

**DATABASE:**

CREATE TABLE PERSON(

[user\_id] varchar(25) PRIMARY KEY NOT NULL,

[password] varchar(50) NOT NULL,

fullName varchar(301) NOT NULL,

email varchar(50) NOT NULL,

joined DateTime NOT NULL,

active bit NOT NULL

)

CREATE TABLE FOLLOWING(

[user\_id] varchar(25) NOT NULL ,

[following\_id] varchar(25) NOT NULL,

CONSTRAINT FK\_FOLLOWING\_user\_id FOREIGN KEY ([user\_id]) REFERENCES PERSON([user\_id]),

CONSTRAINT FK\_FOLLOWING\_following\_id FOREIGN KEY ([following\_id]) REFERENCES PERSON([user\_id])

)

CREATE TABLE [dbo].[TWEET](

tweet\_id int IDENTITY(1,1),

[user\_id] [varchar](25) NOT NULL ,

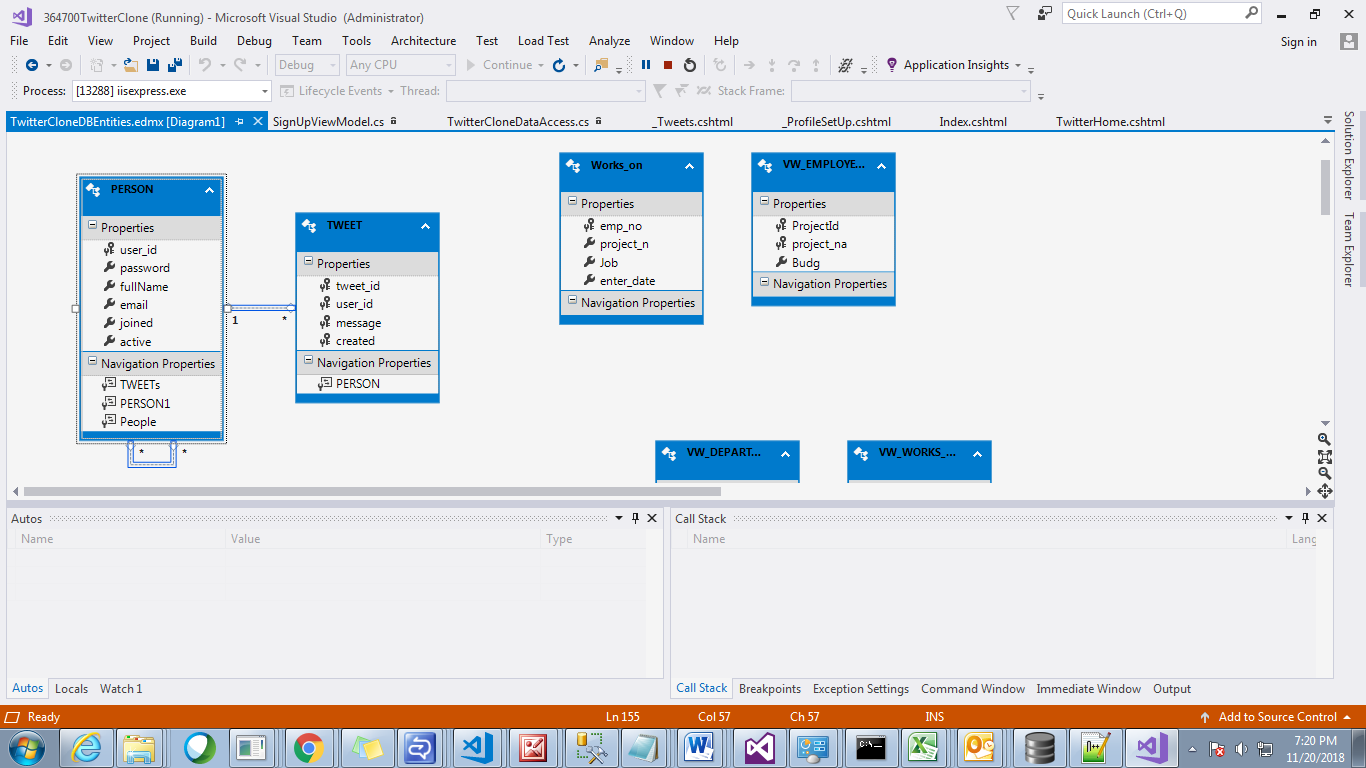
[message] [varchar](140) NOT NULL,

[created] [datetime] NOT NULL,

CONSTRAINT FK\_TWEET\_user\_id FOREIGN KEY ([user\_id]) REFERENCES PERSON([user\_id])

)

**Entity Model:**



**View Model:**

using System;

using System.ComponentModel.DataAnnotations;

using System.Collections.Generic;

namespace DataLayer.ViewModel

{

public class TweetViewModel

{

public TweetMessageViewModel TweetMessageViewModel { get; set; }

public List<TweetMessageViewModel> TweetsViewModel { get; set; }

public Tweetdetails Tweetdetails { get; set; }

}

public class TweetMessageViewModel

{

public int TweetId { get; set; } = 0;

[Required(ErrorMessage ="Tweet Required")]

[MaxLength(140,ErrorMessage ="Tweet message max length is 140 charaters")]

public string TweetMessage { get; set; }

public string userid { get; set; }

public DateTime Created { get; set; }

public bool isUserTweet { get; set; }

}

public class Tweetdetails

{

public int TotalTweets { get; set; } = 0;

public int TotalFollowing { get; set; } = 0;

public int TotalFollowers { get; set; } = 0;

}

}

using System.ComponentModel.DataAnnotations;

using System.Web.Mvc;

namespace DataLayer.ViewModel

{

public class SignUpViewModel

{

[Required(ErrorMessage = "Please Enter User Name")]

[Remote("IsAlreadyExists", "TwitterClone", HttpMethod = "POST", ErrorMessage = "UserId already exists.")]

public string UserName { get; set; }

[Required(ErrorMessage = "Please Enter Password")]

public string Password { get; set; }

[Required(ErrorMessage = "Please Enter Full Name")]

public string FullName { get; set; }

[Required(ErrorMessage = "Please Enter valid Email")]

[EmailAddress]

public string Email { get; set; }

public bool isRegistered { get; set; } = false;

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.ComponentModel.DataAnnotations;

using System.Web.Mvc;

namespace DataLayer.ViewModel

{

public class Login

{

[Required(AllowEmptyStrings = false, ErrorMessage ="Please enter User Name")]

public string UserName { get; set; }

[Required(AllowEmptyStrings = false, ErrorMessage = "Please enter Password")]

public string PassWord { get; set; }

}

}

**DataAccess Layer:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using DataLayer.ViewModel;

using DataLayer.EntityModel;

using System.Data.Entity;

namespace DataLayer.DataAccess

{

public class TwitterCloneDataAccess

{

public static bool UserLogin(Login login)

{

bool isUserExist = false;

using (var \_context = new AngularEntities())

{

var items = \_context.People.Where(x => x.user\_id == login.UserName && x.password == login.PassWord).FirstOrDefault();

isUserExist = (items != null && items.user\_id != null) ? true : false;

}

return isUserExist;

}

public static bool UserAlreadyExists(string userId)

{

bool isUserExist = false;

using (var \_context = new AngularEntities())

{

var items = \_context.People.Where(x => x.user\_id.Trim() == userId.Trim()).FirstOrDefault();

isUserExist = (items == null) ? true : false;

}

return isUserExist;

}

public static SignUpViewModel GetProfile(string userId)

{

SignUpViewModel pr = new SignUpViewModel();

using (var \_context = new AngularEntities())

{

var items = \_context.People.Where(x => x.user\_id.Trim() == userId.Trim()).FirstOrDefault();

if (items != null && items.user\_id != null)

{

pr.UserName = items.user\_id;

pr.Email = items.email;

pr.FullName = items.fullName;

pr.Password = items.password;

}

}

return pr;

}

public static List<string> GetUserList(string UserName)

{

List<string> lstUser = new List<string>();

using (var \_context = new AngularEntities())

{

lstUser = \_context.People.Select(x => x.user\_id).ToList();//.Where(x => x.user\_id.Trim().Contains(UserName))

}

return lstUser != null ? lstUser : new List<string>();

}

public static bool RegisterUser(SignUpViewModel userInfo)

{

bool isRegistered = false;

try

{

using (var \_context = new AngularEntities())

{

var items = \_context.People.Where(x => x.user\_id.ToUpper().Trim() == userInfo.UserName.ToUpper().Trim()).FirstOrDefault();

if (items == null)

{

PERSON user = new PERSON();

user.user\_id = userInfo.UserName;

user.password = userInfo.Password;

user.fullName = userInfo.FullName;

user.email = userInfo.Email;

user.joined = DateTime.Now;

user.active = true;

\_context.People.Add(user);

}

else

{

items.password = userInfo.Password;

items.fullName = userInfo.FullName;

}

\_context.SaveChanges();

isRegistered = true;

}

}

catch (Exception ex)

{

isRegistered = false;

}

return isRegistered;

}

public static Tweetdetails GetTweetDetails(string userName)

{

Tweetdetails td = new Tweetdetails();

try

{

}

catch (Exception ex)

{

}

return td;

}

public static Tweetdetails GetTweetMessageDetails(string userName)

{

Tweetdetails tmv = new Tweetdetails();

try

{

using (var \_context = new AngularEntities())

{

var tweets = \_context.TWEETs.Where(x => x.user\_id.Trim().ToUpper() == userName.ToUpper().Trim());

var follows = \_context.People.Where(x => x.user\_id.Trim().ToUpper() == userName.ToUpper().Trim()).FirstOrDefault();

if (tweets != null)

tmv.TotalTweets = tweets.Count();

if (follows != null)

{

tmv.TotalFollowers = follows.People.Count();

tmv.TotalFollowing = follows.PERSON1.Count();

}

}

}

catch (Exception ex)

{

}

return tmv;

}

public static List<TweetMessageViewModel> GetTweets(string userName)

{

List<TweetMessageViewModel> lsttmv = new List<TweetMessageViewModel>();

try

{

using (var \_context = new AngularEntities())

{

var users = \_context.People.Where(x => x.user\_id.Trim().ToUpper() == userName.ToUpper().Trim());

var tweets = \_context.TWEETs.Where(x => x.user\_id.Trim().ToUpper() == userName.ToUpper().Trim());

lsttmv = tweets.Select(t =>

new TweetMessageViewModel

{

TweetId = t.tweet\_id,

TweetMessage = t.message,

userid = t.user\_id,

Created = t.created,

isUserTweet = t.user\_id.Trim().ToUpper() == userName.ToUpper().Trim() ? true : false

}).ToList();

}

}

catch (Exception ex)

{

}

return lsttmv;

}

public static bool SaveTweet(TweetMessageViewModel tweetMsg)

{

bool IsSuccess = false;

try

{

using (var \_context = new AngularEntities())

{

var tweets = \_context.People.Where(x => x.user\_id.Trim().ToUpper() == tweetMsg.userid.Trim().ToUpper()).FirstOrDefault();

var tweet = tweets.TWEETs.Where(x => x.tweet\_id == tweetMsg.TweetId).FirstOrDefault();//\_context.TWEETs.Where(x => x.tweet\_id == tweetMsg.TweetId).FirstOrDefault();

if (tweet == null && tweetMsg.TweetId == 0)

{

tweet = new TWEET();

tweet.user\_id = tweetMsg.userid;

tweet.message = tweetMsg.TweetMessage;

tweet.created = DateTime.Now;

\_context.Entry(tweet).State = EntityState.Added;

}

else

{

tweet.message = tweetMsg.TweetMessage;

}

IsSuccess = true;

\_context.SaveChanges();

}

}

catch (Exception ex)

{

}

return IsSuccess;

}

public static string DeleteTweet(int tweetId)

{

string strUserName = string.Empty;

try

{

using (var \_context = new AngularEntities())

{

var tweet = \_context.TWEETs.Where(x => x.tweet\_id == tweetId).FirstOrDefault();//\_context.TWEETs.Where(x => x.tweet\_id == tweetMsg.TweetId).FirstOrDefault();

if (tweet != null && tweetId != 0)

{

strUserName = tweet.user\_id;

\_context.TWEETs.Remove(tweet);

//\_context.Entry(tweet).State = EntityState.Deleted;

}

\_context.SaveChanges();

}

}

catch (Exception ex)

{

}

return strUserName;

}

public static TweetMessageViewModel EditTweet(int tweetId)

{

TweetMessageViewModel \_editTweet = new TweetMessageViewModel();

try

{

using (var \_context = new AngularEntities())

{

var tweet = \_context.TWEETs.Where(x => x.tweet\_id == tweetId).FirstOrDefault();//\_context.TWEETs.Where(x => x.tweet\_id == tweetMsg.TweetId).FirstOrDefault();

\_editTweet.userid = tweet.user\_id;

\_editTweet.TweetMessage = tweet.message;

\_editTweet.TweetId = tweet.tweet\_id;

\_editTweet.Created = tweet.created;

}

}

catch (Exception ex)

{

}

return \_editTweet;

}

}

}

**Common Utilites:**

using System.Text;

using System.Security.Cryptography;

namespace CommonUtilities

{

public static class PasswordMD5Encryption

{

public static string MD5Hash(string text)

{

MD5 md5 = new MD5CryptoServiceProvider();

//compute hash from the bytes of text

md5.ComputeHash(ASCIIEncoding.ASCII.GetBytes(text));

//get hash result after compute it

byte[] result = md5.Hash;

StringBuilder strBuilder = new StringBuilder();

for (int i = 0; i < result.Length; i++)

{

//change it into 2 hexadecimal digits

//for each byte

strBuilder.Append(result[i].ToString("x2"));

}

return strBuilder.ToString();

}

}

}

**Business Layer:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using DataLayer.ViewModel;

using DataLayer.DataAccess;

using CommonUtilities;

namespace BusinessLayer

{

public class TwitterCloneBusinessLayer

{

public bool UserLogin(Login login)

{

login.PassWord = PasswordMD5Encryption.MD5Hash(login.PassWord);

return TwitterCloneDataAccess.UserLogin(login);

}

public bool IsAlreadyExists(string UserId)

{

return TwitterCloneDataAccess.UserAlreadyExists(UserId);

}

public List<string> GetUserList(string UserName)

{

return TwitterCloneDataAccess.GetUserList(UserName);

}

public SignUpViewModel RegisterUser(SignUpViewModel userInfo)

{

userInfo.Password = PasswordMD5Encryption.MD5Hash(userInfo.Password);

var isRegistered = TwitterCloneDataAccess.RegisterUser(userInfo);

userInfo.isRegistered = isRegistered;

return userInfo;

}

public SignUpViewModel EditProfile(string UserId)

{

var profile = TwitterCloneDataAccess.GetProfile(UserId);

// profile.Password = PasswordMD5Encryption.MD5Hash(profile.Password);

return profile;

}

public TweetViewModel GetTweet(string userName,int tweetId)

{

TweetViewModel tvm = new TweetViewModel();

tvm.TweetMessageViewModel = new TweetMessageViewModel();

var Tweets = TwitterCloneDataAccess.GetTweets(userName);

var tweetDetails = TwitterCloneDataAccess.GetTweetMessageDetails(userName);

if (tweetId == 0)

tvm.TweetMessageViewModel.userid = userName;

else

{

tvm.TweetMessageViewModel = Tweets.Where(x => x.TweetId == tweetId).FirstOrDefault();

}

tvm.TweetsViewModel = Tweets;

tvm.Tweetdetails = tweetDetails;

return tvm;

}

public bool SaveTweet(TweetMessageViewModel tweetMsg)

{

return TwitterCloneDataAccess.SaveTweet(tweetMsg);

}

public string DeleteTweet(int tweetId)

{

return TwitterCloneDataAccess.DeleteTweet(tweetId);

}

public TweetMessageViewModel EditTweet(int tweetId)

{

return TwitterCloneDataAccess.EditTweet(tweetId);

}

}

}

**Controller:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.Mvc;

using DataLayer.ViewModel;

using BusinessLayer;

using System.Threading.Tasks;

namespace TwitterClone.Controllers

{

public class TwitterCloneController : Controller

{

public TwitterCloneBusinessLayer \_tcBLayer = null;

public TwitterCloneController()

{

\_tcBLayer = new TwitterCloneBusinessLayer();

}

// GET: TwitterClone

public ActionResult Index()

{

return View(new Login());

}

[HttpPost]

[ValidateAntiForgeryToken]

public ActionResult Login(Login userLogin)

{

if (ModelState.IsValid)

{

var isUser = \_tcBLayer.UserLogin(userLogin);

if (!isUser)

ModelState.AddModelError("UserName", "InValid User");

else

{

SetTempUser(userLogin.UserName);

//TempData["userLogin"] = userLogin;

return RedirectToAction("TwitterHome");

}

}

return View("Index", userLogin);

}

public ActionResult signUp()

{

return View(new SignUpViewModel());

}

[HttpPost]

public ActionResult RegisterUser(SignUpViewModel regUser)

{

if (ModelState.IsValid)

{

regUser = \_tcBLayer.RegisterUser(regUser);

if (!regUser.isRegistered)

ModelState.AddModelError("UserName", "Something went wrong, Please try later.");

}

return View("signUp", regUser);

}

[HttpPost]

public ActionResult IsAlreadyExists(string UserName)

{

bool isExist = false;

isExist = \_tcBLayer.IsAlreadyExists(UserName);

return Json(isExist);

}

public ActionResult TwitterHome(int EdittweetId = 0,string userName=null)

{

Login userLogin = getTempUser(userName);

ViewBag.SignUp = new SignUpViewModel();

ViewBag.UserName = userLogin.UserName;

ViewBag.Users = GetUserList(userLogin.UserName);

//bool isExist = false;

var res = \_tcBLayer.GetTweet(userLogin.UserName, EdittweetId);

return View(res);

}

public ActionResult TwitterHomeRefresh(int EdittweetId = 0, string userName = null)

{

Login userLogin = getTempUser(userName);

ViewBag.UserName = userLogin.UserName;

var res = \_tcBLayer.GetTweet(userLogin.UserName, EdittweetId);

return PartialView("\_Tweets",res.TweetsViewModel);

}

public ActionResult TwitterMessage(TweetViewModel tweetMV)

{

SetTempUser(tweetMV.TweetMessageViewModel.userid);

if (ModelState.IsValid)

{

var status = \_tcBLayer.SaveTweet(tweetMV.TweetMessageViewModel);

//bool isExist = false;

if (status)

{

var res = \_tcBLayer.GetTweet(tweetMV.TweetMessageViewModel.userid, tweetMV.TweetMessageViewModel.TweetId);

//return PartialView("Tweets", res.TweetsViewModel);

return RedirectToAction("TwitterHome", tweetMV);

}

}

return View("TwitterHome", tweetMV);

//return RedirectToAction("TwitterHome");

}

public ActionResult EditTweet(int tweetId)

{

return RedirectToAction("TwitterHome",new { EdittweetId = tweetId });

}

public ActionResult DeleteTweet(int tweetId)

{

SetTempUser(\_tcBLayer.DeleteTweet(tweetId));

return RedirectToAction("TwitterHome"); //, new { EdittweetId = tweetId }

}

public ActionResult HomePage(string UserName)

{

SetTempUser(UserName);

return RedirectToAction("TwitterHome"); //, new { EdittweetId = tweetId }

}

public ActionResult EditProfile(string UserName)

{

SetTempUser(UserName);

return PartialView("\_ProfileSetUp", \_tcBLayer.EditProfile(UserName)); //, new { EdittweetId = tweetId }

}

private Array GetUserList(string UserName)

{

return \_tcBLayer.GetUserList(UserName).ToArray();

}

private void SetTempUser(string userName)

{

Login lgn = new Login();

lgn.UserName = userName;

TempData["userLogin"] = lgn;

TempData.Keep("userLogin");

}

private Login getTempUser(string userName)

{

Login userLogin = null;

if (!string.IsNullOrEmpty(userName))

{

userLogin = new Login();

userLogin.UserName = userName;

}

else

{

userLogin = (Login)TempData["userLogin"];

}

TempData["userLogin"] = userLogin;

TempData.Keep("userLogin");

return userLogin;

}

}

}

**Views:**

**Index.Cshtml**

@model DataLayer.ViewModel.Login

@{

ViewBag.Title = "Twitter Clone";

}

<style>

fieldset.scheduler-border {

border: 1px groove #ddd !important;

padding: 0 1.4em 1.4em 1.4em !important;

margin: 0 0 1.5em 0 !important;

-webkit-box-shadow: 0px 0px 0px 0px #000;

box-shadow: 0px 0px 0px 0px #000;

}

legend.scheduler-border {

font-size: 1.2em !important;

font-weight: bold !important;

text-align: left !important;

width: auto;

padding: 0 10px;

border-bottom: none;

}

.borderless tr td {

border: none !important;

vertical-align: middle !important;

}

</style>

<div class="container container-table">

<div class="row vertical-center-row">

<div class="text-center col-md-6 col-md-offset-3"><h1>Welcome to My Twitter Clone</h1></div>

</div>

</div>

@using (Html.BeginForm("Login", "TwitterClone", FormMethod.Post))

{

@Html.AntiForgeryToken()

<div class="container container-table">

<div class="row vertical-center-row">

<div class="col-md-4 col-md-offset-4">

<fieldset class="scheduler-border">

<legend class="scheduler-border">Login</legend>

<div class="alert-danger">

@Html.ValidationSummary(false)

</div>

<table class="table borderless">

<tr>

<td><label>UserName</label></td>

<td>@Html.TextBoxFor(a => a.UserName, new { @class = "form-control" })</td>

@\*<td>@Html.ValidationMessageFor(a => a.UserName)</td>\*@

</tr>

<tr>

<td>

<label>PassWord</label>

</td>

<td>

@Html.PasswordFor(a => a.PassWord, new { @class = "form-control" })

</td>

<td>

@\*@Html.ValidationMessageFor(a => a.PassWord)\*@

</td>

</tr>

<tr>

<td colspan="2" class="text-right"><input class="btn btn-primary" type="submit" value="Login" /></td>

</tr>

<tr>

<td colspan="2">New User? @Html.ActionLink("Sign Up", "signUp")</td>

</tr>

</table>

</fieldset>

</div>

</div>

</div>

}

**SignUp:**

@model DataLayer.ViewModel.SignUpViewModel

<script src="~/Scripts/jquery-3.3.1.js"></script>

<script src="~/Scripts/jquery.validate.js"></script>

<script src="~/Scripts/jquery.validate.unobtrusive.js"></script>

<style>

fieldset.scheduler-border {

border: 1px groove #ddd !important;

padding: 0 1.4em 1.4em 1.4em !important;

margin: 0 0 1.5em 0 !important;

-webkit-box-shadow: 0px 0px 0px 0px #000;

box-shadow: 0px 0px 0px 0px #000;

}

legend.scheduler-border {

font-size: 1.2em !important;

font-weight: bold !important;

text-align: left !important;

width: auto;

padding: 0 10px;

border-bottom: none;

}

.borderless tr td {

border: none !important;

vertical-align: middle !important;

}

</style>

<div class="container container-table">

<div class="row vertical-center-row">

<div class="text-center col-md-6 col-md-offset-3"><h1>Join My Twitter Clone</h1></div>

</div>

</div>

@using (Html.BeginForm("RegisterUser", "TwitterClone", FormMethod.Post))

{

@Html.AntiForgeryToken()

<div class="container container-table">

<div class="row vertical-center-row">

<div class="col-md-5 col-md-offset-4">

@Html.ActionLink("Back", "Index")

<fieldset class="scheduler-border">

<legend class="scheduler-border">Your Details</legend>

<div class="alert-danger">

@Html.ValidationSummary(false)

</div>

<table class="table borderless">

<tr>

<td><label>UserName</label></td>

<td>@Html.TextBoxFor(a => a.UserName, new { @class = "form-control" })</td>

</tr>

<tr>

<td><label>PassWord</label></td>

<td>@Html.PasswordFor(a => a.Password, new { @class = "form-control" })</td>

</tr>

<tr>

<td><label>Full Name</label></td>

<td>@Html.TextBoxFor(a => a.FullName, new { @class = "form-control" })</td>

</tr>

<tr>

<td><label>Email</label></td>

<td>@Html.TextBoxFor(a => a.Email, new { @class = "form-control" })</td>

</tr>

<tr>

<td colspan="2" class="text-right"><input class="btn btn-primary" type="submit" value="Login" /></td>

</tr>

</table>

@if (Model.isRegistered)

{

<div class="success">

User Registration successful.

</div>

}

</fieldset>

</div>

</div>

</div>

}

**Twitter Home page:**

@model DataLayer.ViewModel.TweetViewModel

@{

ViewBag.Title = "Twitter Home";

}

<link rel="stylesheet" href="//code.jquery.com/ui/1.12.1/themes/base/jquery-ui.css">

<link rel="stylesheet" href="/resources/demos/style.css">

<script src="https://code.jquery.com/jquery-1.12.4.js"></script>

<script src="https://code.jquery.com/ui/1.12.1/jquery-ui.js"></script>

<style type="text/css">

ul li {

float: left;

}

ul li:not(:last-child) {

border-right: 1px solid grey;

margin-right: 0px;

padding-right: 0px;

}

.clsAlignMiddle {

vertical-align: middle !important;

margin-top: 15px;

}

.navbar-nav > li > a {

padding-top: 0px;

padding-bottom: 0px;

}

.nav > li > a {

padding: 10px 10px;

}

.input, select, textarea {

max-width: 100%;

}

.col-md-12 {

padding-left: 0px !important;

}

#imaginary\_container {

margin-top: 20%; /\* Don't copy this \*/

}

.stylish-input-group .input-group-addon {

background: white !important;

}

.stylish-input-group .form-control {

border-right: 0;

box-shadow: 0 0 0;

border-color: #ccc;

}

.stylish-input-group button {

border: 0;

background: transparent;

}

</style>

<script type="text/javascript">

var userId = '@ViewBag.UserName';

var autoSource = @Html.Raw(Json.Encode(@ViewBag.Users));

$(function () {

$('#divProfile').hide();

$('#divHome').show();

$("#tags").autocomplete({

source: autoSource,

minLength:3

});

setInterval('Refresh()', 10000);

});

function Refresh() {

$.ajax(

{

url: "/TwitterClone/TwitterHomeRefresh",

data: { EdittweetId:0, UserName: userId },

cache: false,

type: 'GET',

contentType: "application/json",

success: function (html) {

$('#divProfile').hide();

$('#divHome').show();

//$('#divHome').empty();

//$('#divHome').html(html);

}

});

}

function onProfileSetUp(value) {

$.ajax({

url: "/TwitterClone/EditProfile",

//data: { dttime: searchTxt },

data: { UserName: userId },

cache: false,

type: 'GET',

contentType: "application/json",

success: function (html) {

$('#divProfile').show();

$('#divHome').hide();

$('#divProfile').empty();

$('#divProfile').html(html);

}

});

}

function onHomePage(value) {

$.ajax({

url: "/TwitterClone/HomePage",

//data: { dttime: searchTxt },

data: { UserName: value },

cache: false,

type: 'GET',

contentType: "application/json",

success: function (html) {

$('#divProfile').hide();

$('#divHome').show();

//$('#divHome').html(html);

}

});

}

</script>

@using (Html.BeginForm("TwitterMessage", "TwitterClone", FormMethod.Post))

{

@Html.AntiForgeryToken()

<div class=".container-fluid">

<div class="d-inline-block col-md-3">

<div class="col-md-12 text-left">

<div id="imaginary\_container">

<div class="input-group stylish-input-group">

<input type="text" id="tags" class="form-control" placeholder="Search">

<span class="input-group-addon">

<button>

<span class="glyphicon glyphicon-search"></span>

</button>

</span>

</div>

</div>

</div>

<br />

<br />

<br />

<div class="clsAlignMiddle">

@if (Model.Tweetdetails != null)

{

<div><a>@Model.Tweetdetails.TotalTweets</a> Tweets</div>

<div><a>@Model.Tweetdetails.TotalFollowing</a> Following</div>

<div><a>@Model.Tweetdetails.TotalFollowers</a> Followers</div>

}

</div>

</div>

<div class="d-inline-block col-md-9">

<div class="d-inline-block col-md-5">

<h1>My Twitter Clone</h1>

</div>

<div class="d-inline-block col-md-3 clsAlignMiddle"><h4>Welcome @ViewBag.UserName !</h4></div>

<div class="d-inline-block col-md-4 clsAlignMiddle">

<div class="d-inline-block">

<div class="d-inline-block navbar-collapse collapse">

<ul class="nav navbar-nav">

<li>

<a onclick="onHomePage('@Model.TweetMessageViewModel.userid')">Home</a>

@\*@Html.ActionLink("Home", "HomePage", new { UserName = Model.TweetMessageViewModel.userid })\*@

</li>

<li><a onclick="onProfileSetUp('@Model.TweetMessageViewModel.userid')">Profile</a></li>

<li>@Html.ActionLink("Signout", "Index")</li>

</ul>

</div>

</div>

</div>

<br />

<br />

<div id="divHome">

@{Html.RenderPartial("\_Tweets", Model);}

</div>

<div id="divProfile">

@{Html.RenderPartial("\_ProfileSetUp", new DataLayer.ViewModel.SignUpViewModel());}

</div>

</div>

</div>

}

**Tweets Partial View:**

@model DataLayer.ViewModel.TweetViewModel

<div class="form-group text-left col-md-12">

@Html.ValidationSummary(false)

<label for="exampleTextarea text-left">What are you doing?</label>

</div>

<div class="form-group text-left">

@Html.TextAreaFor(a => a.TweetMessageViewModel.TweetMessage, new { @class = "form-control", id = "exampleTextarea", rows = "3" })

@Html.HiddenFor(a => a.TweetMessageViewModel.userid)

@Html.HiddenFor(a => a.TweetMessageViewModel.TweetId)

</div>

<div class="text-right">

<input class="btn btn-primary" type="submit" value="Tweet" />

</div>

<br />

<br />

<br />

<div class="form-group">

<table class="table">

@foreach (var item in Model.TweetsViewModel)

{

<tr>

<td>

@Html.DisplayFor(modelItem => item.userid)

</td>

<td>

@Html.DisplayFor(modelItem => item.TweetMessage)

</td>

<td>

@Html.DisplayFor(modelItem => item.Created)

</td>

@if (item.isUserTweet)

{

<td>

@Html.ActionLink("Edit", "EditTweet", new { tweetId = item.TweetId }) |

@Html.ActionLink("Delete", "DeleteTweet", new { tweetId = item.TweetId })

</td>

}

else

{

<td></td>

}

</tr>

}

</table>

</div>

**Partial view \_pagesetup**

@model DataLayer.ViewModel.SignUpViewModel

<style>

fieldset.scheduler-border {

border: 1px groove #ddd !important;

padding: 0 1.4em 1.4em 1.4em !important;

margin: 0 0 1.5em 0 !important;

-webkit-box-shadow: 0px 0px 0px 0px #000;

box-shadow: 0px 0px 0px 0px #000;

}

legend.scheduler-border {

font-size: 1.2em !important;

font-weight: bold !important;

text-align: left !important;

width: auto;

padding: 0 10px;

border-bottom: none;

}

.borderless tr td {

border: none !important;

vertical-align: middle !important;

}

</style>

@using (Html.BeginForm("RegisterUser", "TwitterClone", FormMethod.Post))

{

@Html.AntiForgeryToken()

if (Model != null)

{

<div class="container container-table">

<div class="row text-left">

<div class="col-md-6 col-md-offset-4">

<fieldset class="scheduler-border">

<legend class="scheduler-border">Your Details</legend>

<div class="alert-danger">

@Html.ValidationSummary(false)

</div>

<table class="table borderless">

<tr>

<td colspan="2">

@Html.ActionLink("Delete Account", "DeleteAccount")

</td>

</tr>

<tr>

<td><label>UserName</label></td>

<td>@Html.TextBoxFor(a => a.UserName, new { @class = "form-control", diasbled = "disabled", @readonly = "readonly" })</td>

</tr>

<tr>

<td><label>PassWord</label></td>

<td>@Html.PasswordFor(a => a.Password, new { @class = "form-control" })</td>

</tr>

<tr>

<td><label>Full Name</label></td>

<td>@Html.TextBoxFor(a => a.FullName, new { @class = "form-control" })</td>

</tr>

<tr>

<td><label>Email</label></td>

<td>@Html.TextBoxFor(a => a.Email, new { @class = "form-control" })</td>

</tr>

<tr>

<td colspan="2" class="text-right"><input class="btn btn-primary" type="submit" value="Login" /></td>

</tr>

</table>

@if (Model.isRegistered)

{

<div class="success">

User Registration successful.

</div>

}

</fieldset>

</div>

</div>

</div>

}

}