Seanca 3

Views

- View is an Asp.net web page that contains presentation logic.
- Views are present in “Views\controllername” folder.
- The view’s file extension should be “.cshtml”.

View can contains two types of presentation logic:

1. **Server side presentation logic:**
   This executes on server. It requires to use a “view engine”.
2. **Client side presentation logic:**
   This executes on client (browser), after server side presentation logic.

- For every controller, there should be a folder in the “Views” folder, where the controller name and folder name should be SAME.
- It is recommended to maintain the same name for both view and action method. When you return a view

Syntax

```
<html>
<head>
<title>Title here</title>
</head>
<body>
Content here
</body>
</html>
```

View Name

- It is used to specify the view name while calling a view from controller. By default, it is recommended to maintain the action method name and view name as same. In case of you have a different name for the view, then you need to specify the view name in the controller, to call it.

Syntax of Action method with ViewResult with Name

```
public ViewResult Actionname()
{
 return View("view name");
}
```

View Engine

- A view engine provides a set of rules to write c# code (server side code) in the view.
- View Engine is also responsible to render the view as html.

ASP.NET MVC supports two types of view engines:

1. ASPX View Engine
2. Razor View Engine

ASPX View Engine

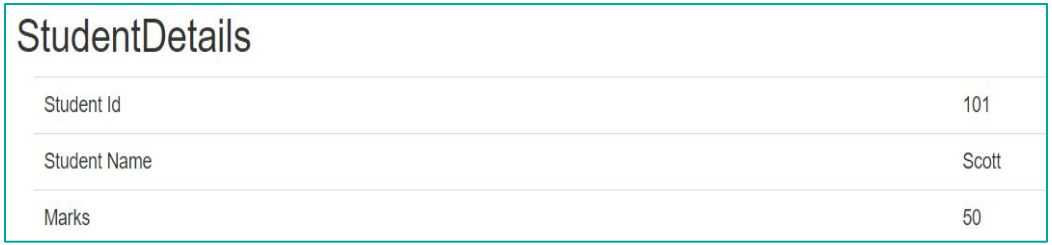
```
<%
    c#.net code
%>
```

Razor View Engine

```
@{
    c#.net code
}
```

Razor Expressions

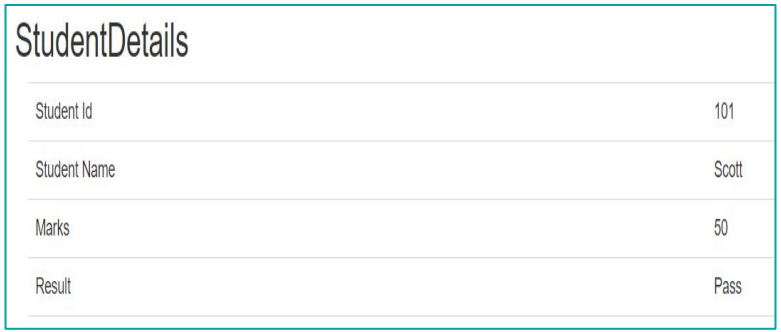
```
<table class="table">
        <tr>
            <td>Student Id</td>
            <td>@ViewBag.StudentId</td>
        </tr>
        <tr>
            <td>Student Name</td>
            <td>@ViewBag.StudentName</td>
        </tr>
        <tr>
            <td>Marks</td>
            <td>@ViewBag.Marks</td>
        </tr>
    </table>
```



Razor Code Blocks

Based on the mark, show if student passes/fails

```
@{
    string StudentResult;
    if (ViewBag.Marks >=35)
    {
        StudentResult = "Pass";
    }
    else
    {
        StudentResult = "Fail";
    }
}
```



Razor If

Display different output based on condition:

```
@if (StudentResult == "Pass")
  {
   <span class="text-success">Congratulations</span>
   }
    else
    {
    <span class="text-danger">Better luck next time!</span>
    }
```

- You should always use { }

Razor For

Show a list of number of all semesters for the student

```
<td>Semesters</td>
  <td>
  <ul>
 @for (int i = 1; i <= ViewBag.NoOfSemesters; i++)
    {
   <li>@i</li>
    }
   </ul>
  </td>
```

Razor Foreach

Reading data from arrays or collections

```
<td>Subjects</td>
   <td>
    <ul>
  @foreach (var subject in ViewBag.Subjects)
    {
  <li>@subject</li>
     }
   </ul>
   </td>
```

Assignment

Receive the "amount" value as input from query string. For example, the url is "localhost:1234/home/index?amount=60743".

Based on the given "amount" value, generate the highest currency denominations. Consider, you have denominations of 1000, 500, 100, 50, 10, 5, and 1.

For example, if the input is "60743", output should be:
1000 X 60 = 60000
500 X 1 = 500
100 X 2 = 200
10 X 4 = 40
1 X 3 = 3

Denomination	Count	Amount
1000	60	60000
500	1	500
100	2	200
10	4	40
1	3	3

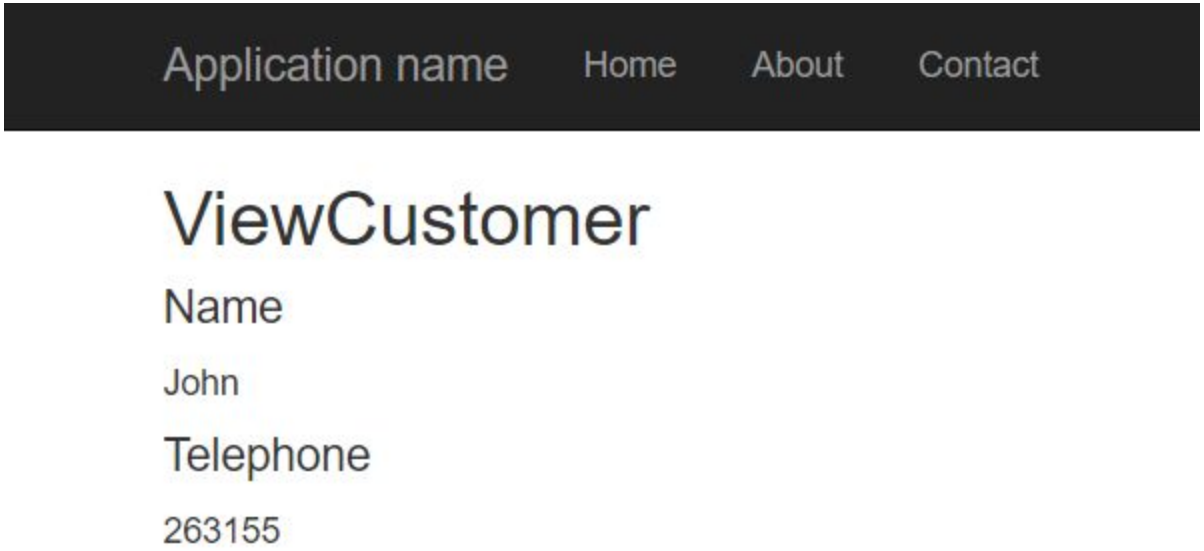
Models

- Model is a class that contains structure of the data to the displayed in the view.
- Contains business logic.
- Model class contains properties to define fields.

For example, “Employee” model class contains properties like “EmpID”, “EmpName” and “Salary”. Controller creates an object for the model class & pass them to the view. Then the view can access data from model object.

- Models are present in “Models” folder.
- Model classes are present in “Projectname.Models” namespace.

Show Customer Data Example



Customer Model

```
public class Customer
  {
     public String Id { get; set; }
     public string Name  {get; set;}
     public string Telephone { get; set; }
  }
```

Customer View

```
@model  Seanca3ModelExample.Models.Customer
@{
    ViewBag.Title = "ViewCustomer";
  }

<h2>ViewCustomer</h2>
<h4>Name</h4>
@Model.Name
<h4>Telephone</h4>
@Model.Telephone
```

Interfaces

- An interface is a language construct that is similar to a class (in terms of syntax) but is fundamentally different.
- An interface is simply a declaration of the capabilities (or services) that a class should provide.

```
public interface ITaxCalculator
{
int Calculate();
}
```

- An interface can only declare methods and properties, but not fields (because fields are about implementation detail).
- Members of an interface do not have access modifiers.

Interfaces Advantages

- Testability

It improves unit testing, testing different piece of code, that are independent.

- Extensibility

Changing code is easier, you can extend the solution with new features without a lot of changes

Abstract classes vs Interfaces

- An abstract class can have non abstract methods, interfaces can not
- An abstract class can have variables, an interface can not
- An abstract class can have constructor
- A class can implement any number of interfaces but a subclass can at most use only one abstract class.

Example

```
public class VideoEncoder
{
    private readonly IList<INotificationChannel> _notificationChannels;

    public VideoEncoder()
    {
        _notificationChannels = new List<INotificationChannel>();
    }

    public void Encode(Video video)
    {
        // Video encoding logic
        // ...

        foreach (var channel in _notificationChannels)
            channel.Send(new Message());
    }

    public void RegisterNotificationChannel(INotificationChannel channel)
    {
        _notificationChannels.Add(channel);
    }
```

```
public class MailNotificationChannel : INotificationChannel
{
    public void Send(Message message)
    {
        Console.WriteLine("Sending mail...");
    }
}

public interface INotificationChannel
{
    void Send(Message message);
}
```

Assignment

Show a table of products in a view called Index.cshtml. A product has an Id, Name and Price

Solution must contain:

- ✔ A Product class Model
- ✔ A Product controller
- ✔ An action method called that creates a list of Products and show them to the view.

Products

Product ID	Product Name	Rate
101	AC	45000
102	Mobile	38000
103	Bike	94000

Assignment

- Create details link for each project name, that opens a view which shows all details of that product

QUESTIONS