Creating Real Time MVC Projects Step by Step with n-layer architecture

## **Introduction**

Model view controller (MVC) is a software architectural pattern for developing web applications. It divides a given application into three interconnected parts.

**Model**- The model represents the data. The model does not depend on the controller or the view.It is a virtual representation of data used to perform any opperaton in the project.

**View**- This is responsible for displaying the UI to the user.

**Controller**- It mainly handle the incomming request coming from browser and process these requests to implement View.

 The MVC design pattern decouples these major components allowing us for code reusability and parallel development.

In this pattern Controller receives all incomming requests from browser  and then works with the Model to identify the data used to create a view. The View then uses the data and generate a final presentable UI.

Here in this article we will learn how to create a real time project using MVC architecture.

## **Table of contents**

**Creating multi-layer MVC project**

1. Setting up project layers.
2. creating UI,BLL,DAL for the project.
3. Understanding the flow between the layers.

**Dynamic Home Page Design for the Project**

1. Creating Master page for the project
2. A slider in the Homepage.
3. Dynamicly data to populate in Home page.

**Working with Database**

1. Creating Sample Table Structure,
2. Writting multi-purpose Store procedure.

**Master Details Entry for the Project**

1. Creating a form for Patient Entry.
2. Displaying the patient entry using a MVC web grid.
3. Template entry of for the project.
4. Displaying all template with(Searching,Sorting) using  Jquery DataTable.

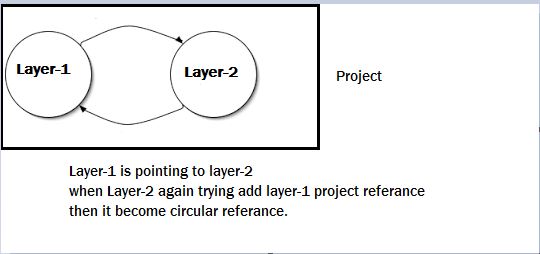
## **Setting Up Project Layers**

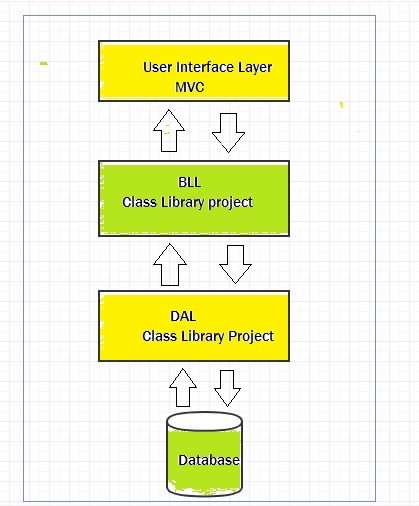
Before setting up project layer let me tell you all what a layer exactly in a project.

**Layers**refer to the internal architecture of a project,For ex:  You divide your project into different layers like Data access layer, Business logic layer,User Interface layer etc.. So they are internal to the project, and these layers interact with each other internally to form the entire working component.  
The main benefits of the layered architectural style are:  
Abstraction,Isolation,Reusability, Testability.

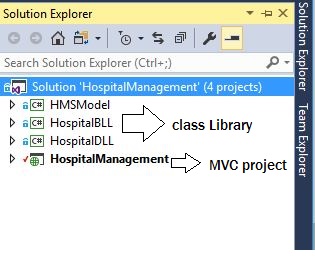
while creating project layers please make focus on following things-

1. Beware of Circular Dependency
2. Each layer should have their own Responsibility(SRP).
3. The layers which are commonly used try to make these layers  as parallel dependant.

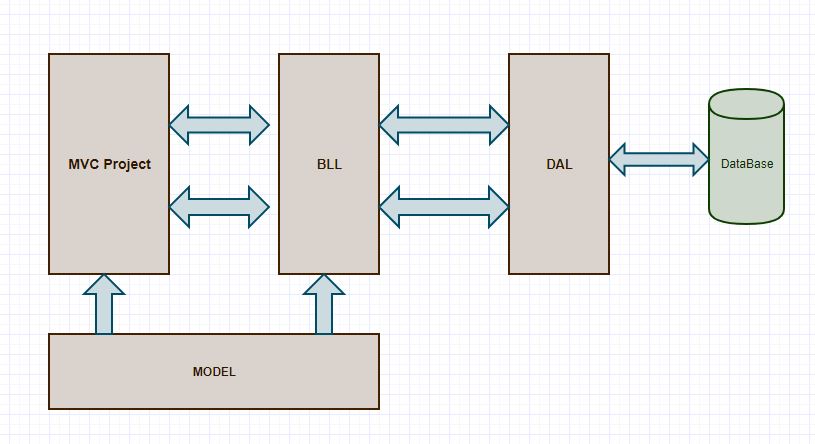
So lets see what exactly Circular dependency is and how it will affect the project architecture .  
  


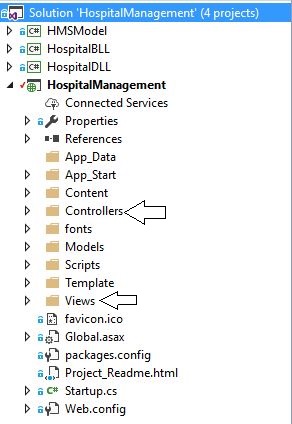
If you try to  reference two class libraries with each other,then Visual Studio throws circular referance error.So while working on project  please avoid this scenario.  
If we have 2 layers in the project,The Layer-1 is referance to the Layer-2 ,that mean some resources of  layer-2 will be consumed by Layer-1, similarly at any point of time Layer-2 wants any resources from Layer-1 then  we need to referance it to layer-1.  
If we will do this a Circular dependency error will arise,so to overcome it we need to create an intermediate module between the 2 layers.This will be clearly describe while start working on project.  
Here is how our project flow look alike.  
  


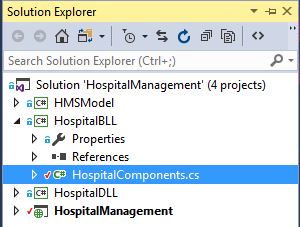
So as per the above sketch diagram i have added the projects layer as shown below.



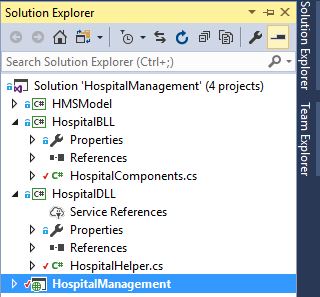
In this diagram you can find out how the project layers are related with each other,



Here is the UI or MVC layer of the application.In this you will find a Template Folder which  basically  contain a template a Home Page for this project.If you want there are several templates available in internet  which you can download and use as for your project.  
  
I have a BussinessLogicLayer called (**HospitalBLL**).This Layer  is the intermediate layer between the main Project and the DataAccess Layer(**HospitalDLL**).The BussinessLogicLayer  contains all the Bussiness rules, The business logic layer (BLL) contains logic specific to the business domain. Also Business logic is used to maximize reuse opportunities.



As it is very specific to the bussiness we will work on it as per our  requirement.Here in this diagram i have shown the bussiness component.We will start coading on it while interacting with the project.

I have added a class as HospitalComponent.cs, but basically the BLL should be based on mainly Controller requirement.A single controller can have one BLL or may be 2-3 controllers have one BLL component.  
  
 Now Lets check what is this Data Access Layer.Data Access Layer is mainly used to interact with the DB,so its contains all the opperation related to the DataBase.  
  


Hide   Shrink Image 12 for Creating Real Time  MVC Projects Step by Step with  n-layer architecture   Copy Code

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Data;

using System.Data.SqlClient;

using System.Configuration;

using HMSMODEL;

namespace HospitalDLL

{

    public class HospitalHelper

    {

        SqlCommand cmd;

        DataTable dt;

        SqlDataAdapter ad;

        public static SqlConnection connect()

        {

            string myconnection = ConfigurationManager.ConnectionStrings["connectHMS"].ToString();

            SqlConnection connection = new SqlConnection(myconnection);

            if(connection.State==ConnectionState.Open)

            {

                connection.Close();

            }

            else

            {

                connection.Open();

            }

            return connection;

        }

        public DataTable getData( string str)

        {

            ad = new SqlDataAdapter(str, HospitalHelper.connect());

            dt = new DataTable();

            ad.Fill(dt);

            return dt;

        }

        public bool DML(SqlCommand cmd)

        {

            int result = 0;

            using (connect())

            {

                using (cmd)

                {

                    cmd.Connection = connect();

                    result = cmd.ExecuteNonQuery();

                }

            }

           if(result>0)

            {

                return true;

            }

           else

            {

                return false;

            }

        }

        public DataTable GetALL(SqlCommand cmd)

        {

            ad = new SqlDataAdapter();

            DataTable dt = new DataTable();

            using (connect())

            {

                using (cmd)

                {

                    cmd.Connection = connect();

                    ad.SelectCommand = cmd;

                    ad.Fill(dt);

                }

            }

            return dt;

        }

    }

}

**Project Description:**

**Iclinic**is a platform which bridges the gap between the doctors,patients and clinic. This platform is concerned about end to end consultation between the  doctors and patients via the medium of digitating clinical Ticketing system.  
In order to achive this facility we have  provided a portal where a patient can register his symptoms,get a online ticket, later can consult with any of the collabarated clinic.During his/her visit to clinic they need to quote the registration id which will be later process by doctor.  
Here in this article we will mainly focus on creating Home page,Master entry forms,etc. Here we have a dynamic Home Page where we have(Slider,dynamic content update,etc).  
Let us see the creating of Home Page with Slider.

Lets check the steps to create a Carousal slider.  
  
A carousal slider actually contains 3 parts.

1. The Banners
2. The Indecators
3. The Controls  
     
     
     
     
     
   **Indecators**are specific circular Icons which the sliding/fading kind of effects inside the banner.These are represented as a floating content over the banner images.Indecators are represented in a ordered list manner.We need to fix the first list item as active,thus making the slidding effect to be started from the first image and later slidded to second,third and soon.

Hide   Copy Code

<ol class="carousel-indicators">

                        <li data-target="#responsive-slider" data-slide-to="0" class="active"></li>

                        <li data-target="#responsive-slider" data-slide-to="1"></li>

                        <li data-target="#responsive-slider" data-slide-to="2"></li>

<li data-target="#responsive-slider" data-slide-to="3"></li>

  <li data-target="#responsive-slider" data-slide-to="4></li>

                    </ol>

As we have thought to fit 3 images in banner for that purpose we have created 3 indecators,the process of which is described above.  
  
The **Banner** area contains a set of images which will be shown to the users with an animated effect.We can give as much images as we want for the banner area, which will be later processed by indecators.The process is shown below.

Hide   Shrink Image 15 for Creating Real Time  MVC Projects Step by Step with  n-layer architecture   Copy Code

<div class="slides" data-group="slides">

                        <ul>

                            <li>

                                <div class="slide-body" data-group="slide">

                                    <img src="~/Template/img/2.jpg" alt="">

                                    <div class="caption header" data-animate="slideAppearUpToDown" data-interval="20" data-length="300">

                                </div>

                            </li>

                            <li>

                                <div class="slide-body" data-group="slide">

                                    <img src="~/Template/img/1.jpg" alt="">

                                    <div class="caption header" data-animate="slideAppearDownToUp" data-interval="20" data-length="300">

                                    </div>

                                </div>

                            </li>

                            <li>

                                <div class="slide-body" data-group="slide">

                                    <img src="~/Template/img/10.jpg" alt="">

                                    <div class="caption header" data-animate="slideAppearUpToDown" data-interval="20" data-length="300">

                                    </div>

                                </div>

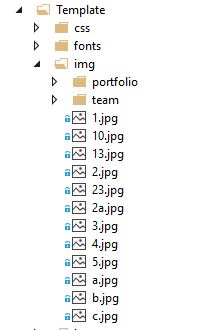
                            </li>

<li>                                 <div class="slide-body" data-group="slide">                                     <img src="~/Template/img/10.jpg" alt="">                                     <div class="caption header" data-animate="slideAppearUpToDown" data-interval="20" data-length="300">                                                                         </div>                                 </div>                             </li>

                        </ul>

                    </div>

The first part contain all the slider image with the delay time mentioned in "**data-interval**".  
Here is the image referances inside the project.



The final step is to create controls for the carousel. This can be achived by using a class called "data-slide" with specifically two arguments "previous & next". When being rendered, the content is shown as two indicators over the banner. The process is shown below-

Hide   Copy Code

<a class="slider-control left" href="#" data-jump="prev"><i class="fa fa-angle-left fa-2x"></i></a>

<a class="slider-control right" href="#" data-jump="next"><i class="fa fa-angle-right fa-2x"></i></a>

As we can see, two controls as left and right angle icon, are being shown over the banner.

The complete corousel format is being shown below, for further clarification.

Hide   Shrink Image 18 for Creating Real Time  MVC Projects Step by Step with  n-layer architecture   Copy Code

<!-- Responsive slider - START -->

    <div class="slider">

        <div class="container">

            <div class="row">

                <div class="responsive-slider" data-spy="responsive-slider" data-autoplay="true">

                    <ol class="carousel-indicators">

                        <li data-target="#responsive-slider" data-slide-to="0" class="active"></li>

                        <li data-target="#responsive-slider" data-slide-to="1"></li>

                        <li data-target="#responsive-slider" data-slide-to="2"></li>

                    </ol>

                    <div class="slides" data-group="slides">

                        <ul>

                            <li>

                                <div class="slide-body" data-group="slide">

                                    <img src="~/Template/img/2.jpg" alt="">

                                    <div class="caption header" data-animate="slideAppearUpToDown" data-interval="20" data-length="300">

                                </div>

                            </li>

                            <li>

                                <div class="slide-body" data-group="slide">

                                    <img src="~/Template/img/1.jpg" alt="">

                                    <div class="caption header" data-animate="slideAppearDownToUp" data-interval="20" data-length="300">

                                    </div>

                                </div>

                            </li>

                            <li>

                                <div class="slide-body" data-group="slide">

                                    <img src="~/Template/img/10.jpg" alt="">

                                    <div class="caption header" data-animate="slideAppearUpToDown" data-interval="20ssssssss" data-length="300">

                                    </div>

                                </div>

                            </li>

                        </ul>

                    </div>

                    <a class="slider-control left" href="#" data-jump="prev"><i class="fa fa-angle-left fa-2x"></i></a>

                    <a class="slider-control right" href="#" data-jump="next"><i class="fa fa-angle-right fa-2x"></i></a>

                </div>

            </div>

        </div>

    </div>

    <!-- Responsive slider - END -->

Actually in this project we have used a template,if you want to use you can download any template and use in your project.  
Now our next thing is to create a dynamic notification which will give information to the user in the Home Page.

Here is the table for the above requirements.

Hide   Copy Code

USE [HMS]

GO

*/\*\*\*\*\*\* Object:  Table [dbo].[tbl\_Template]    Script Date: 04-09-2017 09:54:47 \*\*\*\*\*\*/*

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[tbl\_Template](

    [Header] [nchar](250) NOT NULL,

    [Status] [bit] NOT NULL,

    [id] [int] IDENTITY(1,1) NOT NULL,

    [Description] [nvarchar](max) NULL

) ON [PRIMARY] TEXTIMAGE\_ON [PRIMARY]

GO

Now create a multipurpose stored procedure to perform all functionality.

Hide   Copy Code

USE [HMS]

GO

*/\*\*\*\*\*\* Object:  StoredProcedure [dbo].[sp\_SaveTemplates]    Script Date: 04-09-2017 11:17:16 \*\*\*\*\*\*/*

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

 ALTER procedure [dbo].[sp\_SaveTemplates]

 (

 @Header nvarchar(300),

 @Description nvarchar(max),

 @status bit

 )

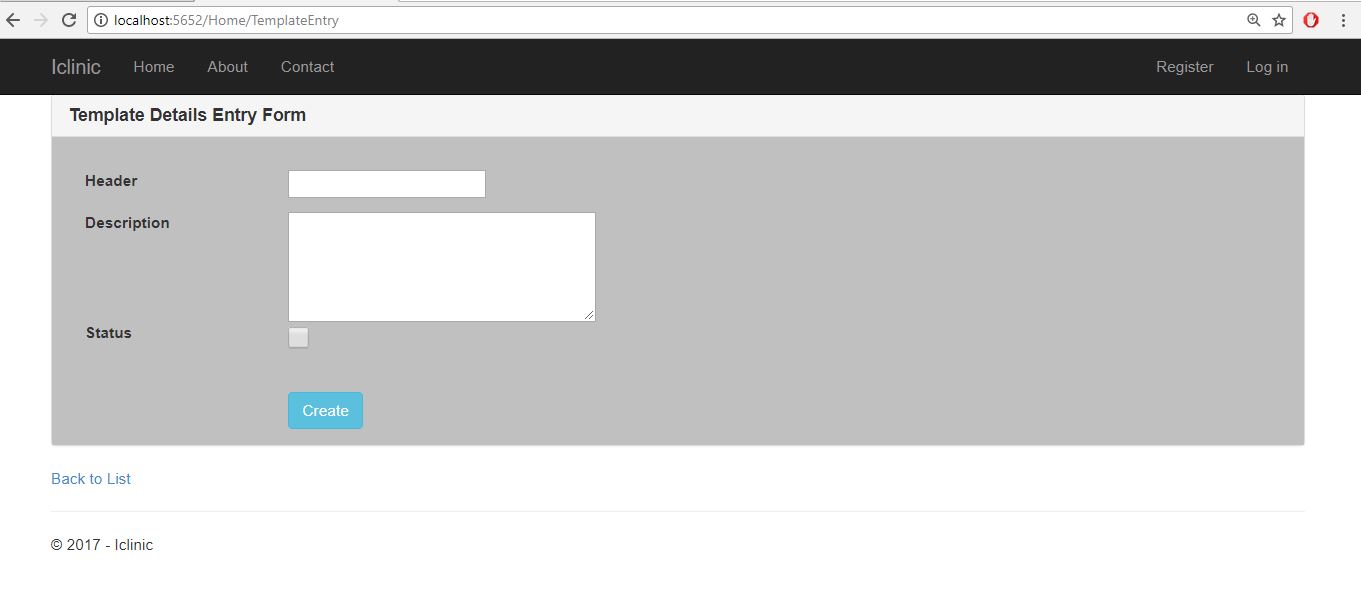
 AS

 Begin

 INSERT INTO tbl\_Template(Header,Description,Status) VALUES(@Header, @Description,@Status)

 End

**Template entry Form:-**This form is used to dynamically entry and template for the patient and shows there details in Home page.

So now we will have the following form for template entry.Here is the image given below.  


Here is the code for the following view.

Hide   Shrink Image 21 for Creating Real Time  MVC Projects Step by Step with  n-layer architecture   Copy Code

@model HMSMODEL.Template

@{

    ViewBag.Title = "TemplateEntry";

}

<script src="https://code.jquery.com/jquery-1.11.1.min.js"></script>

<script src="https://cdn.datatables.net/1.10.4/js/jquery.dataTables.min.js"></script>

<link rel="stylesheet" href="https://cdn.datatables.net/1.10.4/css/jquery.dataTables.min.css">

<style type="text/css">

    .textbox {

        width: 600px;

        height: 100px

    }

    .Checkbox {

        width: 20px;

        height: 20px;

        display: block;

        background: url("link\_to\_image");

    }

</style>

<style type="text/css">

    .panel-body {

        background-color: #C0C0C0;

    }

</style>

@using (Html.BeginForm("TemplateEntry", "Home", FormMethod.Post))

{

    @Html.AntiForgeryToken()

    <div class="panel panel-default">

        <div class="panel-heading">

            <h3 class="panel-title"><b>Template Details Entry Form</b></h3>

        </div>

        <div class="panel-body">

            <div class="row; ba">

                <div class="span6">

                    @Html.ValidationSummary(true)

                    <div class="form-group">

                        @Html.LabelFor(model => model.Header, new { @class = "control-label col-md-2" })

                        <div class="col-md-10">

                            @Html.EditorFor(model => model.Header)

                            @Html.ValidationMessageFor(model => model.Header)

                        </div>

                    </div>

                    <br />

                    <br />

                    <div class="form-group">

                        @Html.LabelFor(model => model.Description, new { @class = "control-label col-md-2" })

                        <div class="col-md-10">

                            @Html.TextAreaFor(model => model.Description, new { @class = "textbox" })

                            @Html.ValidationMessageFor(model => model.Description)

                        </div>

                    </div>

                    <br />

                    <br />

                    <div class="form-group">

                        @Html.LabelFor(model => model.Status, new { @class = "control-label col-md-2" })

                        <div class="col-md-10">

                            @Html.CheckBoxFor(model => model.Status, new { @class = "Checkbox" })

                            @Html.ValidationMessageFor(model => model.Status)

                        </div>

                    </div>

                </div>

                <br />

                <br />

                <br />

                <div class="form-group">

                    <br />

                    <br />

                    <div class="col-md-offset-2 col-md-10">

                        <input type="submit" value="Create" class="btn btn-info" />

                    </div>

                </div>

            </div>

        </div>

    </div>

}

<div>

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

</div>

Here is the Home controller method where  data is posted for saving.

Hide   Copy Code

 HospitalBLL.HospitalComponents obj = new HospitalBLL.HospitalComponents();

This is the object of our bussiness Logic Layer.

And Here is the Template entry GET  and POST Method.

Hide   Copy Code

 public ActionResult TemplateEntry()

        {

            return View();

        }

        [HttpPost]

        public ActionResult TemplateEntry(Template temp)

        {

            bool x = obj.tabEntery(temp);

            if(x==true)

            {

                ViewBag.result = "Data Saved Successfully";

                return View();

            }

            else

            {

                return View();

            }

        }

Now here is the HMS Model layer where we have the Template Classs.

Hide   Copy Code

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace HMSMODEL

{

    public class Template

    {

        public int id { get; set; }

        public string  Header { get; set; }

        public bool Status { get; set; }

        public string Description { get; set; }

    }

}

Now this is the "tabEntery" method on HospitalBLL.

Hide   Copy Code

 public bool tabEntery(Template temp)

        {

            cmd = new SqlCommand("sp\_SaveTemplates");

            cmd.CommandType = CommandType.StoredProcedure;

            cmd.Parameters.AddWithValue("@Header", temp.Header);

            cmd.Parameters.AddWithValue("@Description", temp.Description);

            cmd.Parameters.AddWithValue("@Status", temp.Status);

           bool result=  \_hospitalhelper.DML(cmd);

            if(result==true)

            {

                return true;

            }

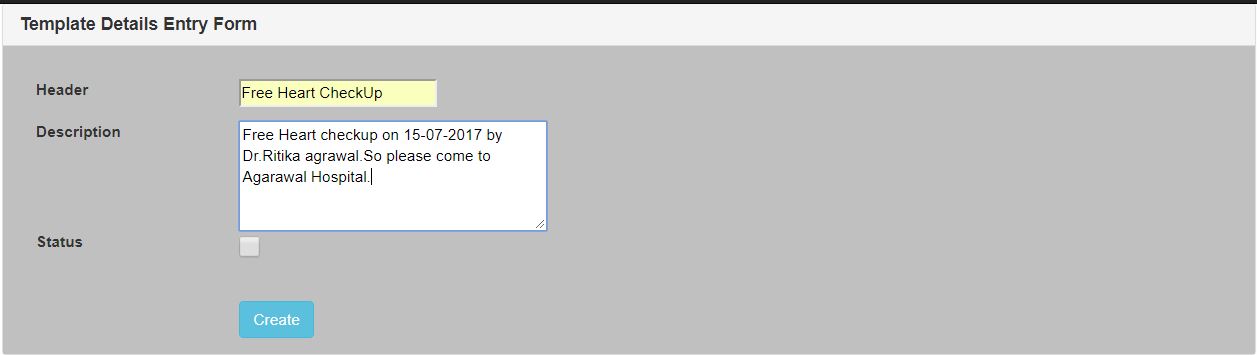
            else

            {

                return false;

            }

        }



Now after creating templates if you want to fetch all template in a dynamic manner,you have the following code.

Hide   Copy Code

        HospitalBLL.HospitalComponents obj = new HospitalBLL.HospitalComponents();

        DataTable dt;

         public ActionResult getData()

        {

            dt = new DataTable();

            dt = obj.getTemplateData();

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

            for (int i = 0; i < dt.Rows.Count; i++)

            {

                Template temp1 = new Template();

                temp1.id = Convert.ToInt32(dt.Rows[i]["id"]);

                temp1.Header = dt.Rows[i]["Header"].ToString();

                temp1.Description = dt.Rows[i]["Description"].ToString();

                list.Add(temp1);

            }

            var data = list;

            return Json(new { data = data }, JsonRequestBehavior.AllowGet);

        }

now here is how we will design the view to show the Templates in index page.

Hide   Shrink Image 23 for Creating Real Time  MVC Projects Step by Step with  n-layer architecture   Copy Code

  @model IEnumerable<HMSMODEL.Template>

@{

    ViewBag.Title = "ViewTemplate";

}

<script src="~/Scripts/jquery-1.10.2.min.js"></script>

<br />

<div style="width:90%; margin:0 auto; background-color:blanchedalmond">

    <table id="myTable">

        <thead>

            <tr>

                <th>ID</th>

                <th>Header</th>

                <th>Description</th>

            </tr>

        </thead>

    </table>

</div>

    @\* CSS for the DataTable \*@

<link href="//cdn.datatables.net/1.10.9/css/jquery.dataTables.min.css" rel="stylesheet" />

<style>

    tr.even {

        background-color:blueviolet;

    }

</style>

@\* Load datatable js \*@

@section Scripts{

    <script src="//cdn.datatables.net/1.10.9/js/jquery.dataTables.min.js"></script>

    <script>

        $(document).ready(function () {

            $('#myTable').DataTable({

                "ajax": {

                    "url": "/home/getData",

                    "type": "GET",

                    "datatype": "json"

                },

                "columns" : [

                        { "data": "id", "autoWidth": true },

                        { "data": "Header", "autoWidth": true },

                        { "data": "Description", "autoWidth": true }

                ],

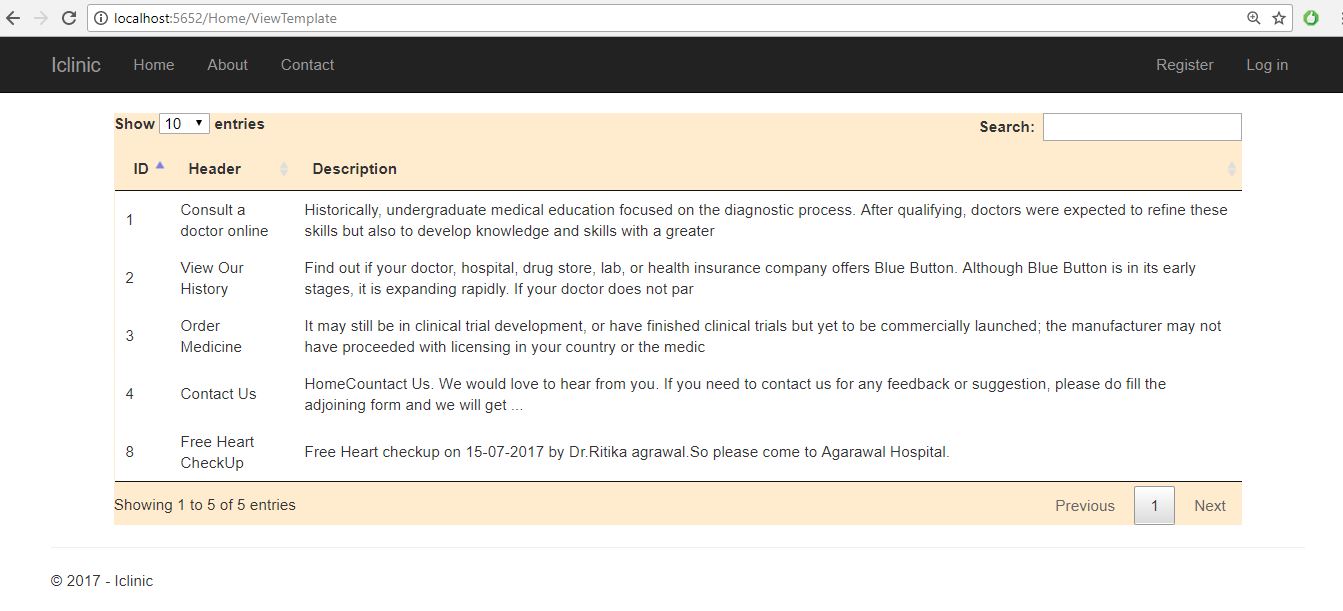
                select: true

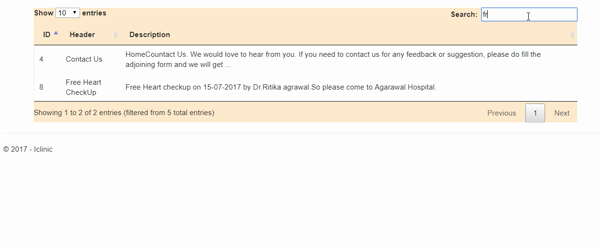
            });

        });

    </script>

    }

  
To design this we have used Jquery DataTable.We have enabled paging,sorting,serching functionality as shown below.



**patient Entry form:**This is the form where patient entry their details to get  consulted with a doctor.

Now lets desing a patient Entry form as follow.



Here is the complete View code.

Hide   Shrink Image 28 for Creating Real Time  MVC Projects Step by Step with  n-layer architecture   Copy Code

@model HMSMODEL.PatientEntry

@{

    ViewBag.Title = "PatientsEntry";

}

    <!-- Bootstrap -->

    <link href="~/Template/css/bootstrap.min.css" rel="stylesheet" />

<style type="text/css">

    .panel-body {

        background-color:#C0C0C0;

    }

</style>

@{

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

    {

        <script type="text/javascript">

            alert("@TempData["alertMessage"]");

        </script>

    }

    else

    {

    }

}

<h4  style="text-align:center;color:aquamarine"></h4>

@using (Html.BeginForm())

{

    @Html.AntiForgeryToken()

    <br />

    <div class="panel panel-default">

        <div class="panel-heading">

            <h3 class="panel-title">Patients Details Entry Form</h3>

        </div>

        <div class="panel-body">

            <div class="row; ba">

                <div class="span6">

                    @Html.ValidationSummary(true)

                    <div class="form-group">

                        @Html.LabelFor(model => model.FirstName, htmlAttributes: new { @class = "control-label col-md-2" })

                        <div class="col-md-10">

                            @Html.EditorFor(model => model.FirstName, new { @class = "form-control", id = "txt\_firstname" })

                            @Html.ValidationMessageFor(model => model.FirstName)

                        </div>

                    </div>

                    <br />

                    <br />

                    <div class="form-group">

                        @Html.LabelFor(model => model.LastName, htmlAttributes: new { @class = "control-label col-md-2" })

                        <div class="col-md-10">

                            @Html.EditorFor(model => model.LastName, new { @class = "form-control" })

                            @Html.ValidationMessageFor(model => model.LastName)

                        </div>

                    </div>

                    <br />

                    <div class="form-group">

                        @Html.LabelFor(model => model.Age, htmlAttributes: new { @class = "control-label col-md-2" })

                        <div class="col-md-10">

                            @Html.EditorFor(model => model.Age, new { @class = "form-control" })

                            @Html.ValidationMessageFor(model => model.Age)

                        </div>

                    </div>

                    <br />

                    <div class="form-group">

                        @Html.LabelFor(model => model.Address, htmlAttributes: new { @class = "control-label col-md-2" })

                        <div class="col-md-10">

                            @Html.EditorFor(model => model.Address, new { @class = "form-control" })

                            @Html.ValidationMessageFor(model => model.Address)

                        </div>

                    </div>

                    <br />

                    <div class="form-group">

                        @Html.LabelFor(model => model.Gender, htmlAttributes: new { @class = "control-label col-md-2" })

                        <div class="col-md-10">

                            Male:   @Html.RadioButton("Gender", "Male")

                            Female: @Html.RadioButton("Gender", "Female")

                        </div>

                    </div>

                    <br />

                    <div class="form-group">

                        @Html.LabelFor(model => model.EntryFee, htmlAttributes: new { @class = "control-label col-md-2" })

                        <div class="col-md-10">

                            @Html.EditorFor(model => model.EntryFee, new { @class = "form-control" })

                            @Html.ValidationMessageFor(model => model.EntryFee)

                        </div>

                    </div>

                    <br />

                    <div class="form-group">

                        @Html.LabelFor(model => model.Description, htmlAttributes: new { @class = "control-label col-md-2" })

                        <div class="col-md-10">

                            @Html.TextAreaFor(model => model.Description, new { @class = "form-control"  })

                            @Html.ValidationMessageFor(model => model.Description)

                        </div>

                    </div>

                    <br />

                    <br />

                    <div class="form-group">

                        <br />

                        <div class="col-md-offset-2 col-md-10">

                            <input type="submit" value="SAVE" class="btn btn-info" onclick="location.href='@Url.Action("PatientsEntry", "PatientsEntry")'" />

                            <input type="submit" id="btn\_Show" value="SHOW" class="btn btn-info" onclick="location.href='@Url.Action("patientDetails", "PatientsEntry")'" />

                        </div>

                    </div>

                </div>

                </div>

        </div>

    </div>

        }

        <div>

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

        </div>

Now here is the Patient entry controller for all functions.

Hide   Shrink Image 29 for Creating Real Time  MVC Projects Step by Step with  n-layer architecture   Copy Code

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.Mvc;

using HMSMODEL;

using System.Data;

namespace HospitalManagement.Controllers

{

    public class PatientEntryController : Controller

    {

        HospitalBLL.HospitalComponents obj = new HospitalBLL.HospitalComponents();

        DataTable dt;

*// GET: PatientEntry*

        public ActionResult PatientsEntry()

        {

            return View();

        }

        [HttpPost]

        public ActionResult PatientsEntry(PatientEntry patient)

        {

            if(ModelState.IsValid)

            {

                bool details = obj.PatientEntry(patient);

                if (details == true)

                {

                    TempData["alertMessage"] = "Success!!!";

                    ModelState.Clear();

                    return View();

                }

                else

                {

                    ViewBag.SuccessMessage = "<p>Please try once!!!!!!</p>";

                    return View();

                }

            }

            else

            {

                ViewBag.SuccessMessage =null;

                return View();

            }

        }

        public ActionResult patientDetails()

        {

            dt = new DataTable();

            dt= obj.GetPatientEntry\_onParticularDate();

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

            for (int i = 0; i < dt.Rows.Count; i++)

            {

                PatientEntry patient = new PatientEntry();

                patient.ID = dt.Rows[i]["ID"].ToString();

                patient.FirstName = dt.Rows[i]["FirstName"].ToString();

                patient.LastName = dt.Rows[i]["LastName"].ToString();

                patient.Date = Convert.ToDateTime(dt.Rows[i]["date"]);

                patient.Address = dt.Rows[i]["Address"].ToString();

                patient.Age = Convert.ToInt32(dt.Rows[i]["Age"]);

                patient.Description = dt.Rows[i]["Description"].ToString();

                list.Add(patient);

            }

            return View(list);

        }

    }

}

The model is given here

Hide   Copy Code

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace HMSMODEL

{

   public class PatientEntry

    {

        public string ID { get; set; }

        public string FirstName { get; set; }

        public string LastName { get; set; }

        public int Age { get; set; }

        public string Address { get; set; }

        public string Description { get; set; }

        public DateTime Date { get; set; }

        public int EntryFee { get; set; }

        public string Gender { get; set; }

    }

}

Here is the complete Bussiniss Logic  for both Template entry and Patient Entry.

Hide   Shrink Image 30 for Creating Real Time  MVC Projects Step by Step with  n-layer architecture   Copy Code

using HMSMODEL;

using HospitalDLL;

using System;

using System.Collections.Generic;

using System.Data;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Data;

using System.Data.SqlClient;

namespace HospitalBLL

{

    public class HospitalComponents

    {

        HospitalHelper \_hospitalhelper = new HospitalHelper();

        DataTable dt = new DataTable();

        SqlCommand cmd;

        public DataTable getTemplateData()

        {

            dt = \_hospitalhelper.getData("select id ,Header ,LEFT(Description,200) as  Description from dbo.tbl\_Template");

            if (dt.Rows.Count > 0)

            {

                return dt;

            }

            else

            {

                return dt;

            }

        }

        public bool tabEntery(Template temp)

        {

            cmd = new SqlCommand("sp\_SaveTemplates");

            cmd.CommandType = CommandType.StoredProcedure;

            cmd.Parameters.AddWithValue("@Header", temp.Header);

            cmd.Parameters.AddWithValue("@Description", temp.Description);

            cmd.Parameters.AddWithValue("@Status", temp.Status);

           bool result=  \_hospitalhelper.DML(cmd);

            if(result==true)

            {

                return true;

            }

            else

            {

                return false;

            }

        }

        public bool editTemplate(Template temp)

        {

            cmd = new SqlCommand("sp\_EditTemplates");

            cmd.CommandType = CommandType.StoredProcedure;

            cmd.Parameters.AddWithValue("@Header", temp.Header);

            cmd.Parameters.AddWithValue("@Description", temp.Description);

            cmd.Parameters.AddWithValue("@Status", temp.Status);

            cmd.Parameters.AddWithValue("@id", temp.id);

            \_hospitalhelper.DML(cmd);

            return true;

        }

        public bool PatientEntry(PatientEntry patent)

        {

            patent.ID = Guid.NewGuid().ToString();

            cmd = new SqlCommand("SP\_PatientEntry");

            cmd.CommandType = CommandType.StoredProcedure;

            cmd.Parameters.AddWithValue("@EntryId", patent.ID);

            cmd.Parameters.AddWithValue("@FirstName", patent.FirstName);

            cmd.Parameters.AddWithValue("@LastName", patent.LastName);

            cmd.Parameters.AddWithValue("@Age", patent.Age);

            cmd.Parameters.AddWithValue("@Address", patent.Address);

            cmd.Parameters.AddWithValue("@Description", patent.Description);

            cmd.Parameters.AddWithValue("@EntryFee",patent.EntryFee);

            cmd.Parameters.AddWithValue("@Gender",patent.Gender);

            cmd.Parameters.AddWithValue("@Date", DateTime.Today);

            cmd.Parameters.AddWithValue("@Opptype","Save");

           bool result= \_hospitalhelper.DML(cmd);

            return result;

        }

        public DataTable GetPatientEntry\_onParticularDate()

        {

            dt = new DataTable();

            cmd = new SqlCommand("SP\_PatientEntry");

            cmd.CommandType = CommandType.StoredProcedure;

            cmd.Parameters.AddWithValue("@Opptype", "Getall");

            dt = \_hospitalhelper.GetALL(cmd);

            return dt;

        }

    }

}

This is the table structure for patient Entry.

Hide   Shrink Image 31 for Creating Real Time  MVC Projects Step by Step with  n-layer architecture   Copy Code

USE [HMS]

GO

*/\*\*\*\*\*\* Object:  Table [dbo].[PatientEntry]    Script Date: 04-09-2017 20:05:15 \*\*\*\*\*\*/*

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[PatientEntry](

    [Id] [int] IDENTITY(1,1) NOT NULL,

    [EntryId] [nvarchar](150) NOT NULL,

    [FirstName] [nvarchar](50) NULL,

    [LastName] [nvarchar](50) NULL,

    [Age] [int] NULL,

    [Address] [nvarchar](350) NULL,

    [Description] [nvarchar](max) NULL,

    [Date] [date] NULL,

    [EntryFee] [int] NULL,

    [Gender] [nvarchar](10) NULL,

 CONSTRAINT [PK\_PatientEntry] PRIMARY KEY CLUSTERED

(

    [EntryId] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY] TEXTIMAGE\_ON [PRIMARY]

GO

Here is the Stored Procedure for all Patient Related opperation.

Hide   Shrink Image 32 for Creating Real Time  MVC Projects Step by Step with  n-layer architecture   Copy Code

USE [HMS]

GO

*/\*\*\*\*\*\* Object:  StoredProcedure [dbo].[SP\_PatientEntry]    Script Date: 04-09-2017 20:03:37 \*\*\*\*\*\*/*

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

ALTER PROCEDURE [dbo].[SP\_PatientEntry]

(

@EntryId as nvarchar(150)=null,

@FirstName as nvarchar(50)=null,

@LastName as nvarchar(50)=null,

@Age as int=null,

@Address as nvarchar(350)=null,

@Description as nvarchar(max)=null,

@Date as Date=null,

@EntryFee as int=null,

@Gender as nvarchar(10)=null,

@Opptype as nvarchar(20)=null

)

AS

BEGIN

if (@opptype='Save')

begin

insert into [dbo].[PatientEntry]([EntryId],[FirstName],[LastName],[Age],[Address],[Description],[Date],[EntryFee],[Gender])

values(@EntryId,@FirstName,@LastName,@Age,@Address,@Description,GETDATE(),@EntryFee,@Gender)

end

if(@Opptype='Getall')

begin

select \* from [dbo].[PatientEntry]

end

END

Here is the code for the View.Here we have used HTML Grid to show the data.

Hide   Shrink Image 33 for Creating Real Time  MVC Projects Step by Step with  n-layer architecture   Copy Code

@model IEnumerable<HMSMODEL.PatientEntry>

@{

    ViewBag.Title = "patientDetails";

}

<script src="~/Scripts/jquery-1.10.2.min.js"></script>

<script src="~/Scripts/bootstrap.min.js"></script>

<link href="~/Scripts/Calender/calendrical.css" rel="stylesheet" />

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

<script src="http://ajax.googleapis.com/ajax/libs/jquery/1.11.3/jquery.min.js"></script>

<style type="text/css">

    .webgrid-table {

        font-family: "Trebuchet MS", Arial, Helvetica, sans-serif;

        font-size: 1.2em;

        width: 300px;

        display: table;

        border-collapse: separate;

        border: solid 1px;

        background-color: purple;

    }

        .webgrid-table td, th {

            border: 3px solid;

            padding: 3px 7px 2px;

            width: 230px;

        }

    .webgrid-header {

        background-color: whitesmoke;

        color: #FFFFFF;

        padding-bottom: 4px;

        padding-top: 5px;

        text-align: left;

        width: 20%;

    }

    .webgrid-footer {

    }

    .webgrid-row-style {

        padding: 3px 7px 2px;

    }

    .webgrid-alternating-row {

        background-color: #EAF2D3;

        padding: 3px 7px 2px;

    }

</style>

<script type="text/javascript">

    $(function () {

        $('.edit-mode').hide();

        $('.edit-user, .cancel-user').on('click', function () {

            var tr = $(this).parents('tr:first');

            tr.find('.edit-mode, .display-mode').toggle();

        });

            $.ajax({

                url: '/Home/Update/',

                data: JSON.stringify(UserModel),

                type: 'POST',

                contentType: 'application/json; charset=utf-8',

                success: function (data) {

                    window.location.href = window.location.href;

                }

            });

        });

    })

</script>

<br />

            <div>

                @{

                    var grid = new WebGrid(Model, canPage: true, rowsPerPage: 10);

                    grid.Pager(WebGridPagerModes.All);

                }

            </div>

            <h3>List of Patients</h3>

            <div>

                @grid.GetHtml(

                   headerStyle: "webgrid-header",

                footerStyle: "webgrid-footer",

                alternatingRowStyle: "webgrid-alternating-row",

                selectedRowStyle: "webgrid-selected-row",

                rowStyle: "webgrid-row-style",

                mode: WebGridPagerModes.All,

             columns: grid.Columns(

                grid.Column("ID", format: @<text>  <span class="display-mode">@item.ID </span> <label id="lbl\_Empid" class="edit-mode">@item.ID</label> </text>, style: "col1Width"),

                grid.Column(columnName: "FirstName", header: "First Name", format: @<text> <span class="display-mode">@item.FirstName <label id="lblfirstName"></label> </span>  <input type="text" id="txt\_firstName" value="@item.FirstName" class="edit-mode" /> </text>, style: "col2Width"),

                grid.Column(columnName: "LastName", header: "LastName", format: @<text> <span class="display-mode">@item.LastName <label id="lbl\_lastname"></label> </span>  <input type="text" id="txt\_lastName" value="@item.LastName" class="edit-mode" /> </text>, style: "col2Width"),

                grid.Column(columnName: "Description", header: "Description", format: @<text> <span class="display-mode">@item.Description <label id="lbl\_desc"></label> </span>  <input type="text" id="txt\_desc" value="@item.Description" class="edit-mode" /> </text>, style: "col2Width"),

                  grid.Column(columnName: "Address", header: "User Address", format: @<text> <span class="display-mode">@item.Address <label id="lbladdress"></label> </span>  <input type="text" id="txt\_address" value="@item.Address" class="edit-mode" /> </text>, style: "col2Width"),

                  grid.Column("Action", format: @<text>

                    <button class="edit-user display-mode" style="background-color: #1E90FF; border-bottom-style: groove">Edit</button>

                    <button class="save-user edit-mode" style="background-color: #FF5733; border-bottom-style: groove">Save</button>

                    <button class="cancel-user edit-mode" style="background-color: #FF5733; border-bottom-style: groove">Cancel</button>

                    <button class="Delete-user display-mode" style="background-color: #1E90FF; border-bottom-style: groove">Delete</button>

                </text>, style: "col3Width", canSort: true)))

            </div>

        </div>

## 

* 1. **TELL ME ABOUT YOUR PROJECT ARCHITECTURE:**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| HTTP |  |  |  | NHS System. | | | |  |  |  |  |  | SOR |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
| Hospital-1 |  |  |  |  | req |  |  | req | |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  | S.L |  | P.L | |  |  | I.L |  |  |  |
|  | WST |  |  |  | resp |  |  | resp | |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| HTTP |  |  |  |  |  |  |  |  |  |  |  |  | WEB- |
|  |  |  |  |  |  |  |  |  |  |  |  | SERVICE |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
| Hospital -2 | Project Architecture (Provider Side). | | | | | | |  |  |  |  |  |  |
|  |  |  |  |  |  |

A part of my project architecture, I am working as provider side. In this provider side having three layer

A). Service Layer.

B). Process Layer.

C). Integration Layer.

Here our NHS interacted with the SOR(DataBase) system.

* Here with respect to the project architecture, Hospitals will send the HTTPs request, by this request going to the provider side (a) Service Layer (b) Process Layer (c) Integration Layer, in this service layer we are validating the data and calling the process layer & send it to the process layer, by this data we perform the converting point logic.

**4. TELL ME ABOUT YOUR PROJECT TECHNICAL FLOW:**

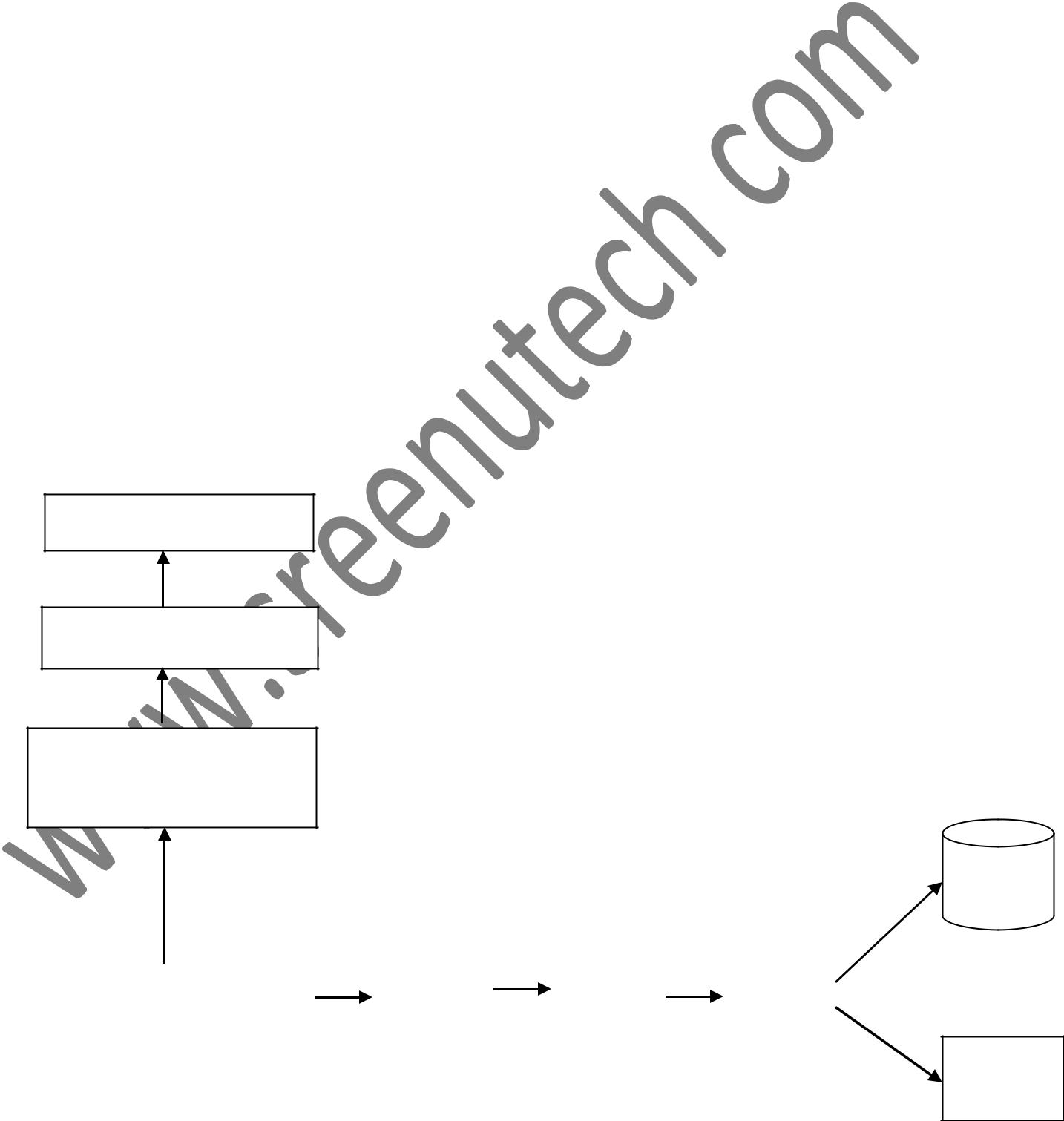
A part of my project technical flow, here first we are taking the business requirement from the client and by this business requirement we prepare the Data Mapping Sheet, and we prepare the Schema file, and by using this XSD file we generate the WSDL, and then create the Artifacts (creating the .java and .classes file).

And by the artifacts create Service layer implementation class, in this class calling the interface of whatever generated by the WSDL file.

Here is the Technical Flow of my project (Partner Integration).

Business Requirement

Data Mapping Sheet



XSD Schema File,

WSDL File

SOR

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Artifacts |  | S.Layer |  | P.Layer |  | I. Layer |
|  |  |  |  |  |  |  |

Web-

services

Fig: - Technical flow of Spine(Patient Details) Service.

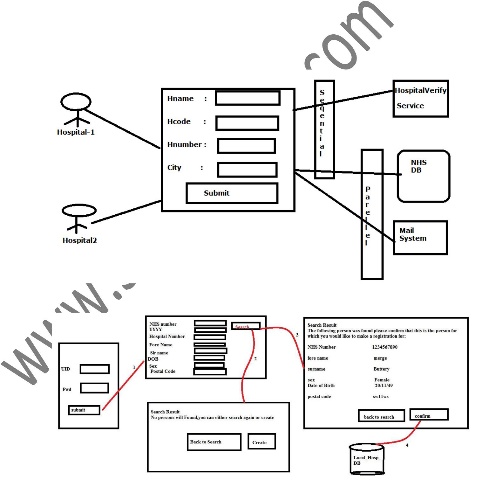
As of our project, here first we take the request and response element by the Data mapping sheet, by this we create the XSD files and then create the WSDL files also. After completion of this work we create the Artifacts and by this create the Layers like-

A). Service Layer. B). Process Layer. C). Integration Layer.

In the Service layer, calling the interface by the implementation class and we prepare the web service request object and then building the request for process layer and then call the process layer, In process layer building the request for Integration layer and calling the Integration layer, In integration layer transfer the request data to the SOR system or web-service for storing the data permanent.

**5. DATA Flow Diagram**

**5.1. Registration Module**



**5.2. Patient Details Module**

**5.3.Import and Export Modules**

1. **How you are handling exception in your project?**

As part of my project we are handling the exception, generally in my project we have handle user defined exception, based on the business requirement we have categorized user defined exception into different types of exception like-

A). Business Exception. B). System Exception. C). Service Exception.

Here, when the user sends the request element (like- fistName, LastName, D.O.B, MobNo and etc.) data, if that data is not valid then we are throwing the some user defined exception. This data not sending to the process layer, here only we are handle the exception.

* 1. Business Exception :- Here business exception means if the User sending the request element , that request element is not valid then that time we handle the business exception. To handle this, here we are writing the some user defined exception class and some Enum class/properties files/XML files. By this class we handle the business exception. In business exception coming the some DataException this one also related to the request data only.

#64/3RT, SR Nagar, Near Community Hall, Hyderabad

Contacts: **+91-8019697596, 040-40061799**

1. System Exception: - This Exception coming at the time of running the service, this one also related to the hardware system, this exception will be handling by the Admin team. This exception we handle in Throw able class exception.
2. Service Exception: - In service exception we handle the service related exception like- DBConnection error.

In my project having layers like- service layer, process layer, Integration layer.

In Integration layer, as per business requirement we are categorized different kind of user defined exceptions.

Some exception we are throwing, the exception in process layer and service layer.

Here, I am explaining the some exceptions.

NumberFormateException, ClassCastException, ClassNotFoundException, FileNotFoundException, NullPointerException and etc.



NullPointerException :- To resolve the NullPointerException before creating the object we can write some if\_else (for avoiding this exception) statement and checkout this, and make sure that object not null, if object whatever you pass is null then that time you will get NullPointerException.

ClassNotFoundException :- At the time of loading the class if that class is not available then that you will get the ClassNotFoundException.

To resolve this Exception you make sure whatever you load the class, that class is available. Another style of handle this exception is you should pass the class otherwise you will get this exception.

1. **What are the exceptions you have used in project during deployment?** A part of our project we are facing the different type of exception like,

A). OutOfMemoryException. B). ResourcesFailedException. C). NullPointerException, and etc.

A). OutOfMemoryException : - To handle this exception, we raise request/ticket to support/admin team. They will resolve issue and update the ticket, in stage server we handled or resolve this issue.

Here have some reasons.

(1). Heap Dumps.

(2). More no. of application. B). ResourceFailedException: -

1. **What is Encapsulation?Where you have used in project?**

Encapsulation is the ability to package data. It is all about packaging related stuff together and hides them from external elements. Encapsulation importance here is for binding data and methods together.

**How you are implemented polymorphism in your project?**

Normally we are developing our application interface base approach for example service and serviceImpl so my controller need to inject serviceImpl to get business functionality so in this case in controller class we are injection service bean by taking service interface reference as below .

**public interface** BankService {

**public void** doTransaction();

}

[@Service](mailto:@Service)

**public class** BankServiceImpl **implements** BankService{

[@Override](mailto:@Override)

**public void** doTransaction() {

// LOGIC

}

}

[@Controller](mailto:@Controller)

**public class** BankController {

[@Autowired](mailto:@Autowired)(required = **true**)

**private** BankService service;

}

Here internally IOC container instantiate my service bean as below approach

BankService service=**new** BankServiceImpl();//Runtime polymorphism

**What is serialization? Have you implement serialization in your project?**

Serialization is a process where we can change state of object to the file over the network or simply we

can transfer our object from one layer to another layer, that’s why java provides streaming API.

In my project I used in pojo class means my pojo class should be implements from Serializable interface because that business object will be transfer over the network, that’s why it’s recommended to implements BO object from Serializable interface.

**How to create web-services project and spring project using maven?**

Simply we have to create one maven project like maven-archetype-webapp then add the dependency from local repository if available else download from central repository in pom.xml.

Add spring dependency along with JAX-WS implementation class dependency in pom.xml

**.how you implement exception handling in your project?**

Answer:

Normally we are developing Spring based application so in Spring to handle exception multiple predefined class is there .so simply in my project we are throwing custom exception from service layer and when my controller call service it will catch that exception by using Spring

Aop , We have to create a class which should be annoted as [@ControllerAdvice](mailto:@ControllerAdvice) and we have to take one method whose return type is Model And View and method should be annoted as

[@ExceptionHandler](mailto:@ExceptionHandler) so in this method we have to write the logic for map the exception. And return the same view which is return by controller class at the time of exception raise. And we have to return some user understandable message by view page.

**Note: Both return logical view should be same**