Become a **Master** in **ASP.NET** (from **Scratch**)

1

Customizing Controllers

Let's start with the first set of slides

Taking Control of Controllers

- Adding Actions
- Model Binding
- Filters
- Vanity URLs
- Controller Best Practices

2

Adding Actions

Let's start with the second set of slides

Adding Actions

- Controllers are classes
- Actions are methods

- Creating an action involves adding a method to a class

Action Signature

- Return Types
  - ActionResult
    - FileResult
    - JsonResult
    - ViewResult
- Parameters
  - Normal parameters
  - MVC model binding

Get and Post

- Create/Update/Delete are typically two step operations
  - Present the form
  - Accept the input
- Create two actions
  - Form presentation via HttpGet (default)
  - Accept data via HttpPost

3

Demo

Model Binding

4

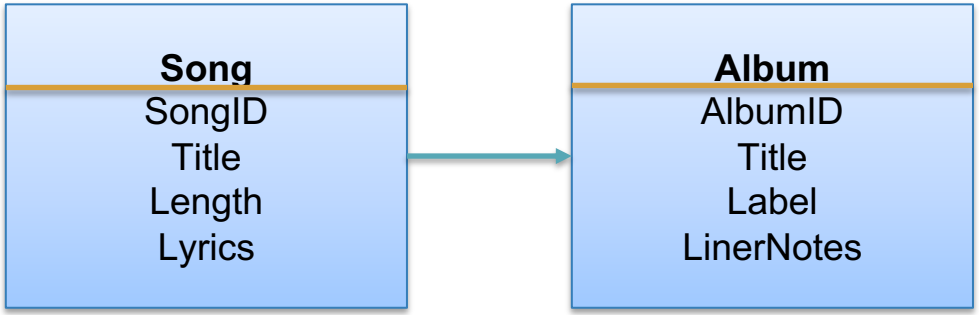
Model Binding

Let’s start with the fourth set of slides

Default Model Binder

- "It just works" – Jon Galloway
- Uses the name attribute of input elements
  - Automatically matches parameter names for simple data types
  - Complex objects are mapped by property name
    - Complex properties use dotted notation

```
<input type="text" name="Album.LinerNotes" />
```



Controlling Model Binding

- Imagine the following model

- Need
  - Create a form to edit everything but the lyrics
- Challenge
  - Default model binder automatically binds all inbound properties

Solutions

- Simplest
  - Use the bind attribute to indicate which properties to bind

```
Edit([Bind(Include = "SongID,Title,Length")]
Song song)
```

- Other solutions
  - Create a view model
  - Create a custom model binder

5

Demo

BindAttribute

6

Filters

Let’s start with the sixth set of slides

Normal Action Execution

User Request

MVC Instantiates Controller

Action is Executed

Model is Combined with View

HTML Returned

Actions with Filters

User Request

MVC Instantiates Controller

Pre-execution Filter Code Executes

Action is Executed

Post-execution Filter Code Executes

Model is Combined with View

HTML Returned

Adding Filters

- Action
- Controller
- Global
  - FilterConfig.cs

Security Filters

- Authorize
  - Control who can access a controller/action
  - Properties
    - Users
    - Roles
- ValidateAntiForgeryToken
  - Defends against cross-site request forgery
  - Requires anti-forgery token to be added to view
- RequireHttps
  - Requries SSL

SSL

- Encrypts traffic and prevents tampering
- Authenticates server

- When to use SSL
  - Asking for sensitive information
  - After authentication

7

Demo

Security Filters

8

Vanity URLs

Let's start with the eighth set of slides

Standard URL

www.mymusicstore.com/App/Album/Details/Display.aspx?ID=42&BandID=64

- Users have no idea what that URL refers to
- Search engines have no idea what that URL refers to
- It’s just plain ugly

Vanity URL

www.mymusicstore.com/Album/Cure/Wish

- User knows information provided by the page
- Search engines know information provided by page
- Don’t underestimate the importance of vanity URLs

MVC Routing

- Vanity URLs are handled by routing
- Routing in MVC controls what controller/action is called based on the URL provided
- Methods for updating routing
  - RouteConfig.cs
  - AttributeRouting

9

Demo

RouteConfig.cs

Attribute Routing

- Attributes control routing/URL
- RouteAttribute

```
[Route("Album/Edit/{id:int}")]
public ActionResult Edit(int id)
```

  - www.mymusicstore.com/Album/Edit/42
  - Calls the Edit action
  - Passes in the ID parameter
  - ID must be an integer

RoutePrefix

- Added to controller
- Adds prefix to all routes

```
[RoutePrefix("Album")]
public class AlbumsController : Controller
{
    [Route("Album/Edit/{id:int}")]
    public ActionResult Edit(int id)
    {
        // code
    }
}
```

10

Demo

Attribute Routing

11

Vanity URLs

Let's start with the eleventh set of slides

Controller Design Guidelines

- High Cohesion
  - Make sure all actions are closely related
- Low Coupling
  - Controllers should know as little about the rest of the system as possible
  - Simplifies testing and changes
  - Repository pattern
    - Wrap data context calls into another object