ASP.NET Development

Slides

https://www.accelebrate.com
(877) 849-1850 ❖ info@accelebrate.com

Customized Technical Training

Accelebrate
ACCELERATED LEARNING, CELEBRATED RESULTS®

On-Site, Customized Private Training

Don't settle for a one-size-fits-all class! Let Accelebrate tailor a private class to your group's goals and experience. Classes can be delivered at your site or online (or a combination of both) worldwide. Visit us at **https://www.accelebrate.com** and contact us at **sales@accelebrate.com** for details.

Public Online Training

Need to train just 1-3 people? Attend one of our regularly scheduled, live, instructor-led public online classes. For course dates, times, outlines, pricing, and registration, visit **https://www.accelebrate.com/public-training-schedule.**

Newsletter

Want to find out about our latest class offerings? Subscribe to our newsletter **https://www.accelebrate.com/newsletter**.

Blog

Get insights from our instructors and staff! Visit our blog, **https://www.accelebrate.com/blog** and join the discussion threads to get feedback from our instructors!

Learning Resources

Get access to tutorials, how-to's, and both past and upcoming webinars in the **Insights** section of our website.

Call us for a training quote!

877 849 1850

aws PARTNER

Microsoft Learning Partner

Accelebrate, Inc. was founded in 2002 with the goal of delivering **private training** that rapidly achieves participants' goals. Each year, our experienced instructors deliver hundreds of classes online and at client sites all over the US, Canada, and abroad. We pride ourselves on our **instructors' real-world experience** and ability to **adapt the training** to your team and their objectives. We offer a wide range of topics, including:

- AWS, Azure, and Cloud Computing
- Angular, React, and Vue
- JavaScript
- Data Science using R, Python, & Julia
- Excel Power Query
- Power BI & Tableau
- .NET & VBA programming
- SharePoint & Microsoft 365
- DevOps & CI/CD
- iOS & Android Development
- PostgreSQL, Oracle, and SQL Server
- Java, Spring, and Groovy
- Agile, DEI, & IT Leadership
- Web/Application Server Admin
- HTML5 & Mobile Web Development
- Adobe & Articulate
- Docker, Kubernetes, Ansible, & Git
- Software Design and Testing
- AND MORE *(see back)*

"Our organization and members love Accelebrate! Every training opportunity has been well-received, well-presented, and easy to set up. Not only is Accelebrate a great training corporation, but the customer service is unmatched."

— Emily, MnCCC

Like us on Facebook + Follow us on Twitter + Watch us on YouTube

Visit our website for a complete list of courses!

Adobe & Articulate
Adobe Captivate
Adobe Presenter
Articulate Storyline / Studio
Camtasia
RoboHelp

AWS, Azure, & Cloud
AWS
Azure
Google Cloud
OpenStack
Terraform
VMware

Big Data
Alteryx
Apache Spark
Teradata
Snowflake SQL

Data Science and RPA
Apache Airflow
Blue Prism
Data Literacy
Django
Julia
Machine Learning
MATLAB
Minitab
Python
R Programming
SPSS
UiPath

Data Visualization
BusinessObjects
Crystal Reports
Excel Power Query
Power BI
PivotTable and PowerPivot
Qlik
Tableau

Database
MongoDB
NoSQL Databases
Oracle
Oracle APEX
PostgreSQL
SQL Server

DevOps, CI/CD & Agile
Agile
Ansible
Apache Maven
Chef
DEI
Docker
Git
IT Leadership
ITIL
Jenkins
Jira & Confluence
Kubernetes
Linux
Microservices
OpenShift
Six Sigma
Software Design

Java
Groovy and Grails
Java & Web App Security
JavaFX
JBoss
Scala
Selenium & Cucumber
Spring Boot
Spring Framework

JS, HTML5, & Mobile
Angular
CSS
D3.js
Flutter
HTML5
iOS/Swift Development
JavaScript
Node.js & Express
React & Redux
Svelte
Swift
Symfony
Xamarin
Vue

Microsoft & .NET
.NET Core
ASP.NET
Azure DevOps
Blazor
C#
Design Patterns
Entity Framework Core
IIS
Microsoft Dynamics CRM
Microsoft 365
Microsoft Power Platform
Microsoft Project
Microsoft SQL Server
Microsoft System Center
Microsoft Windows Server
PowerPivot
PowerShell
VBA
Visual C++/CLI
Visual Studio
Web API

Security
.NET Web App Security
C and C++ Secure Coding
C# & Web App Security
Linux Security Admin
Python Security
Secure Coding for Web Dev
Spring Security

SharePoint
Power Automate & Flow
SharePoint Administrator
SharePoint Developer
SharePoint End User
SharePoint Online
SharePoint Site Owner

SQL Server
Azure SQL Data Warehouse
Business Intelligence
Performance Tuning
SQL Server Administration
SQL Server Development
SSAS, SSIS, SSRS
Transact-SQL

Teleconferencing Tools
Adobe Connect
GoToMeeting
Microsoft Teams
WebEx
Zoom

Web/Application Server
Apache httpd
Apache Tomcat
IIS
JBoss
Oracle WebLogic

Other
C++
Go Programming
Mulesoft
Rust
Salesforce
Sitefinity
UX

*Visit **www.accelebrate.com/newsletter** to sign up and receive our **newsletters** with information about **new courses**, **free webinars**, **tutorials**, and **blog articles**.*

Call us for a training quote! 877 849 1850 (US/Canada) or +1 678 648 3133

ASP.NET Development

Agenda

- Introduction
- Pages and Controls
- Data Validation
- C# Language
- State Management
- Master Pages
- Themes
- Navigation
- Database Connectivity

ASP.NET Development

Agenda

- Application Settings
- Ajax Programming
- Authentication
- ASP.NET MVC
- Web API
- ASP.NET Core

ASP.NET Development

Introduction

- Evolution of Web Development
- Evolution of Microsoft Web Development
- .NET Support Status
- Visual Studio and Visual Studio Code

Introduction

Evolution of Web Development

- Serving of static files
- Dynamic server-side content (CGI, Classic ASP, PHP)
- Dynamic client-side content (JavaScript)
- Web Services
- Asynchronous client-server communication (Ajax)
- Single-page applications [SPAs] (Angular, React)
- WebSockets
- WebAssembly

Introduction

Evolution of Microsoft Web Development

- Classic ASP
  - Dynamic server-side content using VBScript of JScript
- ASP.NET WebForms
  - Server-generated content with event-driven model
  - Postback architecture
  - Ajax-style functionality added later
- Windows Communication Foundation (WCF)
  - XML/SOAP (JSON API support added later)

Introduction

Evolution of Microsoft Web Development

- ASP.NET MVC
  - Developed as an alternative to WebForms
  - Avoids much of the machinery used by WebForms
  - Easier to integrate unit testing and other Web frameworks
- ASP.NET Web API
  - RESTful JSON services
  - Much easier to configure and use vs. WCF

Introduction

Evolution of Microsoft Web Development

- ASP.NET Core
  - Web development framework for .NET Core
  - Version of .NET Core after 3.1 named ".NET 5"
  - WebForms and WCF no longer available (still supported as part of .NET Framework)
  - Applications can be developed in an MVC style or using a more page-focused approach with Razor Pages
  - Web API framework no longer exists as a separate framework

Introduction

Evolution of Microsoft Web Development

- Blazor Server
  - Provides the ability to implement client-side functionality with server-side .NET code
  - Updates are sent to the client via a persistent network connection
- Blazor WebAssembly
  - Uses a WebAssembly version of the .NET CLR to allow the execution of .NET code on the client side

Introduction

.NET Support Status

- Each version of .NET has a lifecycle status
  - Current – Includes the latest features and bug fixes but will only be supported for a short time after the next release
  - LTS (Long-Term Support) – Has an extended support period
  - Preview – Not supported for production use
  - Out of support – No longer supported

`dotnet.microsoft.com/download`

Introduction

Visual Studio and Visual Studio Code

- Visual Studio is available for Windows and macOS
  - Full-featured IDE
- Visual Studio Code is available for Windows, macOS, and Linux
  - Includes IntelliSense and debugging features
  - Thousands of extensions are available for additional functionality

`visualstudio.microsoft.com`

Introduction

Visual Studio and Visual Studio Code

- JetBrains also offers an IDE for .NET development called Rider
- Available for Windows, macOS, and Linux
- Includes advanced capabilities in the areas of refactoring, unit testing, and low-level debugging

www.jetbrains.com/rider

ASP.NET Development

Pages and Controls

- Page Class
- Basic Server Controls
- Other Server Controls

WebForms Architecture

Page Class

- The Page class is the foundation of the user interface of a WebForms application
  - Defined via .aspx file
  - Page directive used to specify a code behind file
- First time it is requested, a page is compiled on the server and cached for subsequent requests
- The Page class defines many useful properties and events

WebForms Architecture

Basic Server Controls

- Server controls use the "<asp:" tag prefix and the have the runat property set to server
- Server controls know how to render themselves to HTML based on their current state
  - Control state can be set as design-time via tag attributes or programmatically at run-time

WebForms Architecture

Basic Server Controls

- Label
- TextBox
- Button
  - An event handler can be added for the OnClick event
  - Command properties can also be used
- Table

WebForms Architecture

Other Server Controls

- Calendar
- CheckBox
  - AutoPostBack properties determines if toggling the control triggers events on the server
- RadioButton
  - Inherits from CheckBox and adds a GroupName property to define a group of mutually exclusive options
- FileUpload
  - Allows the user to select a local file
  - Server-side SaveAs method used to handle the file

WebForms Architecture

Other Server Controls

- HyperLink
  - Can display text or an image
  - Useful when the link needs to be adjusted programmatically
- Image
  - Like HyperLink, do not use unless there is a need to access the control from code
- ImageButton
- ImageMap
  - Can contain hot spots with different postback values

WebForms Architecture

Other Server Controls

- LinkButton
  - Looks like a link but behaves like a button
- ListControl
  - Subclasses include BulletedList, CheckBoxList, DropDownList, ListBox, and RadioButtonList
  - Properties include AutoPostBack, SelectedIndex, and SelectedItem (of type ListItem)

WebForms Architecture

Other Server Controls

- Panel
  - Holds a group of related server controls
  - Useful to hide (or show) the entire group from code

ASP.NET Development

Data Validation

- Introduction
- BaseValidator
- Validator Controls

Data Validation

Introduction

- Any data that arrives from the client should not be trusted
- Client-side validation provides a good user experience but cannot be depended on from a security standpoint
- Data validation classes in .NET can help with performing client-side and server-side validation
  - For client-side validation, JavaScript is injected into the page (uses the jQuery validation library via the AspNet.ScriptManager.jQuery NuGet package)

Data Validation

BaseValidator Class

- All the validator classes inherit from BaseValidator
- Common properties include...
  - ControlToValidate
  - ErrorMessage (used by ValidationSummary control)
  - IsValid
  - SetFocusOnError
  - Text
  - ValidationGroup

Data Validation

Validator Controls

- RequiredFieldValidator
- CompareValidator
  - Can be used to check data type, compare to a constant, or compare to another control
- RangeValidator
- RegualrExpressionValidator
- CustomValidator
  - Includes a ClientValidationFunction property where the name of a custom JavaScript function can be specified

Data Validation

Validator Controls

- ValidationSummary
  - Can display a list of all errors on a page (bulleted list, paragraph, or dialog box)

ASP.NET Development

C# Language

- Types
- Exception Handling
- Type Conversions
- Control Operations
- String Operations
- Collections
- User Defined Types

C# Language

Types

- Value types vs. reference types
- Copy on assignment
- Enumerations
- Implicit typing

C# Language

Exception Handling

- An exception occurs in response to a runtime error
- try-catch-finally block can be used to catch, handle, and optionally re-throw an exception
- Some exceptions may wrap another exception
  - Available via the InnerException property

C# Language

Type Conversions

- Widening conversions
- Narrowing conversions
- Convert class
- Parse / TryParse

C# Language

Control Operations

- Equality (value vs. reference)
- Logical operators
- Decision structures
- Iterative structures

C# Language

String Operations

- Concatenations
- Escape sequences
- Verbatim literals
- String class

C# Language

Collections

- Arrays
- List
- Stack
- Queue
- Concurrent collection classes

C# Language

User Defined Types

- Class vs. Struct
- Fields
- Methods
- Properties

ASP.NET Development

State Management

- Introduction
- Client-Side State
- Server-Side State

State Management

Introduction

- By default, Web applications are stateless
- When state is required, it can be achieved by using client-side state or server-side state

State Management

Client-Side State

- View State
- Control State
- Hidden Fields
- Cookies
- Query String

State Management

Server-Side State

- Session State
- Application State
- Cache

ASP.NET Development

Master Pages

- Introduction
- Content Pages

Master Pages

Introduction

- A master page is a template that contains elements common to every associated content page
- Can define content placeholders for content that will be provided by a content page
- A single application can define multiple master pages
- Refer to the documentation for the order in which page events will fire

Master Pages

Content Pages

- A page can use the MasterPageFile attribute of the Page directive to identify the master page is will use
- The asp:Content element is used to provide content for a placeholder defined in the master page

ASP.NET Development

Themes

- Introduction
- Skin Files
- CSS Files

Themes

Introduction

- A theme allows you to define the default characteristics of an application's page controls
- Defined as a subdirectory under App_Themes
- Can contain skin files and CSS files
- Skin files are used for setting server control properties
- Theme can be selected for an individual page using the Theme property of the Page directive
- Set for an entire application by using the theme property of the <pages> element in Web.config

Themes

Skin Files

- A skin file can contain default skins and named skins

```
<asp:button runat="server" BorderStyle="Double" Font-Names="Arial" />
```

```
<asp:button runat="server" SkinID="skiButton" BorderStyle="Double"
            Font-Names="Arial" />
```

Themes

CSS Files

- CSS files that are part of a theme are sent to the client and applied to the HTML by the browser on the client
- Keep in mind that CSS files are often cached by the browser

ASP.NET Development

Navigation

- SiteMap
- Menu
- TreeView

Navigation

SiteMap

- The SiteMap class stores a hierarchically organized list of SiteMapNode objects
- Several navigation controls use a SiteMap as their source of data
- Makes maintenance easier since modifying the SiteMap will affect all controls that are using it

Navigation

Menu

- Menu control can use a SiteMap or be populated programmatically
- Contains a static menu (top level) and a dynamic menu (pop-up)

Navigation

TreeView

- TreeView can also use a SiteMap or be populated programmatically

ASP.NET Development

Database Connectivity

- Data-Bound Controls
- Data Sources
- Control Filtering
- FormView
- ListView
- DataPager

Database Connectivity

Data-Bound Controls

- ASP.NET WebForms offers several controls that can be bound to a data source
  - BulletedList
  - CheckBoxList
  - DetailsView
  - DropDownList
  - FormView
  - GridView
  - ListBox
  - Repeater

Database Connectivity

Data Sources

- SqlDataSource communicates directly with a relational database
- ObjectDataSouce uses calls class methods

Database Connectivity

Control Filtering

- Parameters used by a data source can be provided by another control, request values or session values

Database Connectivity

FormView

- FormView displays a form for editing a single item provided by a data source
- Uses a collection of templates for different operations

Database Connectivity

ListView

- ListView control can be used to display data in a grid style with the editing capability
- Like a FormView, templates are used for the different modes that an item can be in

Database Connectivity

DataPager

- The DataPager can be used to implement paging behavior when associated with a ListView control
- NextPreviousPagerField control can used to add next/previous navigation controls
- NumericPagerField control can be used to add page number-based navigation

ASP.NET Development

Application Settings

- Web.config File
- WebConfigurationManager

Application Settings

Web.config

- An application's Web.config file is used for storing settings that affect the functionality of the application
  - ASP.NET settings (themes, debugging, etc.)
  - IIS settings (URL rewriting, redirects, etc.)
  - App settings (database connection strings, API keys, etc.)
- Web.config transformations can be used to provide different settings based on the build environment

Application Settings

WebConfigurationManager

- The WebConfigurationManager class makes it easy to access settings in the Web.config file
- Pre-defined <connectionStrings> section should be used for database connection strings

```
WebConfiguration.ConnectionStrings["MyConnStr"]
```


ASP.NET Development

Ajax Programming

- ScriptManager
- UpdatePanel
- UpdateProgress
- Ajax Control Toolkit

Ajax Programming

ScriptManager

- The ScriptManager class can be used to manage the automatic download of Ajax scripts
  - Asynchronous postbacks
  - Partial-page rendering

Ajax Programming

UpdatePanel

- The UpdatePanel can be used to define a page region that will be asynchronously posted back to the server and re-rendered in the browser
- When a postback is triggered within an UpdatePanel, the entire page lifecycle will still execute on the server but only the contents of the panel will update on the client
- Important to check how much extra work is being done beyond what is required

Ajax Programming

UpdateProgress

- The UpdateProgress class can be used to provide the end user with feedback about the progress of an asynchronous postback
- Can be connected to an UpdatePanel
- Content of UpdateProgress displayed when asynchronous request starts and hidden when complete

Ajax Programming

Ajax Control Toolkit

- The Ajax Control Toolkit is an open-source library built on top of the basic Ajax functionality provided by .NET

ASP.NET Development

Authentication

- ASP.NET Identity
- Authorization

Authentication

ASP.NET Identity

- ASP.NET Identity is the membership system used by the project templates for WebForms, ASP.NET MVC, and Web API
  - Can also be added manually via the NuGet package
- The identity system can be used to create a membership database, or you can use a custom data source

Authentication

Authorization

- For a WebForms application, authorization rules can be specified via a Web.config file in the relevant folder

```
<system.web>
  <authorization>
    <allow roles="canEdit"/>
    <deny users="*"/>
  </authorization>
</system.web>
```


Authentication

Authorization

- UI elements can be adjusted based on the authenticated user

```
<ul class="nav navbar-nav">
  <li><a runat="server" id="adminLink" visible="false"
         href="~/Admin/AdminPage">Admin</a></li>
  <li><a runat="server" href="~/">Home</a></li>
  <li><a runat="server" href="~/About">About</a></li>
</ul>
```

```
protected void Page_Load(object sender, EventArgs e)
{
  if (HttpContext.Current.User.IsInRole("canEdit")) {
    adminLink.Visible = true;
  }
}
```

ASP.NET Development

ASP.NET MVC

- Goals of Modern Web Development
- Introduction to MVC
- Request Routing
- Controllers
- Views

ASP.NET MVC

Goals of Modern Web Development

- Design based on standards
  - HTML5, CSS3, JavaScript
- Provide a good experience for different client types
  - Desktop, tablet, phone
  - Can be done client-side (responsive) and/or server-side (adaptive)
- Expose clean, consistent, meaningful URLs
- Leverage client-side frameworks
  - jQuery, Bootstrap, Angular, React, …

ASP.NET MVC

Introduction to MVC

- ASP.NET MVC is a framework for building web applications
- Applies the Model-View-Controller (MVC) design pattern to the ASP.NET framework
- Uses the same underlying plumbing as ASP.NET Web Forms
  - Request, Response, Session, Cache, etc.
- Avoids use of System.Web.UI components
  - Server controls, ViewState, etc.
- Open source

ASP.NET MVC

Introduction to MVC

- The MVC design pattern provides an elegant means of separating concerns within an application
- Has been used in dozens of frameworks on a variety of platforms

ASP.NET MVC

Introduction to MVC

- Model
  - Set of classes that describe the data you're working with as well as reusable business logic
- View
  - The application's user interface (UI)
  - Defines how model data is represented to users
- Controller
  - Set of classes that handle communication from the user, overall application flow, and application-specific logic

ASP.NET MVC

Introduction to MVC

- Each module should focus on what it is good at while maintaining as little knowledge of the other parts as possible
- Model should not know anything about Views and Controllers
- View should not know about the existence of Controllers
- Controller should know about the existence of relevant Views and their individual purpose but avoid going deeper
  - Controller code should provide data to a View but should not be aware of the view's implementation

ASP.NET MVC

Introduction to MVC

- Testing your code for accuracy and errors is at the core of good software development
- Unit testing as a concept should be part of all development
- Testability and a loosely-coupled design go hand-in-hand
- Even if not writing tests, keeping testability in mind helps to create more flexible, maintainable software

ASP.NET MVC

Introduction to MVC

- ASP.NET MVC applications, by default, rely heavily on conventions
- Some examples...
  - Controller classes are expected to end with Controller and are typically located in a folder named Controllers
  - View template files are stored in \Views\ [ControllerName]
- All convention-based defaults can be overridden if desired

ASP.NET MVC

Introduction to MVC

- HTTP request received
- Check is done for a physical file that corresponds to the URL
  - If a file exists, it is returned
- If no physical file exists, the routing engine uses the URL and a list of rules to determine a destination controller
- An instance of the controller class is created and an appropriate action method is invoked
- The action method returns a result that is sent to the client

ASP.NET MVC

Request Routing

- In Global.asax.cs, Application_Start calls RegisterRoutes( ) of RouteConfig class
- RegisterRoutes is responsible for adding routes to the application's RouteTable
- It is possible to add routes using information obtained from an external source (e.g. separate XML file)
  - Avoid expensive operations (e.g. reading from a database) since this code will run whenever the application is restarted
  - Remember that a failure in creating the RouteTable will prevent the application from starting

ASP.NET MVC

Request Routing

- Instead of adding a route in RegisterRoutes, you can add an attribute to the controller action itself
  - Use whichever technique you prefer or use a mixture of both
- If using attribute routing, you must tell MVC to add the attribute-based routes by calling MapMvcAttributeRoutes in RegisterRoutes

```
[Route("product/{id}")]
public ActionResult Details(int id) { ... }
```

```
routes.MapMvcAttributeRoutes();
```


ASP.NET MVC

Controllers

- Controllers are responsible for responding to user requests
- May need to retrieve or make modifications to model objects
- Determines the appropriate response to return
  - HTML, JSON, XML, redirection, error, etc.

ASP.NET MVC

Controllers

- Each controller class typically has a name that ends with the word Controller
  - For example, ProductController
- Controller classes are typically located in a project sub-folder (and namespace) named Controllers
- Controller classes typically inherit from a framework class named Controller
- ASP.NET MVC developers typically define one controller class for each type of model object

ASP.NET MVC

Controllers

- Controller actions can return many different types of results
- ActionResult is the base class for many specialized result types

ActionResult Type	Description
EmptyResult	Represents a null or empty response
ContentResult	Writes the specified content directly to the response as text
JsonResult	Serializes the objects it is given into JSON
RedirectResult	Redirects the user to the given URL
HttpNotFoundResult	Redirects to a URL specified via Routing parameters
ViewResult	Calls into a View engine to render a View
PartialViewResult	Renders a partial View to the response
FileResult	Writes a binary response to the stream

ASP.NET MVC

Views

- In MVC, the View is responsible for providing the user interface to the client
- Transforms model data into a format for presentation to the user
  - Most often HTML but a view can render other formats as well

ASP.NET MVC

Views

- The default view engine behavior is to look for a view with the same name as the action located in a folder under Views with the same name as the controller
- It is possible to specify a different view name or a full path

```
return View("AnotherView");
```

```
return View("~/Views/Stuff/SomeOtherView");
```

ASP.NET MVC

Views

- Razor provides a streamlined syntax for expressing views
- Minimizes the amount of syntax and extra characters
- Understands the transitions between code and markup

```
<ul>
<% foreach (Course course in Model) { %>
  <li><%: course.Number %> is named <%: course.Title %>.</li>
<% } %>
</ul>
```

```
<ul>
@foreach (Course course in Model) {
  <li>@course.Number is named @course.Title.</li>
}
</ul>
```


ASP.NET MVC

Views

- The output from a Razor expression is automatically HTML encoded
- Disable using Html.Raw( )

```
<span>@Html.Raw(course.Description)</span>
```

ASP.NET MVC

Views

- ASP.NET MVC provides a collection of helper methods by defining extension methods for classes HtmlHelper, UrlHelper, and AjaxHelper
- Make it easier to generate common pieces of view content and help with maintenance by dynamically generating content based on things like the routing configuration or model fields
- Properties of WebViewPage make it easy to access helpers

```
public HtmlHelper<object> Html { get; set; }
public AjaxHelper<object> Ajax { get; set; }
public UrlHelper Url { get; set; }
```


ASP.NET MVC

Views

- WebViewPage contains several properties used for accessing data provided by a controller

```
public ViewDataDictionary ViewData { get; set; }
public dynamic ViewBag { get; }
public object Model { get; }
```

ASP.NET Development

Web API

- Introduction
- ApiController
- HTTP Status Code

Web API

Introduction

- ASP.NET Web API is a framework for building HTTP services on top of the .NET Framework
- Result of collaboration between ASP.NET MVC and WCF teams
- Web API is its own stand-alone framework and technically not part of ASP.NET MVC
- Very often used in conjunction with ASP.NET MVC

Web API

Introduction

- Standard ASP.NET MVC controllers can be use to create an HTTP service but that approach has some shortcomings
- WCF can also be used to create HTTP services but can be complex and difficult to configure
- Web API represents a middle ground
  - More natural dispatching to actions based on HTTP verb
  - Accepting and generating different content types
  - Content type negotiation
  - Hosting outside of the ASP.NET runtime stack

Web API

ApiController

- Like ASP.NET MVC, Web API maps HTTP requests to controller actions
- Controllers inherit from ApiController
- Actions directly render the resulting model object as the response (typically no views)
- Supports automatic content negotiation via the request Accept header (XML and JSON)

Web API

ApiController

```
public class CourseController : ApiController
{
  // GET api/<controller>
  public IEnumerable<Course> Get() { ... }

  // GET api/<controller>/5
  public Course Get(int id) { ... }

  // POST api/<controller>
  public void Post([FromBody]Course course) { ... }

  // PUT api/<controller>/5
  public void Put(int id, [FromBody]Course course) { ... }

  // DELETE api/<controller>/5
  public void Delete(int id) { ... }
}
```


Web API

HTTP Status Codes

- When using Web API, your API methods should return the proper HTTP status codes when appropriate
  - Default is 200 (OK)

```
throw new HttpResponseException(HttpStatusCode.NotFound);
```

```
public HttpResponseMessage Post(Product item)
{
  item = repository.Add(item);
  var response =
    Request.CreateResponse<Product>(HttpStatusCode.Created, item);

  string uri = Url.Link("DefaultApi", new { id = item.Id });
  response.Headers.Location = new Uri(uri);
  return response;
}
```

ASP.NET Development

ASP.NET Core

- Introduction
- Application Startup
- Middleware and Services
- Application Configuration
- Controllers and Views
- Error Handling
- Web APIs
- Security

ASP.NET Core

Introduction

- Single stack for Web UI and Web APIs
- Modular architecture distributed as NuGet packages
- Flexible, environment-based configuration
- Built-in dependency injection support
- Support for using an MVC-based architecture or a more page-focused architecture by using Razor Pages
- Blazor allows for the implementation of client-side functionality using .NET code

ASP.NET Core

Application Startup

- When an ASP.NET Core application is launched, the first code executed is the application's Main method
  - Generated by the compiler if using top-level statements
- Code in the Main method is used to…
  - Create a WebApplication object
  - Configure application services
  - Configure the request processing pipeline
  - Run the application

ASP.NET Core

Middleware

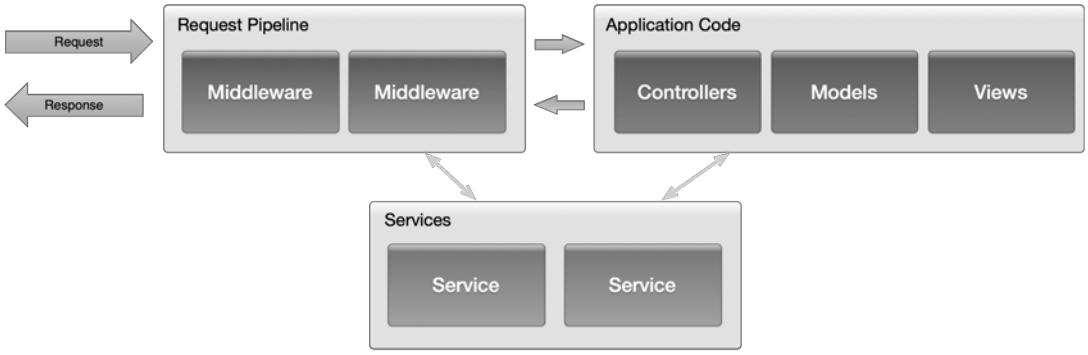
- ASP.NET uses a modular request processing pipeline
- The pipeline is composed of middleware components
- Each middleware component is responsible for invoking the next component in the pipeline or short-circuiting the chain
- Examples of middleware include…
  - Request routing
  - Handling of static files
  - User authentication
  - Response caching
  - Error handling

ASP.NET Core

Services

- ASP.NET Core also includes the concept of services
- Services are components that are available throughout an application via dependency injection
- An example of a service would be a component that accesses a database or sends an email message

ASP.NET Core



ASP.NET Core

Application Configuration

- The default WebApplicationBuilder adds providers to read settings (in the order shown) from:
  - appsettings.json
  - appsettings.{Environment}.json
  - User secrets
  - System environment variables
  - Command-line arguments
- Values read later override ones read earlier

ASP.NET Core

Application Configuration

- The configuration API provides the ability to read from the constructed collection of name-value pairs
- An object of type IConfiguration is available to be used via dependency injection

```
public class HomeController : ControllerBase
{
  public HomeController(IConfiguration configuration)
  {
    _emailServer = configuration["EmailServer"];
  }
}
```

ASP.NET Core

Controllers and Views

- Controllers and Views in ASP.NET Core work much the same as in ASP.NET MVC (with some new features and capabilities)

ASP.NET Core

Error Handling

- Handle errors as best you can when they occur
- Record the error information and/or send a notification
- Provide the user with an appropriate response
  - Do not reveal information that a malicious user could potentially use against you (e.g., database schema information)
  - Give the user some options (e.g., link to visit the home page in the case of a 404)
  - Use static content whenever possible to avoid an error page that itself produces an error

ASP.NET Core

Web APIs

- ASP.NET Core includes a class named ControllerBase
  - Includes many properties and methods for handling HTTP requests
- The Controller class inherits from ControllerBase and adds support for views
- If creating a controller that does not have any views, you should inherit directly from ControllerBase

ASP.NET Core

Web APIs

- An API controller should be decorated with the ApiController attribute

```
[ApiController]
public class ProductApiController : ControllerBase
```

  - Automatic HTTP 400 responses for validation failures
  - Problem details for error status codes

ASP.NET Core

OpenAPI / Swagger

- OpenAPI is a specification for describing REST APIs
- Swagger is a collection of tools that work with OpenAPI
  - SwaggerDocument objects expose data about the API in JSON format (openapi.json)
  - Swagger UI is a dynamically generated web-based UI that can be used to view and test API methods

ASP.NET Core

Testing APIs

- API endpoints that are exposed via GET are easy to test using a web browser
- For other verbs, it can be helpful to have a tool that can be used to craft custom HTTP requests
  - Postman application is very popular (getpostman.com)
  - Many other options are available

Web APIs

Cross-Origin Resource Sharing (CORS)

- Browser security prevents a web page from making Ajax requests to another domain
- CORS is a W3C standard that allows a server to relax this policy
- A server can explicitly allow some cross-origin requests
- CORS is configured in ASP.NET Core via a service and middleware

```
services.AddCors();
```

```
app.UseCors(builder =>
    builder.WithOrigins("https://example.com"));
```


ASP.NET Core

Security

- Authentication
- Authorization
- Secrets Management