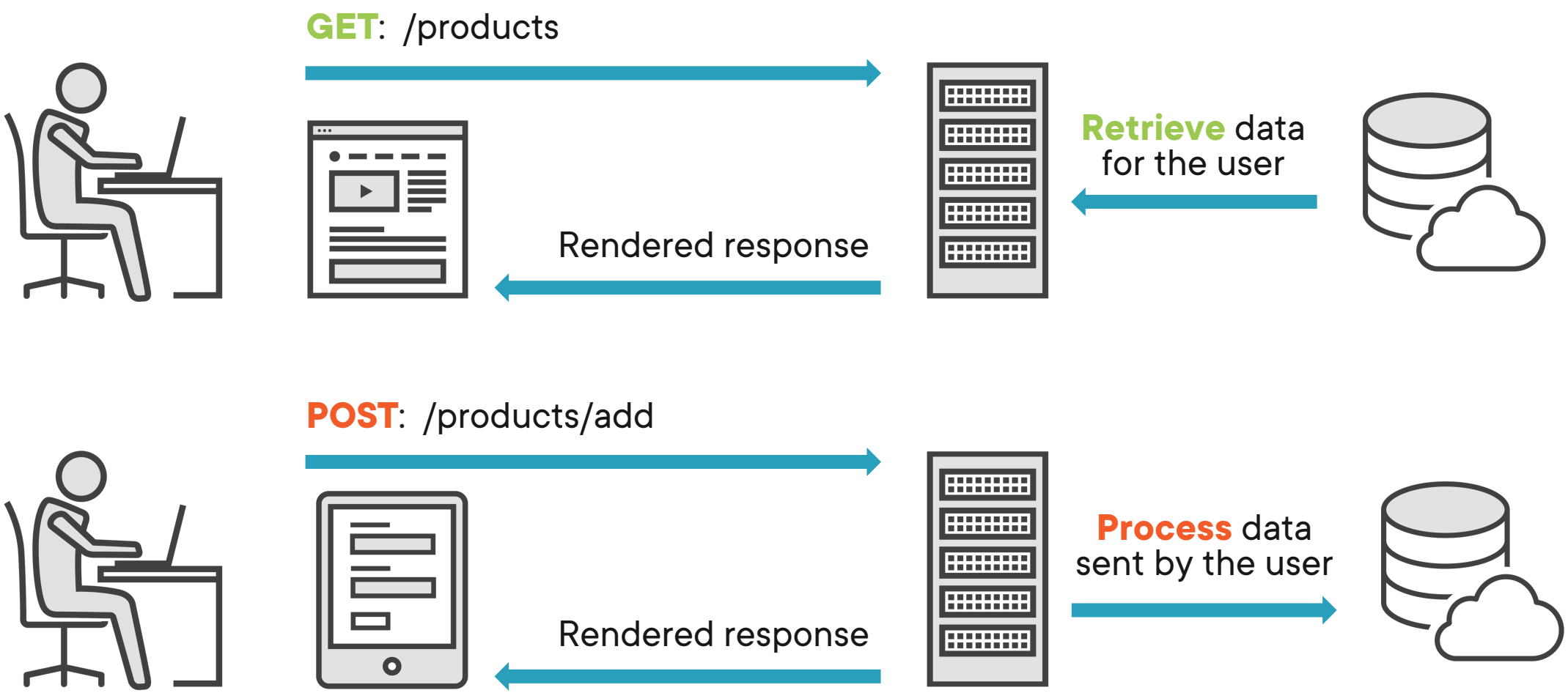
Working with Forms, Binding and Validation

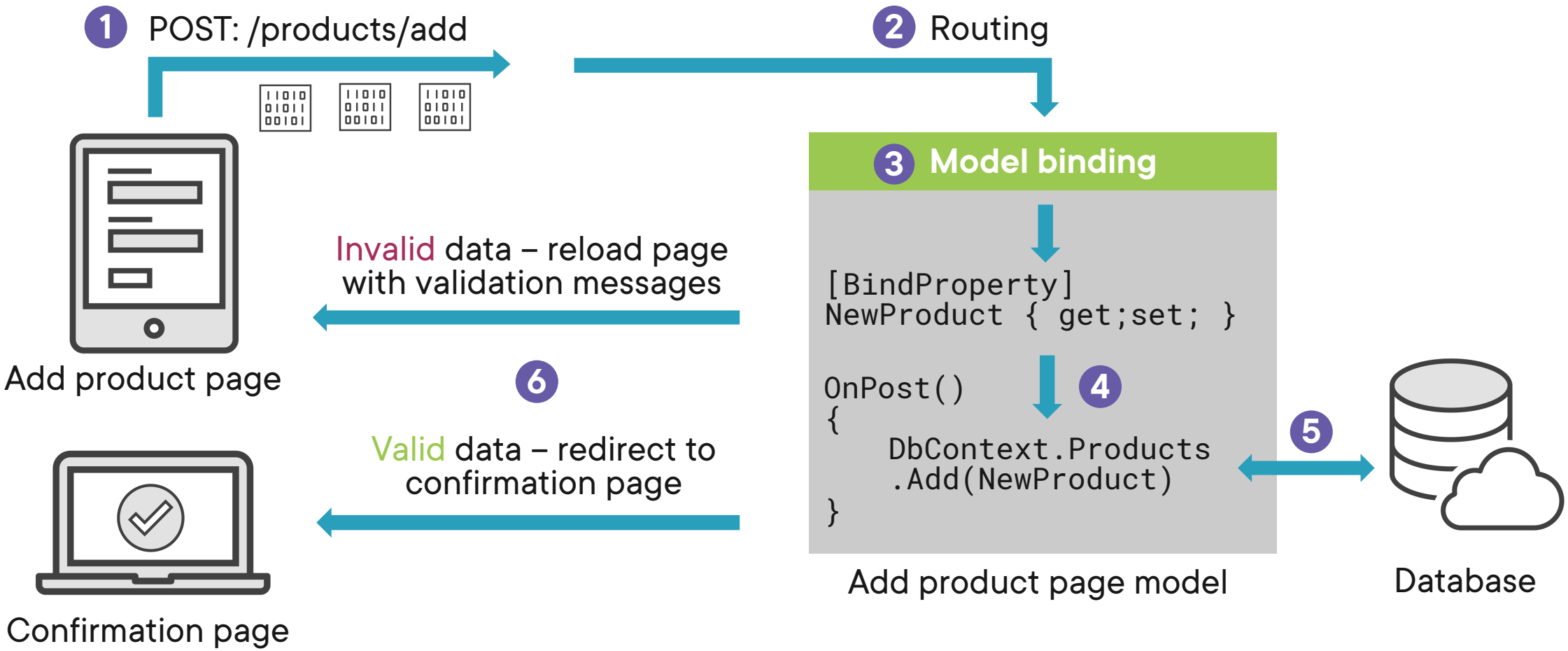
Alex Wolf
.NET Developer

www.thecodewolf.com

Getting vs. Sending Data



The Razor Pages Form Workflow

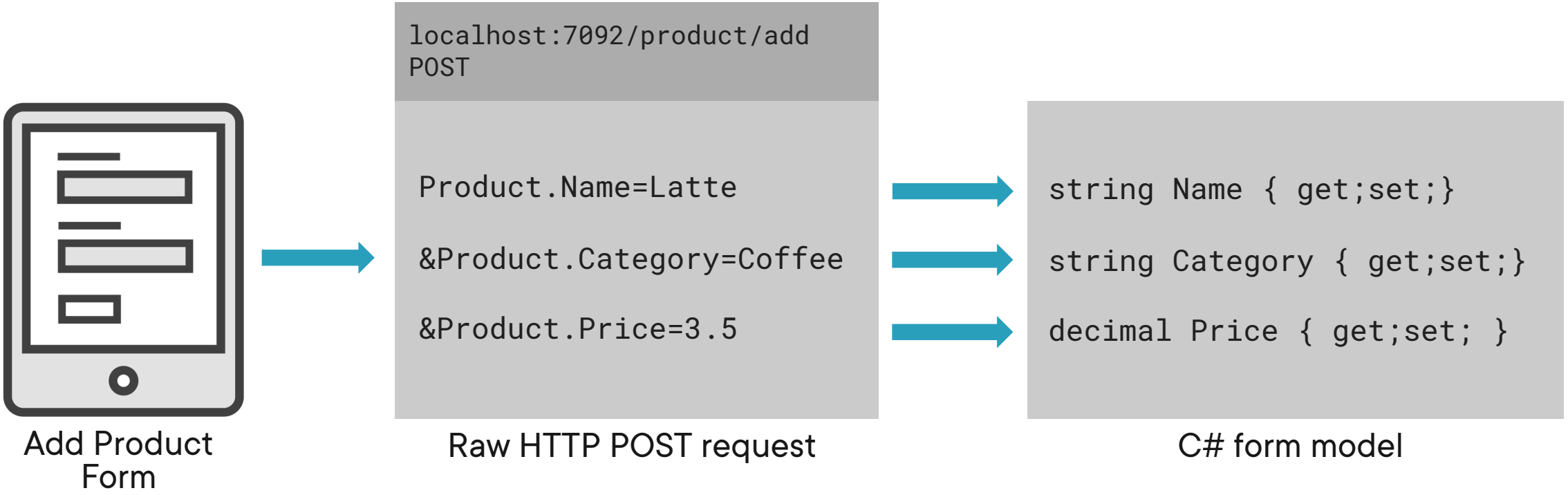


Understanding Model Binding

Model Binding

The process of mapping incoming request data to matching C# objects

Model Binding Posted Form Data



Applying Model Binding

AddProduct.cshtml.cs

```
[BindProperty]
public Product Product { get; set; }

public void OnPost()
{
    // Use Product property here
}
```

AddProduct.cshtml.cs

```
public void OnPost(Product product)
{
    // Use product parameter here
}
```

Tag Helpers can streamline the model binding process.

Form Tag Helpers

Form

Input

Select

Textarea

Label

Validation and more!

```
<form asp-page="AddItem">


<label asp-for="Item.Name">Name</label>
<input asp-for="Item.Name" />


<label asp-for="Item.Type">Price</label>
<select asp-for="Item.Type">
  <option value="Coffee">Coffee</option>
  <option value="Food">Food</option>
</select>


<label asp-for="Item.Desc">Desc</label>
<textarea asp-for="Item.Desc" />


<button type="submit">Submit</button>

</form>
```

◀ Form tag helper

◀ Label and input tag helper

◀ Select tag helper (drop downs)

◀ Textarea tag helper

Other Model Binding Features

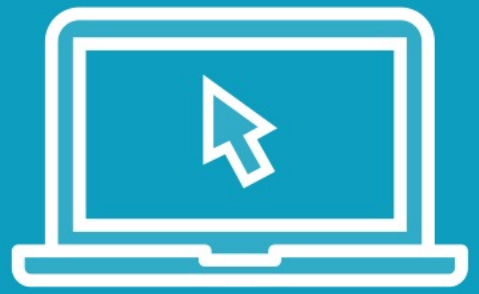
Bind URL data, JSON, file uploads and other sources

Data validation

Configurable conventions

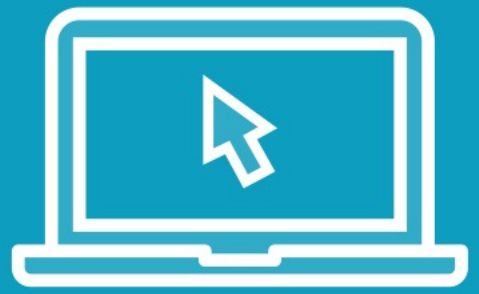
Custom components

Demo



Building the add product form using tag helpers

Demo



Handling the form submission

Understanding Server-side Form Validation

Model Validation

Enforces the formatting and structure of incoming data during Model Binding

Model Validation Benefits

Improves security

Reduces bugs

Helps the user

Simplifies code

Client and Server Validation

Client-side validation

Runs in the browser before the user submits a form

Server-side validation

Runs on the server after the form data is received

```
[Required]
public string Name { get; set; }


[MinLength(30)]
[MaxLength(500)]
public string Description { get; set; }


[EmailAddress]
public string Email { get; set; }


[Phone]
public string Phone { get; set; }
```

- Must not be empty
- Must be at least 30 and less than 500 characters
- Must be formatted as an email address
- Must be formatted as a phone number

```
[BindProperty]
public Product NewProduct { get; set; }

public void OnPost()
{
    if (ModelState.IsValid)
    {
        dbContext.Add(NewProduct);
    }

    // If there are validation errors,
    reload the page and display them
}
```

- Bound property – validation occurs during the binding process
- Check if there are validation errors
- If the data is valid, process it
- If the data is not valid, reload the page and display validation messages

Model and Property Level Validation

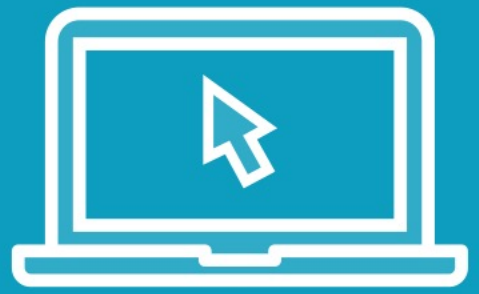
```
<!-- Show all validation errors -->
<div asp-validation-summary="All"></div>

<!- Show the validation errors for this property -->
<p asp-validation-for="NewProduct.Name"></p>
```

Validation Tag Helpers

Tag helpers can show individual validation messages or a summarized list

Only the asp-validation-summary tag helper can show model-level errors

Demo



Adding server-side form validation

Working with Action Results

Action Results

Enable page handler methods to return different response types

```
[BindProperty]
public Product NewProduct { get; set; }

public IActionResult OnPost()
{
    // Save product to database

    return Page();
}
```

Using Action Results

Specify IActionResult as the return type of a page model handler method

Return an action result using one of the action result helper methods

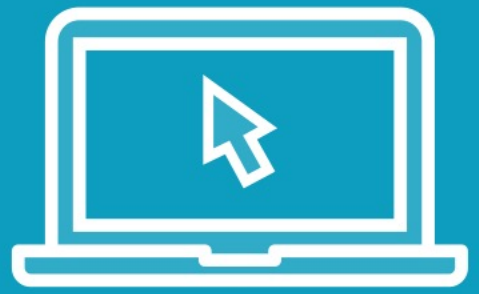
Available Action Result Helpers

Action result helper method	Purpose
return Page()	Reloads the page (default behavior)
return RedirectToPage("Index")	Redirects to the specified page
return RedirectToRoute()	Redirects to a specified route
return Forbid()	Returns a 403 forbidden
return File()	Returns a downloadable file

```
[BindProperty]
public Product NewProduct { get; set; }

public IActionResult OnPost()
{
    if (ModelState.IsValid)
    {
        dbContext.Add(NewProduct);
        return RedirectToPage("All");
    }

     return Page();
}
```

- Form data populated through model binding
- Change handler return type to IActionResult
- Use action result to redirect on successful submissions
- Use action result to reload the page if there are validation issues

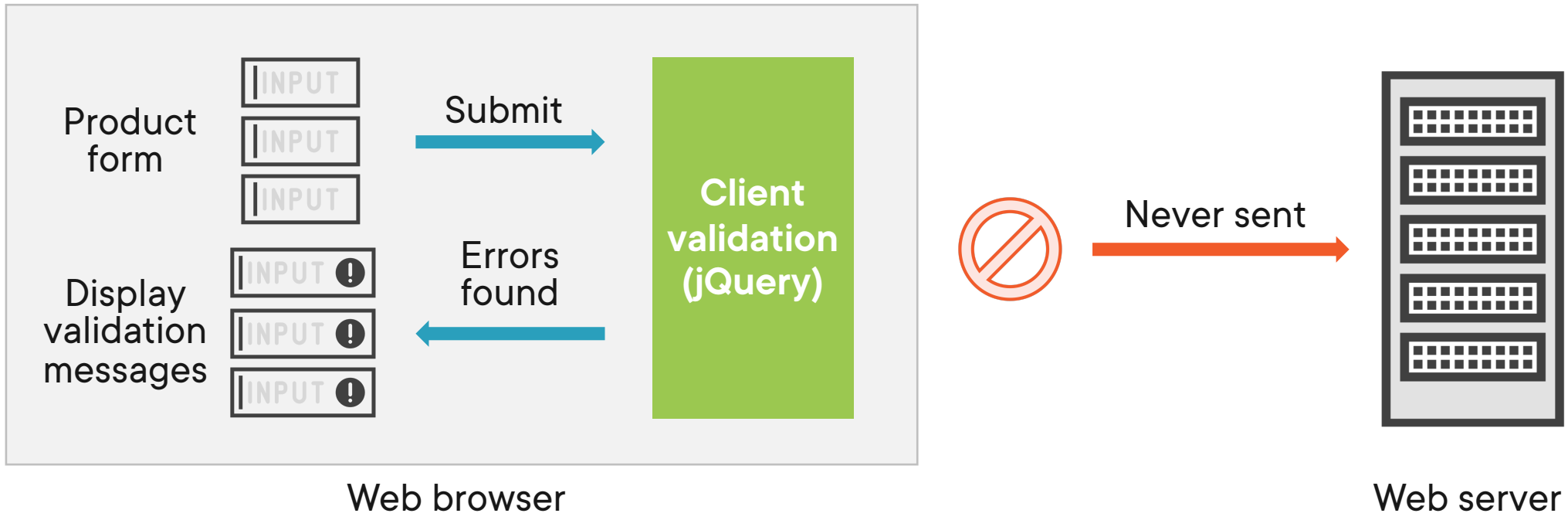
Demo



Improving form workflows with action results

Exploring Client-side Validation

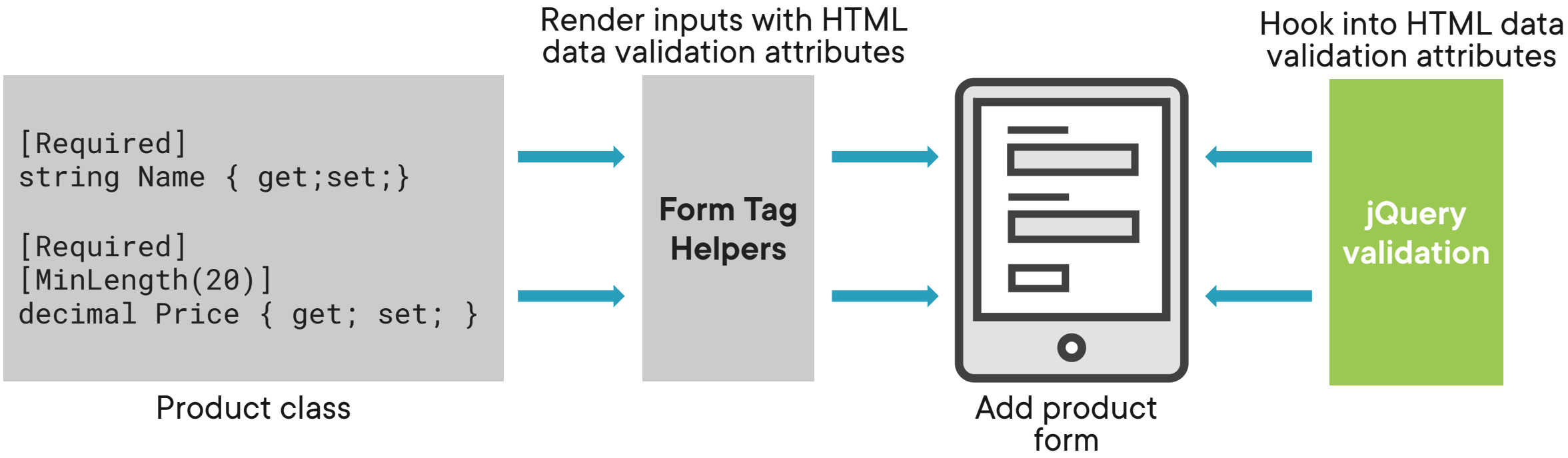
The Client-side Validation Workflow



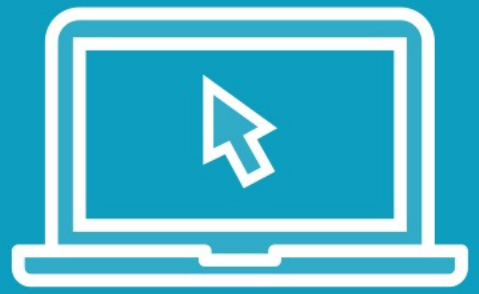
Client-side validation is primarily for user experience, not security

Server-side validation should always be implemented for security reasons

Adding Client-side Validation to Razor Pages



Demo



Adding client-side form validation

Summary

Razor Pages can process forms using handler methods such as OnPost

Form tag helpers streamline building forms that integrate with the model binding process

Model binding maps incoming request data to matching C# objects

Model validation occurs during the binding process to enforce rules about our data

Tag helpers can retrieve and display model validation messages to the user

Action results can help manage response types, such as redirects, status codes and more

Razor Pages can provide client-side validation using Tag Helpers and JavaScript integration