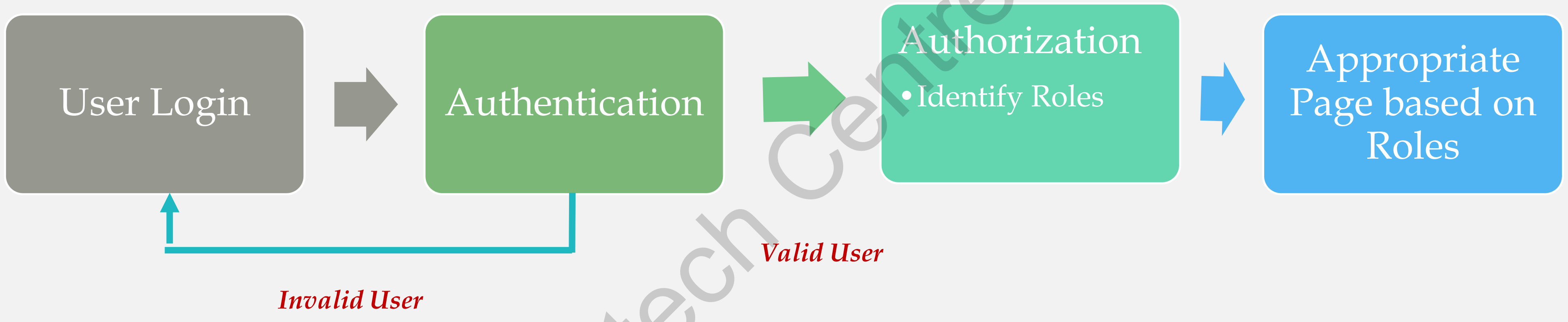
Session 10

Authorization, Selectors, and Helpers in MVC

Session Overview

- Explain authorization and roles with respect to MVC applications
- Describe ASP.NET selectors
- List various helpers in ASP.NET

Role-based and View-based Authorization



Authorization indicates what a user is allowed to do

Role-based Authorization

Adding Role Checks

- Role-based authorization check is done by developers against a controller or an action in a controller.
- This announces roles of which the existing user must be a member in order to get access to the target resource.

Policy-based Role Checks

- Policy syntax can be utilized to define role requirements.
- A developer executes a policy at the initial stage that is included in authorization service configuration.

View-based Authorization

Developers can access authorization service in MVC views using dependency injection.

To inject a service into a view, use @inject directive.

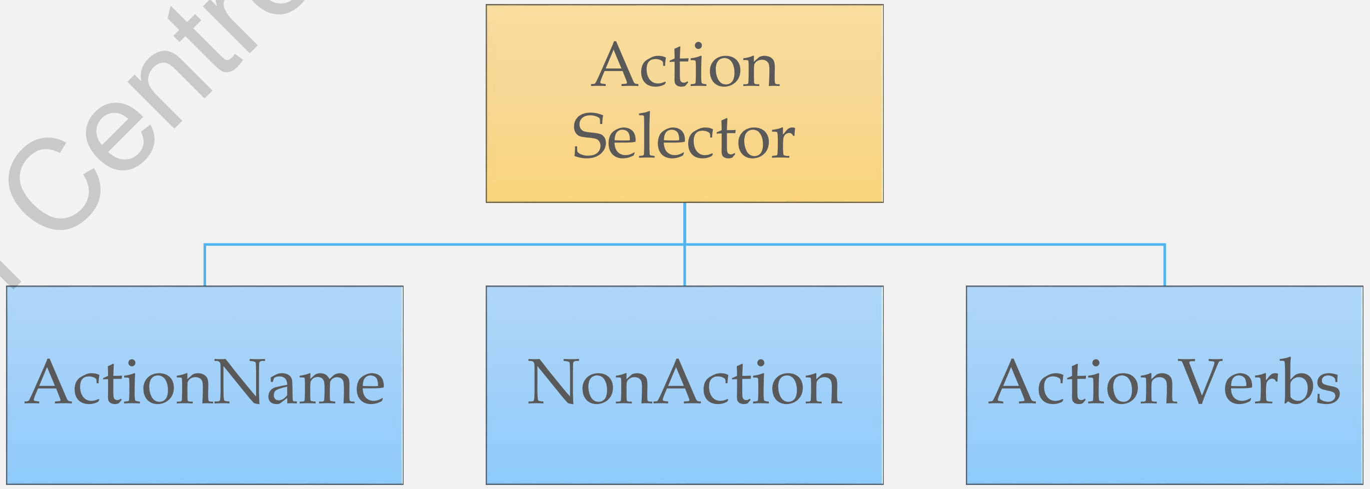
To implement authorization service in all the views at a time, add the `@inject` directive into the `_ViewImports.cshtml` file of the Views directory.

ASP.NET Selectors

Properties related to action methods are known as action selectors.

An action selector is used to determine the action method that gets invoked in response to a request.

The action method to be selected is determined by the Routing engine.



ActionName, NonAction, and ActionVerbs

The ActionName attribute can be used to define an action name other than the method name.

NonAction attribute spots the public method of controller class as non-action method.

When there is a need to regulate the selection of an action method as per the HTTP request method, the `ActionVerbs` selector is applied. Some `ActionVerbs` supported by the MVC framework are `HttpGet`, `HttpPut`, `HttpPost`, `HttpDelete`, `HttpOptions`, and `HttpPatch`.

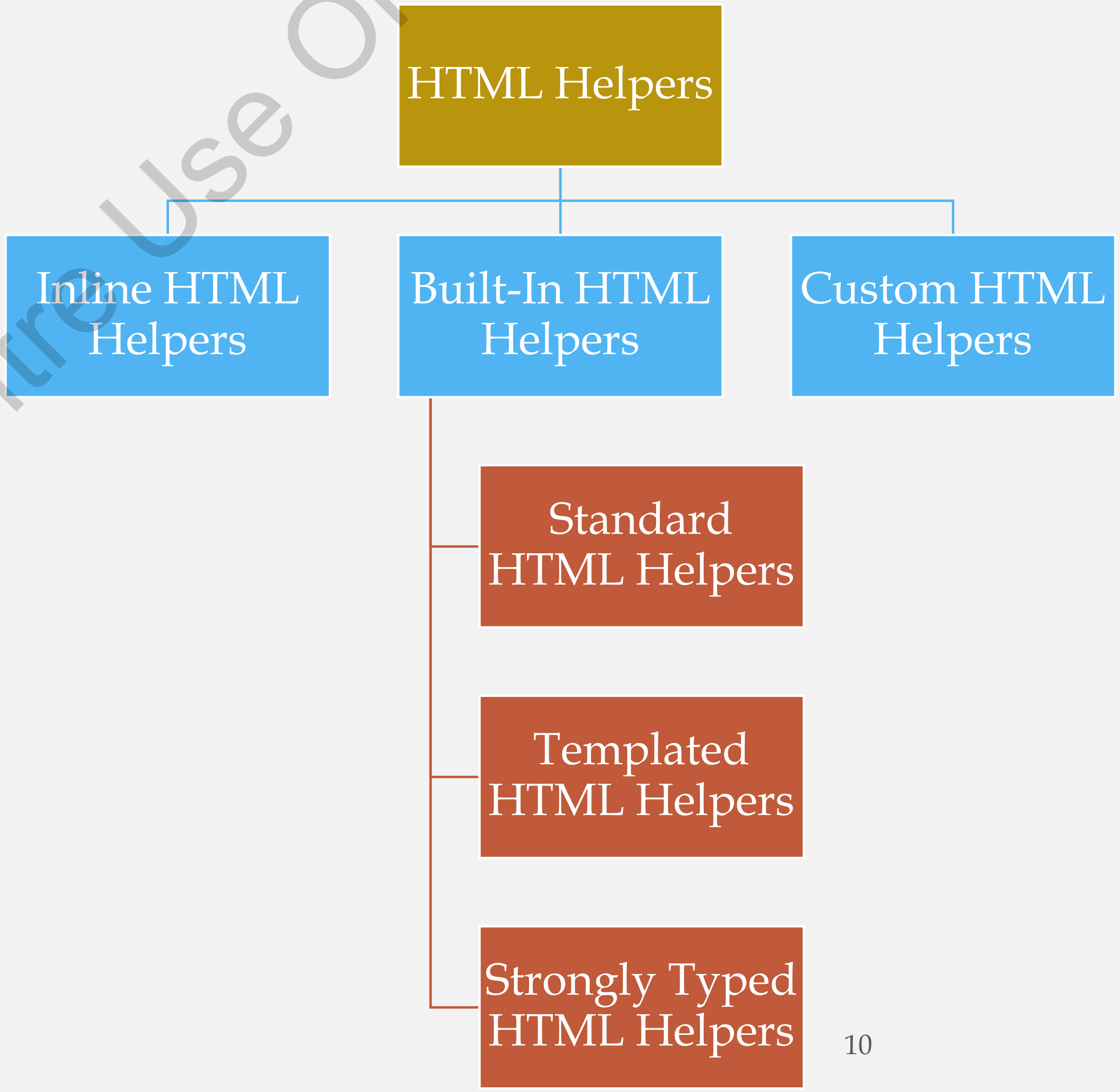
HTTP Methods

Method	Description
GET	Used for fetching data from the server. Parameters are added in the query string
POST	Used for generating a new resource
PUT	Used for appending an existing resource
HEAD	Used as GET method with the only difference being that server does not throwback message body
OPTIONS	Used for representing a request for data for communication options supported by browsers
DELETE	Used for deleting an existing resource
PATCH	Used for updating the resource either fully or partially

ASP.NET Helpers

Helpers are reusable components that include code and markup to perform a monotonous or complex task.

An HTML helper is a procedure that returns an HTML string.



Inline ASP.NET Helpers

@helper tag helps to develop inline HTML helper tags in the same view.

Standard HTML Helpers

Element	Example

TextBox	`@Html.TextBox("Tb1", "val")`
	`Output: <input id="Tb1" name="Tb1" type="text" value="name" />`

TextArea	`@Html.TextArea("Ta1", "val", 5, 20, null)`
	`Output: <textarea cols="20" id="Ta1" name="Ta1" rows="5">val</textarea>`

Password	`@Html.Password("Pwd1", "val")`
	`Output: <input id="Pwd1" name="Pwd1" type="password" value="pwd" />`

Hidden Field	`@Html.Hidden("Hdn1", "val")`
	`Output: <input id="Hdn1" name="Hdn1" type="hidden" value="hid" />`

CheckBox	`@Html.CheckBox("Ckb1", false)`
	`Output: <input id="Ckb1" name="Ckb1" type="checkbox" value="true" /> <input name="myCkb" type="hidden" value="false" />`

RadioButton	`@Html.RadioButton("Rb1", "val", true)`
	`Output: <input checked="checked" id="Rb1" name="Rb1" type="radio" value="opt1" />`

Drop-down list	`@Html.DropDownList ("Ddl1", new SelectList(new [] {"Asia", "Europe"}))`
	`Output: <select id="Ddl1" name="Ddl1"> <option>A</option> <option>E</option> </select>`

Multiple-select	`@Html.ListBox("Lb1", new MultiSelectList(new [] {"Tennis", "Chess"}))`
	`Output: <select id="Lb1" multiple="multiple" name="Lb1"> <option>Tennis</option> <option>Chess</option> </select>`

Strongly Typed HTML Helpers (1-2)

Element	Example

TextBox	`@Html.TextBoxFor(m=>m.MidName)`
	`Output: <input id="MidName" name="MidName" type="text" value="MidName-val" />`

TextArea	`@Html.TextArea(m=>m.Add, 5, 15, new{}))`
	`Output: <textarea cols="15" id="Add" name=" Add" rows="5">Addvalue</textarea>`

Password	`@Html.PasswordFor(m=>m.Pwd)`
	`Output: <input id="Pwd" name="Pwd" type="password"/>`

Hidden Field	`@Html.HiddenFor(s=>msSno)`
	`Output: <secret no=" Sno" name=" Secret No" type="hidden" value="Sno-val" />`

CheckBox	`@Html.CheckBoxFor(s=>s.IsAccepted)`
	`Output: <input id="Ps1" name="Ps1" type="checkbox" value="true" /> <input name="myPs" type="hidden" value="false" />`

RadioButton	`@Html.RadioButtonFor(s=> s.IsApproved, "val")`
	`Output:<input checked="checked" id="Br1" name="Br1" type="radio" value="value" />`

Strongly Typed HTML Helpers (2-2)

Element	Example
Drop-down list	`@Html.DropDownListFor(m => m.Occupation, new SelectList(new []{"Student", "Employee"}))` Output: `<select id="Occp" name="Occup"> <option>Student</option> <option>Employee</option> </select>`
Multiple-select	`@Html.ListBoxFor(d => d.Designation, new MultiSelectList(new [] {"Devops", "Developer"}))` Output: `<select id="Designation" multiple="multiple" name="Designation"> <option>Devops</option> <option>Developer</option> </select>`

Templated HTML Helpers

Templated Helper	Example
Display	`Html.Display("Name")`
DisplayFor	`Html.DisplayFor(m => m. Name)`
Editor	`Html.Editor("Name")`
EditorFor	`Html.EditorFor(m => m. Name)`

Templated Helpers

HTML elements, which are required to deliver, are identified by templated helpers as per properties of the model class. Developers should use `DataType` attribute of `DataAnnotation` class to set up proper HTML elements with templated HTML helpers.

Custom HTML Helpers

- One can create static methods in a utility class to form custom helper methods.
- An alternative is to develop an extension method on the HtmlHelper class.

Summary

- Authorization indicates what a user is allowed to do.
- The `AuthorizeAttribute` attribute and its different parameters control the authorization in MVC.
- Developers can use `IsInRole` property on the `ClaimsPrincipal` class for the desired roles depending upon the requirement.
- Developers can use Policy property to apply policies on the `AuthorizeAttribute` attribute to implement policy-based authorization.
- Action selectors in ASP.NET are features applicable to action methods.
- An HTML Helper is a procedure that returns an HTML string.