Understanding URL Routing and Page Flow

Alex Wolf

www.crywolfcode.com

Exploring Action Results

Action Results

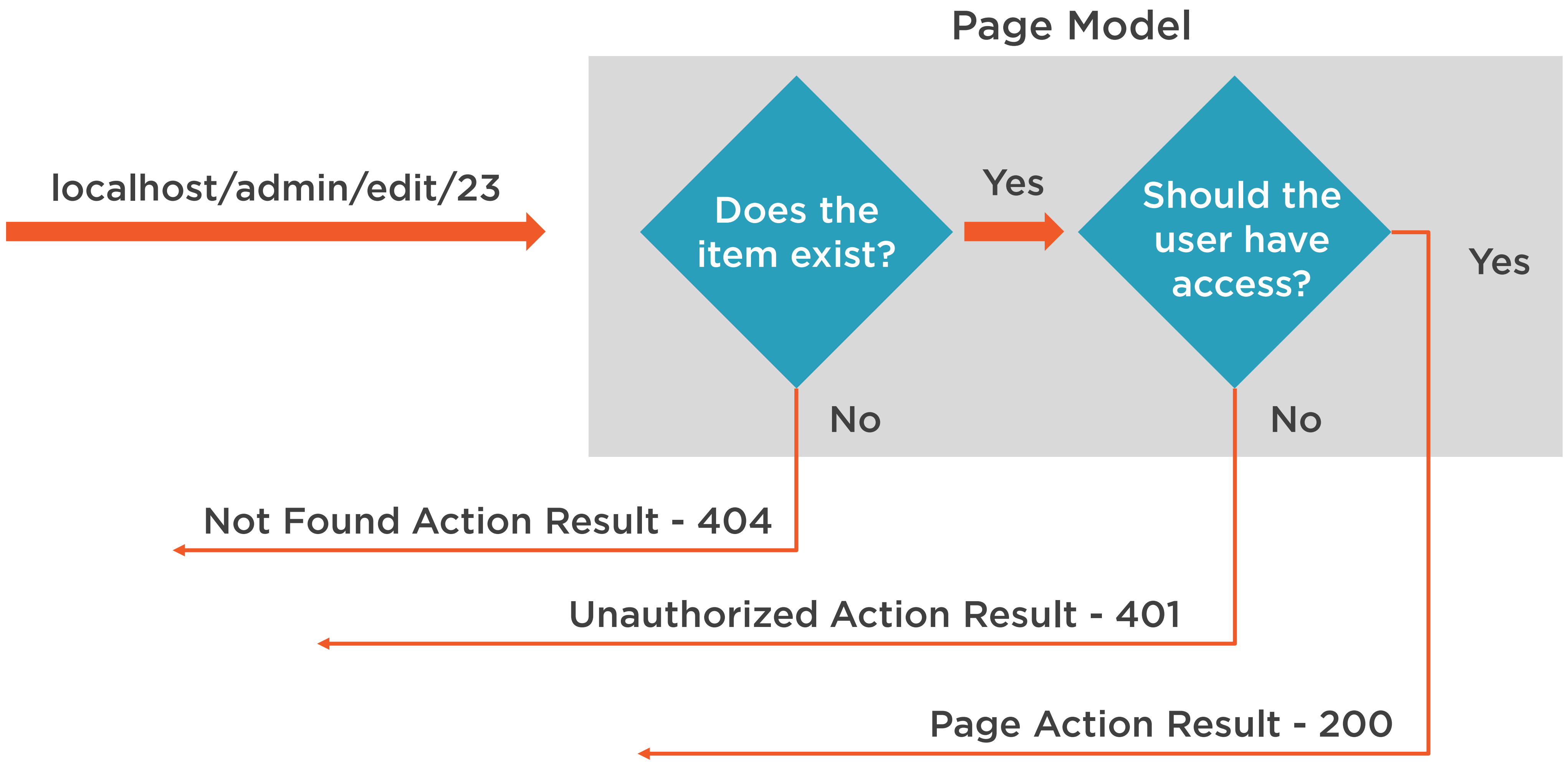
Components that generate different responses for a page.

```
public class ContactModel : PageModel
{
    public void OnGet()
    {
        // Returns void
    }

    public IActionResult OnPost()
    {
        return new PageResult();
    }
}
```

◀ Simple handler method that returns void

◀ Action Results can generate different types of responses

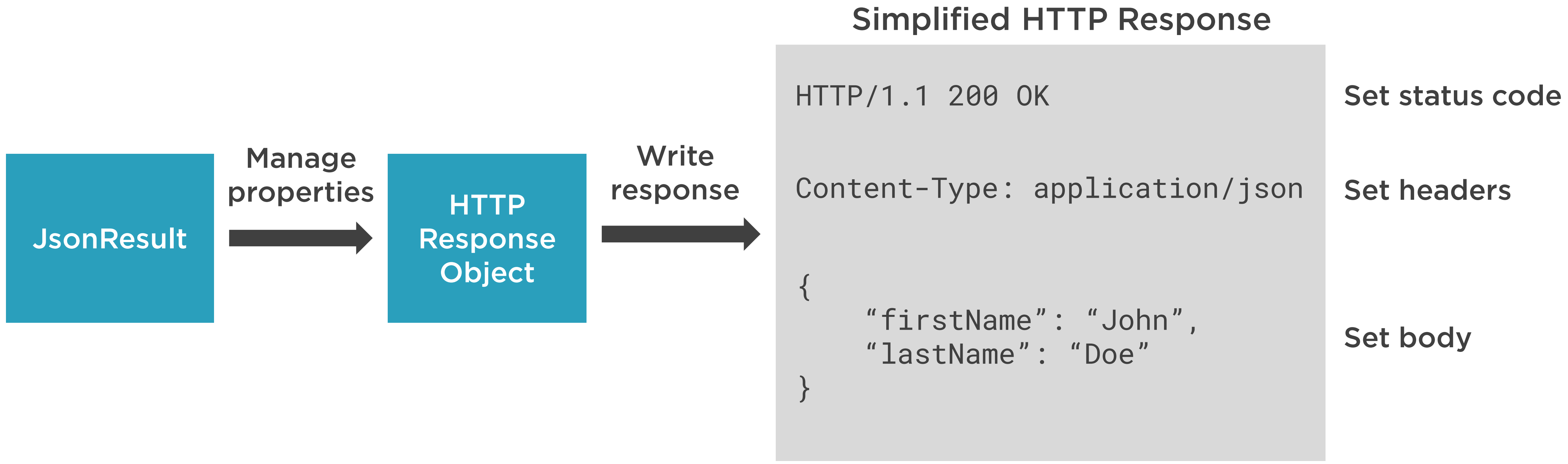
Action Result Workflows


Page Model
localhost/admin/edit/23
Does the item exist?
Yes
Should the user have access?
Yes
No
No
Not Found Action Result - 404
Unauthorized Action Result - 401
Page Action Result - 200

Action Result Types

Action Result Type	Generated Response
RedirectToPageResult	301 redirect to another page
ContentResult	A simple string of content
JsonResult	JSON formatted data
Many more!	Status codes, different content formats, etc.

Understanding the HTTP Response Object



```
HTTP/1.1 200 OK

Content-Type: application/json

{
    “firstName”: “John”,
    “lastName”: “Doe”
}
```

Action Result Helper Methods

Helper Method	Generated Action Result
return Page()	PageResult
return NotFound()	NotFoundResult
return Unauthorized()	JsonResult
Many more!	Various results

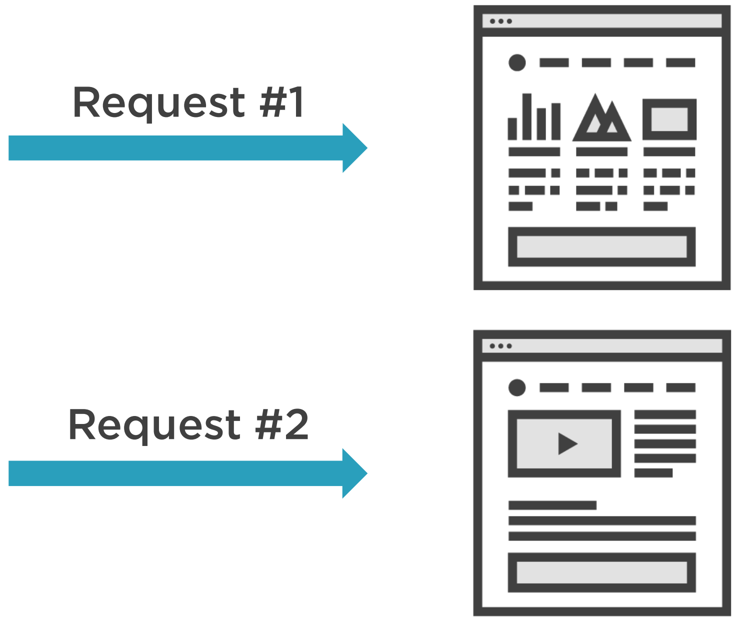
Demo

Demo: Improving Page Flow with Action Results

Understanding Routing Conventions

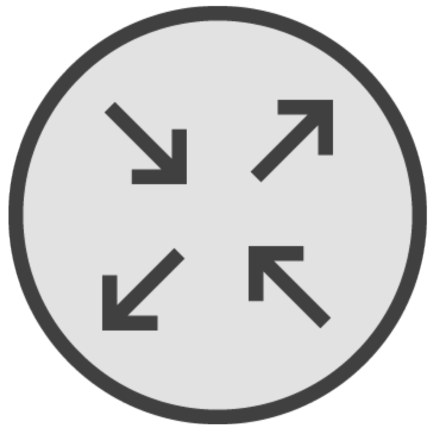
A note on Middleware.

The Primary Purposes of Routing



URL Mapping

Route incoming requests to the right Razor Page



URL Generation

Create links and URLs that point to the right places

Convention vs. Configuration

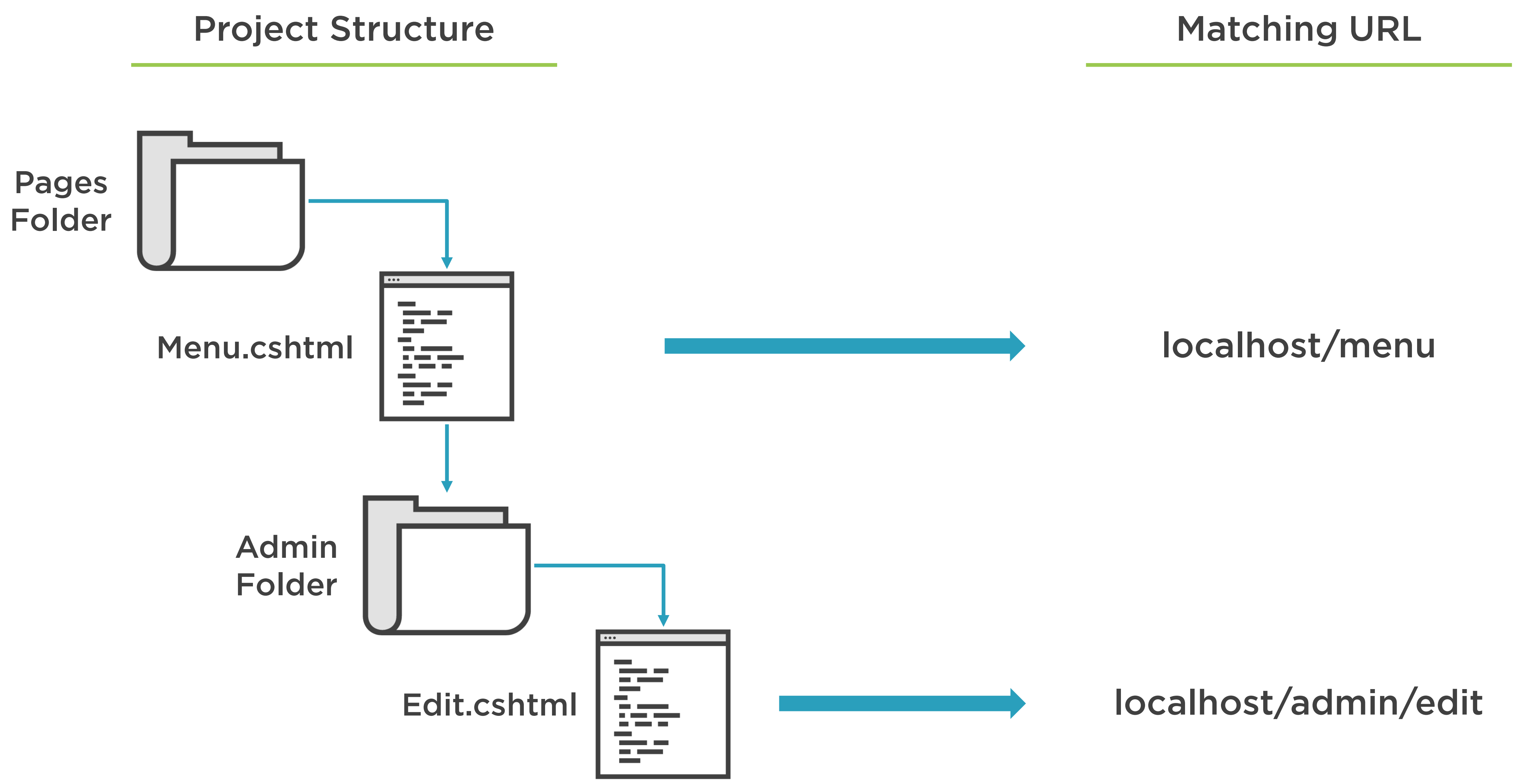
Convention

Relies on assumed naming and structural patterns

Configuration

Relies on custom templates and settings

Route Mapping Conventions



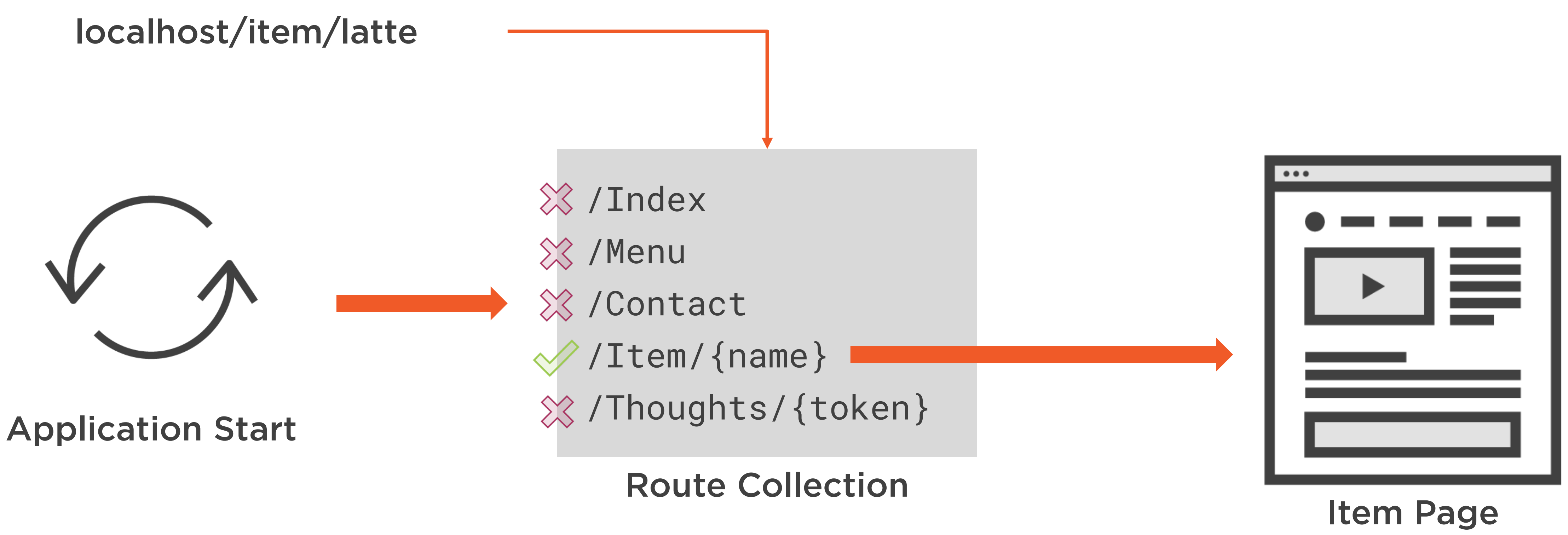
```
@page "/menu-item/{name}"


<div class="container">
    <!-- Page markup -->
</div>
```

Custom Route Templates

Route templates can be declared next to the page directive

The templates can include static segments, variables, and constraints

Technical Routing Concepts



Tag Helpers and Routing

Anchor Tag Helper

```
<a class="nav-link" asp-page="Item" asp-route-id="@Item.Name">Text</a>
```

Page with custom route

/menu-item/{name}

Rendered HTML

```
<a class="nav-link" href="/menu-item/mocha-latte">Text</a>
```

```
services.AddMvc().AddRazorPagesOptions(opts =>
{
    opts.Conventions.AddPageRoute("/Index", "Wired");
    opts.Conventions.AddPageRoute("/Index", “Home");
    opts.RootDirectory = "/Features";
});
```

Application Routing Configurations

We can add routes and configurations at the application level

Demo

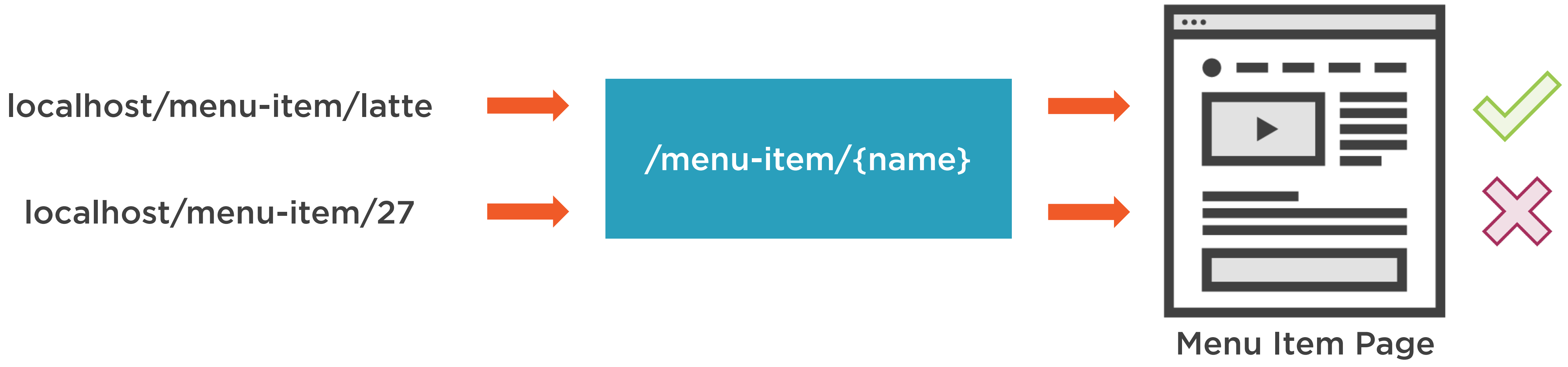
Demo: Working with Route Templates

Introducing Route Constraints

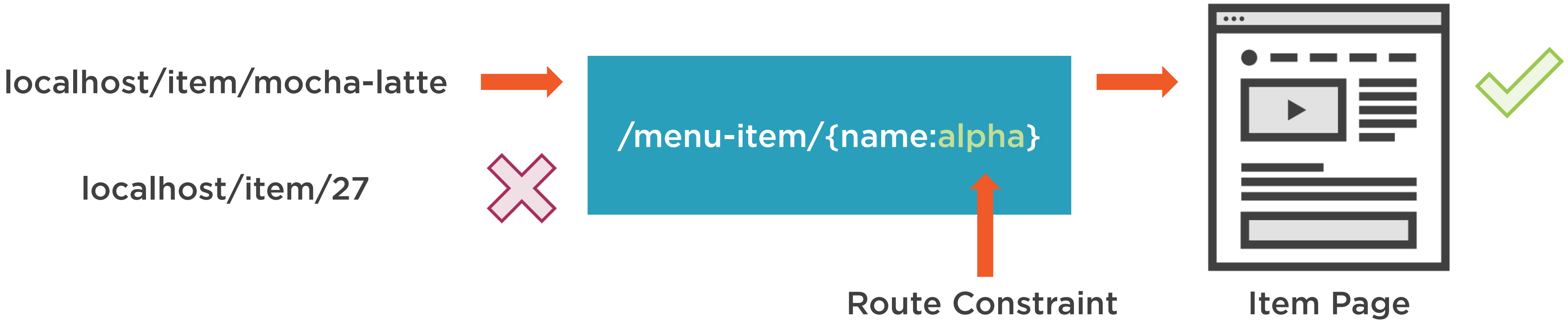
Route Constraints

Apply rules to determine whether the content of a URL matches a route

The Need for Route Rules



Enforcing URL Content Rules



Included Route Constraints

Route Constraint	URL Parameter Rule
double	Must be int parseable
minlength(n)	Must be at least n characters
datetime	Must be a date time
guid	Must be formatted as guid
Many more!	Other data formats

```
public interface IRouteConstraint {

    public bool Match(HttpContext httpContext, IRouter route, string
    routeKey, RouteValueDictionary values, RouteDirection routeDirection)

    {

        // Custom logic

    }

}
```

The IRouteConstraint Interface

Can be used to create custom route constraints

The Match method determines if a URL segment is valid.

Demo

Demo: Adding Route Constraints

Demo

Demo: Creating a Custom Route Constraint

Summary

Action Results can help streamline and customize Razor Page responses

Routing maps incoming URLs to Razor Pages, and helps generate application links

Routing is convention based to provide useful defaults

Custom templates and configurations can further extend Routing behavior

Route Constraints can enforce rules around the content of a URL

Custom Route Constraints can be created using the IRouteConstraint interface