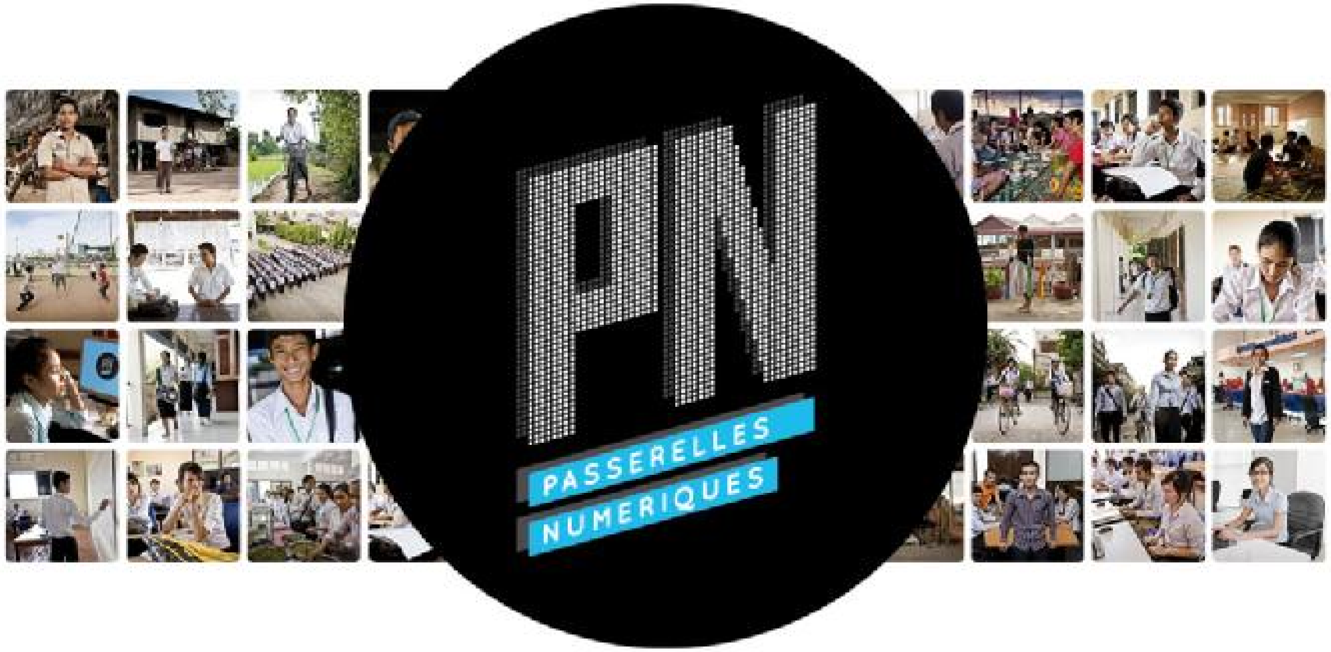


Microsoft®

.NET

UNIT 11:
CONTROLLER

CONTENTS

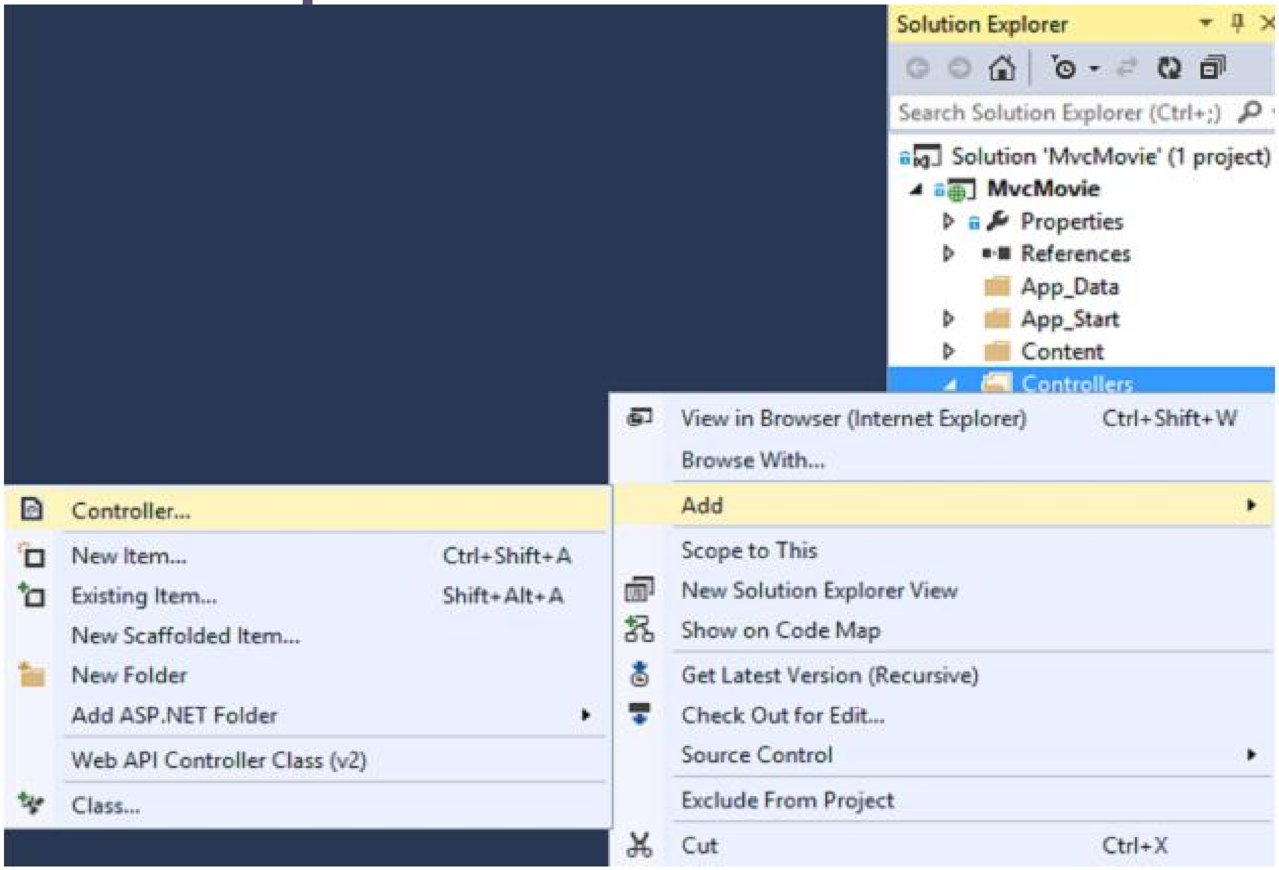
- Introduction to Controller
- Action Method
- Write a controller and action methods
- Parameters in controller actions

Introduction to Controller

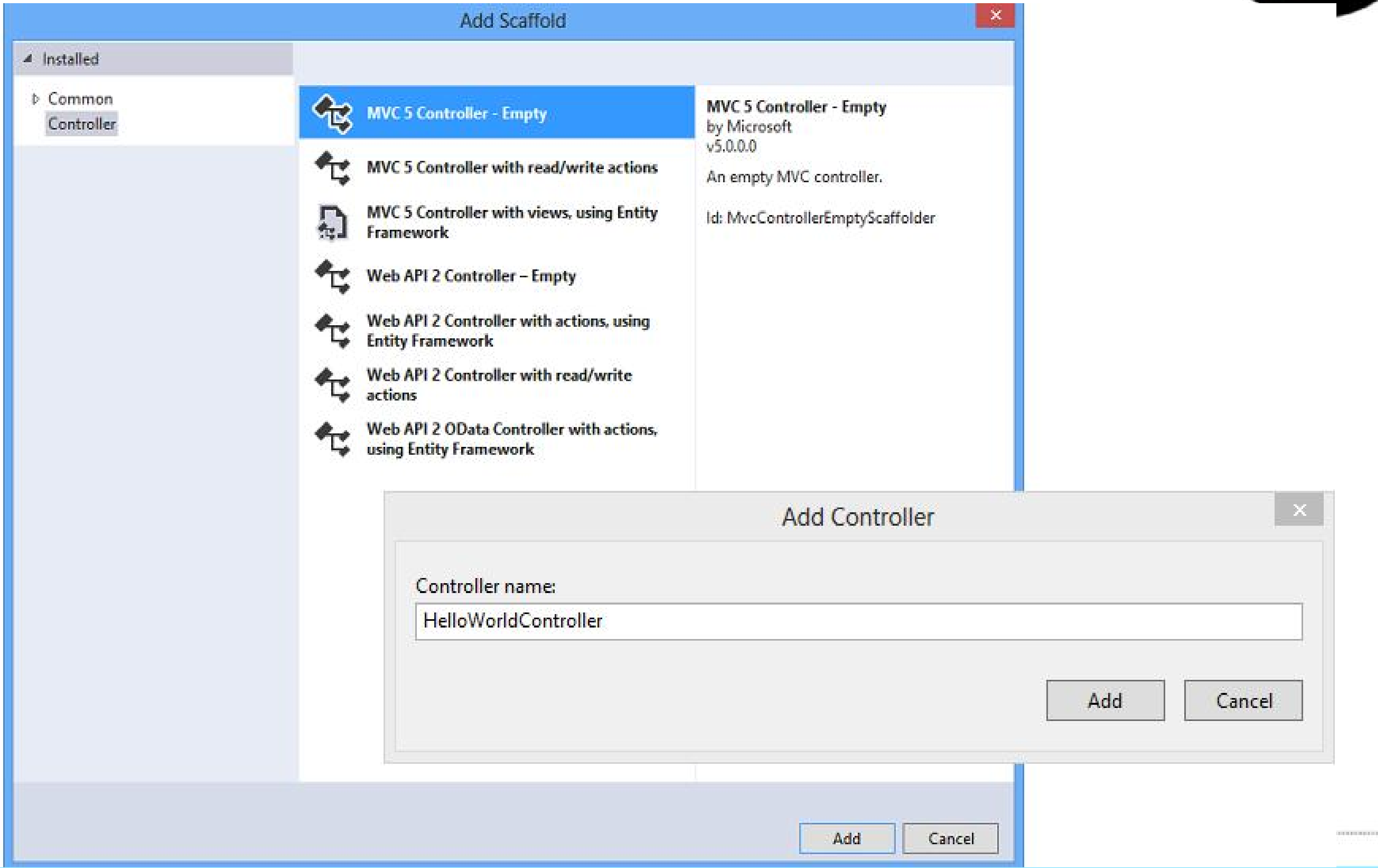
- Controllers are responsible for responding to user input, often making changes to the model in response to user input.
- Controllers are concerned with the flow of the application, working with data coming in, and providing data going out to the relevant view.
- A Controller class name must end with "Controller".

Add a Controller

- Right-click the Controllers folder within the Solution Explorer and select the Add ->Controller



Add a Controller


Add Scaffold
Installed
Common
Controller
MVC 5 Controller - Empty
MVC 5 Controller with read/write actions
MVC 5 Controller with views, using Entity Framework
Web API 2 Controller – Empty
Web API 2 Controller with actions, using Entity Framework
Web API 2 Controller with read/write actions
Web API 2 OData Controller with actions, using Entity Framework
Add
Cancel
Add Controller
Add
Cancel

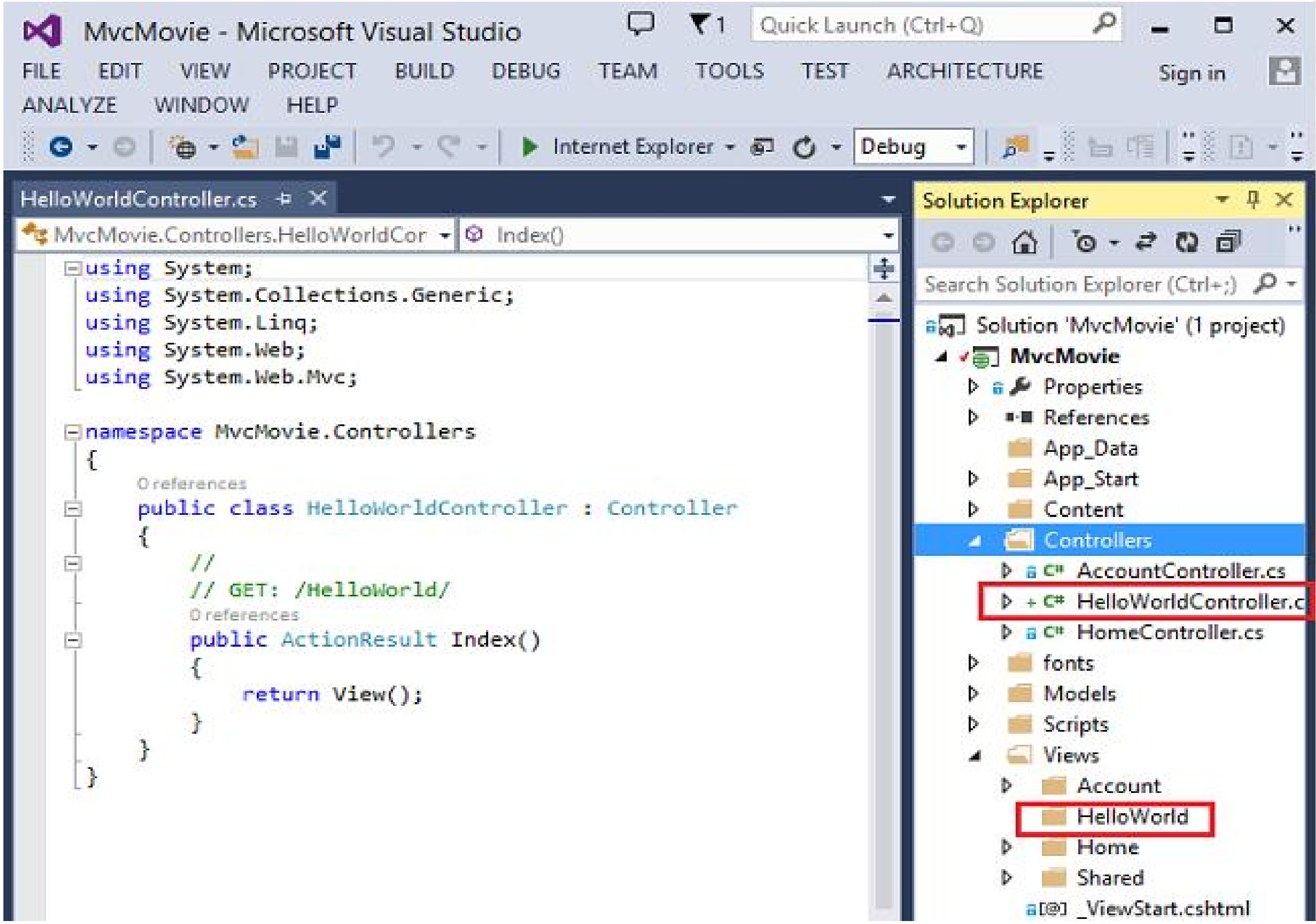
MVC 5 Controller - Empty
by Microsoft
v5.0.0.0

An empty MVC controller.

Id: MvcControllerEmptyScaffolder

Controller name: HelloWorldController

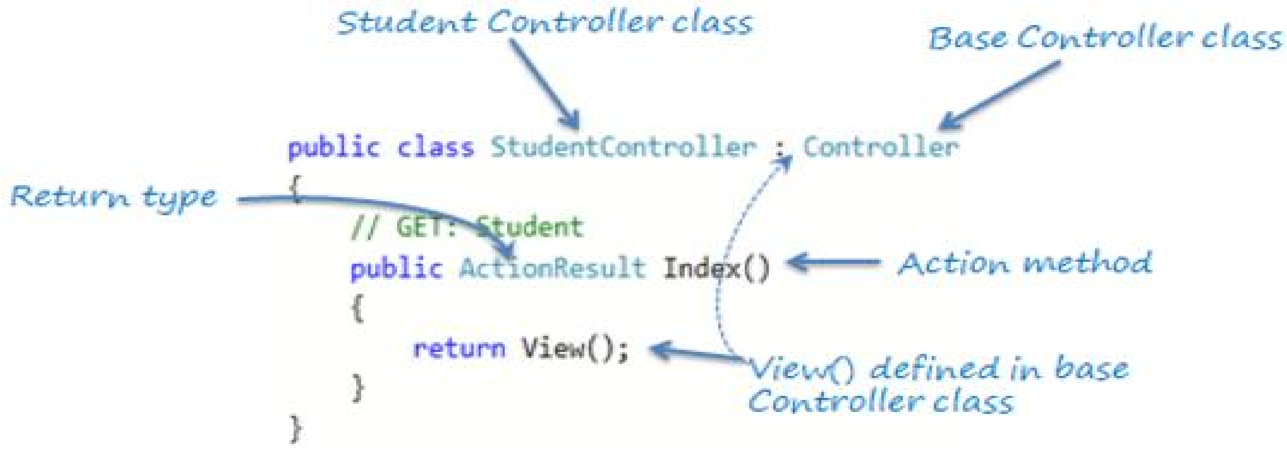
Add a Controller



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

namespace MvcMovie.Controllers
{
    0 references
    public class HelloWorldController : Controller
    {
        //
        // GET: /HelloWorld/
        0 references
        public ActionResult Index()
        {
            return View();
        }
    }
}
```

Action Method

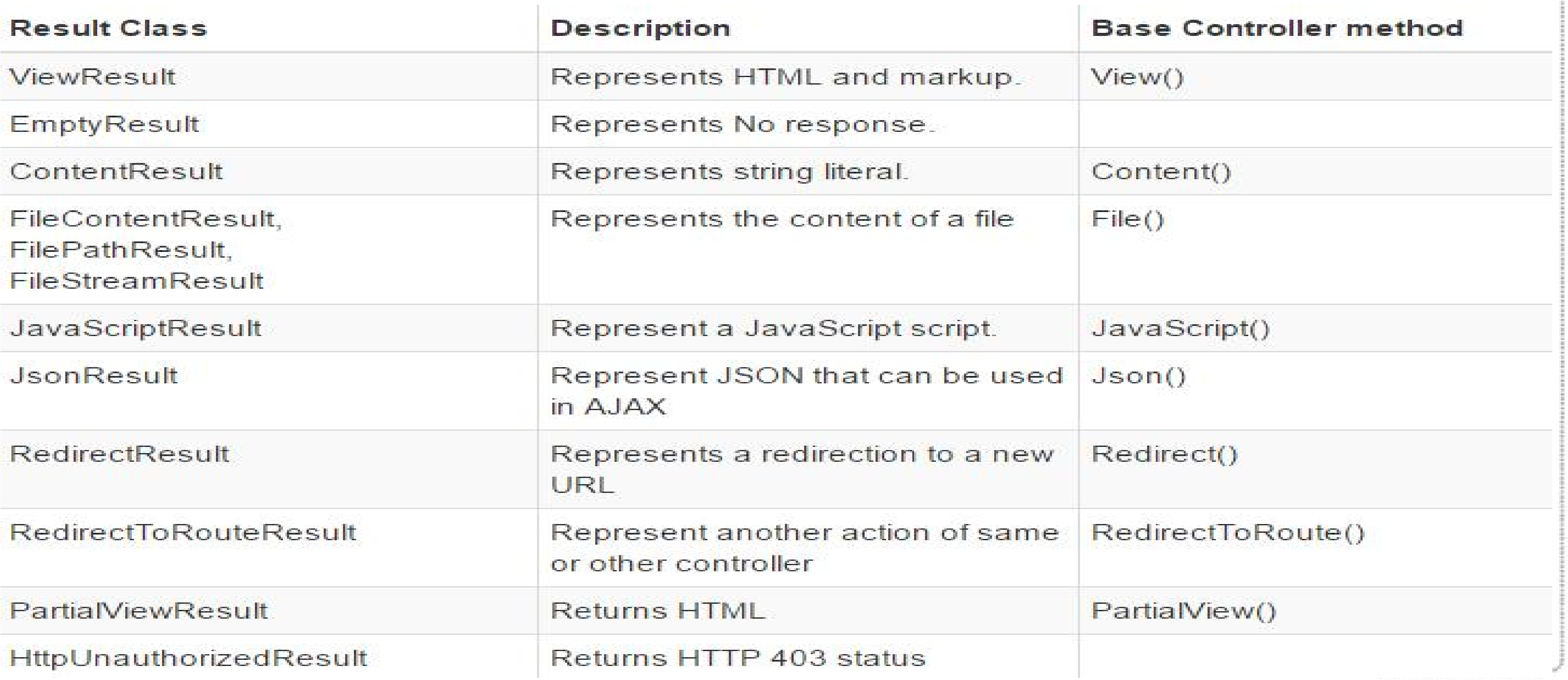
- Action Method: is simply a public method inside controller which accepts user’s request and returns some response.
- Action method must be public. It cannot be private or protected, cannot be a static method



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

Action Method

- **ActionResult class is a base class of all the result classes, which can be return from an action methods**



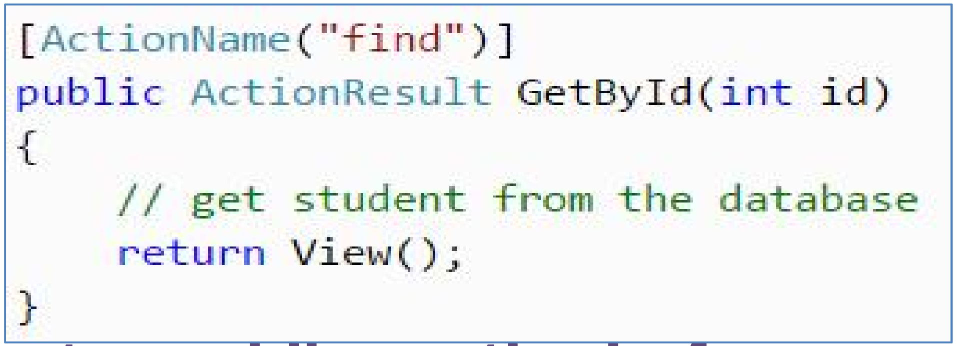
Result Class	Description	Base Controller method
ViewResult	Represents HTML and markup.	View()
EmptyResult	Represents No response.	
ContentResult	Represents string literal.	Content()
FileContentResult, FilePathResult, FileStreamResult	Represents the content of a file	File()
JavaScriptResult	Represent 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 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. Some action selectors attribute:
  - ActionName
  - NonAction
  - ActionVerbs

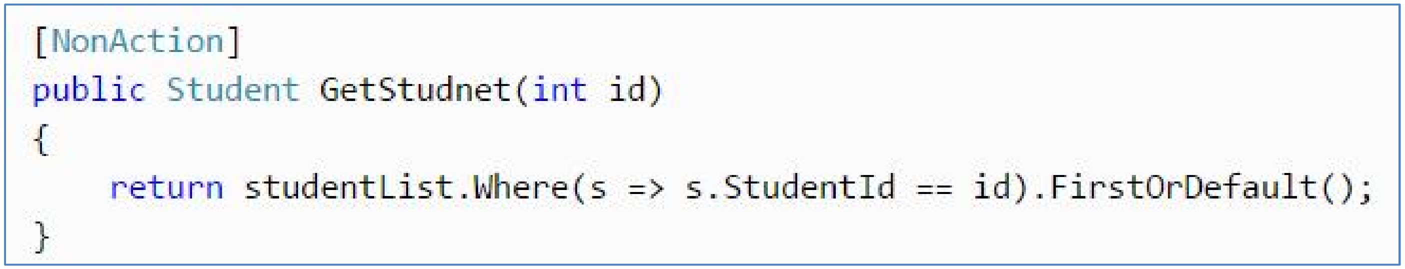
Action Selectors

- **ActionName: allows us to specify a different action name than the method name**



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

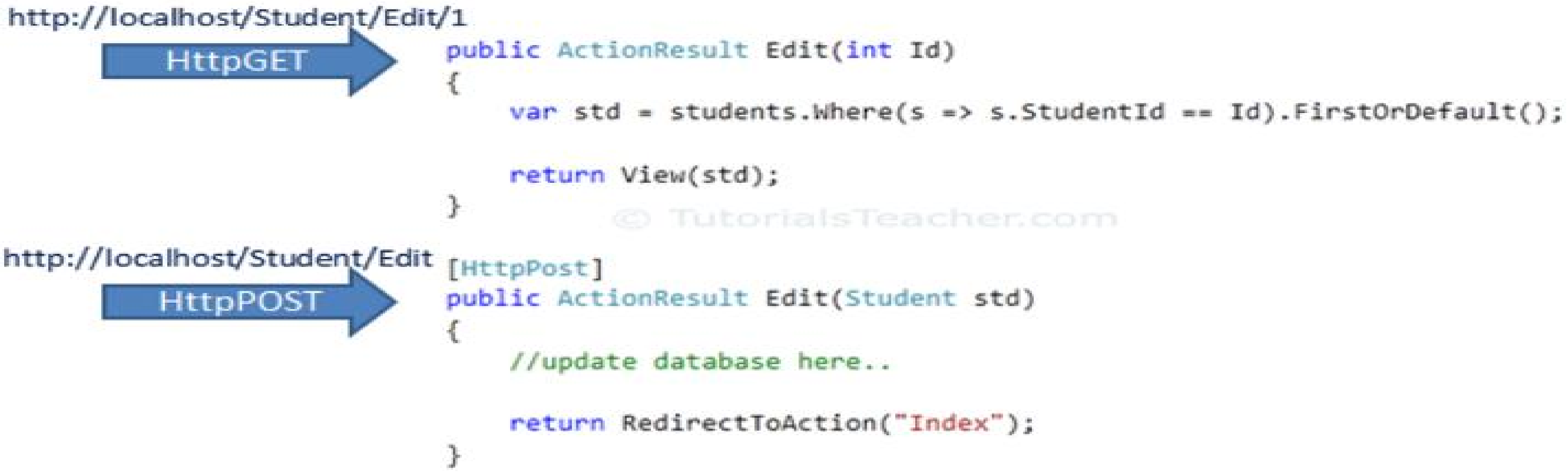
- **NonAction: indicates that a public method of a Controller is not an action method**



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

Action Selectors

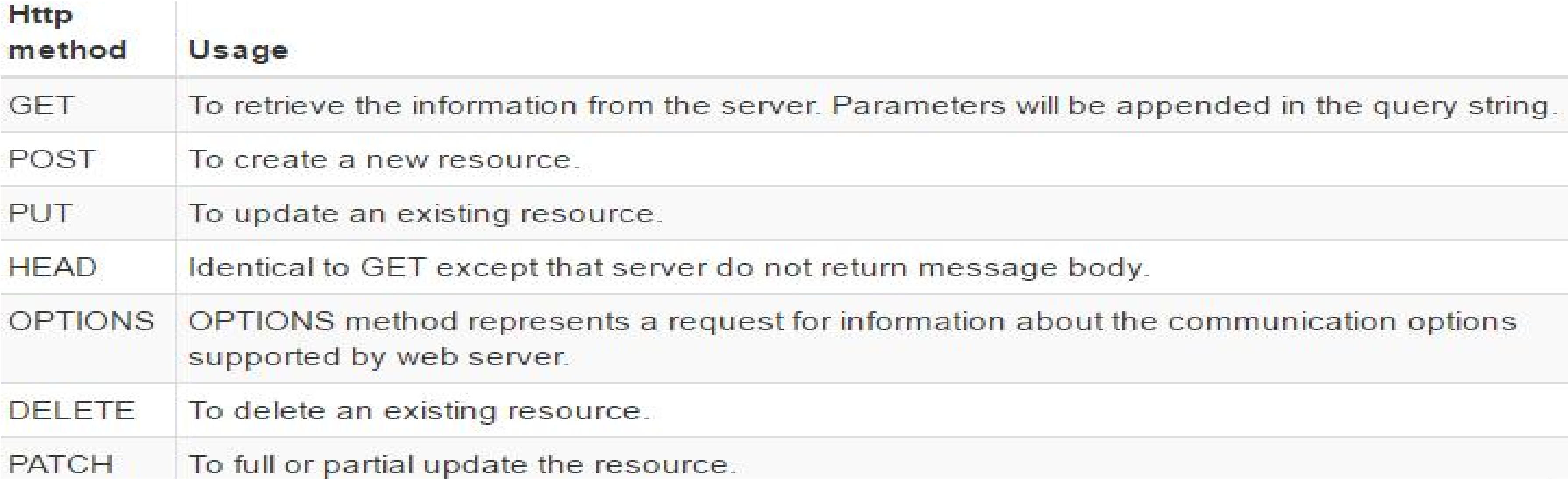
- ActionVerbs: is used when you want to control the selection of an action method based on a Http request method.
- If you do not apply any attribute then it considers it a GET request by default.


http://localhost/Student/Edit/1
HttpGET
http://localhost/Student/Edit
HttpPOST

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

    return View(std);
}

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

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

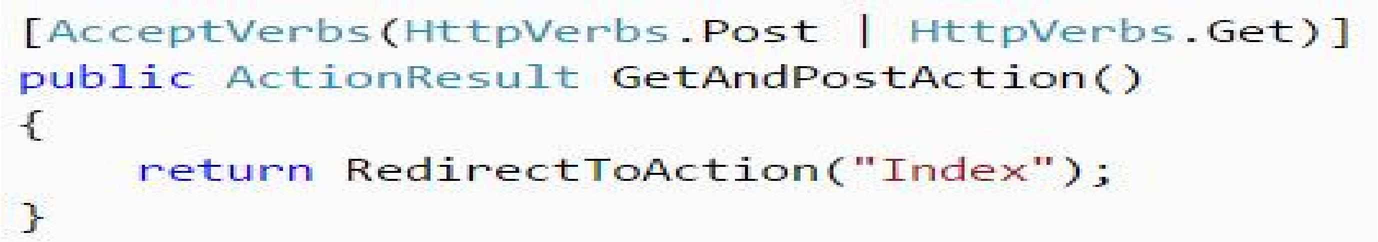
Action Selectors

- **Some Http request method:**



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.

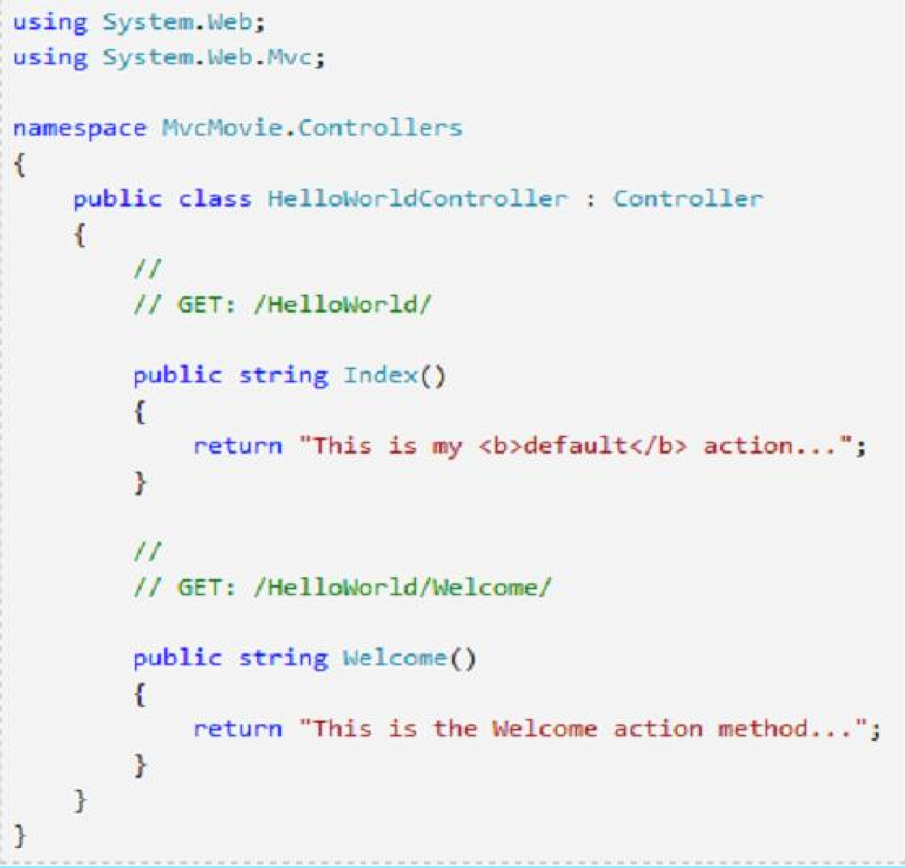
- **You can also apply multiple http verbs using AcceptVerbs attribute.**



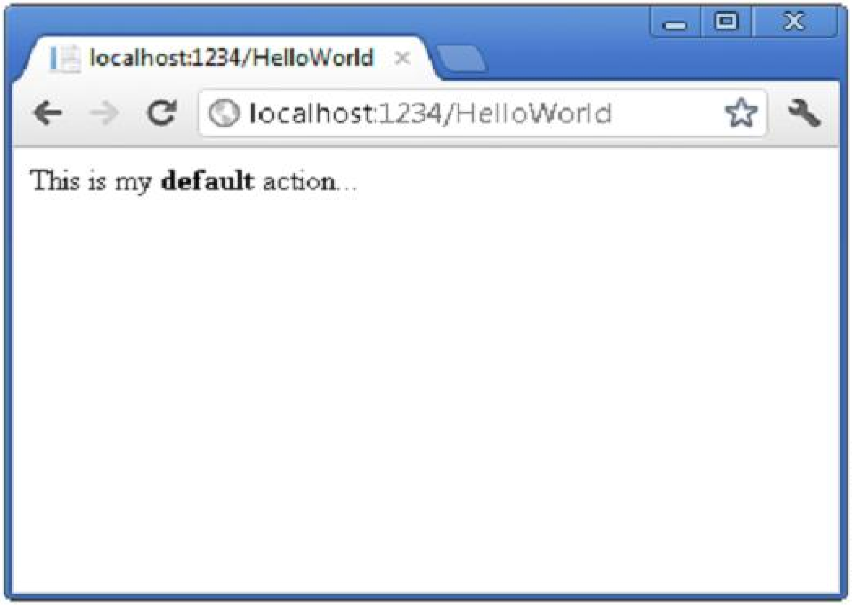
```
[AcceptVerbs(HttpVerbs.Post | HttpVerbs.Get)]
public ActionResult GetAndPostAction()
{
    return RedirectToAction("Index");
}
```

Write a controller and action methods

- **Testing Controller**
- **The way routing maps the URL to actions.**



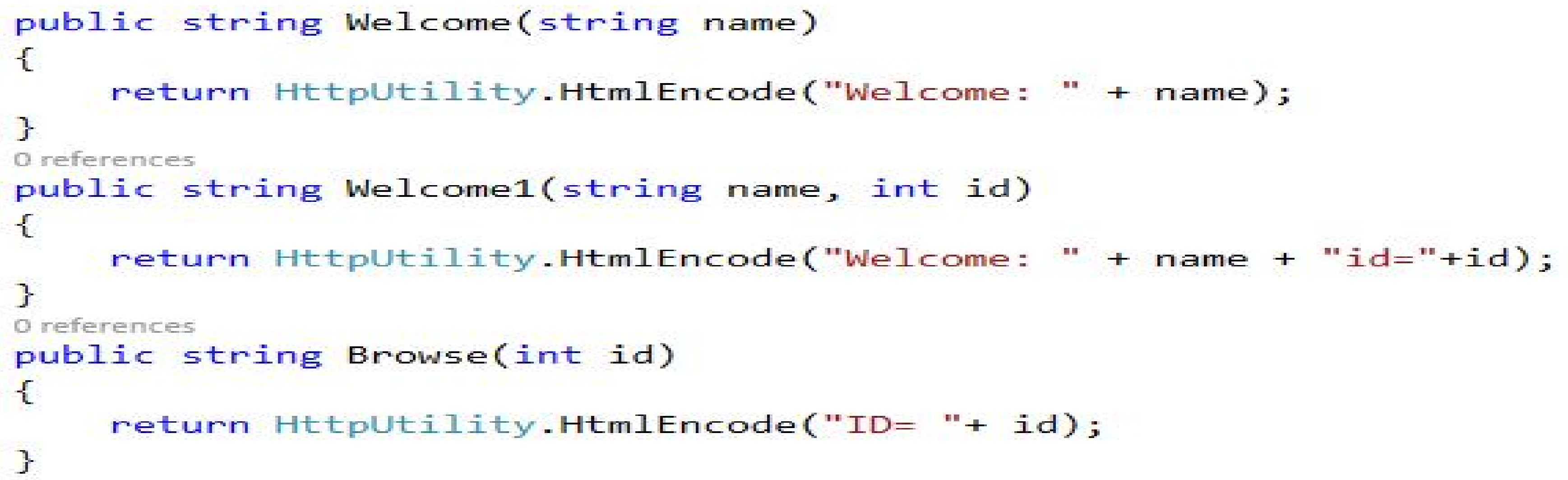
```
using System.Web;
using System.Web.Mvc;

namespace MvcMovie.Controllers
{
    public class HelloWorldController : Controller
    {
        //
        // GET: /HelloWorld/

        public string Index()
        {
            return "This is my <b>default</b> action...";
        }

        //
        // GET: /HelloWorld/Welcome/

        public string Welcome()
        {
            return "This is the Welcome action method...";
        }
    }
}
```



This is my **default** action...

Parameters in action method

- Every action methods can have input parameters as normal methods. It can be primitive data type or complex type parameters



```
public string Welcome(string name)
{
    return HttpUtility.HtmlEncode("Welcome: " + name);
}
0 references
public string Welcome1(string name, int id)
{
    return HttpUtility.HtmlEncode("Welcome: " + name + "id="+id);
}
0 references
public string Browse(int id)
{
    return HttpUtility.HtmlEncode("ID= "+ id);
}
```

http://localhost:42998/demo/welcome?name=abc

http://localhost:42998/demo/welcome1/2?name=abc

http://localhost:42998/demo/browse/2