T18 ASP.NET MVC Transactions, stored procedures

# Tasks

1. Implement web page that will allow creation of new employee.
2. Implement web page for editing employee details. First implement it using plain SQL UPDATE and then using stored procedure for update of employee details.
3. Add link to /Employee/Index page at the top navigation menu.
4. Improve employee edit web page: show ReportsTo as a dropdown with a list of employee names

# Implementation

# Insertion of a new employee instructions

* 1. First we will add method in EmployeeManager that will allow insertion of employees:

public void CreateEmployee(Employee emp)

{

using (DbConnection conn = new SqlConnection(ConnectionStr))

{

using (DbCommand cmd = conn.CreateCommand())

{

cmd.CommandText = @"INSERT INTO Employee(

[LastName]

,[FirstName]

,[ReportsTo]) VALUES (

@LastName,

@FirstName,

)"; /\*add missing fields yourself!\*/

DbParameter pLn = cmd.CreateParameter();

pLn.DbType = System.Data.DbType.String;

pLn.ParameterName = "@LastName";

pLn.Value = emp.LastName;

cmd.Parameters.Add(pLn);

DbParameter pFn = cmd.CreateParameter();

pFn.DbType = System.Data.DbType.String;

pFn.ParameterName = "@FirstName";

pFn.Value = emp.FirstName;

cmd.Parameters.Add(pFn);

/\*add missing fields yourself!\*/

conn.Open();

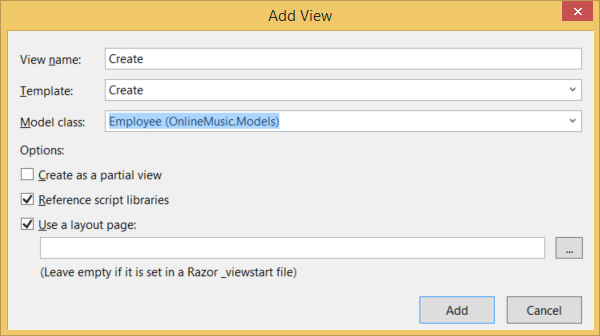
cmd.ExecuteNonQuery();

}

}

}

* 1. Add view for insertion(creation) of a new employee. Open EmployeeController select Create() method and right click on it. In pop up menu select Add View. View Name: Create, Template: Create, Model class: Employee



* 1. Modify method:

public ActionResult Create(FormCollection collection)

to look like this:

[HttpPost]

public ActionResult Create(Employee emp)

{

EmployeeManager manager = new EmployeeManager ();

manager.CreateEmployee(emp);

return RedirectToAction("Index");

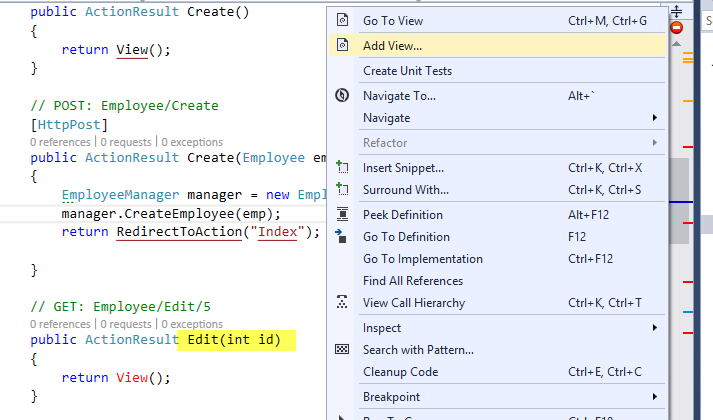
}

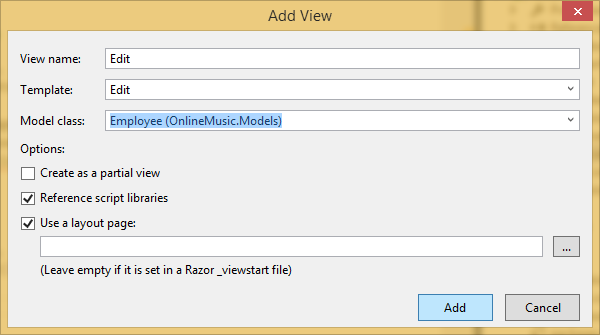
* 1. Now run web application and navigate to http://localhost/Employee/Create
  2. Populate values for First/Last name and Title and press Create button. Check if employee was inserted in database.

# Edit employee instructions

## Add View

* + 1. Right click inside Edit() of EmployeeController and click Add View





## Configure Get Method

* + 1. Inside EmployeeManager class prepare method that helps to retrieve specific employee data from database

public Employee GetEmployeeById(int employeeId)

{

Employee result = null;

using (DbConnection conn = new SqlConnection(ConnectionStr))

{

using (DbCommand cmd = conn.CreateCommand())

{

cmd.CommandText = @" "; //change text here

cmd.AddParameter("EmployeeId", DbType.Int32, employeeId);

conn.Open();

using (DbDataReader rdr = cmd.ExecuteReader())

{

if (rdr.Read()) //notice if

{

result = new Employee()

{

EmployeeId = rdr.GetInt32(0),

LastName = rdr.GetString(1),

FirstName = rdr.GetString(2),

ReportsTo = rdr.IsDBNull(3)

? (int?)null : rdr.GetInt32(3),

BirthDate = rdr.IsDBNull(4) ? (DateTime?)null : rdr.GetDateTime(4),

Email = rdr.GetString(5)

};

}

}

}

}

return result;

}

* + 1. Call the function created above from EmployeeController’s Get Edit method

public ActionResult Edit(int id)

{

EmployeeManager manager = new EmployeeManager();

Employee employee = manager.GetEmployeeById(id);

return View(employee);

}

* + 1. Run the project and see how employee details are fetched from database.

## Use SQL Update statement to save data

* + 1. Extension method to simplify addition of parameters, in /DAL folder create new class DbCommandExtensionMethods and the following code into it:

public static class DbCommandExtensionMethods

{

public static IDbDataParameter AddParameter(this IDbCommand cmd, string name, System.Data.DbType type, object value, ParameterDirection direction = ParameterDirection.Input)

{

var p = cmd.CreateParameter();

p.ParameterName = name;

p.Direction = direction;

if(value == null)

{

p.Value = DBNull.Value;

} else

{

p.DbType = type;

p.Value = value;

}

cmd.Parameters.Add(p);

return p;

}

}

* + 1. Inside EmployeeManager add following method

public void UpdateEmployee(Employee emp)

{

using (DbConnection conn = new SqlConnection(ConnectionStr))

{

using (DbCommand cmd = conn.CreateCommand())

{

//Set command text

cmd.CommandText = @" ";

cmd.AddParameter("@FirstName", DbType.String, emp.FirstName);

//Add other parameters

conn.Open();

cmd.ExecuteNonQuery();

}

}

}

* + 1. Modify HttpPost Edit method of EmployeeController to use Employee manager

[HttpPost]

public ActionResult Edit(Employee emp)

{

EmployeeManager manager = new EmployeeManager();

manager.UpdateEmployee(emp);

return RedirectToAction("Index");

}

* + 1. Run the solution and navigate to localhost/Employee/Edit/1

## Use SQL Stored procedure to save data

* + 1. Create the stored procedure in SQL Server Management Studio

CREATE PROCEDURE udpUpdateEmployee

@EmployeeId int

, @LastName nvarchar(200)

, @FirstName nvarchar(200)

, @ReportsTo int

, @BirthDate datetime

, @Email nvarchar(200)

AS

BEGIN

UPDATE [dbo].[Employee]

SET [LastName] = @LastName

,[FirstName] = @FirstName

,[ReportsTo] = @ReportsTo

,[BirthDate] = @BirthDate

,[Email] = @Email

WHERE [EmployeeId] = @EmployeeId;

END

* + 1. Add new method in EmployeeManager that used stored procedure above to update employee details

public void UpdateEmployeeStoredProc(Employee emp)

{

using (DbConnection conn = new SqlConnection(ConnectionStr))

{

using (IDbCommand cmd = conn.CreateCommand())

{

cmd.CommandText = @"udpUpdateEmployee";

cmd.CommandType = CommandType.StoredProcedure;

var pFn = cmd.CreateParameter();

pFn.DbType = System.Data.DbType.String;

pFn.ParameterName = "@FirstName";

pFn.Value = emp.FirstName;

cmd.Parameters.Add(pFn);

var pLn = cmd.CreateParameter();

pLn.DbType = System.Data.DbType.String;

pLn.ParameterName = "@LastName";

pLn.Value = emp.LastName;

cmd.Parameters.Add(pLn);

var pR = cmd.CreateParameter();

pR.ParameterName = "@ReportsTo";

if (emp.ReportsTo.HasValue)

{

pR.DbType = System.Data.DbType.Int32;

pR.Value = emp.ReportsTo;

}

else

{

pR.Value = DBNull.Value;

}

cmd.Parameters.Add(pR);

cmd.AddParameter("@BirthDate", System.Data.DbType.DateTime, emp.BirthDate);

cmd.AddParameter("@Email", System.Data.DbType.String, emp.Email);

cmd.AddParameter("@EmployeeId", System.Data.DbType.Int32, emp.EmployeeId);

conn.Open();

cmd.ExecuteNonQuery();

}

}

}

* + 1. Modify post Edit method in EmployeeController to call method that in turn calls stored procedure

[HttpPost]

public ActionResult Edit(Employee emp)

{

EmployeeRepository empRepo = new EmployeeRepository();

//empRepo.UpdateEmployee(emp);

empRepo.UpdateEmployeeStoredProc(emp);

return RedirectToAction("Index");

}

* + 1. Run the solution and navigate to localhost/Employee/Edit/1 and see the magic happen

1. Adding link to employees link in top navigation
   * 1. Open \_Layout.cshtml under Views/Shared and replace navigation bar code

<ul class="nav navbar-nav">

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

**<li>@Html.ActionLink("Employees List", "Index", "Employee")</li>**

<li>@Html.ActionLink("About", "About", "Home")</li>

<li>@Html.ActionLink("Contact", "Contact", "Home")</li>

</ul>

# Add DropDownList in Edit view

* 1. Drop down list for ReportsTo (using ViewBag to pass additional data to view):
     1. Add line in bold to EmployeeController:

public ActionResult Edit(int id)

{

EmployeeRepository empRepo = new EmployeeRepository();

Employee emp = empRepo.GetEmployeeById(id);

**ViewBag.ReportsToOptions = empRepo.GetAllEmployees();**

return View(emp);

}

* + 1. In Edit.cshtml view of Employee folder comment out the line that generates EditorFor ReportsTo property and add the following line:

@Html.**DropDownListFor**(model => model.ReportsTo, new SelectList(ViewBag.ReportsToOptions, "EmployeeId", "LastName", Model.ReportsTo), "--Select--", new { htmlAttributes = new { @class = "form-control" } })

* 1. Drop down list for ReportsTo (using EmployeeEditViewModel to pass additional data to view):
     1. Add Model in Models folder:

public class EmployeeEditViewModel

{

public Employee Emp { get; set; }

public IList<Employee> ReportsToOptions;

}

* + 1. In EmployeeController generate another pair of methods for Edit that uses the model created above:

#region EmployeeEditViewModel

// GET: Employee/Edit2/5

public ActionResult Edit2(int id)

{

EmployeeRepository empRepo = new EmployeeRepository();

Employee emp = empRepo.GetEmployeeById(id);

EmployeeEditViewModel empModel = new EmployeeEditViewModel();

empModel.Emp = emp;

empModel.ReportsToOptions = empRepo.GetAllEmployees();

return View(empModel);

}

// POST: Employee/Edit2/5

[HttpPost]

public ActionResult Edit2(Employee emp)

{

EmployeeRepository empRepo = new EmployeeRepository();

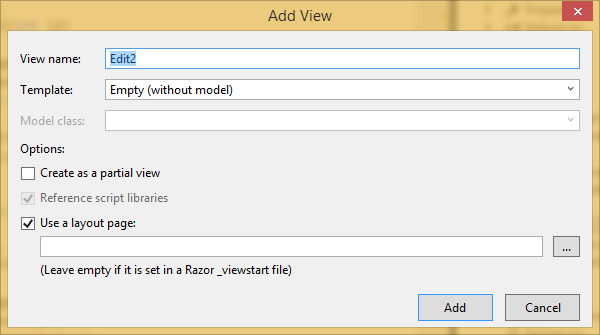
empRepo.UpdateEmployee(emp);

return RedirectToAction("Index");

}

#endregion

* + 1. Add View Edit2.cshtml that uses not employee class, but uses EmployeeEditViewModel class.



* + 1. Add the following code inside Edit2.cshtml

@model OnlineMusic.Models.EmployeeEditViewModel

@{

ViewBag.Title = "Edit2";

}

<h2>Edit</h2>

@using (Html.BeginForm())

{

@Html.AntiForgeryToken()

<div class="form-horizontal">

<h4>Employee</h4>

<hr />

@Html.ValidationSummary(true, "", new { @class = "text-danger" })

@Html.HiddenFor(model => model.Emp.EmployeeId)

<div class="form-group">

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

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

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

@Html.ValidationMessageFor(model => model.Emp.LastName, "", new { @class = "text-danger" })

</div>

</div>

<div class="form-group">

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

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

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

@Html.ValidationMessageFor(model => model.Emp.FirstName, "", new { @class = "text-danger" })

</div>

</div>

<div class="form-group">

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

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

@Html.DropDownListFor(model => model.Emp.ReportsTo, new SelectList(Model.ReportsToOptions, "EmployeeId", "LastName", Model.Emp.ReportsTo), "--Select--", new { htmlAttributes = new { @class = "form-control" } })

@Html.ValidationMessageFor(model => model.Emp.ReportsTo, "", new { @class = "text-danger" })

</div>

</div>

<div class="form-group">

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

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

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

@Html.ValidationMessageFor(model => model.Emp.BirthDate, "", new { @class = "text-danger" })

</div>

</div>

<div class="form-group">

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

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

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

@Html.ValidationMessageFor(model => model.Emp.Email, "", new { @class = "text-danger" })

</div>

</div>

<div class="form-group">

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

<input type="submit" value="Save" class="btn btn-default" />

</div>

</div>

</div>

}

<div>

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

</div>

@section Scripts {

@Scripts.Render("~/bundles/jqueryval")

}

# Home work

1. Create web page for editing customers