UNIT 5
ASP.NET AJAX and MVC

(10 Hours) — Advanced .NET Technology (CACS408)

Note: This PDF may contain errors — please cross-check important points with your syllabus, prescribed textbook and teacher's notes before the exam.

Note: The syllabus typo "ASPoNET" means ASP.NET. The AJAX part (Web Forms UpdatePanel, ScriptManager) belongs to the .NET Framework; the MVC part is shown in ASP.NET Core MVC style, which is what current courses and the next unit use. Differences from classic ASP.NET MVC 5 are noted.

5.1 Introduction to ASP.NET AJAX

AJAX (Asynchronous JavaScript and XML) is a technique that lets a web page exchange data with the server in the background and update only part of the page, without a full reload. It uses JavaScript's **XMLHttpRequest** (or fetch) and exchanges XML, JSON or HTML. **ASP.NET AJAX** adds server controls and a client library so developers can use AJAX without writing much JavaScript.

bcalink.com

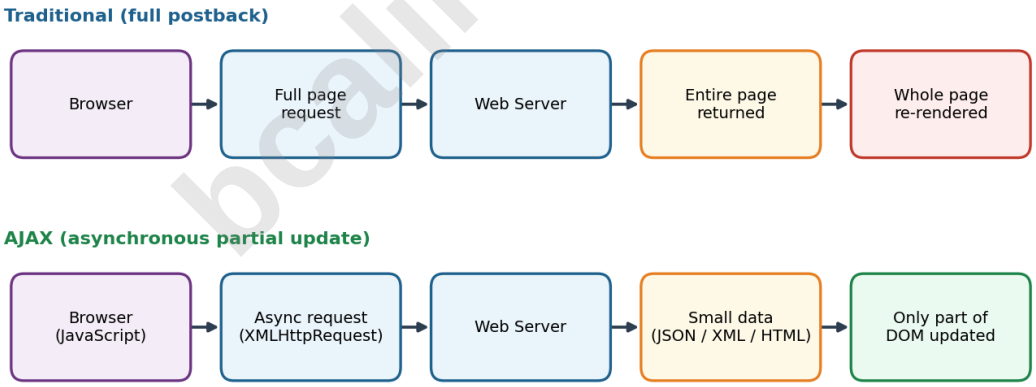


Fig 5.1: Traditional postback vs AJAX request

- **Benefits:** faster, smoother user experience; less bandwidth; no page flicker; scroll position and focus preserved.
- **Drawbacks:** needs JavaScript; browser Back button/bookmarking is harder; harder to debug; search engines may not see dynamic content.

5.2 ASP.NET AJAX Server Controls

Control	Purpose
ScriptManager	Mandatory, one per page. Loads the AJAX scripts and coordinates partial-page updates.
UpdatePanel	Marks a region that refreshes asynchronously (partial-page rendering).
UpdateProgress	Shows a “loading…” message while an asynchronous postback runs.
Timer	Triggers postbacks at set intervals (live clock, auto-refresh).
ScriptManagerProxy	Adds scripts/services in content pages when the ScriptManager is on the master page.

An UpdatePanel has a **ContentTemplate** holding the controls, optional **Triggers** (`AsyncPostBackTrigger` makes an outside control refresh the panel; `PostBackTrigger` forces a full postback), and properties `UpdateMode` (Always or Conditional) and `ChildrenAsTriggers`.

```
<asp:ScriptManager ID="sm1" runat="server" />

<asp:UpdatePanel ID="up1" runat="server" UpdateMode="Conditional">
  <ContentTemplate>
    <asp:Label ID="lblTime" runat="server" />
    <asp:Button ID="btnTime" runat="server" Text="Get Time"
                OnClick="btnTime_Click" />
  </ContentTemplate>
  <Triggers>
    <asp:AsyncPostBackTrigger ControlID="tmr1" EventName="Tick" />
  </Triggers>
</asp:UpdatePanel>

<asp:Timer ID="tmr1" runat="server" Interval="5000" OnTick="btnTime_Click" />

<asp:UpdateProgress ID="upg1" runat="server" AssociatedUpdatePanelID="up1">
  <ProgressTemplate>Loading...</ProgressTemplate>
</asp:UpdateProgress>

// code-behind
protected void btnTime_Click(object sender, EventArgs e)
{
    lblTime.Text = DateTime.Now.ToLongTimeString();   // only the panel refreshes
}
```

Program 5.1: ScriptManager, UpdatePanel, Timer and UpdateProgress

bcalink.com

5.3 ASP.NET AJAX Server Data

Besides partial-page rendering, ASP.NET AJAX lets client script call **server-side data and services** directly and get the result as JSON:

- **Page methods:** static methods marked `[WebMethod]` in a page, called from JavaScript (needs `EnablePageMethods="true"` on the ScriptManager).

- **ASMX / WCF web services:** a service class marked `[ScriptService]` is registered under ScriptManager <Services>, and a JavaScript proxy class is generated.
- **Application services:** built-in authentication, role and profile services available from the client.
- **AJAX Control Toolkit:** extra controls/extenders (AutoComplete, Calendar, ModalPopup, Accordion, …).

```
// StudentService.asmx.cs
[System.Web.Script.Services.ScriptService]
public class StudentService : System.Web.Services.WebService
{
    [System.Web.Services.WebMethod]
    public string GetName(int id)
    {
        return "Student #" + id;       // returned to the client as JSON
    }
}

<asp:ScriptManager ID="sm1" runat="server">
  <Services><asp:ServiceReference Path="StudentService.asmx" /></Services>
</asp:ScriptManager>
```

Program 5.2: Exposing server data as a script service

5.4 ASP.NET AJAX Client-Side Libraries

The **Microsoft AJAX Library** (MicrosoftAjax.js) is the JavaScript framework that ships with the ScriptManager. It adds an object-oriented layer to JavaScript (namespaces, classes, interfaces, events), browser compatibility and networking support.

Part	Description
Type system	Namespaces (Type.registerNamespace), classes, inheritance, interfaces and enums in JavaScript.
Sys.Application	Client page lifecycle: init, load and unload events.
Sys.WebForms.PageRequestManager	Controls partial postbacks (events such as beginRequest, endRequest).
Sys.Net (WebRequest, WebServiceProxy)	Makes asynchronous HTTP and web-service calls.
Sys.UI / $get, $find, $create	DOM helpers: $get('id') is shorthand for document.getElementById; $find gets components.
Sys.Serialization.JavaScriptSerializer	Converts objects to and from JSON.

bcalink.com

```
<script type="text/javascript">
  // call the page method / service from the client
  function getStudent() {
    var id = $get("txtId").value;
    StudentService.GetName(id, onSuccess, onError);
  }
  function onSuccess(result) { $get("lblName").innerHTML = result; }
  function onError(error)    { alert(error.get_message()); }

  // run code when a partial postback finishes
  Sys.WebForms.PageRequestManager.getInstance().add_endRequest(function () {
      $get("status").innerHTML = "Updated";
  });
</script>
```

Program 5.3: Client-side script calling the service

Note: Modern projects usually replace the Microsoft AJAX Library with jQuery or plain JavaScript fetch() calling JSON endpoints (Web API / MVC actions).

5.5 Introduction to ASP.NET MVC

ASP.NET MVC is a framework for building web applications using the **Model–View–Controller** pattern, which separates an application into three parts:

- **Model:** data and business rules (classes, validation, database access).
- **View:** the user interface; HTML generated with Razor syntax from model data.
- **Controller:** receives the request, works with the model, and chooses the view to return.

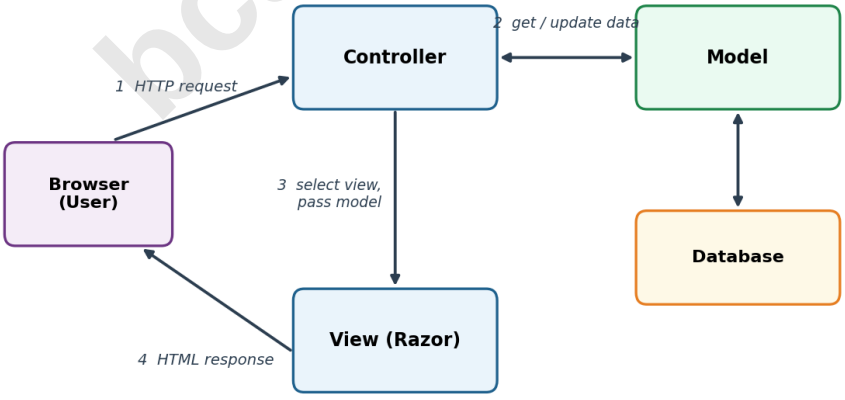


Fig 5.2: MVC request flow

Point	ASP.NET Web Forms	ASP.NET MVC
Pattern	Page controller, event-driven	MVC, request/action-driven
State	ViewState and postbacks	Stateless; no ViewState

bcalink.com

Point	ASP.NET Web Forms	ASP.NET MVC
HTML control	Limited (server controls generate HTML)	Full control over markup
Testing	Hard to unit test	Easy unit testing of controllers and models
URLs	Map to .aspx files	Clean routing URLs: /Student/Details/5
Separation of concerns	Weak (code-behind mix)	Strong

Project structure and routing

Folder / file	Purpose
Controllers/	Controller classes (HomeController.cs, StudentController.cs).
Models/	Model and view-model classes.
Views/	Razor views (.cshtml) in one folder per controller; Views/Shared holds _Layout.cshtml and partial views.
wwwroot/ (Core)	Static files: CSS, JavaScript, images.
Program.cs, appsettings.json	Startup configuration, services, middleware, routing, settings.

Routing maps a URL to a controller action. The default convention route is `{controller=Home}/{action=Index}/{id?}`, so `/Student/Details/5` calls `StudentController.Details(5)`. Attribute routing uses `[Route("...")]`.

```
// Program.cs (ASP.NET Core MVC)
var builder = WebApplication.CreateBuilder(args);
builder.Services.AddControllersWithViews();       // register MVC services
var app = builder.Build();

app.UseStaticFiles();
app.UseRouting();
app.MapControllerRoute(
    name: "default",
    pattern: "{controller=Home}/{action=Index}/{id?}");
app.Run();
```

Program 5.4: Enabling MVC and the default route

5.6 Model

A **model** is a plain C# class (POCO) representing data. **Data annotations** add validation and display rules. In real applications a model is usually mapped to a database table with Entity Framework Core through a `DbContext`.

bealink.com

```
using System.ComponentModel.DataAnnotations;

public class Student
{
    public int Id { get; set; }

    [Required(ErrorMessage = "Name is required")]
    [StringLength(50, MinimumLength = 2)]
    public string Name { get; set; }

    [Range(16, 60, ErrorMessage = "Age must be between 16 and 60")]
    public int Age { get; set; }

    [EmailAddress, Display(Name = "E-mail")]
    public string Email { get; set; }
}

// EF Core context
public class CollegeContext : DbContext
{
    public CollegeContext(DbContextOptions<CollegeContext> o) : base(o) { }
    public DbSet<Student> Students { get; set; }
}
```

Program 5.5: Model with data annotations and DbContext

5.7 Controller

A **controller** is a class ending in "Controller" that inherits from `Controller`. Its public methods are **action methods**. Actions return an `IActionResult` (older MVC: `ActionResult`).

Return type	Produces
View()	Renders a Razor view as HTML (ViewResult).
RedirectToAction() / Redirect()	HTTP redirect to another action / URL.
Json(obj)	JSON data (JsonResult) — often used with AJAX.
Content(text)	Plain text.
File(...)	File download.
NotFound() / BadRequest()	HTTP 404 / 400 status result.

Data is passed to views with a strongly typed **model** (best), or with `ViewBag` / `ViewData` (current request) and `TempData` (survives one redirect). `[HttpGet]` and `[HttpPost]` select the action by HTTP verb.

bcalink.com

```
public class StudentController : Controller
{
    private static List<Student> list = new List<Student>();

    public IActionResult Index()                // GET /Student
    {
        ViewBag.Title = "Student List";
        return View(list);                      // pass model to the view
    }

    [HttpGet]
    public IActionResult Create() { return View(); }     // show empty form

    [HttpPost]
    [ValidateAntiForgeryToken]
    public IActionResult Create(Student s)               // model binding
    {
        if (!ModelState.IsValid) return View(s);         // validation failed
        s.Id = list.Count + 1;
        list.Add(s);
        TempData["msg"] = "Student added";
        return RedirectToAction("Index");
    }

    public IActionResult Details(int id)                 // GET /Student/Details/3
    {
        var s = list.FirstOrDefault(x => x.Id == id);
        if (s == null) return NotFound();
        return View(s);
    }
}
```

Program 5.6: Controller with Index, Create (GET/POST) and Details

5.8 Razor View

Razor is a compact syntax for mixing C# with HTML in **.cshtml** files. The `@` symbol switches from HTML to C#. Views are placed in `Views/<ControllerName>/<Action>.cshtml`.

Razor syntax	Meaning
`@Model.Name`	Output an expression (HTML-encoded automatically).
`@{ var x = 5; }`	C# code block.
`@if / @foreach / @for`	Control structures that generate markup.
`@model Student`	Declares the strongly typed model (top of view); use `@model List<Student>` for lists.
`@Html.TextBoxFor(m => m.Name)`	HTML helper generating an input element.
`<input asp-for="Name" />`	ASP.NET Core Tag Helper (modern alternative to HTML helpers).
`@Html.Partial / <partial>`	Inserts a reusable partial view.
`@RenderBody() / _Layout.cshtml`	Layout (master page) with a placeholder where each view is inserted.

bcalink.com

```
@* Views/Student/Index.cshtml *@
@model List<Student>
<h2>@ViewBag.Title</h2>
<p class="text-success">@TempData["msg"]</p>
<a asp-action="Create">Add New</a>
<table class="table">
  <tr><th>Id</th><th>Name</th><th>Age</th><th></th></tr>
  @foreach (var s in Model)
  {
    <tr>
      <td>@s.Id</td><td>@s.Name</td><td>@s.Age</td>
      <td><a asp-action="Details" asp-route-id="@s.Id">Details</a></td>
    </tr>
  }
</table>

@* Views/Student/Create.cshtml *@
@model Student
<form asp-action="Create" method="post">
  <div asp-validation-summary="All"></div>
  <label asp-for="Name"></label>
  <input asp-for="Name" /> <span asp-validation-for="Name"></span>
  <label asp-for="Age"></label>
  <input asp-for="Age" /> <span asp-validation-for="Age"></span>
  <button type="submit">Save</button>
</form>
```

Program 5.7: Razor views for listing and creating students

Note: In MVC 5 (.NET Framework) the same views use HTML helpers such as @Html.TextBoxFor and @Html.ActionLink instead of Tag Helpers, and controllers return ActionResult.

bcalink.com

Key Takeaways — Unit 5

✓ AJAX = asynchronous partial-page updates using XMLHttpRequest/fetch with JSON or XML.

✓ ScriptManager is required; UpdatePanel (ContentTemplate + Triggers), UpdateProgress and Timer give partial rendering.

✓ Server data for AJAX: [WebMethod] page methods and [ScriptService] web services; Microsoft AJAX Library is the client side.

✓ MVC = Model (data/rules), View (Razor UI), Controller (handles request, returns result).

✓ Default route: {controller=Home}/{action=Index}/{id?}.

✓ Controller actions return IActionResult (View, Redirect, Json, NotFound…); use [HttpGet]/[HttpPost] and ModelState.IsValid.

✓ Models use data annotations; Razor views use @ syntax, @model, layouts and Tag/HTML helpers.

— End of Unit 5 —