summary>

/// <returns></returns>

[HttpGet]

public ActionResult myproperty()

{

var model = new UserMyPropertiesModel();

model.index = 0;

return View(model);

}

/// <summary>

///

/// </summary>

/// <param name="model"></param>

/// <returns></returns>

[HttpPost]

public object removePropertyFromList(UserDeletePropertyFromList model)

{

bool status = false;

string message = "Failed";

try

{

Tuple<bool, string> t = \_repository.removePropertyFromList(model);

status = t.Item1;

message = t.Item2;

if (status)

{

return Json(new { Status = status, Message = message }, JsonRequestBehavior.AllowGet);

}

else

return Json(new { Status = status, Message = message }, JsonRequestBehavior.AllowGet);

}

catch (Exception ex)

{

return new { Status = status, Message = ex.Message };

}

}

/// <summary>

///

/// </summary>

/// <param name="model"></param>

/// <returns></returns>

public PartialViewResult RefreshViewPropertiesList(UserMyPropertiesModel model)

{

return PartialView("~/Views/User/\_pv\_User\_Myproperties\_list.cshtml", model);

}

public PartialViewResult RefreshMyJobsListPagination(UserMyJobsPaginationModel model)

{

return PartialView("~/Views/User/\_pv\_UserMyJobsPagination.cshtml", model);

}

public PartialViewResult RefreshViewPropertiesListPagination(UserMyPropertiesPaginationModel model)

{

return PartialView("~/Views/User/\_pv\_UserMypropertiesPagination.cshtml", model);

}

/// <summary>

///

/// </summary>

/// <param name="id"></param>

/// <returns></returns>

[HttpGet]

public ActionResult viewproperty(string id)

{

var model = new UserViewProperty();

model.propertyId = Convert.ToInt64(id);

return View(model);

}

/// <summary>

///

/// </summary>

/// <param name="id"></param>

/// <returns></returns>

[HttpGet]

public ActionResult editproperty(string id)

{

var model = new EditUserPropertyModel();

model.PropertyId = id;

model.UserId = Convert.ToString(\_AccountDetails == null ? 0 : \_AccountDetails.Id);

return View(model);

}

/// <summary>

///

/// </summary>

/// <param name="model"></param>

/// <returns></returns>

[HttpPost]

public object editproperty(EditUserPropertyModel model)

{

bool status = false;

string message = "Failed";

try

{

RootObject1 bsObj = JsonConvert.DeserializeObject<RootObject1>(model.PropertyImagesString);

model.PropertyImagesList = bsObj.PropertyImagesList;

Tuple<bool, string, string> t = \_repository.editproperty(model);

status = t.Item1;

message = t.Item2;

if (status)

{

return Json(new { Status = status, Message = message }, JsonRequestBehavior.AllowGet);

}

else

return Json(new { Status = status, Message = message }, JsonRequestBehavior.AllowGet);

}

catch (Exception ex)

{

return new { Status = status, Message = ex.Message };

}

}

[HttpGet]

public ActionResult editjob(string id)

{

var model = new EditUserJobModel();

model.JobId = id;

model.UserId = Convert.ToString(\_AccountDetails == null ? 0 : \_AccountDetails.Id);

return View(model);

}

[HttpPost]

public object editjob(EditUserJobModel model)

{

bool status = false;

string message = "Failed";

try

{

Tuple<bool, string> t = \_repository.editjob(model);

status = t.Item1;

message = t.Item2;

if (status)

{

return Json(new { Status = status, Message = message, jobid = model.JobId }, JsonRequestBehavior.AllowGet);

}

else

return Json(new { Status = status, Message = message }, JsonRequestBehavior.AllowGet);

}

catch (Exception ex)

{

return new { Status = status, Message = ex.Message };

}

}

/// <summary>

///

/// </summary>

/// <param name="id"></param>

[HttpGet]

public void Refreshtimezone(string id)

{

EntitiesLibrary.Data.BaseReference obj = new EntitiesLibrary.Data.BaseReference(id);

}

/// <summary>

///

/// </summary>

/// <returns></returns>

[HttpGet]

public ActionResult myjobs()

{

var model = new UserMyJobsModel();

model.userId = \_AccountDetails == null ? 0 : \_AccountDetails.Id;

model.index = 0;

//model.JobStatus = ClassLibrary.Enum.JobStatus.Pending;

model.JobStatus = ClassLibrary.Enum.JobStatus.fetchAll;

model.UserProperties = new List<UserPorpertyModel>();

var properties = \_entities.tUserProperties.Where(z => z.UserId == model.userId && z.IsActive && z.IsDynamic == false).ToList();

if (properties.Count > 0)

{

foreach (var item in properties)

{

UserPorpertyModel UserProperty = new UserPorpertyModel();

UserProperty.propertyId = item.Id;

UserProperty.propertyName = item.PropertyName ?? string.Empty;

model.UserProperties.Add(UserProperty);

}

}

model.propertyId = 0;

model.workType = ClassLibrary.Enum.WorkType.All;

return View(model);

}

/// <summary>

///

/// </summary>

/// <param name="model"></param>

/// <returns></returns>

public PartialViewResult RefreshMyJobsList(UserMyJobsModel model)

{

//model.UserProperties = new List<UserPorpertyModel>();

//var properties = \_entities.tUserProperties.Where(z => z.UserId == model.userId && z.IsActive && z.IsDynamic == false).ToList();

//if (properties.Count > 0)

//{

// foreach (var item in properties)

// {

// UserPorpertyModel UserProperty = new UserPorpertyModel();

// UserProperty.propertyId = item.Id;

// UserProperty.propertyName = item.PropertyName ?? string.Empty;

// model.UserProperties.Add(UserProperty);

// }

//}

//model.propertyId = 0;

return PartialView("~/Views/User/\_pv\_User\_Myjobs\_list.cshtml", model);

}

[HttpGet]

public ActionResult viewjob(string id)

{

var model = new UserViewJobModel();

model.userId = \_AccountDetails == null ? 0 : \_AccountDetails.Id;

model.jobId = Convert.ToInt64(id);

return View(model);

}

[HttpGet]

public object removeJob(string id)

{

bool status = false;

string message = "Failed";

try

{

Tuple<bool, string> t = \_repository.removeJob(id);

status = t.Item1;

message = t.Item2;

if (status)

{

return Json(new { Status = status, Message = message }, JsonRequestBehavior.AllowGet);

}

else

return Json(new { Status = status, Message = message }, JsonRequestBehavior.AllowGet);

}

catch (Exception ex)

{

return new { Status = status, Message = ex.Message };

}

}

[HttpGet]

public ActionResult counteroffer(string id)

{

var model = new UserJobCounterOfferModel();

model.userId = \_AccountDetails == null ? 0 : \_AccountDetails.Id;

model.index = 0;

model.jobId = Convert.ToInt64(id);

return View(model);

}

[HttpPost]

public object userDeleteCounterOffer(UserDeleteCounterOffer model)

{

bool status = false;

string message = "Failed";

try

{

Tuple<bool, string> t = \_repository.userDeleteCounterOffer(model);

status = t.Item1;

message = t.Item2;

if (status)

{

return Json(new { Status = status, Message = message }, JsonRequestBehavior.AllowGet);

}

else

return Json(new { Status = status, Message = message }, JsonRequestBehavior.AllowGet);

}

catch (Exception ex)

{

return new { Status = status, Message = ex.Message };

}

}

public PartialViewResult RefreshCounterOfferList(UserJobCounterOfferModel model)

{

return PartialView("~/Views/User/\_pv\_User\_Counteroffer\_list.cshtml", model);

}

public object checkCounterOfferListCount(UserJobCounterOfferModel model)

{

bool status = false;

string message = "Failed";

try

{

Tuple<bool, string, long> t = \_repository.checkCounterOfferListCount(model);

status = t.Item1;

message = t.Item2;

if (status)

{

return Json(new { Status = status, Message = message, Count = t.Item3 }, JsonRequestBehavior.AllowGet);

}

else

return Json(new { Status = status, Message = message }, JsonRequestBehavior.AllowGet);

}

catch (Exception ex)

{

return new { Status = status, Message = ex.Message };

}

}

public PartialViewResult RefreshCounterOfferListPagination(UserJobCounterOfferPaginationModel model)

{

return PartialView("~/Views/User/\_pv\_UserJob\_CounterOfferPagination.cshtml", model);

}

[HttpGet]

public ActionResult counterofferdetail(string id, string jobid)

{

var model = new CounterOfferDetailModel();

model.userId = \_AccountDetails == null ? 0 : \_AccountDetails.Id;

model.counterOfferId = Convert.ToInt64(id);

model.jobId = Convert.ToInt64(jobid);

return View(model);

}

[HttpPost]

public object acceptCounterOffer(AcceptCounterOffer model)

{

bool status = false;

string message = "Failed";

try

{

Tuple<bool, string> t = \_repository.acceptCounterOffer(model);

status = t.Item1;

message = t.Item2;

if (status)

{

return Json(new { Status = status, Message = message }, JsonRequestBehavior.AllowGet);

}

else

return Json(new { Status = status, Message = message }, JsonRequestBehavior.AllowGet);

}

catch (Exception ex)

{

return new { Status = status, Message = ex.Message };

}

}

[HttpPost]

public object submitjobfeedback(SubmitJobFeedbackModel model)

{

bool status = false;

string message = "Failed";

try

{

Tuple<bool, string> t = \_repository.submitjobfeedback(model);

status = t.Item1;

message = t.Item2;

if (status)

{

return Json(new { Status = status, Message = message }, JsonRequestBehavior.AllowGet);

}

else

return Json(new { Status = status, Message = message }, JsonRequestBehavior.AllowGet);

}

catch (Exception ex)

{

return new { Status = status, Message = ex.Message };

}

}

[HttpGet]

public ActionResult settings()

{

var model = new SettingsModel();

model.Id = \_AccountDetails.Id;

model.Status = Convert.ToString(\_AccountDetails.PushStatus);

return View(model);

}

[HttpPost]

public object RefreshPushStatus(SettingsModel model)

{

bool status = false;

string message = "Failed";

try

{

Tuple<bool, string> t = \_repository.RefreshPushStatus(model);

status = t.Item1;

message = t.Item2;

if (status)

{

return Json(new { Status = status, Message = message }, JsonRequestBehavior.AllowGet);

}

else

return Json(new { Status = status, Message = message }, JsonRequestBehavior.AllowGet);

}

catch (Exception ex)

{

return new { Status = status, Message = ex.Message };

}

}

[HttpPost]

public object changePassword(ChangePasswordModel model)

{

bool status = false;

string message = "Failed";

try

{

Tuple<bool, string> t = \_repository.changePassword(model);

status = t.Item1;

message = t.Item2;

if (status)

return Json(new { Status = status, Message = message }, JsonRequestBehavior.AllowGet);

else

return Json(new { Status = status, Message = message }, JsonRequestBehavior.AllowGet);

}

catch (Exception ex)

{

return new { Status = status, Message = ex.Message, };

}

}

[HttpGet]

public ActionResult profile()

{

return View();

}

[HttpGet]

public ActionResult mycleaners()

{

var model = new MyCleanersViewModel();

model.userId = \_AccountDetails.Id;

model.index = 0;

return View(model);

}

[HttpPost]

public object RemoveSelectedCleaners(RemoveSelectedCleaners model)

{

bool status = false;

string message = "Failed";

try

{

Tuple<bool, string> t = \_repository.RemoveSelectedCleaners(model);

status = t.Item1;

message = t.Item2;

if (status)

{

return Json(new { Status = status, Message = message }, JsonRequestBehavior.AllowGet);

}

else

return Json(new { Status = status, Message = message }, JsonRequestBehavior.AllowGet);

}

catch (Exception ex)

{

return new { Status = status, Message = ex.Message };

}

}

public PartialViewResult RefreshSelectedCleanersList(MyCleanersViewModel model)

{

return PartialView("~/Views/User/\_pv\_list\_selectedcleaners.cshtml", model);

}

public PartialViewResult RefreshSelectedCleanersListPagination(MyCleanersViewModelUserMyPropertiesPaginationModel model)

{

return PartialView("~/Views/User/\_pv\_list\_selectedcleaners\_pagination.cshtml", model);

}

[HttpPost]

public object postUserSelectedCleaner(PostSelectedCleaner model)

{

bool status = false;

string message = "Failed";

try

{

Tuple<bool, string> t = \_repository.postUserSelectedCleaner(model);

status = t.Item1;

message = t.Item2;

if (status)

return Json(new { Status = status, Message = message }, JsonRequestBehavior.AllowGet);

else

return Json(new { Status = status, Message = message }, JsonRequestBehavior.AllowGet);

}

catch (Exception ex)

{

return new { Status = status, Message = ex.Message, List = new { } };

}

}

[HttpPost]

public object refreshMyCleanersDropdown(RefreshMyCleanersDropdown model)

{

bool status = false;

string message = "Failed";

try

{

Tuple<bool, string, List<ClassLibrary.PostModel.CleanerResponse>> t = \_repository.refreshMyCleanersDropdown(model);

status = t.Item1;

message = t.Item2;

if (status)

return Json(new { Status = status, Message = message, List = t.Item3 }, JsonRequestBehavior.AllowGet);

else

return Json(new { Status = status, Message = message, List = t.Item3 }, JsonRequestBehavior.AllowGet);

}

catch (Exception ex)

{

return new { Status = status, Message = ex.Message, List = new { } };

}

}

[HttpPost]

public object changeSelectedCleaner(UpdateJobsSelectedCleaner model)

{

bool status = false;

string message = "Failed";

try

{

Tuple<bool, string> t = \_repository.changeSelectedCleaner(model);

status = t.Item1;

message = t.Item2;

if (status)

return Json(new { Status = status, Message = message }, JsonRequestBehavior.AllowGet);

else

return Json(new { Status = status, Message = message }, JsonRequestBehavior.AllowGet);

}

catch (Exception ex)

{

return new { Status = status, Message = ex.Message, List = new { } };

}

}

}

}