9. Model Binding

Project: Add New Employee

- To implement this feature, which part of the project should be changed?
  - Model?
  - View?
  - Controller?
  - Repository?

82

Project: Update the Repository

```
public interface IEmployeeRepository
{
    //罗列所有员工信息
    List<Employee> SelectAll();
    //新增员工
    void Insert(Employee emp);
}
```

```
public class EmployeeRepository:IEmployeeRepository
{
      //新增员工
      public void Insert(Employee emp)
   {
       data.Add(emp);
   }
}
```

83

Project: Update the Controller（一）

- To add an employee, how many cycles of interaction would happen between the browser and the server?
  - 2 cycles
  - In the first cycle, the application provide a form to let the user input new employee info.
  - In the second cycle, the user submits the form, and the application add the employee to the collection.
- So, how many action methods do we need?

84

Project: Update the Controller（二）

```
public class EmployeeManagerController :
Controller
{
    ……
    //处理get请求
    public IActionResult Insert()
    {
        return View();
    }
    //处理post请求
    [HttpPost]
    public IActionResult Insert(Employee model)
    {
        repo.Insert(model);
        //重定向到List Action方法
        return RedirectToAction("List");
    }
}
```

85

Project: Implement the Insert View

```
@model Employee

<h2>Insert New Employee</h2>

<form asp-controller="EmployeeManager" asp-action="Insert"
method="post">
    <table>
        <tr>
            <td><label>ID:</label></td>
            <td><input type="text" asp-for="EmployeeId" /></td>
        </tr>
        <tr>
            <td><label>Name:</label></td>
            <td><input type="text" asp-for="EmployeeName" /></td>
        </tr>
        <tr></tr>
        <tr></tr>
    </table>
    <button type="submit">Save</button>
</form>
```

86

Project: Update the List View

```
……
<form>
    <button asp-action="Insert">Insert</button>
</form>
```

Employee Info List

Employee ID	Employee Name	Employee Title	Employee Country
1	Adam	manager	USA
2	Bob	president	UK
3	Cathy	sale	USA

Insert

87

Project: Run the Application

Employee Info List

Employee ID	Employee Name	Employee Title	Employee Country
1	Adam	manager	USA
2	Bob	president	UK
3	Cathy	sale	USA

Insert

Insert New Employee

ID: 4
Name: David
Title: Accountant
Country: CN
Save

Employee Info List

Employee ID	Employee Name	Employee Title	Employee Country
1	Adam	manager	USA
2	Bob	president	UK
3	Cathy	sale	USA
4	David	Accountant	CN

Insert

88

10. Tag Helpers

Building Forms with Tag Helpers (一)

- Displaying dynamic data is an important aspect of many web applications as well as needing to display data to the user.
- Model binding is how you accept the data sent by a user in a request and convert it to C# objects.
- The final aspect is how to build the HTML forms that users use to send this data in a request.
- ASP.NET Core provides a feature to achieve this, called Tag Helpers.

90

Building Forms with Tag Helpers (二)

- Tag Helpers are additions to Razor syntax that you use to customize the HTML generated in your templates.
  - Tag Helpers can be added to an otherwise-standard HTML element, such as an <input>, to customize its attributes based on your C# model.
  - Tag Helpers can also be standalone elements and can be used to generate completely customized HTML.

91

Enabling the Built-in Tag Helpers

- The built-in tag helpers are all defined in the Microsoft.AspNetCore.Mvc.TagHelpers namespace.
- Add an @addTagHelpers directive to enable it.

```
@using WebApp.Models
@addTagHelper *, Microsoft.AspNetCore.Mvc.TagHelpers
@using WebApp.Components
@addTagHelper *, WebApp
```

92

Creating forms using Tag Helpers

- The FormTagHelper transforms form elements so they target an action method.

```
<form asp-controller="EmployeeManager"
      asp-action ="Insert"
      method="post">
```

⇩

```
<form action="/EmployeeManager/Insert" method="post">
```

93

The Label Tag Helper

- The Label Tag Helper class is used to generate the caption (the visible text) and the for attribute for a <label> element, based on the properties in the Model Class.

```
public class UserModel
{
    [Display(Name = "Your name")]
    public string FirstName { get; set; }
    public string Email { get; set; }
}
```

⇔

```
<label asp-for="FirstName"></label>
<label asp-for="Email"></label>
```

⇩

```
<label for="FirstName">Your name</label>
<label for="Email">Email</label>
```

94

The Input and Textarea Tag Helpers (一)

- There's such a wide array of possible input types.
- The Input Tag Helper uses information based on both the type of the property (bool, string, int, and so on) and any DataAnnotations attributes applied to it to determine the type of the input element to generate.

95

The Input and Textarea Tag Helpers (二)

- The value of the type attribute is determined by the type of the view model property specified by the asp-for attribute.

C# Type	input Element type Attribute
byte, sbyte, int, uint, short, ushort, long, ulong	number
float, double, decimal	text, with additional attributes for model validation
bool	checkbox
string	text
DateTime	datetime

96

The Input and Textarea Tag Helpers (三)

- A more elegant—and reliable approach—is to apply one of the attributes to the property in the C# model class.

Attribute	input Element type Attribute
[HiddenInput]	hidden
[Text]	text
[Phone]	tel
[Url]	url
[EmailAddress]	email
[DataType(DataType.Password)]	password
[DataType(DataType.Time)]	time
[DataType(DataType.Date)]	date

97

The Input and Textarea Tag Helpers (四)

- Consider the Email property that was decorated with the [EmailAddress] attribute.

```
<input asp-for="Input.Email" />
```

⇩

```
<input type="email" id="Input_Email" name="Input.Email"
value="test@example.com" data-val="true"
data-val-email="The Email Address field is not a valid e-mail address."
Data-val-required="The Email Address field is required."
/>
```

98

Formatting input Element Values

- The asp-format attribute is used to specify how that data value is formatted.

```
<input asp-for="Dec" asp-format="{0:0.000}" />
```

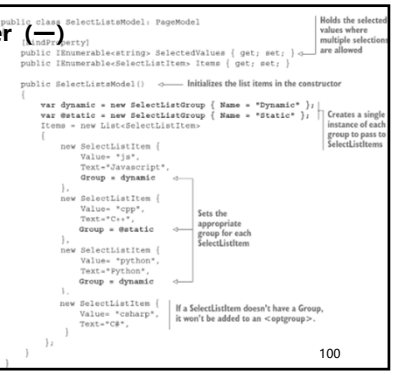
⇩

```
<input type="text" id="Dec" name="Dec" value="1.200">
```

99

The Select Tag Helper (一)

- Use a <select> element to display a drop-down list or a list box.
- The SelectListItem object defines a Group property that specifies the SelectList-Group the item belongs to.



100

The Select Tag Helper (二)

- The Select Tag Helper exposes the asp-for and asp-items attributes that you can add to <select> elements.

```
@model SelectListsModel
<select asp-for="Input.SelectedValue1"
    asp-items="Model.Items"></select>
<select asp-for="Input.SelectedValue2"
    asp-items="Model.Items" size="4"></select>
<select asp-for="Input.MultiValues"
    asp-items="Model.Items"></select>
```

Creates a standard drop-down select list by binding to a standard property in asp-for

Creates a single-select list box of height 4 by providing the standard HTML size attribute

Creates a multiselect list box by binding to an IEnumerable property in asp-for

101

The Select Tag Helper (三)

- The rendered HTML

```
<select id="SelectedValues" name="SelectedValues"
multiple="multiple">
  <optgroup label="Dynamic">
    <option value="js">JavaScript</option>
    <option value="python">Python</option>
  </optgroup>
  <optgroup label="Static">
    <option value="cpp">C++</option>
  </optgroup>
  <option value="csharp">C#</option>
</select>
```

102

The Validation Message Tag Helpers (一)

- The Input Tag Helper generates the necessary data-val-* validation attributes on input elements themselves.
- Use asp-validation-for attribute to display the validation messages.

```
<span asp-validation-for="Email"></span>
```

Email

The Email field is required.

103

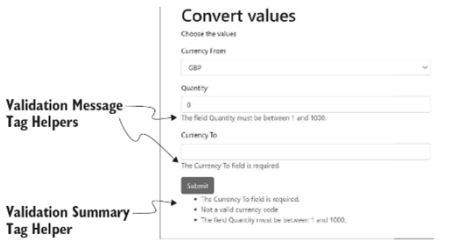
The Validation Message Tag Helpers (二)

- The Validation Summary Tag Helper is applied to a using the asp-validationsummary attribute and providing a ValidationSummary enum value.
  - None—Don't display a summary.
  - ModelOnly—Display only errors that are not associated with a property.
  - All—Display errors associated with either a property or the model.

104

The Validation Message Tag Helpers (三)

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



105

The Validation Message Tag Helpers (四)

```
[HttpPost]
public IActionResult OnPost()
{
    if(Input.CurrencyFrom == Input.CurrencyTo)
    {
        ModelState.AddModelError(
            string.Empty,
            "Cannot convert currency to itself");
    }
    if (!ModelState.IsValid)
    {
        return Page();
    }

    //store the valid values somewhere etc
    return RedirectToPage("Checkout");
}
```

Can't convert currency to itself

Adds model-level error, not tied to a specific property, by using empty key

If there are any property-level or model-level errors, displays them

106

Generating links with the Anchor Tag Helper

- The Anchor Tag Helper class is used to transform the href attributes of a elements.
- Values for segment variables are defined using asp-route-[name] attributes.

```
<a asp-action="index" asp-controller="home"
asp-route-id="@Model?.ProductId">Select</a>
```

⇩

```
<a href="/Home/index/3">Select</a>
```

107

Cache-busting with the Append Version Tag Helper (一)

- For performance reasons, browsers often cache files locally and reuse them for subsequent requests rather than calling the application every time a file is requested.
- But what happens if it does change? You want to make sure users get the updated assets as soon as they're available.
- One of the most common ways for handling it is to use a cache-busting query string.

108

Cache-busting with the Append Version Tag Helper (二)

- A cache-busting query string adds a query parameter to a URL, such as ?v=1.
  - Browsers will cache the response and use it for subsequent requests to the URL.
- When the resource changes, the query string is also changed, such as to ?v=2.
  - Browsers will see this as a request for a new resource and make a fresh request.

109

Cache-busting with the Append Version Tag Helper (三)

- The biggest problem with this approach is that it requires you to update a URL every time the resource changes.
- The asp-append-version attribute will load the file being referenced and generate a unique hash based on its contents. This is then appended as a unique query string to the resource URL.

```
<img src="/images/city.png" asp-append-version="true"/>
```

⇩

```
<img src="/images/city.png?v=KaMNDSZFAJufRcRDpKh0K_IIPNc7E">
```

110

Project: Implement Employee Updation

- What's the difference between inserting an employee and updating an existed employee?

111

Project: Modify the Repository Code

```
public interface
IEmployeeRepository
{
    //根据id查找员工
    Employee SelectbyId(int id);
    //更新员工
    void Update(Employee emp);
}
```

```
public Employee SelectbyId(int id)
{
    Employee e = data.Find(e=>e.EmployeeId.Equals(id));
    return e;
}
public void Update(Employee emp)
{
    Employee e = data.Find(e => e.EmployeeId.Equals(emp.EmployeeId));
    e.EmployeeName = emp.EmployeeName;
    e.Title = emp.Title;
    e.Country = emp.Country;
}
```

112

Project: Modify the Controller Code

```
//处理get请求，将待更新的员工信息提供给视图
public IActionResult update(int id)
{
    return View(repo.SelectbyId(id));
}

//处理post请求，更新员工信息
[HttpPost]
public IActionResult update(Employee model)
{
    repo.Update(model);
    return RedirectToAction("List");
}
```

113

Project: Modify the List View

```
<td>
    <form>
        <input type="hidden" name="id"
value="@e.EmployeeId" />
        <button asp-action="Update" >Update</button>
    </form>
</td>
```

Employee Info List

Employee ID	Employee Name	Employee Title	Employee Country	
1	Adam	manager	USA	Update
2	Bob	president	UK	Update
3	Cathy	sale	USA	Update

Insert

114

Project: Create the Update View

- The update view is similar to insert view

```
<tr>
    <td><label>ID:</label></td>
    <td><input type="text" asp-for="EmployeeId" readonly/></td>
</tr>
```

Update Employee Info

ID: 1
Name: Adam
Title: manager
Country: USA
Save

115

Project: Run the Application

Employee Info List

Employee ID	Employee Name	Employee Title	Employee Country	
1	Adam	manager	USA	Update
2	Bob	president	UK	Update
3	Cathy	sale	USA	Update

Insert

Update Employee Info

ID: 1
Name: Ad
Title: manager
Country: USA
Save

Employee Info List

Employee ID	Employee Name	Employee Title	Employee Country	
1	Ad	manager	USA	Update
2	Bob	president	UK	Update
3	Cathy	sale	USA	Update

Insert

116

Project: Implement Employee Deletion

- What's the similarity and difference between deletion and updation?

117

Project: Modify the Repository Code

```
public interface IEmployeeRepository
{
    //删除员工
    void Delete(int id);
}
```

```
public void Delete(int id)
{
    Employee e = this.data.Find(e => e.EmployeeId.Equals(id));
    this.data.Remove(e);
}
```

118

Project: Modify the Controller Code

```
public IActionResult delete(int id)
{
    repo.Delete(id);
    return RedirectToAction("List");
}
```

- Does the deletion need a view?

119

Project: Modify the List View

```
<form>
    <input type="hidden" name="id" value="@e.EmployeeId" />
    <button asp-action="Update">Update</button>
    <button asp-action="Delete">Delete</button>
</form>
```

Employee Info List

Employee ID	Employee Name	Employee Title	Employee Country	
1	Adam	manager	USA	Update Delete
2	Bob	president	UK	Update Delete
3	Cathy	sale	USA	Update Delete

Insert

120

Project: Run the Application

Employee Info List

Employee ID	Employee Name	Employee Title	Employee Country	
1	Adam	manager	USA	Update Delete
2	Bob	president	UK	Update Delete
3	Cathy	sale	USA	Update Delete

Insert

Employee Info List

Employee ID	Employee Name	Employee Title	Employee Country	
2	Bob	president	UK	Update Delete
3	Cathy	sale	USA	Update Delete

Insert

121

THANK YOU