

# Controller in ASP.NET MVC

Tran Khai Thien

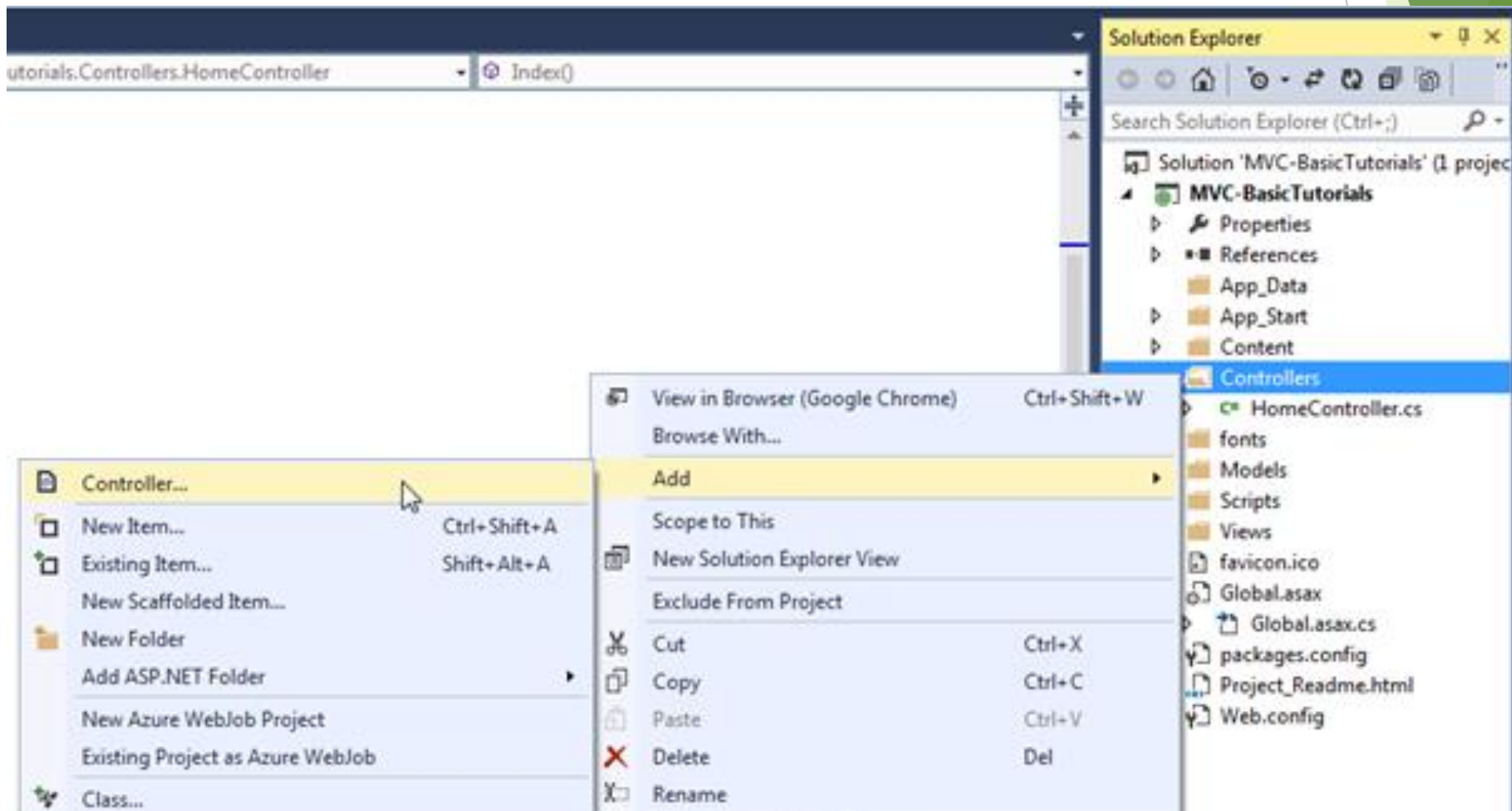
# Introduction

In this lesson, you will learn about the **Controller** in ASP.NET MVC.

- ▶ The **Controller** in MVC architecture handles any incoming URL request. Controller is a class, derived from the base class `System.Web.Mvc.Controller`. Controller class contains **public methods** called **Action** methods. Controller and its action method ***handles incoming browser requests, retrieves necessary model data and returns appropriate responses.***
- ▶ In ASP.NET MVC, every controller class name must end with a word "Controller". For example, controller for home page must be HomeController and controller for student must be StudentController. Also, every controller class must be located in Controllers folder of MVC folder structure.

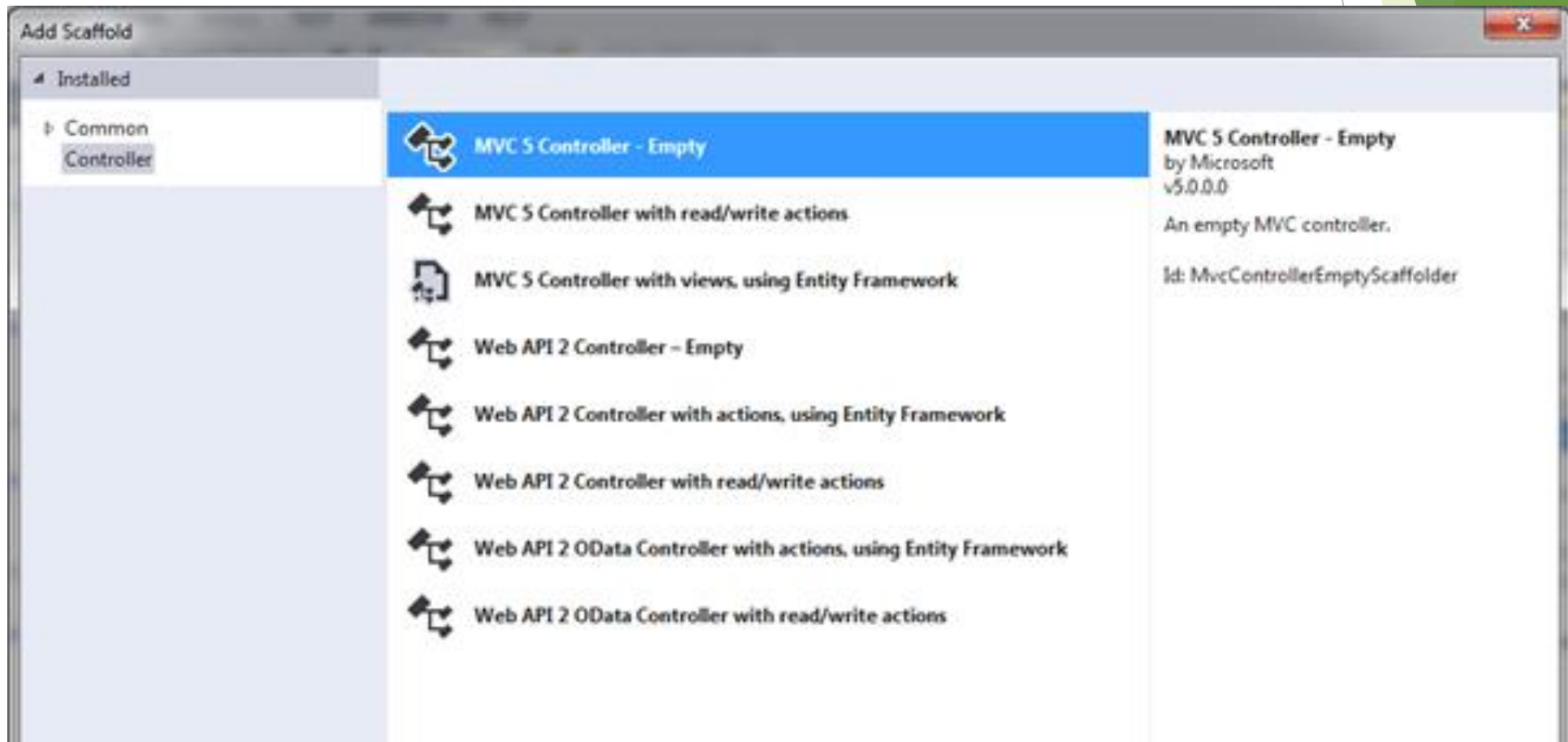
# Adding New Controller

- In the Visual Studio, right click on the **Controllers** folder -> select **Add** -> click on **Controller**.



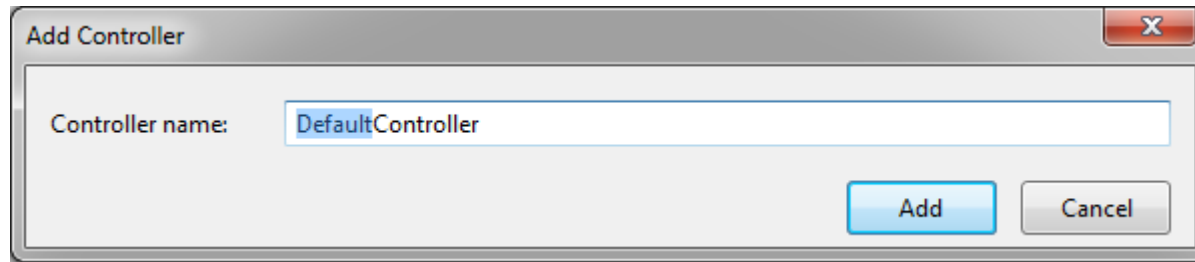
# Adding New Controller

- This opens Add Scaffold dialog as shown below. (Visual Studio 2013 introduced the Add New Scaffold Item dialog. This dialog replaced the Add View/Add Controllers dialog seen in the earlier version of Visual Studio.)

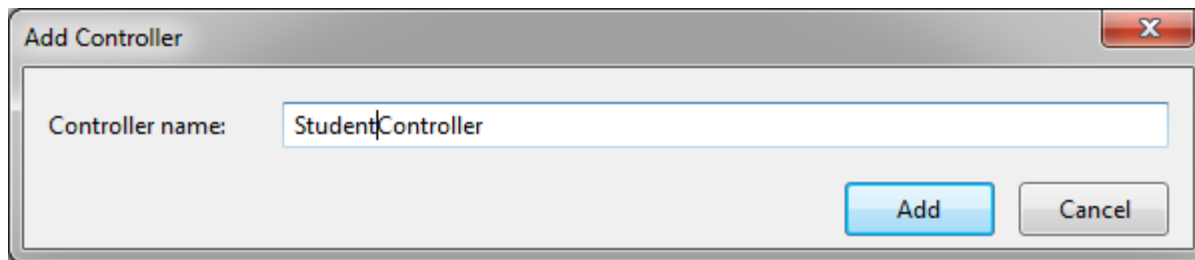


# Adding New Controller

- Add Scaffold dialog contains different templates to create a new **controller**. We will learn about other templates later. For now, select "**MVC 5 Controller - Empty**" and click **Add**. It will open Add Controller dialog as shown below:



- In the Add Controller dialog, enter the name of the controller. Remember, controller name must end with Controller. Let's enter StudentController and click **Add**.



# Adding New Controller

- This will create StudentController class with Index method in StudentController.cs file under Controllers folder, as shown below.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;

namespace MVC_BasicTutorials.Controllers
{
    public class StudentController : Controller
    {
        // GET: Student
        public ActionResult Index()
        {
            return View();
        }
    }
}
```

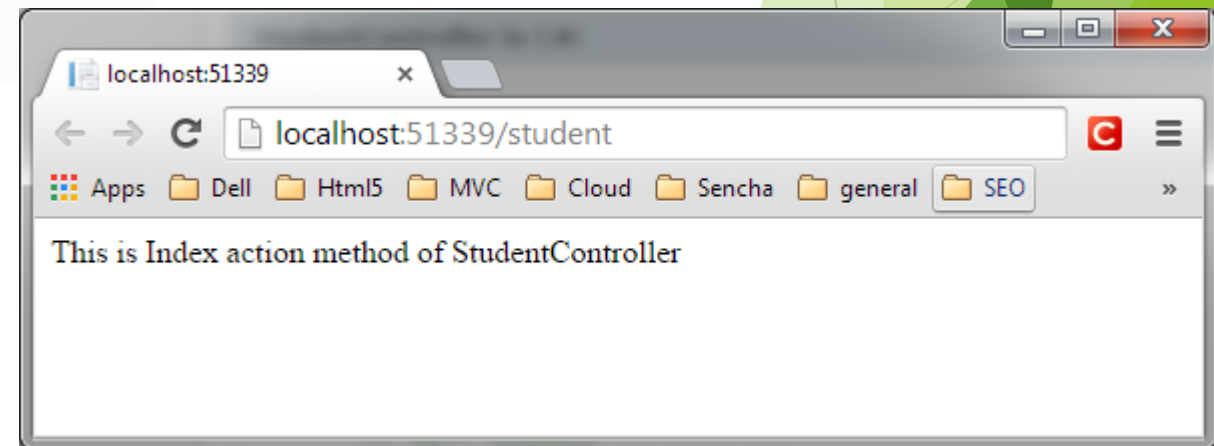
As you can see, the StudentController class is derived from Controller class. Every controller in MVC must derived from this abstract Controller class. This base Controller class contains helper methods that can be used for various purposes.

# Adding New Controller

- ▶ Now, we will return a dummy string from Index action method of above StudentController. Changing the return type of Index method from ActionResult to string and returning dummy string is shown below. You will learn about ActionResult in the next section.

```
namespace MVC_BasicTutorials.Controllers
{
    public class StudentController : Controller
    {
        // GET: Student
        public string Index()
        {
            return "This is Index action method of StudentController";
        }
    }
}
```

We have already seen in the routing section that the URL request *http://localhost/student* or *http://localhost/student/index* is handled by the Index() method of StudentController class.



# Controller

## Points to Remember :

- ▶ A Controller handles incoming URL requests. MVC routing sends request to appropriate controller and action method based on URL and configured Routes.
- ▶ All the public methods in the Controller class are called **Action** methods.
- ▶ A Controller class must be derived from ***System.Web.Mvc.Controller*** class.
- ▶ A Controller class name must end with "Controller".
- ▶ New controller can be created using different scaffolding templates. You can create custom scaffolding template also.

**Q: What is the relationship between StudentController and Student?**



# Action method

- ▶ All the public methods of a Controller class are called **Action methods**. They are like any other normal methods with the following restrictions:
  1. Action method must be public. It cannot be private or protected
  2. Action method cannot be overloaded
  3. Action method cannot be a static method.
- ▶ The following is an example of Index action method of StudentController

The diagram illustrates the structure of the `StudentController` class and its `Index()` method. It shows the following components and relationships:

- Student Controller class**: Points to the `public class StudentController` declaration.
- Base Controller class**: Points to the `: Controller` inheritance part of the class declaration.
- Return type**: Points to the `ActionResult` type in the `Index()` method signature.
- Action method**: Points to the `Index()` method signature.
- View() defined in base Controller class**: Points to the `return View();` statement inside the `Index()` method body.

```
public class StudentController : Controller
{
    // GET: Student
    public ActionResult Index()
    {
        return View();
    }
}
```

# Default Action method

- ▶ Every controller can have default action method as per configured route in RouteConfig class. By default, Index is a default action method for any controller, as per configured default root as shown below.

```
routes.MapRoute(  
    name: "Default",  
    url: "{controller}/{action}/{id}/{name}",  
    defaults: new { controller = "Home",  
                    action = "Index",  
                    id = UrlParameter.Optional  
    });
```

- ▶ However, you can change the default action name as per your requirement in RouteConfig class.

# ActionResult

- MVC framework includes various result classes, which can be return from an action methods. There result classes represent different types of responses such as html, file, string, json, javascript etc. The following table lists all the result classes available in ASP.NET MVC.

Result Class	Description
ViewResult	Represents HTML and markup.
EmptyResult	Represents No response.
ContentResult	Represents string literal.
FileContentResult/ FilePathResult/ FileStreamResult	Represents the content of a file
JavaScriptResult	Represent a JavaScript script.
JsonResult	Represent JSON that can be used in AJAX
RedirectResult	Represents a redirection to a new URL
RedirectToRouteResult	Represent another action of same or other controller
PartialViewResult	Returns HTML from Partial view
HttpUnauthorizedResult	Returns HTTP 403 status

# ActionResult

- The Index() method of StudentController in the above figure uses View() method to return ViewResult (**which is derived from ActionResult**). The View() method is defined in base Controller class. It also contains different methods, which automatically returns particular type of result as shown in the below table.

Result Class	Description	Base Controller method
ViewResult	Represents HTML and markup.	View()
EmptyResult	Represents No response.	
ContentResult	Represents string literal.	Content()
FileContentResult, FilePathResult, FileStreamResult	Represents the content of a file	File()
JavaScriptResult	Represent a JavaScript script.	JavaScript()
JsonResult	Represent JSON that can be used in AJAX	Json()
RedirectResult	Represents a redirection to a new URL	Redirect()
RedirectToRouteResult	Represent another action of same or other controller	RedirectToRoute()
PartialViewResult	Returns HTML	PartialView()
HttpUnauthorizedResult	Returns HTTP 403 status	

# Action method Parameters

- Every action methods can have input parameters as normal methods. It can be primitive data type or complex type parameters as shown in the below example.

```
[HttpPost]
public ActionResult Edit(Student std)
{
    // update student to the database

    return RedirectToAction("Index");
}

[HttpDelete]
public ActionResult Delete(int id)
{
    // delete student from the database whose id matches with specified id

    return RedirectToAction("Index");
}
```

# Action method

## Points to Remember

- ▶ All the public methods in the Controller class are called Action methods.
- ▶ Action method has following restrictions.
  - Action method must be public. It cannot be private or protected.
  - Action method cannot be overloaded.
  - Action method cannot be a static method.
- ▶ ActionResult is a base class of all the result type which returns from Action method.
- ▶ Base Controller class contains methods that returns appropriate result type e.g. View(), Content(), File(), JavaScript() etc.

# Action Selectors

- ▶ Action selector is the **attribute** that can be applied to the action methods. **It helps routing engine to select the correct action method to handle a particular request.** MVC 5 includes the following action selector attributes:
  1. ActionName
  2. NonAction
  3. ActionVerbs

# Action Selectors

## ActionName

- ▶ ActionName attribute allows us to specify a different action name than the method name. Consider the following example.

```
public class StudentController : Controller
{
    public StudentController()
    {
    }

    [ActionName("find")]
    public ActionResult GetById(int id)
    {
        // get student from the database
        return View();
    }
}
```

In the example, we have applied ActionName("find") attribute to GetById action method. So now, action name is "find" instead of "GetById". This action method will be invoked on *http://localhost/student/find/1* request instead of *http://localhost/student/getbyid/1* request.



# Action Selectors

## NonAction

- ▶ NonAction selector attribute indicates that a public method of a Controller is not an action method. Use NonAction attribute when you want public method in a controller but do not want to treat it as an action method.
- ▶ For example, the GetStudent() public method cannot be invoked in the same way as action method in the following example.

```
public class StudentController : Controller
{
    public StudentController()
    {
    }

    [NonAction]
    public Student GetStudent(int id)
    {
        return studentList.Where(s => s.StudentId == id).FirstOrDefault();
    }
}
```

# Action Selectors

## Points to Remember:

- ▶ MVC framework routing engine uses Action Selectors attributes to determine which action method to invoke.
- ▶ Three action selectors attributes are available in MVC 5
  - ActionName
  - NonAction
  - ActionVerbs
- ▶ ActionName attribute is used to specify different name of action than method name.
- ▶ NonAction attribute marks the public method of controller class as non-action method. It cannot be invoked.

# ActionVerbs

- ▶ The ActionVerbs selector is used when you want to control the selection of an action method **based on a Http request method**. For example, you can define two different action methods with the same name but one action method responds to an HTTP Get request and another action method responds to an HTTP Post request.
- ▶ MVC framework supports different ActionVerbs, such as `HttpGet`, `HttpPost`, `HttpPut`, `HttpDelete`, `HttpOptions` & `HttpPatch`. You can apply these attributes to action method to indicate the kind of Http request the action method supports. If you do not apply any attribute then it considers it a **GET request by default**.

# Action Verbs

- The following figure illustrates the HttpGET and HttpPOST action verbs.

http://localhost/Student/Edit/1

HttpGET

```
public ActionResult Edit(int Id)
{
    var std = students.Where(s => s.StudentId == Id).FirstOrDefault();

    return View(std);
}
```

http://localhost/Student/Edit

HttpPOST

```
[HttpPost]
public ActionResult Edit(Student std)
{
    //update database here..

    return RedirectToAction("Index");
}
```

# Action Verbs

- The following table lists the usage of some http methods:

Http method	Usage
GET	To retrieve the information from the server. Parameters will be appended in the query string.
POST	To create a new resource.
PUT	To update an existing resource.
HEAD	Identical to GET except that server do not return message body.
OPTIONS	OPTIONS method represents a request for information about the communication options supported by web server.
DELETE	To delete an existing resource.
PATCH	To full or partial update the resource.

# Action Verbs

The example shows different action methods which support different Action Verbs

```
public class StudentController : Controller
{
    public ActionResult Index()
    {
        return View();
    }

    [HttpPost]
    public ActionResult PostAction()
    {
        return View("Index");
    }

    [HttpPut]
    public ActionResult PutAction()
    {
        return View("Index");
    }

    [HttpDelete]
    public ActionResult DeleteAction()
    {
        return View("Index");
    }
}
```

```
[HttpHead]
public ActionResult HeadAction()
{
    return View("Index");
}

[HttpOptions]
public ActionResult OptionsAction()
{
    return View("Index");
}

[HttpPatch]
public ActionResult PatchAction()
{
    return View("Index");
}
```

# ActionVerbs

- You can also apply multiple http verbs using AcceptVerbs attribute. GetAndPostAction method supports both, GET and POST ActionVerbs in the following example:

```
[AcceptVerbs(HttpVerbs.Post | HttpVerbs.Get)]  
public ActionResult GetAndPostAction()  
{  
    return RedirectToAction("Index");  
}
```

# ActionVerbs

## Points to Remember:

- ▶ ActionVerbs are another Action Selectors which selects an action method based on request methods e.g POST, GET, PUT etc.
- ▶ Multiple action methods can have same name with different action verbs. Method overloading rules are applicable.
- ▶ Multiple action verbs can be applied to a single action method using AcceptVerbs attribute.



# Question #1

Which of the following is TRUE?

1. Action method can be static method in a controller class.
2. Action method can be private method in a controller class.
3. Action method can be protected method in a controller class.
4. Action method must be public method in a controller class.

# Question #2

Which of the followings are ActionSelectors?

1. ActionName
2. NonAction
3. ActionVerbs
4. All of the above

# Question #3

Which is the default http method for an action method?

1. `HttpPost`
2. `HttpGet`
3. `HttpPut`
4. `HttpDelete`

# Question #4

Which of the following is a default route pattern in MVC?

1. `"/{action}/{controller}/{id}"`
2. `"{controller}/{id}"`
3. `"{controller}/{action}/{id}"`
4. `"{controller}/{action}"`

# Question #5

What is Action Method?

Action method is simply a public method inside controller which accepts user's request and returns some response.

# Question #6

In Asp.Net MVC two things are very important.

1. How we name something?
2. Where we keep something?

**Why?**

# Question #7

What about non-public methods?

```
[NonAction]
public string SimpleMethod()
{
    return "Hi, I am not action method";
}
```

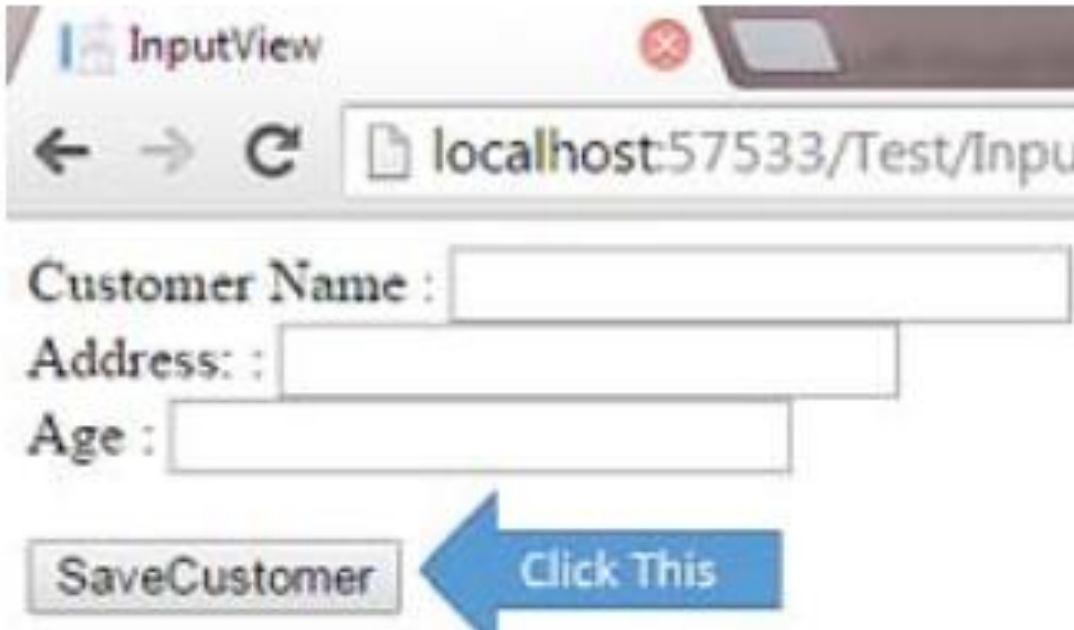


# Question #8

What is User interaction Logic?

Q1: Did you ever gave a thought what happens when end user hits a URL on a browser?

Q2:



The screenshot shows a web browser window with the title 'InputView'. The address bar displays 'localhost:57533/Test/Inpu'. The form contains three input fields labeled 'Customer Name :', 'Address: :', and 'Age :'. Below these fields is a button labeled 'SaveCustomer'. A blue arrow points to the 'SaveCustomer' button with the text 'Click This'.



# Question #9

What will happen if we try to return an object from an action method?

```
namespace WebApplication1.Controllers {  
    public class Customer{  
        public string CustomerName { get; set; }  
        public string Address { get; set; }  
    }  
    public class TestController : Controller{  
        public Customer GetCustomer(){  
            Customer c = new Customer();  
            c.CustomerName = "Customer 1";  
            c.Address = "Address1";  
            return c;  
        }  
    }  
}
```



```
public override string ToString(){  
    return this.CustomerName+"|"+this.Address;}  
}
```

