Technical Documentation – Create your CV Online

People can use the code through

Github Repository:

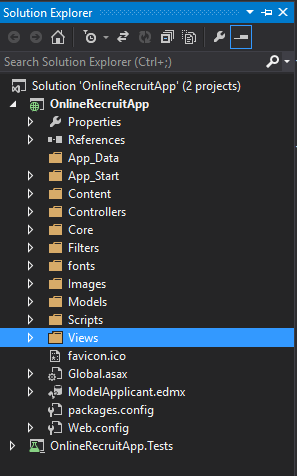
<https://github.com/shikhatest24/SpikeTest.git>

Technologies used:

* ASP.NET MVC4
* SQL server 2012
* C#
* Repository Pattern
* Object Relation Model as Entity Framework
* AngularJS
* Twitter Bootstrap in User Interface
* HTML5 and Razor view Engine
* Unit Testing

Project Flow:

IDE- Visual Studio 2013🡪 Create a Web Project 🡪 MVC4🡪OnlinerecruitApp

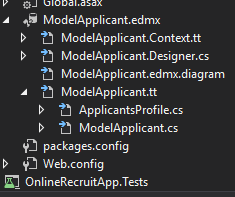


Install Boostrap Packages by using the following command:

PM> Install-Package twitter.bootstrap.mvc4

Create database “resume” and table “ApplicantsProfile” in SQLSERVER 2012

Create an edmx file :Model Name is ApplicantsProfile



App\_Start Folder : Route.Config calls the actionResult Index and Bundle.Config contains all the bundled css and js files (MVC5 Bundling) in Route.Config , first Actionresult called is “Index” from ApplicantController

public class RouteConfig

{

public static void RegisterRoutes(RouteCollection routes)

{

routes.IgnoreRoute("{resource}.axd/{\*pathInfo}");

routes.MapRoute(

name: "routes",

url: "{route}/{id}",

defaults: new { controller = "Applicant", action = "Index", id = UrlParameter.Optional },

constraints: new { route = @"^(login)$" }

);

routes.MapRoute(

name: "Default",

url: "{controller}/{action}/{id}",

defaults: new { controller = "Applicant", action = "Index", id = UrlParameter.Optional }

);

}

}

Content Folder for the references of images/Js/css/angular js and bootstrap files.

Controller Folder: Provides a layer of Controller (We can use Areas as well for different modules)

Code for Applicant Controller : Different Actionresult will in turn communicate the db through the Repositories

public class ApplicantController : Controller

{

private IAppRepository \_applicantRepository;

public ApplicantController()

{

this.\_applicantRepository = new AppRepository(new ProfileContext());

}

public ActionResult Index()

{

return View();

}

// Get method Action to show a view page for adding a New Applicant

public ActionResult AddApplicant()

{

return View();

}

// a json model to save the data through model into the database after communicating the JS files in Scripts/angularJS folder

public JsonResult save(ApplicantsProfile ap)

{

try

{

if (ModelState.IsValid)

{

\_applicantRepository.InsertApplicant(ap);

\_applicantRepository.Save();

return Json("Index");

}

}

catch (DataException)

{

ModelState.AddModelError("", "Unable to save changes. Try again, and if the problem persists see your system administrator.");

}

return Json(ap);

}

public ActionResult ApplicantDetails()

{

return View();

}

// returns a json model to show list of all newly added Applicants

public JsonResult Applicantlist()

{

var applicants = from ApplicantsProfile in \_applicantRepository.GetApplicants()

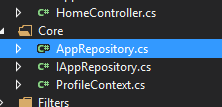
select ApplicantsProfile;

return Json(applicants, JsonRequestBehavior.AllowGet);

}

}

Core Folder : Implements repository Pattern

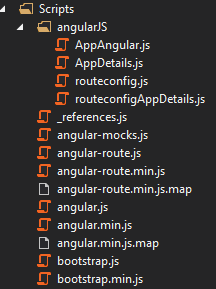


ProfileContext: defines a context on model ApplicantsProfile

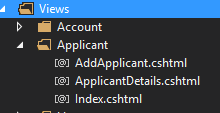
IAppRepository: declares interface for Adding new applicant, show list of applicants.

AppRepository: defines the interface in this layer which in turn will communicate with the edmx model.

Scripts Folder: Added AppAngular.Js and routeconfig.js files to implement Page load and show Details and save the details of an Applicant



Views Folder🡪 Applicant Folder🡪 added 3 views Index AddApplicant and ApplicantDetails



Validations Done : (Applying validations in angular js is in process)

[Display(Name = "First Name")]

[Required(ErrorMessage = "Name is required")]

public string FirstName { get; set; }

Output:

