CSE3200 : Software Development - V

Introduction to ASP.NET MVC
LAB - 3

Outline

- Introduction
- Understanding MVC
- Folder Structure
- Controller
- ActioinResult
- Models
- Views
- Razor
- ViewBag, ViewData
- Strongly Typed View
- Shared Views
- Layout Views
- Partial Views
- Conventional URL Routing
- Attribute URL Routing
- Model Binding
- EF Code-First Approach
- EF Database-First Approach
- HTML Helpers
- Validations

Introduction to MVC

- ASP.NET is a web application framework from Microsoft
- It is open source
- Applies the general Model-View-Controller Pattern
- Separates the data access logic from display logic
- Popular MVC Frameworks: ASP.NET MVC, Ruby on Rails, Express
- MVC has 3 main aspects:
  - **Model**
  - **View**
  - **Controller**

Understanding MVC Pattern

- **Models -** A set of classes that describes the data you are working with
  - Domain Model
  - View Model

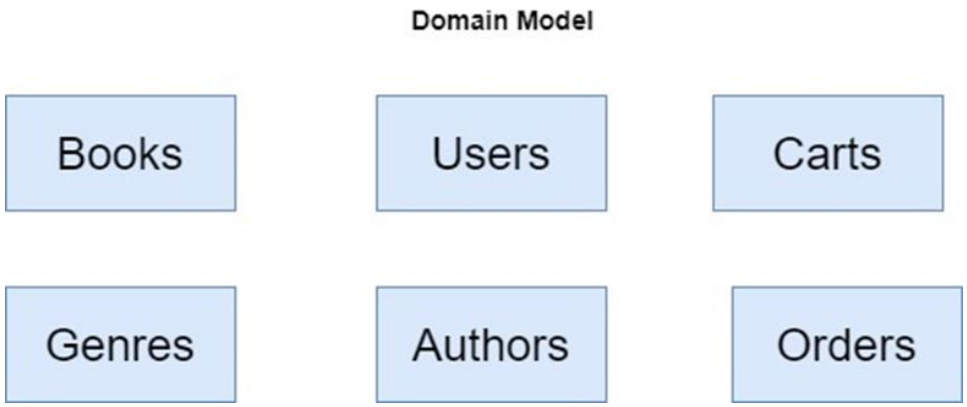
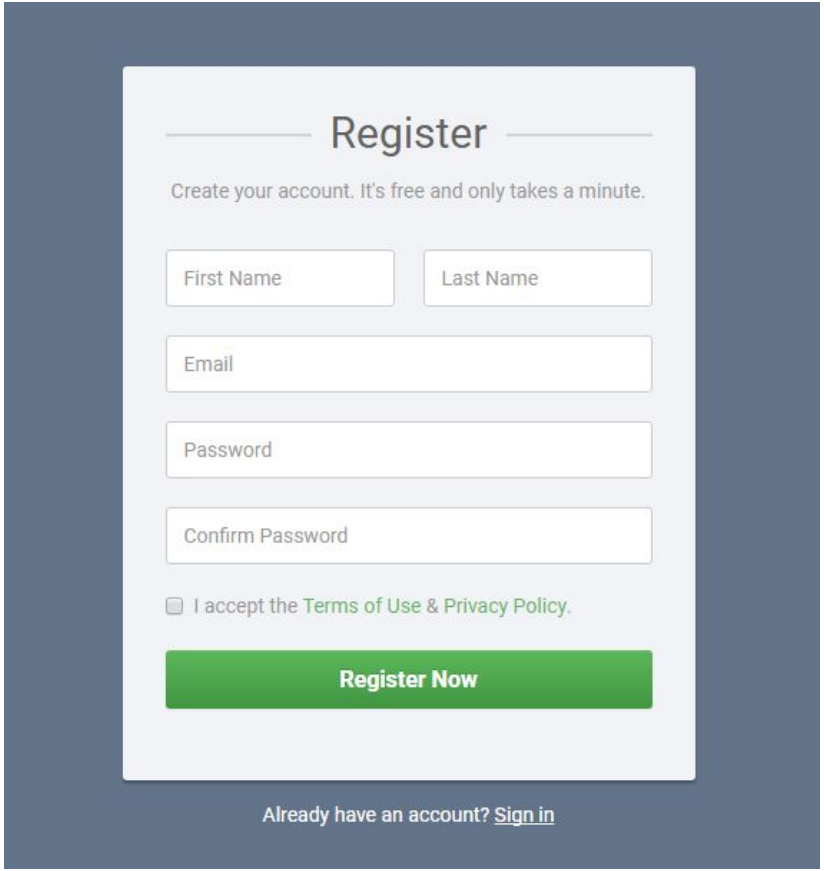


Fig: Example of a Domain Model

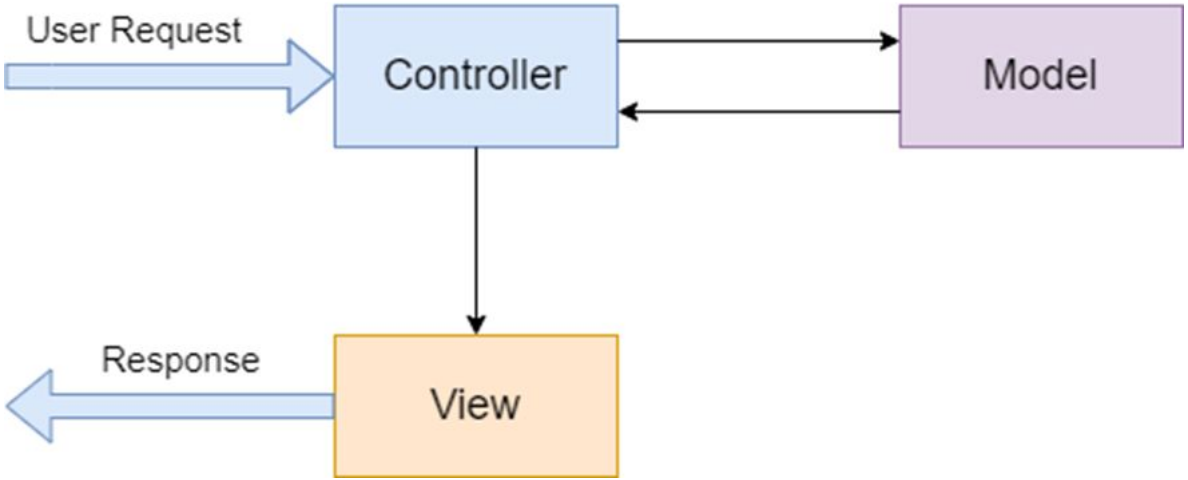
Understanding MVC Pattern

- **View -** Defines how the application's UI will be displayed
- The HTML Markup that we display to the user
- Reads data from Model



Understanding MVC Pattern

- **Controllers** - a set of classes that receives user request, fetch suitable resources for the task and select proper view to respond back to user
- Controller receives request from browser, call the model, call the view



Folder Structure of MVC

PorjectFolder:

- **\App_Start** - Contains the files that needs to be executed on the first request
- **App_Data** - Contains SQL Server Local DB database files
- **\Controllers** - Contains all controller classes
- **\Models** - Contains all model classes
- **\Views** - Contains all views
- **\Views\web.config** - Contains configuration settings for all views
- **\Global.asax** - Contains application level and session level events
- **\packages.config** - Contains list of NuGet packages currently installed in the project
- **\Web.config** - Contains web application configuration settings, that needs to be initiallized on each request

Controller

- Controller is a class
- Optionally, it’s a public class
- Controller should be inherited from “System.Web.MVC.Controller” class
- Controller’s name should have suffix “Controller”. Ex - ProductController
- All the methods in controller class are by default **Action Method**
- It is common to write the return type of Action Methods as ActionResult

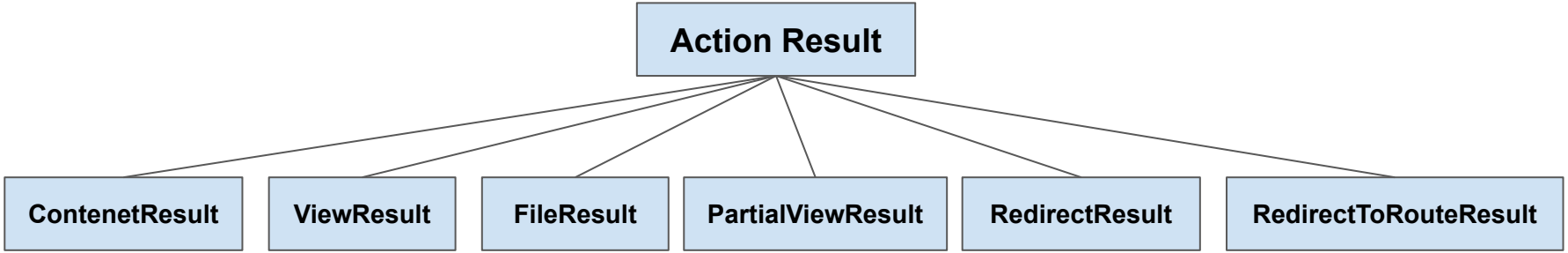
Controller

```
namespace MvcApplication1.Controllers
{
    public class ProductController : Controller
    {
        //
        // GET: /Products/

        public ActionResult Index()
        {
            // Add action logic here
            return View();
        }
    }
}
```

ActionResult

- ActionResult is a class that represents **“result of action method”**
- It is recommended to define action methods return type as **“ActionResult”**
- ActionResult is an abstract class that has several child classes



Methods of different types of Action Result

ContentResult	Content(string Content, string ContentType)
ViewResult	View(string ViewName)
FileResult	File(string FilePath, string ContentType)
JsonResult	Json(object data, JsonRequestBehavior behavior)
RedirectResult	Redirect(string url)
RedirectToRouteResult	RedirectToAction(string ActionName, string Controllername)
PartialViewResult	PartialView(string ViewName)

Models

- Model is a class that defines structure of the data that you want to store/display
- It contains business logic (e.g. validation)
- Model will be called by Controller and View
- Domain Model: Represents the structure of the data you want to store in database table (e.g. user information)
- View Model: Represents the structure of the data you want to display to user (e.g. login page)

Model

```
public class User
{
    0 references
    public int Id { get; set; }
    1 reference
    public string Name { get; set; }
    0 references
    public string Email { get; set; }
    0 references
    public string Address { get; set; }
}
```

View & Razor View

- View is a combination of HTML and C# code
- C# code written within @{} symbol
- Razor View Engine provides set of syntaxes to write C# code in view
- Razor View Engine is responsible to render the view as html
- File extension is .cshtml (.vbhtml)

Razor Syntax

```
@{
    int age = ViewBag.age;
}
```

Fig: Razor Block

```
@if (age < 18)
{
    <div class="alert alert-dismissible alert-danger">
        <strong>You are not allowed to use this site</strong>
    </div>
}
else{
    <div class="alert alert-dismissible alert-success">
        <strong>You can use this site</strong>
    </div>
}
```

Fig: Razor if-else

Razor Syntex

```
@for (int i = 0; i < size; i++)
{
    <p>@i</p>
}
```

Fig: Razor for

```
@foreach (var user in ViewBag.users)
{
    <tr class="table-primary">
        <td>@user.ID</td>
        <td>@user.Name</td>
        <td>@user.Email</td>
    </tr>
}
```

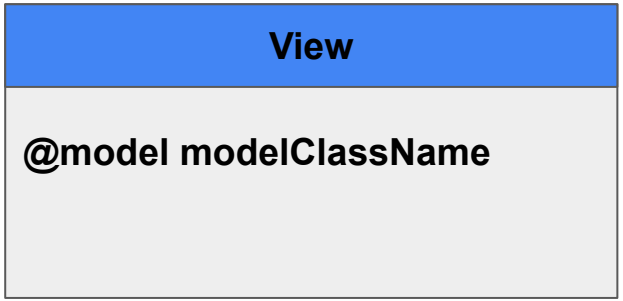
Fig: Razor foreach

Passing Data From Controller to View

- ViewBag
- ViewData
- model

Strongly Typed Views

- Views that is associated with specific type of model class is called strongly typed view
- Strongly typed views have to specify the model class name with @model derivative at the top of the view
- Strongly type view can receive object of that model from the controller



Shared Views

- Shared views are present in the “Views\Shared” folder
- Shared views can be called from any controller
- The views that belong to multiple controllers are created as shared views
- It first searches the view in “Views/ControllerName” folder. If no view is found it searches in the “Views/Shared” folder

Layout Views

- Layout views contains the common parts of UI. Such as logo, haeder, footer, menubar, sidebar etc.
- **@RenderBody()** method represents the reserved area for the actual content of the view

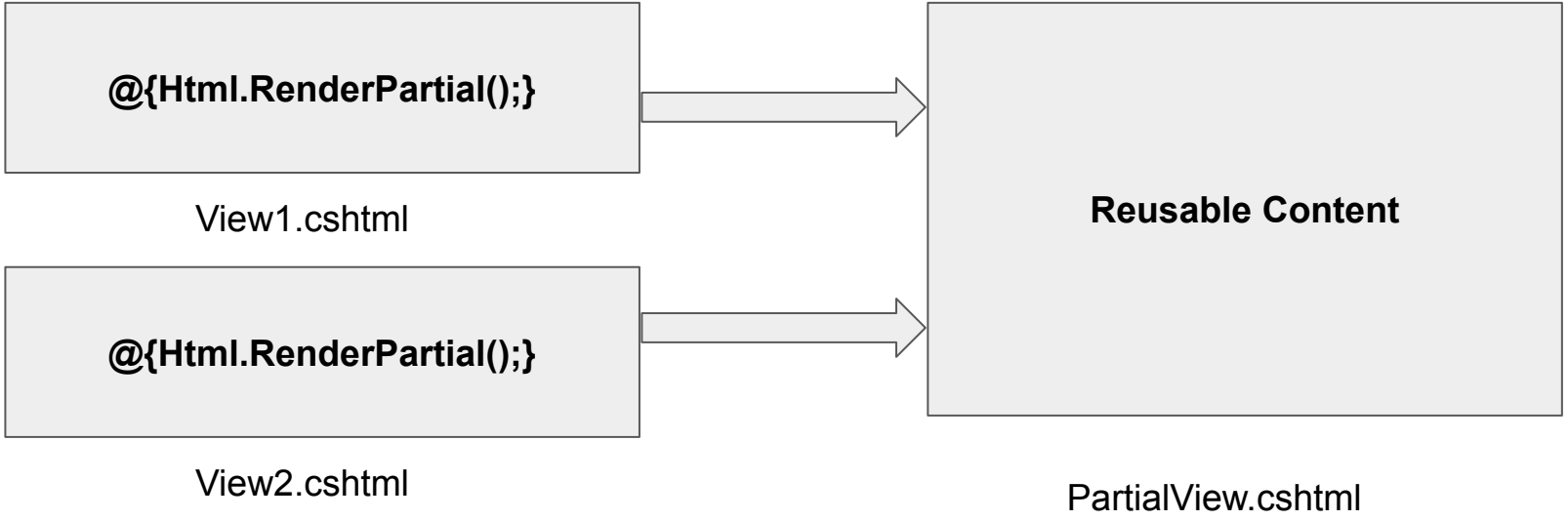
Layout Views

- Data can be shared from a normal view to layout view using Viewbag
- **_ViewStart.cshtml** in Views folder defines the default layout view of all the views of a folder
- There can be multiple layout views in a project (e.g. one layout for user section and one section for admin section)

```
@{
    Layout = "~/Views/Shared/_Layout.cshtml";
}
```

Partial Views

- Partial view is a small view that contains the content that can be shared among multiple views



URL Routing

- URL Routing is a pattern matching system that monitors the incoming request URL and figure out what to do with that
- It allows you to create the meaningful URLs, instead of mapping to physical files
- Route is a URL pattern which includes literals/parameters
- Literal is fixed, whereas parameter is variable
- Ex - Users/Details/{userid}

/Users/Index	/Users/Contact	/Users/Details/1

Attribute Routing

- Conventional Routing is difficult for developers to understand which route for which action methods
- Some routes for multiple action methods, some for other. Overall it looks cumbersome
- To overcome this, Attribute Routing is introduced in MVC5
- Attribute routing should be enabled using route.MapMvcAttributesRoutes() in RouteConfig.cs

```
[“url”]

Public ActionResult MethodName()
{

}
```