ASP.NET MVC

MVC 5

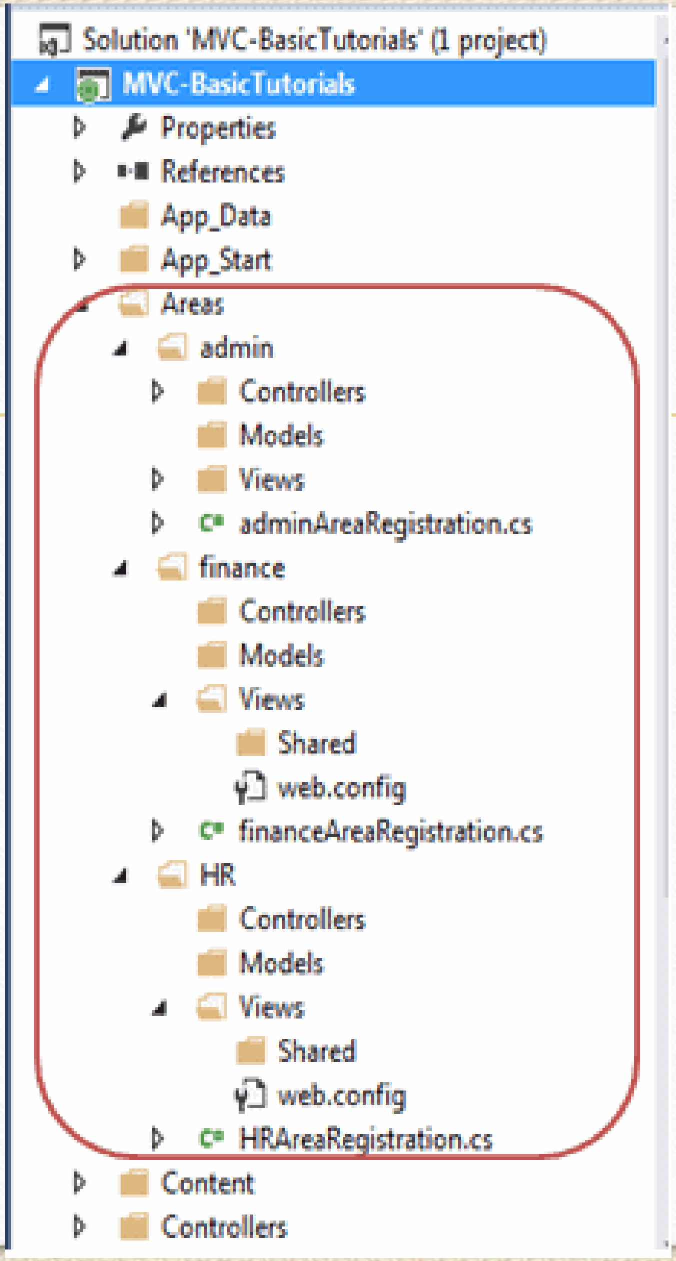
MVC Core

Eng. Basma Hussien

MVC Areas

Areas

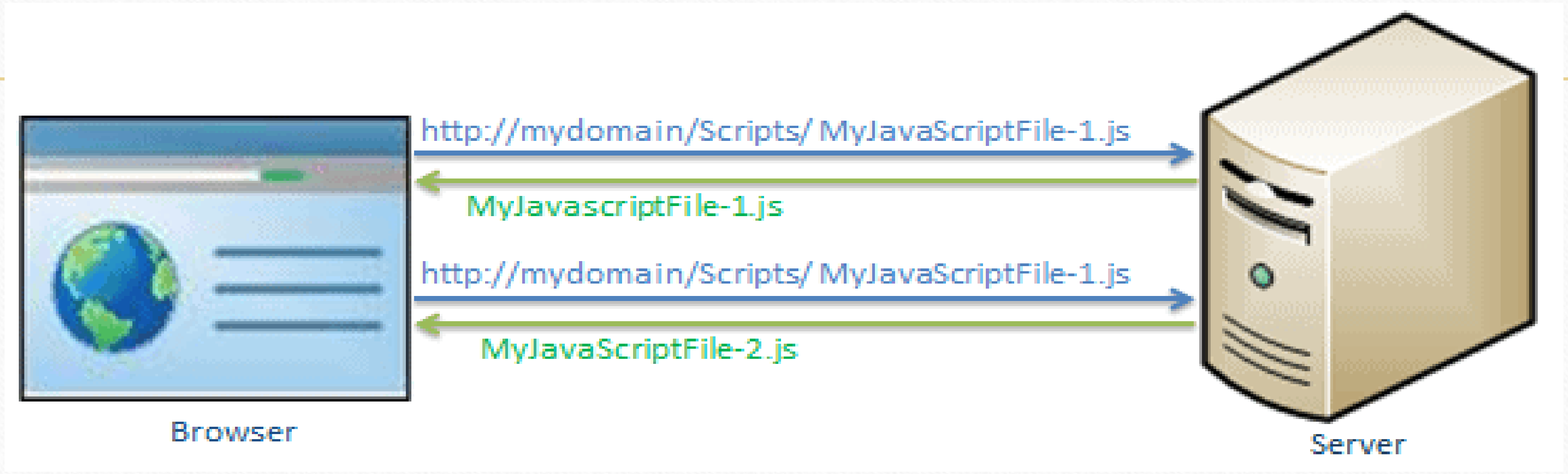
- Area allows us to partition the large application into smaller units where each unit contains a separate MVC folder structure, same as the default MVC folder structure.
- For example, a large enterprise application may have different modules like admin, finance, HR, marketing, etc. So an Area can contain a separate MVC folder structure for all these modules.
- The AreaRegistration class overrides the ***RegisterArea*** method to map the routes for the area.
- Area Route: ***baseurl/areaname/controllername/{actionname}***



```
Solution 'MVC-BasicTutorials' (1 project)
◢ MVC-BasicTutorials
    ▷ Properties
    ▷ References
      App_Data
    ▷ App_Start
    ◢ Areas
        ◢ admin
            ▷ Controllers
              Models
            ▷ Views
            ▷ adminAreaRegistration.cs
        ◢ finance
              Controllers
              Models
            ◢ Views
                  Shared
                  web.config
            ▷ financeAreaRegistration.cs
        ◢ HR
              Controllers
              Models
            ◢ Views
                  Shared
                  web.config
            ▷ HRAreaRegistration.cs
    ▷ Content
    ▷ Controllers
```

Bundling and Minification

Bundling

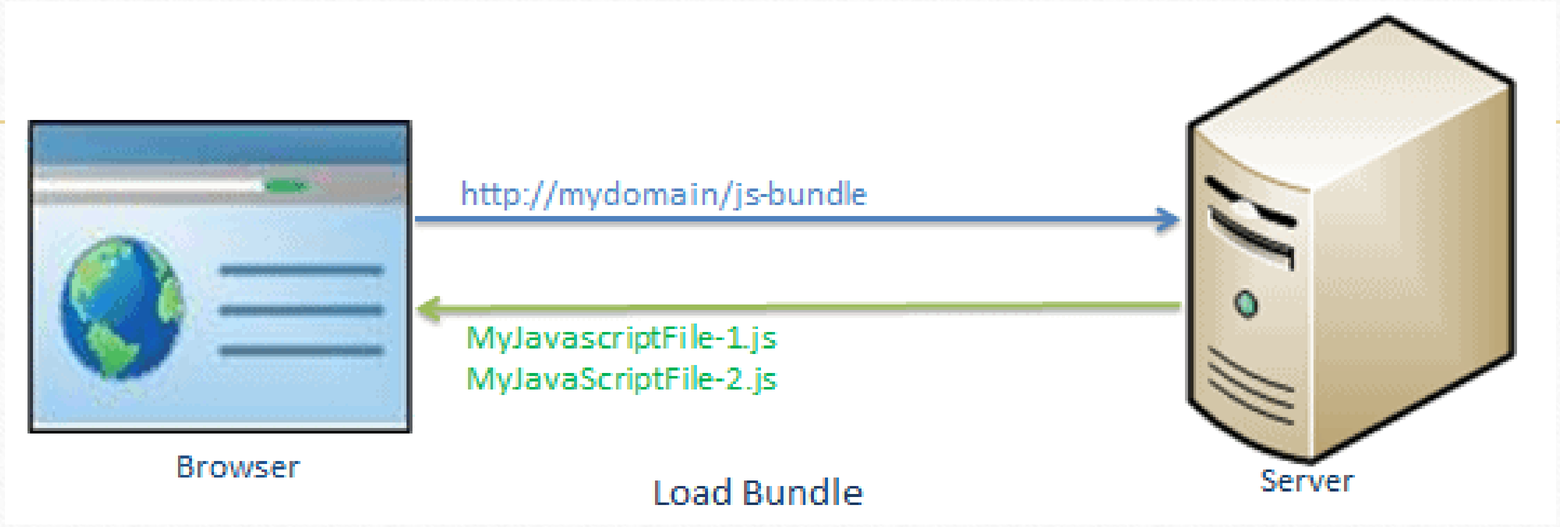

http://mydomain/Scripts/ MyJavaScriptFile-1.js
MyJavascriptFile-1.js
http://mydomain/Scripts/ MyJavaScriptFile-1.js
MyJavaScriptFile-2.js
Browser
Server

- In the above figure, the browser sends two separate requests to load two different JavaScript file MyJavaScriptFile-1.js and MyJavaScriptFile-2.js.

Bundling and Minification

- ASP.NET MVC 4 (and later) supports the same bundling and minification framework included in ASP.NET 4.5.
- This system reduces requests to your site by combining several individual script references or CSS request into a single request.
- It also “minifies” the requests through a number of techniques, such as shortening variable names and removing whitespace and comments.

Bundling (Cont.)



- The bundling technique in ASP.NET MVC allows us to load more than one JavaScript file, MyJavaScriptFile-1.js and MyJavaScriptFile-2.js in one request

Minification

- Minification technique optimizes script or CSS file size by removing unnecessary white space and comments and shortening variable names to one character.

```
sayHello = function(name){
    //this is comment
    var msg = "Hello" + name;
    alert(msg);
}
```

After Minification:

```
sayHello=function(n){var t="Hello"+n;alert(t)}
```

Bundle Types

MVC 5 includes following bundle classes in ***System.web.Optimization*** namespace:

- **ScriptBundle**: ScriptBundle is responsible for JavaScript minification of single or multiple script files.
- **StyleBundle**: StyleBundle is responsible for CSS minification of single or multiple style sheet files.
- **DynamicFolderBundle**: Represents a Bundle object that ASP.NET creates from a folder that contains files of the same type.

ScriptBundle class

- The ScriptBundle class represents a bundle that does JavaScript minification and bundling.
- You can create style or script bundles in ***BundleConfig*** class under ***App_Start*** folder in an ASP.NET MVC project.

```
using System.Web;
using System.Web.Optimization;

public class BundleConfig
{
    public static void RegisterBundles(BundleCollection bundles)
    {
        bundles.Add(new ScriptBundle("~/bundles/bs-jq-bundle").Include(
                                "~/Scripts/bootstrap.js",
                                "~/Scripts/jquery-3.3.1.js"));
    }
}
```

- In the above example, we created a new bundle by creating an instance of the ScriptBundle class and specified the virtual path and bundle name in the constructor.
- The ~/bundles/ is a virtual path and bs-jq-bundle is a bundle name.
- Then, we added two js files, bootstrap.js, and jquery-3.3.1.js in this bundle. The bundles.Add() method is used to add new bundles into the BundleCollection.

Including Bundles in webpages

- To include the above bs-jq-bundle in your webpage, use Scripts.Render() method in the layout view

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <meta    name="viewport"    content="width=device-width,    initial-
scale=1.0">
    <title>@ViewBag.Title</title>
    @Scripts.Render("~/bundles/bootstrap")
</head>
<body>
    @*html code removed for clarity *@
</body>
</html>
```

Error Handling

Data Annotations
&
Validation

DataAnnotation

- Data Annotations are nothing but certain validations that we put in our models to validate the input from the user.

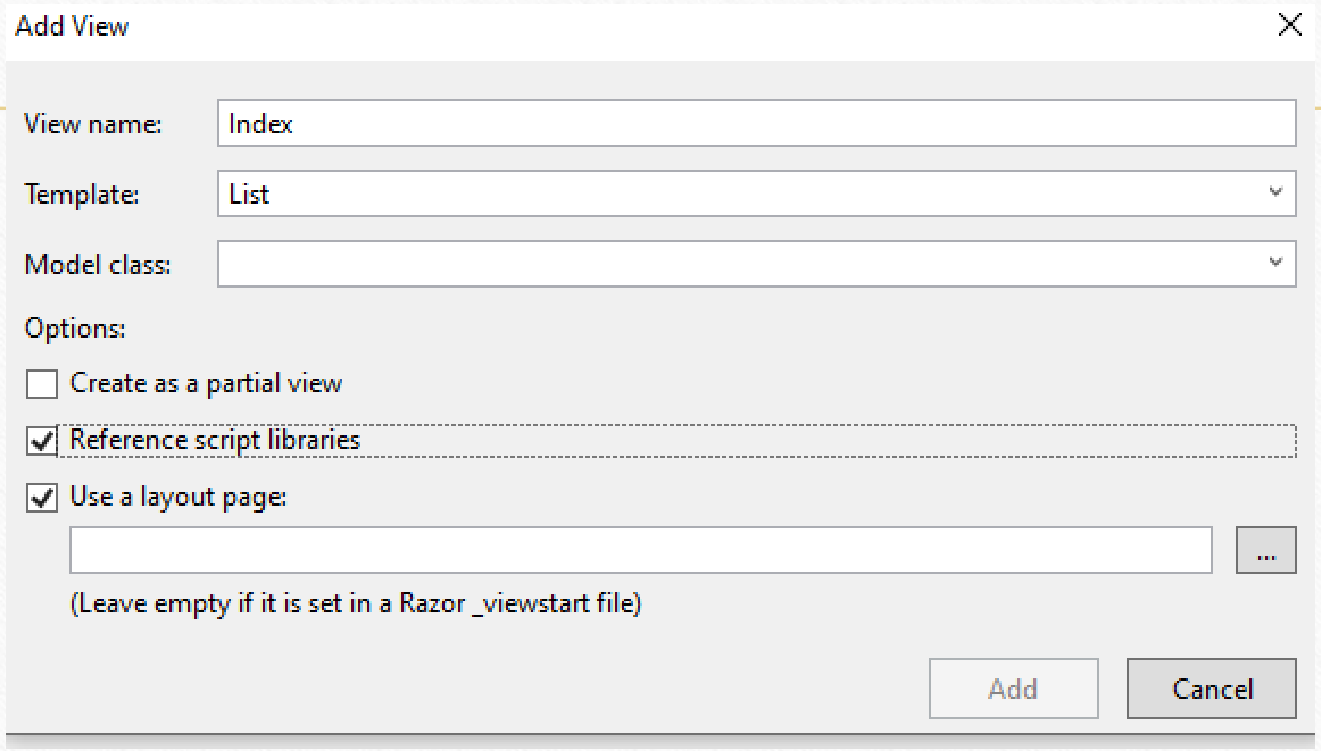
- ASP.NET MVC provides a unique feature in which we can validate the models using the Data Annotation attribute. Import the following namespace to use data annotations in the application.

System.ComponentModel.DataAnnotations

HTMLHelpers & DataAnnotation

- @Html helpers (Like "@Html.EditorFor") looks at DataAnnotation and make it like metadata about this field, and encode it into "data-...", so that the "jquery.validate.js" could apply cliet-side validation according to these metadata
-
- "jquery.validate.js": jquery validation plugin to support client-side validation
- "jquery.validate.unobtrusive.js": acts like a bridge between jquery validate attributes and jquery validation plugins

Enable Client-Side Validation


Add View

View name: Index

Template: List

Model class:

Options:

☐ Create as a partial view

✓ Reference script libraries

✓ Use a layout page:

(Leave empty if it is set in a Razor _viewstart file)

to enable client-side validation while adding a **strongly typed view** -> check: Reference script libraries

Enable Client-Side Validation

to enable client-side validation at any view page :

```
@section Scripts
{
    @Scripts.Render("~/bundles/jqueryval")
}
```

- this will add at the bottom of the view (after jquery and bootstrap librarys):
- this will render anything in the bundle "jqueryval"

Demo