Module 4

Developing Controllers

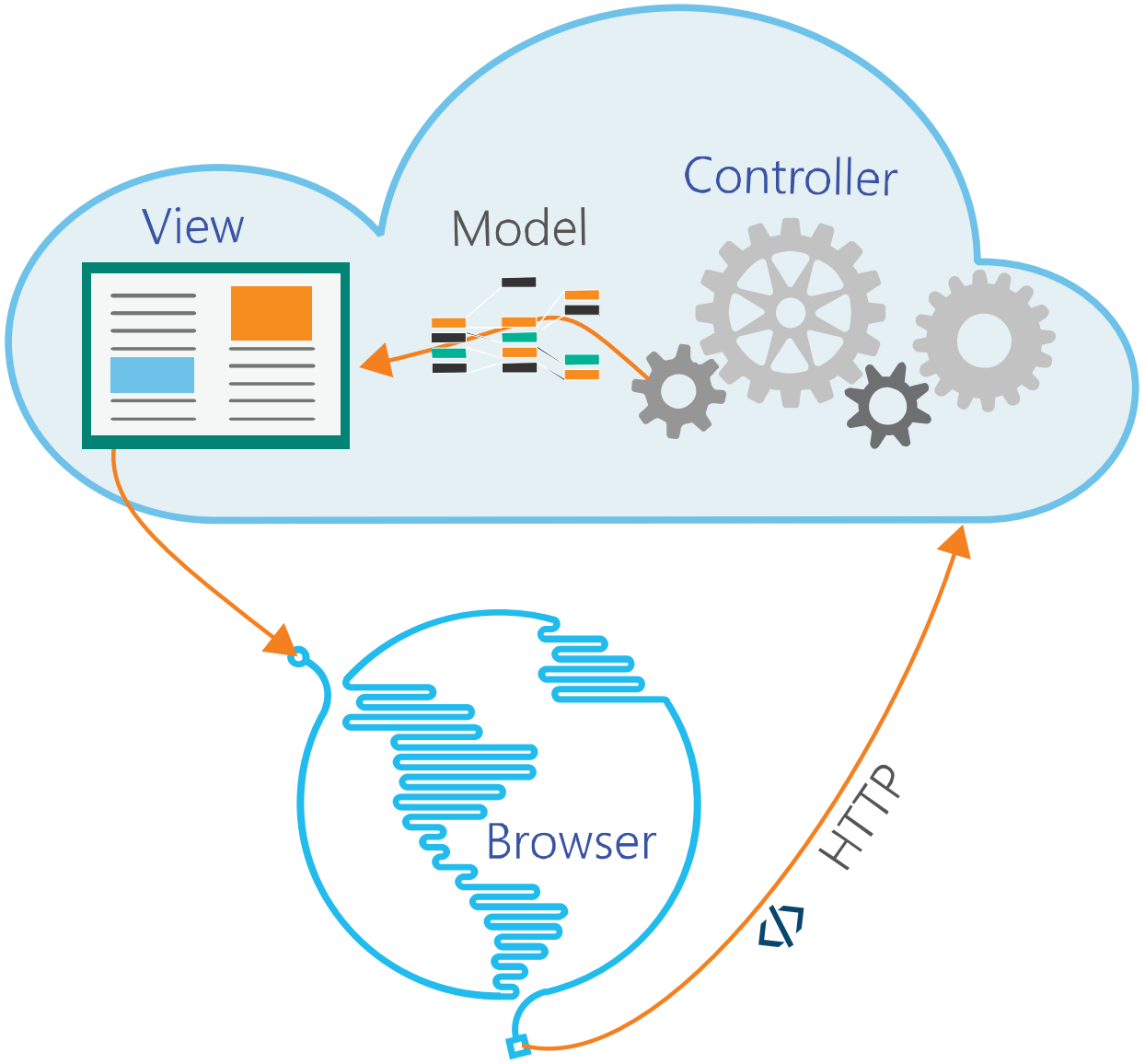
Module Overview

- Writing Controllers and Actions
- Configuring Routes
- Writing Action Filters

Lesson 1: Writing Controllers and Actions

- Responding to User Requests
- Writing Controller Actions
- Using Parameters
- Using ViewBag and ViewData to Pass Information to Views
- Demonstration: How to Write Controllers and Actions

Responding to User Requests



Writing Controller Actions

- An action is a public method of the controller class
- Actions can return objects that implement the **IActionResult** interface

```
public class HomeController : Controller
{
    public IActionResult Index()
    {
        return View();
    }
}
```

Action Result Types

Actions can return different types of results:

- ViewResult
- ContentResult
- RedirectToActionResult
- RedirectToRouteResult
- StatusCodeResult

Return a ContentResult Object

An action that returns a ContentResult:

```
public class AnotherController : Controller
{
    public ContentResult AnotherAction()
    {
        return Content("some text");
    }
}
```

Using Parameters

- The model binders obtain parameters from a user request and pass them to action methods
- There are several ways to retrieve parameters, including:
  - The Request property
  - The FormCollection object
  - Routing

Using ViewBag and ViewData to Pass Information to Views

- Models are the best way to pass data from controllers to views
- In some cases, you may want to augment the model with additional data without modifying it

Using the ViewBag Property

Adding Information:

```
ViewBag.Message = “some text”;

ViewBag.ServerTime = DateTime.Now;
```

Retrieving Information:

```
<p>

   Message is: @ViewBag.Message

</p>

<p>

   Server time is: @ViewBag.ServerTime.ToString()

</p>
```

Demonstration: How to Write Controllers and Actions

In this demonstration, you will learn how to:

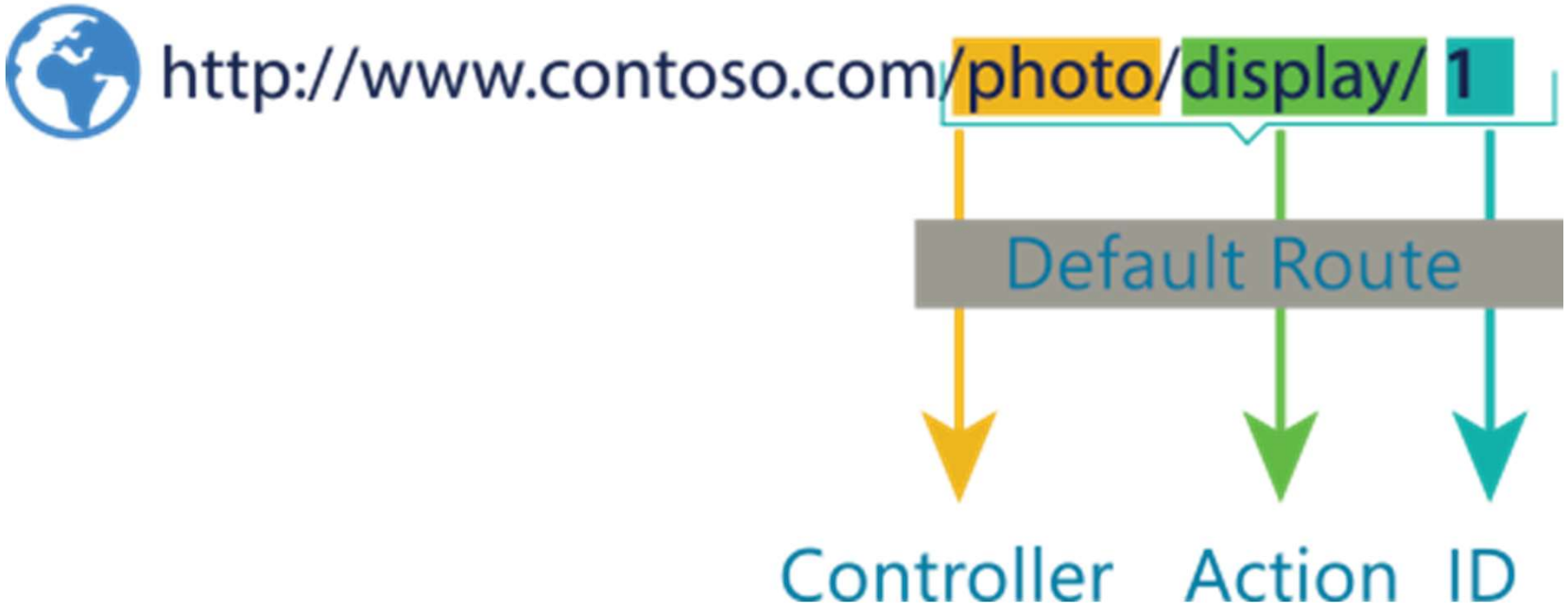
- Add a controller to an MVC application
- Add an action that creates a model and passes it to a view
- Add another action that gets a parameter
- Add another action that returns a ContentResult object
- Use the RouteData.Values collection to access the action method data
- Use the ViewBag and ViewData properties to pass data from a controller to a view

Lesson 2: Configuring Routes

- The ASP.NET Core Routing Engine
- Discussion: Why Add Routes?
- What Is Search Engine Optimization?
- Configuring Routes by Using Convention-Based Routing
- Using Routes to Pass Parameters
- Configuring Routes by Using Attributes
- Demonstration: How to Add Routes

The ASP.NET Core Routing Engine

- Routing determines which controller and action should be called to handle a request
- Routing can be configured centrally in the Startup.cs file and locally by using attributes



Discussion: Why Add Routes?

Why add routes?

- To make URLs easier for site visitors to understand
- To improve search engine rankings

What Is Search Engine Optimization?

- Most users find web applications by using search engines
- It is important to ensure that your web application appears at the top of the search engine results

Configuring Routes by Using Convention-Based Routing

- Convention-based routes might contain the following properties: name, template, defaults, constraints and dataTokens
- Custom routes can be added as shown below:

```
app.UseMvc(routes =>
  {
      routes.MapRoute(
          name: "default",
          template: "{controller}/{action}/{param}",
          defaults: new { controller = "Some", action = "ShowParam" },
          constraints: new { param = "[0-9]+" });
  });
```

- Routes should be added in the appropriate order

Using Routes to Pass Parameters

You can access the values of segment variables by using one of two methods:

- **Using the RouteData.Values collection**

```
public IActionResult Print()
{
    string id = (string)RouteData.Values["id"];
    return Content("id: " + id);
}
```

- **Using model binding to pass appropriate parameters to actions**

```
public IActionResult Print(string id)
{
    return Content("id: " + id);
}
```

Configuring Routes by Using Attributes

- Attribute-based routing allows you to configure routes using attributes
- It allows you to define your routes in the same file as the controller that they refer to

```
[Route("Some")]
public IActionResult SomeMethod()
{
    return Content("Some method");
}
```

- Convention-based routing and attribute-based routing can be used in the same application

Demonstration: How to Add Routes

In this demonstration, you will learn how to:

- Add convention-based routes to an MVC application
- Add attribute-based routes to actions
- Add an attribute-based route with Http[Verb] attribute

Lesson 3: Writing Action Filters

- What Are Action Filters?
- Creating and Using Action Filters
- Demonstration: How to Create and Use Action Filters

What Are Action Filters?

- Filters are MVC classes that can be used to manage cross-cutting concerns in your web application

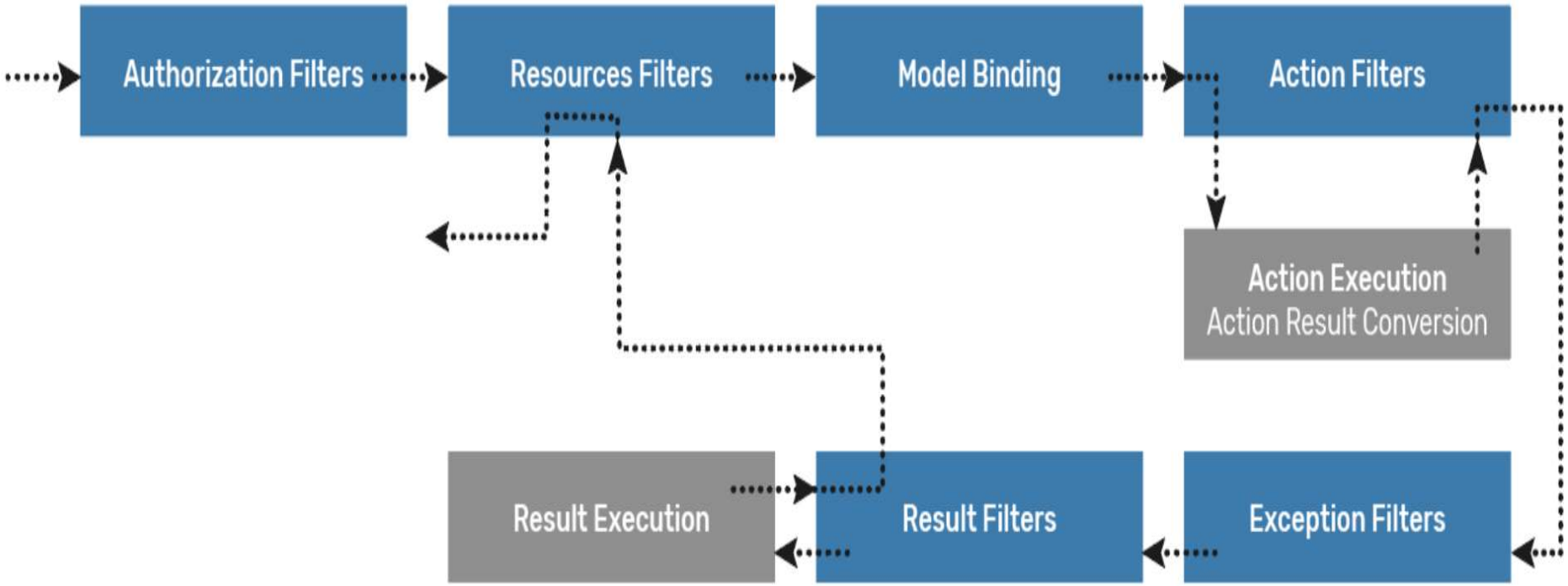

Authorization

Logging


Caching

- You can apply a filter to an action by annotating the action method with an appropriate attribute

Filter Types



Creating and Using Action Filters

Sample Action Filter

```
public class SimpleActionFilter : ActionFilterAttribute {

    public override void OnActionExecuting
        (ActionExecutingContext filterContext) {

        Debug.WriteLine("This Event Fired: OnActionExecuting");

    }

    public override void OnActionExecuted
        (ActionExecutedContext filterContext) {

        Debug.WriteLine("This Event Fired: OnActionExecuted");

    }

}
```

Demonstration: How to Create and Use Action Filters

In this demonstration, you will learn how to:

- Add a filter to an MVC application
- Annotate an action with a filter

Lab: Developing Controllers

- Exercise 1: Adding Controllers and Actions to an MVC Application
- Exercise 2: Configuring Routes by Using the Routing Table
- Exercise 3: Configuring Routes by Using Attributes
- Exercise 4: Adding an Action Filter

Estimated Time: 70 minutes

Lab Scenario

You have been asked to add controllers to a new application. The controllers should include actions that return a view. You have also been asked to add an action that returns a photo as a .jpg file to show on a webpage and add an action that redirects to another action in another controller. Additionally, you are asked to configure routes in a variety of ways.

The members of your development team are new to ASP.NET Core MVC and they find the use of controller actions confusing. Therefore, you need to help them by adding a component that displays action parameters in an external file whenever an action runs. To achieve this, you will add an action filter.

Lab Review

- You decided to add a new action to the CityController controller. The action gets two parameters of type int. Are the routes that are currently configured in the application sufficient to handle the requests to this action or will you need to configure a new route?
- A new controller with an action is added to the WorldJourny MVC application. You want to ensure that every access to the new action is written to the external file. How can you achieve this?

Module Review and Takeaways

- Best Practice
- Common Issues and Troubleshooting Tips