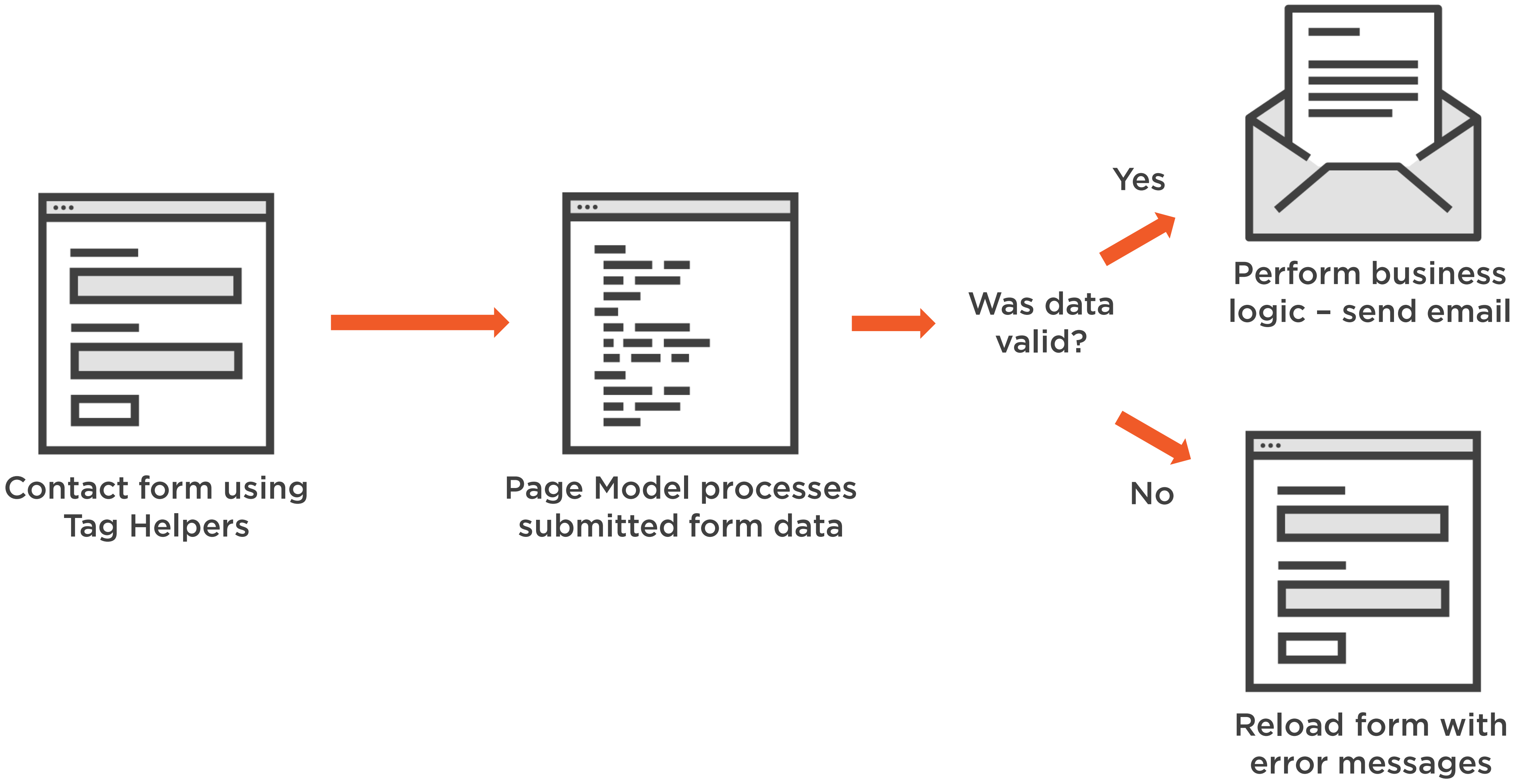
Working with Forms, Binding, and Validation

Alex Wolf

www.crywolfcode.com

Understanding Form Workflows

Handling Forms in Razor Pages


Contact form using Tag Helpers
Page Model processes submitted form data
Was data valid?
Yes
Perform business logic – send email
No
Reload form with error messages

```
public class ContactModel : PageModel
{
    public void OnGet()
    {
        // Page load logic
    }

    public void OnPost()
    {
        // Page submission logic
    }
}
```

- Run logic to load the page and display it to the user
- Run logic to process data submitted back to the Page Model

Page Model Request Handling Conventions

HTTP Request Type	Page Model Convention	Common Purpose
GET	OnGet()	Load page
POST	OnPost()	Create or process new data
PUT	OnPut()	Update existing data
DELETE	OnDelete()	Delete data

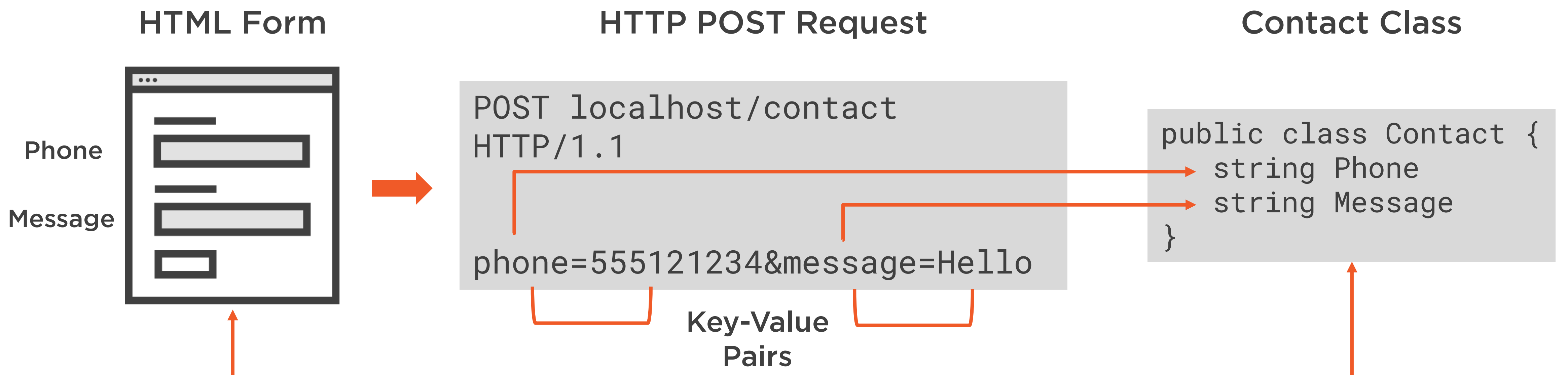
```
public class ContactModel : PageModel
{
    public void OnGet()
    {
        // Page load logic
    }

    public void OnPostSubscribe()
    {
        // Custom form logic
    }
}
```

◀ Run logic to load the page and display it to the user

◀ Custom handler that subscribes the user to a mailing list

Understanding Form Tag Helpers

Handling Forms in Razor Pages



```
POST localhost/contact
HTTP/1.1

phone=555121234&message=Hello
```

```
public class Contact {
  string Phone
  string Message
}
```

Tag Helpers help manage this naming relationship

Types of HTML Form Tag Helpers

Form	Label	Select
Input	Textarea	Option

Form Tag Helper Examples

Tag Helper

Rendered HTML

```
<input asp-for=“FirstName” />
```

→

```
<input type=“text” name=“FirstName” />
```

```
<select asp-for=“Rating”
asp-options=“RatingChoices” />
```

→

```
<select name=“Rating”>
        <option>Poor</option>
        <option>Fair</option>
        <option>Great</option>
</select>
```

HTML Helpers vs. Tag Helpers

HTML Helper

```
@Html.TextBoxFor(x => x.FirstName, new { @class = "gray", id = "Name" })
```

Interrupts standard HTML

Awkward Lambda Expression

Verbose attribute customization

Tag Helper

```
<input type=“text” asp-for=“FirstName” class=“gray” id=“Name” />
```

Demo

Demo: Building the Contact Form

Exploring Model Binding

Model Binding

A process that maps incoming request data to C# objects.

Handling Request Data

Sample HTTP Requests

Model Binding

Query params

```
GET localhost/search?orderby=desc
&type=active
HTTP/1.1
```

→ **Extract values from URL**

Key-value form data

```
POST localhost/contact
HTTP/1.1

phone=55121234&message=Hello
```

→ **Process form encoded values**

JSON & XML data

```
POST localhost/contact
HTTP/1.1

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

→ **Deserialize blocks of data**

Handling Forms in Razor Pages

HTML Contact Form

Name
Message

HTTP POST Request

```
POST localhost/contact
HTTP/1.1

name=John
&message=Hello
```

```
public class Contact {
    string Name
    string Message
}
```

Model Binding creates Contact instance

Contact Page Model

```
public class ContactModel
{
    [BindProperty]
    public Contact contact

    public void OnGet()
    {
    // Use Contact instance
    }
}
```

Model Binding Data Sources

URL	Route Data	HTTP Headers
Request Body	Files	Custom

```
public class ContactModel : PageModel
{
    public void OnGet(int id)
    {
        // Filter display by id
    }

    public void OnPost(Contact contact)
    {
        // Use bound contact object
    }
}
```

- Simple binding from URL
- Bind complex data from request body

```
public class Contact
{
    [BindRequired]
    public string Name { get; set; }
    [BindNever]
    public string Id { get; set; }
}
public class ContactModel : PageModel
{
    public void OnPost
    ([FromQuery]Contact contact)
}
```

◂ Value must be provided for incoming data to be valid

◂ Never bind this value, even if it’s provided

◂ Limit binding data source to just the URL

Demo

Demo: Handling a Submitted Form

Understanding Model Validation

Model Validation

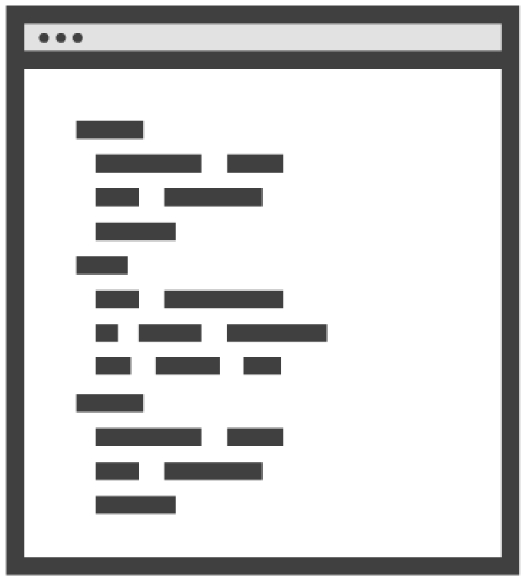
Enforces rules that determine what data can be bound to a model.

The Two Types of Model Validation

Client Side Validation

Improve user experience

Server Side Validation



Enforce security

```
public class Contact
{
    [Required]
    public string Name { get; set; }
    [Email]
    public string Email { get; set; }
    [Phone]
    public string Phone { get; set; }
    [MinLength(25)]
    public string Message { get; set; }
}
```

- Value cannot be null
- String must be formatted like an email
- String must be formatted like a phone number
- Message string must be at least 25 characters long

```
public class ContactModel : PageModel
{
    public void OnPost(Contact contact)
    {
        if(ModelState.IsValid) {
        // Continue logic
        }
    }
}
```

◀ Check if any validation errors occurred during Model Binding

Property vs. Model Validation

Property Validation

Applies to only one data point on a form or object

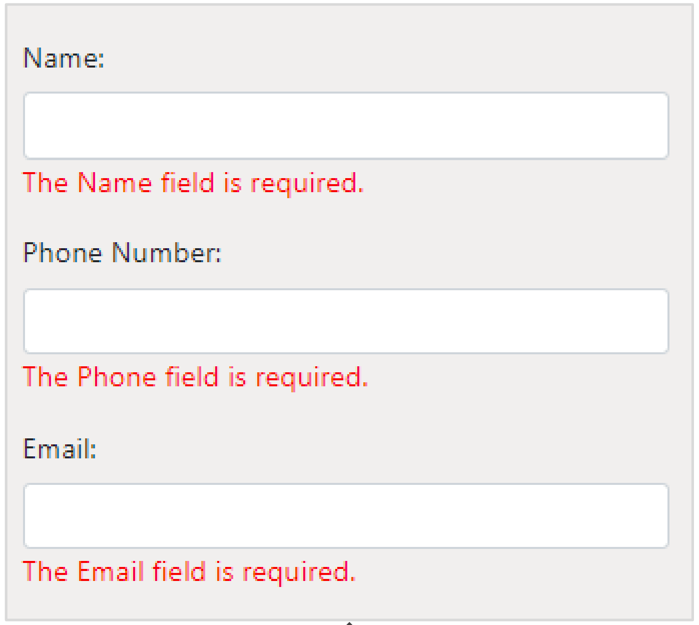
Model Validation

Applies to the overall state of the data, across properties

Form Tag Helper Examples

Property Only Validation Tag Helper

```
<span asp-validation-for="Name">
```



Name:

The Name field is required.

Phone Number:

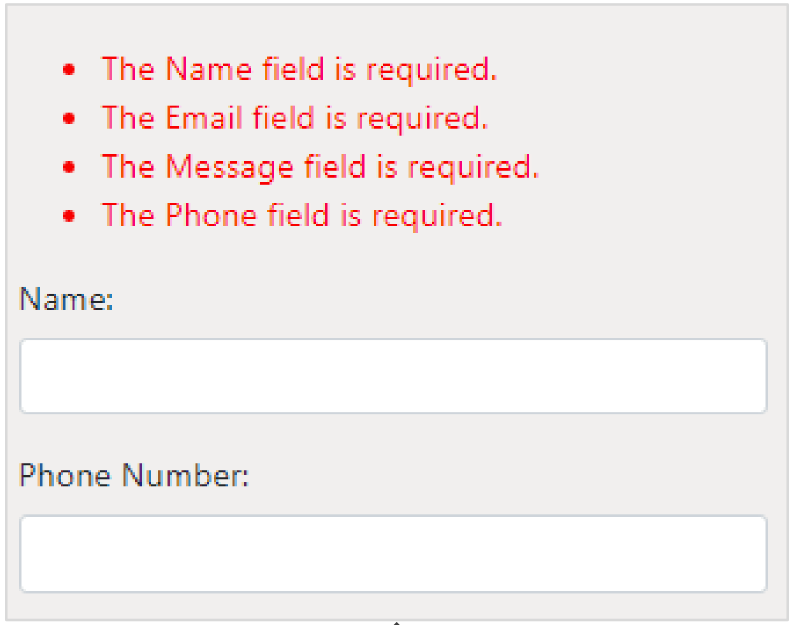
The Phone field is required.

Email:

The Email field is required.

Property or Model Validation Tag Helper

```
<div asp-validation-summary="All">
```



- The Name field is required.
- The Email field is required.
- The Message field is required.
- The Phone field is required.

Name:

Phone Number:

Model State

Client Side Validation with Tag Helpers

Initial Form Load

Tag Helper Output Before Errors

```
<span class="field-validation-valid" data-valmsg-
for="Contact.Name" data-valmsg-replace="true"></span>
```

```
<span class="field-validation-valid" data-valmsg-
for="Contact.Email" data-valmsg-replace="true"></span>
```

Demo

Demo: Improving Form Data with Validation

Demo

Demo: Working with Multiple Forms

Summary

Razor Pages provide a convention driven approach to handling form submissions

Form Tag Helpers make it easier to build forms that integrate with Razor Pages

Model Binding offers an easy way to map incoming request data to C# objects

Model Validation ensures the incoming data has proper values and formatting

Razor Pages provide support for multiple forms with assistance from Tag Helpers