Controller and Action

Lesson Objectives

- *Controller Overview*
- *Action in MVC*
- *Action selector*

Section 1

CONTROLLER OVERVIEW

Controller

- Is basically a C# class which is inherited from the **System.Web.Mvc.Controller**.
- Is the component which will interact with both Models and Views.

Controller

- Contains action methods which are responsible for handling the incoming URL and control flow logic
- Can access and use the model class to pass the data to the views.

Add controller into MVC

- A class that derived from the base class System.Web.Mvc.Controller
- Always end with a word "Controller"
- Inside Controllers folder

Add controller into MVC

- In the Visual Studio
  - ✓ Step 1: right click on the Controller folder
  - ✓ Step 2: select **Add**
  - ✓ Step 3: click on **Controller..**

Add controller into MVC

- ✓ Step 4: select **MVC 5 Controller - Empty**
- ✓ Step 5: click Add
- ✓ Step 6: give controller name such as **StudentController**
- ✓ Step 7: click Add button on the dialog

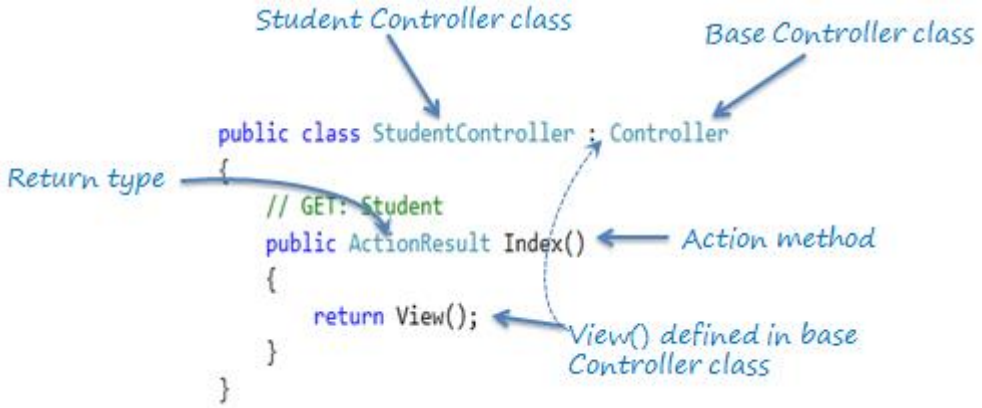
Section 2

ACTION METHOD

Action method

- All the public methods of a Controller class are called Action methods.
  - ✓ Action method must be public. It cannot be private or protected
  - ✓ Action method cannot be overloaded
  - ✓ Action method cannot be a static method.

Action method example



Practice time

- Create action that:
  - ✓ Inside Student controller
  - ✓ Has name: GradeBook
  - ✓ Retrieve name and mark from URL
  - ✓ Return string as format: “Hi, {name}. Your mark is {mark}.”

Practice time

- Run to check your result
- Add your route to match URL

ActionResult

- The ActionResult class is a base class of all the result classes
- There are various result classes, that derived from ActionResult class
- Each result classes represent different types of responses

ActionResult

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()

ActionResult

Result Class	Description	Base Controller Method
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
  - ✓ Primitive data often comes from route
  - ✓ Complex data often comes from form data
- Action method can include Nullable type parameters.

Action Method Parameters

- By default, the values for action method parameters are retrieved from the request's data collection.
- The data collection includes name/values pairs for form data or query string values or cookie values.

Action Method Parameters

- Model binding automatically maps the URL query string or form data collection to the action method parameters if both names are matching.
- Data type is converted automatically

Section 3

ACTION SELECTORS

Action selector

- 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.

Action selector

- MVC 5 includes the following action selector attributes:
  - ✓ ActionName
  - ✓ NonAction
  - ✓ ActionVerbs

ActionName

- ActionName attribute allows to specify a different action name than the method name.

```
[ActionName("Index")]
0 references
public ViewResult ViewAllStudent()
{
    //// Get all students from application context
    //// Return view to show students
    return View();
}
```

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.

```
[NonAction]
0 references
public decimal CalculateGpa(decimal sqlCourse, decimal nplCourse,
    decimal efCourse, decimal mvcCourse)
{
    return (sqlCourse + nplCourse + efCourse + mvcCourse) / 4;
}
```

ActionVerbs

- Used to control the selection of an action method based on a Http request method.
- MVC framework supports different ActionVerbs
  - ✓ HttpGet,
  - ✓ HttpPost,
  - ✓ HttpPut,
  - ✓ HttpDelete,
  - ✓ HttpOptions
  - ✓ HttpPatch.

ActionVerbs

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.

HttpGet vs HttpPost

Send less, Get more

- Creates a query string of the name-value pair
- Fast and quick but **not secure**
- Limited length and mostly it is limited to 255 characters
- Can carry only string data

- Creates readable URL, can be cached, bookmarked, copy

Send more, Get less (or not)

- Passes the name and value pairs in the body of the HTTP request
- More secured but **slower**
- No maximum limit. In fact, it depends on configuration
- Can carry both string and binary data
- Not applicable

ActionVerbs

- Apply these attributes to action method to indicate the kind of Http request the action method supports.
- Default is HttpGet.

ActionVerbs

```
[HttpGet]
0 references
public ActionResult CreateStudent()
{
    return View();
}
```

```
[HttpPost]
0 references
public ActionResult CreateStudent(Student student)
{
    //// Save student to database
    return RedirectToAction("Index");
}
```

ActionVerbs

- Multiple action verbs can be applied to a single action method using AcceptVerbs attribute.

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

Thank you