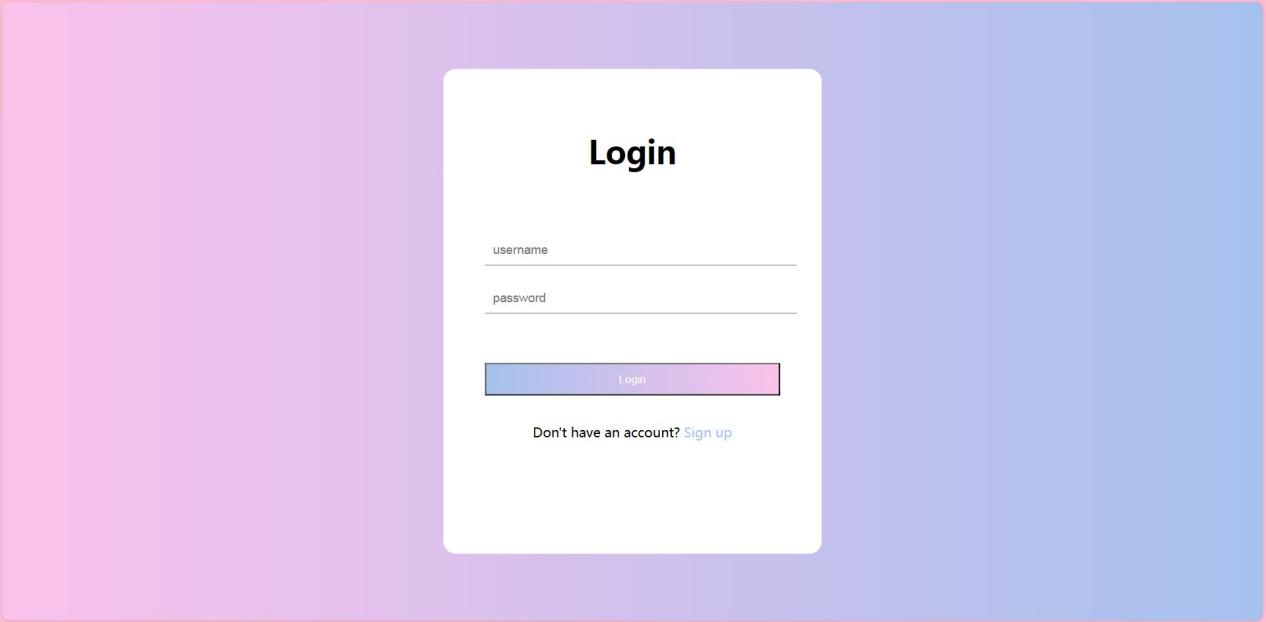
**Name:Yantao Zeng**

**ID:22009333**

**Video link:**

**<https://www.bilibili.com/video/BV1ub4y1g7DV/?spm_id_from=333.999.list.card_archive.click&vd_source=1ed41f2136a1e0a6150048db5b30edf9>**

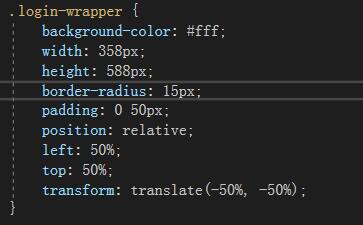
**Login**



This is my login page, you can enter any user name and password, and then click "login" to login.



This is the main code for my Login page, which is essentially just clicking Login, and then login is a hyperlink to go to the home page.



White background, width 358 pixels, height 588 pixels, rounded corner 15 pixels.

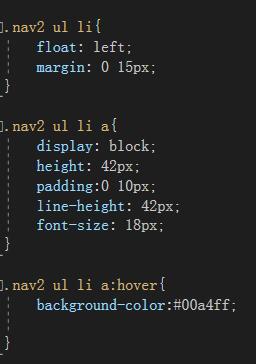
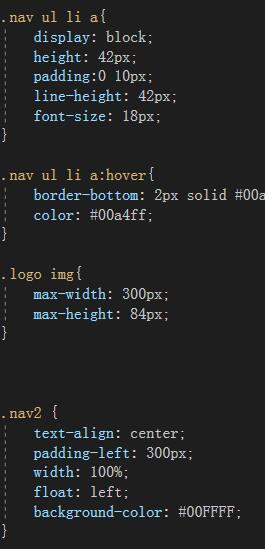
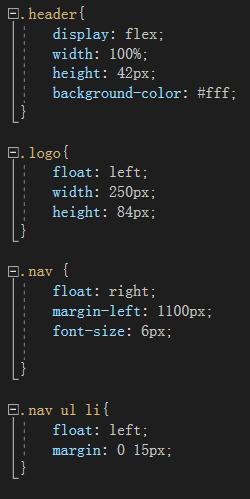
Located in the center of the page, via left: 50%; top: 50%; transform: translate(-50%, -50%); Realized.

**Navigation bar**





This is my navigation bar page.



Overall width Settings (.w) :

The width is 1000 pixels.

Center align horizontally.

Header container style (.header) :

Use Flex layout to arrange header containers horizontally. The width is 100%. The height is 42 pixels. The background color is white.

Logo style (.logo) :

Float to the left. The width is 250 pixels. The height is 84 pixels.

Navigation bar style (.nav) :

Float to the right. The left margin is 1100 pixels. The font size is 6 pixels.

Navigation bar link style (.nav ul li a) :

Displays as block-level elements. The height is 42 pixels. The left and right inner margins are 10 pixels. The row height is 42 pixels. The font size is 18 pixels.

Link style when mouse-over (.nav ul li a:hover) :

The bottom border is a 2 pixel wide solid #00a4ff (blue) line. The text color is #00a4ff (blue).

Logo Image Style (.logo img) :

The maximum width is 300 pixels. The maximum height is 84 pixels.

Navigation bar Style 2 (.nav2) :

Center the text. Float to the left. The left inner margin is 300 pixels. The width is 100%. The background color is #00FFFF (Cyan).

Navigation bar menu Item Style 2 (.nav2 ul li) :

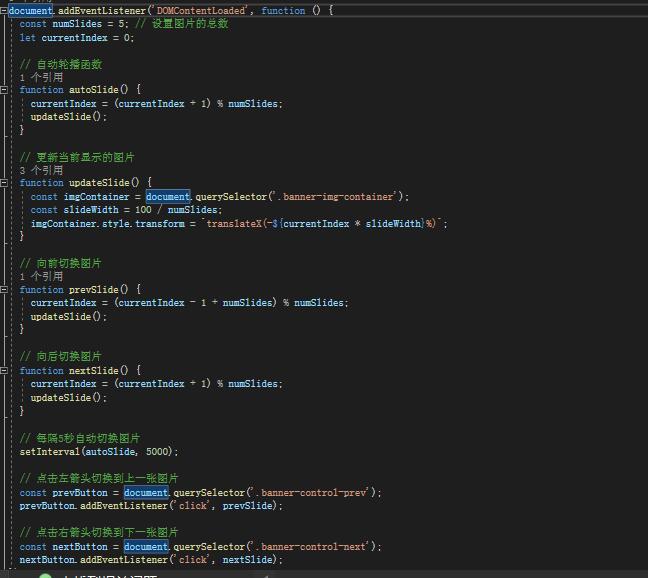
Float to the left. The left and right margins are 15 pixels.

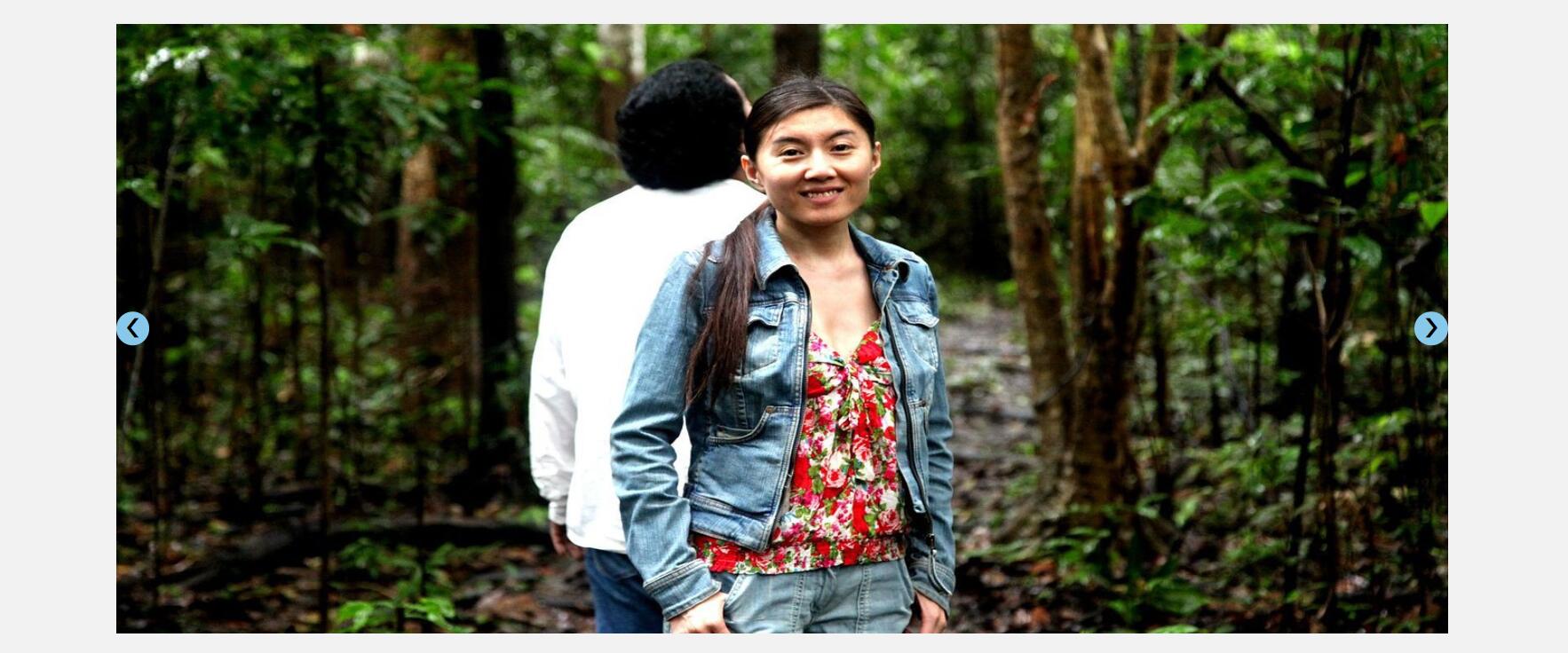
Navigation bar Link Style 2 (.nav2 ul li a) :

Displays as block-level elements. The height is 42 pixels. The left and right inner margins are 10 pixels. The row height is 42 pixels.

The font size is 18 pixels.

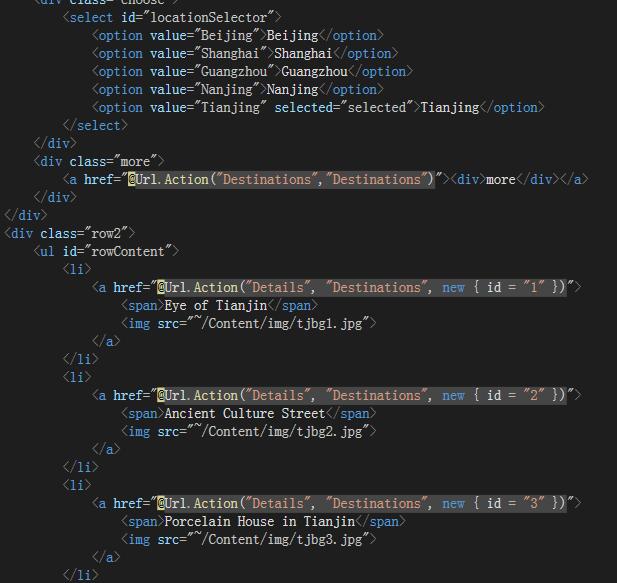
Link style 2 (.nav2 ul li a:hover) when hovering: the background color is #00a4ff (blue).

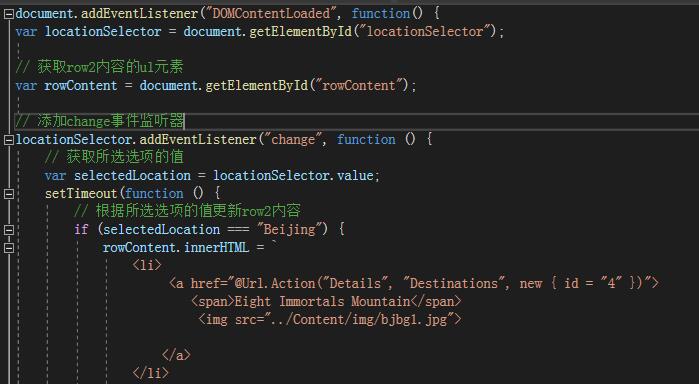
**Picture carousel**



Mainly written in js files.See the caption in the picture.

**Drop down selector**

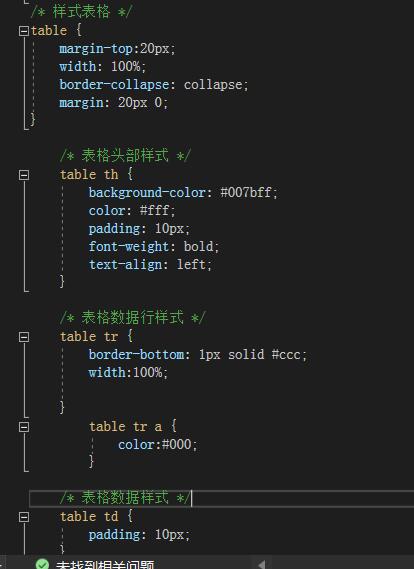




The change of the drop-down selector mainly listens to the locationSelector in my cshtml file, and then replaces the contents of my row2 with the contents of the js file, where the contents of the image in the js file should be implemented with the absolute path.

**Price page**







Overall Settings for the <table> element:

The top margin is 20 pixels, the width is 100%, and the border is merged when overlapping.

Table header style:

The background color of the header is #007bff, the text color is white, the inner margin is 10 pixels, and the text is left justified.

Table row style:

The bottom border is a solid line #ccc 1 pixel wide with a width of 100%.

Table row hover style:

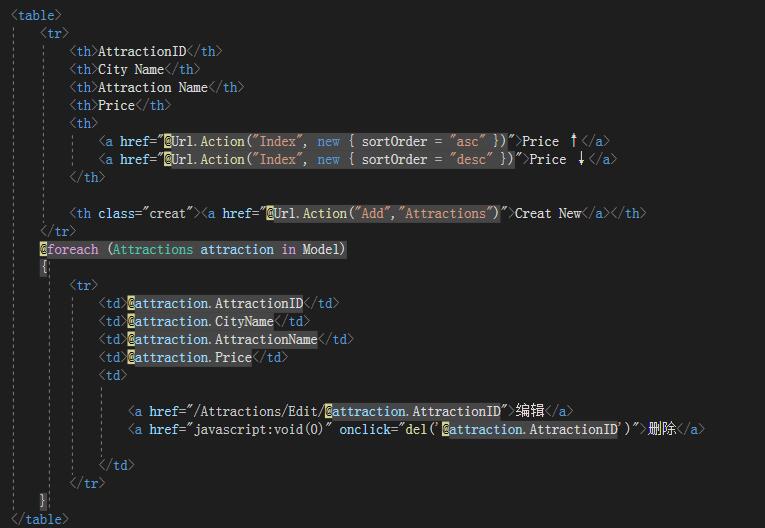
When you hover, the background color changes to #f0f0f0,

Edit and delete link styles:

The link text color is #007bff, and the text is not underlined

Link hover style:

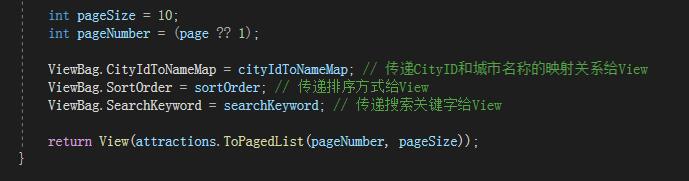
The underline is displayed when the mouse is hovering.

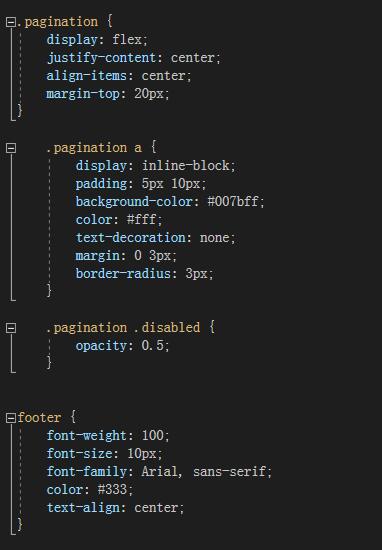


The Ticket page reads AttractionID, CityName, AttractionName, and Price from the database.

**Page breaks and footers**







Paging style (.pagination) :

Use Flex layout to center page elements horizontally and vertically. The upper margin is 20 pixels.

Pagination style (.pagination a) :

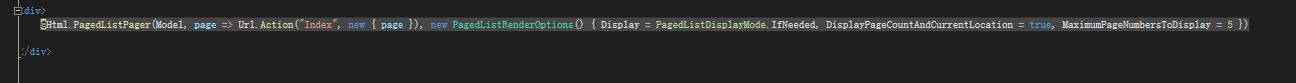
Displays paginated links as inline block elements so that they are arranged horizontally. The inner margin is 5 pixels above and below, 10 pixels around. The background color is #007bff (dark blue). The text color is white. Text is not underlined. The left and right margins are 3 pixels. The border radius is 3 pixels to achieve rounded corners.

Disable status style (.pagination.disabled) :

When a paged link is not available, reduce its transparency to 0.5.

footer style:

font weight: 100. The font size is 10 pixels. Use Arial fonts or, if not available, sans-serif. The text color is #333 (dark gray). Center the text.



The paging function has ten data per page, which is on the first page by default.@Html.PagedListPager(Model, page =&gt; Url.Action("Index", new { page }), new PagedListRenderOptions() { ... }) : This is a block of code that uses the @Html.PagedListPager helper method to generate pagers.

Model is the paging data model passed to the view.

page =&gt; Url.action ("Index", new {page}) is a lambda expression that generates the URL for each paginated link. The url.action method is used here to generate the URL, where "Index" is the Action method in the controller, and new {page} is used to pass the current page number as the route argument.

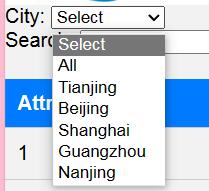
new PagedListRenderOptions() { ... A PagedListRenderOptions object is created to configure the appearance and behavior of the pager.

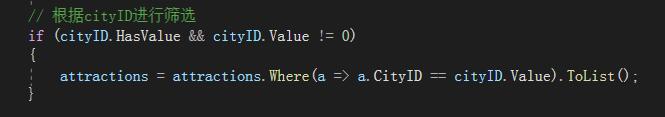
Display = PagedListDisplayMode. IfNeeded set the Display mode to "if needed", namely only shows how page in the pager.

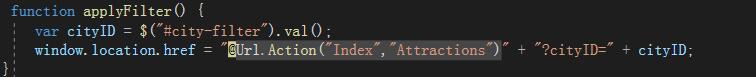
DisplayPageCountAndCurrentLocation = true enabled according to the current page and the function of the total number of pages.

MaximumPageNumbersToDisplay = 5 set the maximum number page Numbers to show is 5, more than the number of page Numbers will be omitted.

**Filter selector**

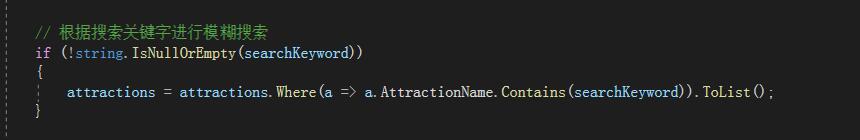






**Search function**

26

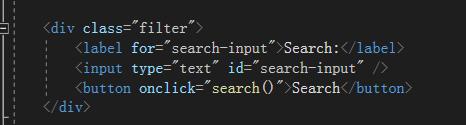


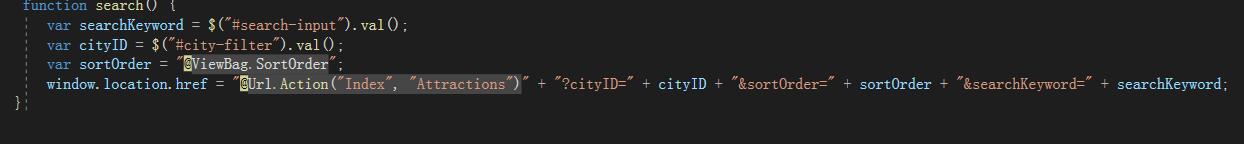
<div> : This is a <div> tag with a style class named "filter" added to it by using the class attribute. It is used to organize and contain other elements of the search function.

&lt; label for="search-input"&gt; Search:&lt; /label&gt; : This is a... label&gt; Tag, which is associated with the input box below by using the for attribute. It displays a text label that identifies the search input field.

&lt; input type="text" id="search-input" /&gt; : This is an input field element that is set to "text" by using the type attribute to indicate that the user can enter text into it. It is also set to "search-input" by using the id attribute, as opposed to the &lt above; label&gt; Label for association.

&lt; button onclick="search()"&gt; Search&lt; /button&gt; : This is a button element that executes a JavaScript function called "search()" when the user clicks on it. When a button is clicked, some sort of search action is usually triggered. The text displayed on the button is "Search", which is used as the label for the button.





var searchKeyword = $("#search-input").val(); This line of code uses the jQuery selector $("#search-input") to get the value of the input field element with the id "search-input" and assigns it to the variable searchKeyword. This is to get the keywords entered by the user in the search input box.

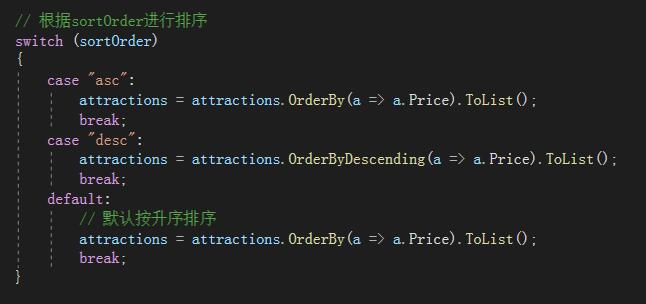
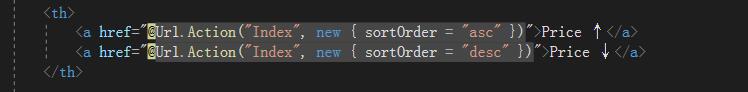
var cityID = $("#city-filter").val(); Similar to the first line, this line of code takes the value of a drop-down box with the id "city-filter" (possibly a city selector) and assigns it to the variable cityID. This is to get the city ID selected by the user.

var sortOrder = "@ViewBag.SortOrder"; Using Razor syntax (commonly used in ASP.NET MVC), @ViewBag.SortOrder is a variable passed by the back-end server to the front-end. It is used to get the sort order, which may be generated dynamically from the server side.

window.location.href = "@Url.Action("Index", "Attractions")" + "? cityID=" + cityID + "&sortOrder=" + sortOrder + "&searchKeyword=" + searchKeyword; This line of code builds a new URL and navigates the browser's current page to that URL with window.location.href. The Url is generated by calling url.action, and includes the controller ("Attractions") and Action ("Index") information, as well as the city ID, sorting order and search keywords collected from the previous as the query parameters passed.

**Sort function**

30



switch (sortOrder) : Multi-branch selection based on the value of the sortOrder variable.

case "asc": : If the value of sortOrder is "asc" (ascending), the following statement block is executed:

attractions.OrderBy(a =&gt; A. rice) : List attractions in ascending order of price.

.tolist () : Converts the sorted result to a list.

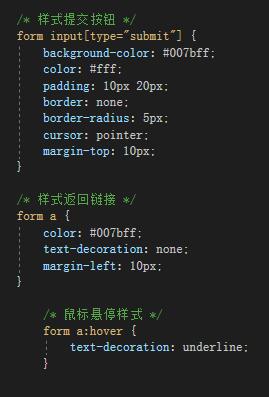
case "desc": : If the value of sortOrder is "desc" (descending), the following blocks are executed:

attractions.OrderByDescending(a =&gt; A. rice) : Attractions are listed in descending order of price.

.tolist () : Converts the sorted result to a list.

**Edit page**





form: This style defines the style of the entire form container, including margins, margins, borders, background colors, and border corners.

form div: This style defines the style of the elements inside the form (div elements) and sets the top and bottom margins between them.

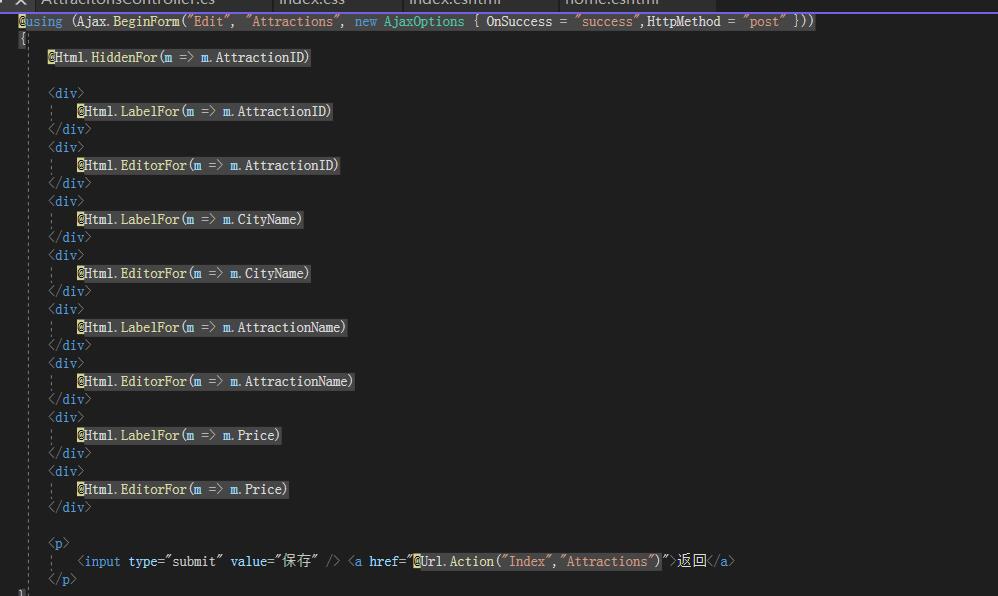
form label: This style defines the style of the form label, including boldness, color, and display as block-level elements.

form input[type="text"]: This style defines the style of the text input field, including width, margins, margins, borders, and border corners.

form input[type="submit"]: This style defines the style of the submit button, including the background color, text color, padding, border, border round, and mouse pointer shape.

form a: This style defines the style of the return link, including color, text decoration, and left margin.

form a:hover: This style defines the style of the return link when the mouse is hovering, including text decorated with underscores.



This is an ASP.NET MVC view that uses Ajax technology to edit information about Attractions. Here's an explanation of the code:

@using (Ajax.BeginForm("Edit", "Attractions", new AjaxOptions { OnSuccess = "success", HttpMethod = "post" })): This is a form start tag using Ajax. It specifies the target action for the form (the "Edit" action is in the Attractions controller) and defines some Ajax options, among which OnSuccess specifies "success" as the JavaScript function to be called on success. HttpMethod specifies that the form submission method is POST.

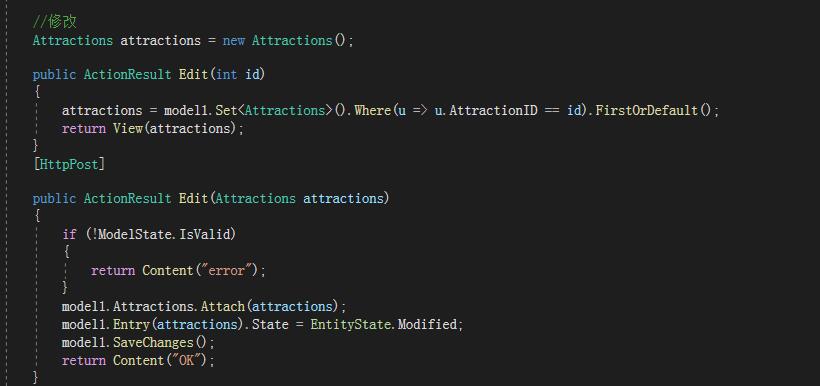
@Html.HiddenFor(m =&gt; m.AttractionID): This is a hidden field used to store the value of AttractionID in the form. It is sent to the server when the form is submitted.

@Html.LabelFor(m =&gt; m.AttractionID): This is an HTML tag used to display the AttractionID tag.

@Html.EditorFor(m =&gt; m.AttractionID): This is an HTML tag used to generate the AttractionID input field. Here the user can edit the value of AttractionID.

Similarly, for the other attributes (CityName, AttractionName, Price), @Html.LabelFor and @Html.EditorFor are used to generate corresponding labels and input fields.

<p>&lt; input type="submit" value=" Save "/&gt; <a href> Returns </a></p>: This is the form's submit button and a return link. Clicking the "Save" button triggers the submission of the form, while the "back" link points to the "Index" action of the Attractions controller.



First, a attractions object named Attractions is created. This could be an entity class used to represent a tourist attraction.

The Edit(int id) method is a GET method that displays the edit page. It receives an id parameter, through which it obtains the corresponding Attractions object from the database and assigns it to the attractions variable. The object is then passed to the view template and returned to the view page for the user to edit.

The [HttpPost] tag indicates that the following method is a POST method that handles the commit of the edit operation.

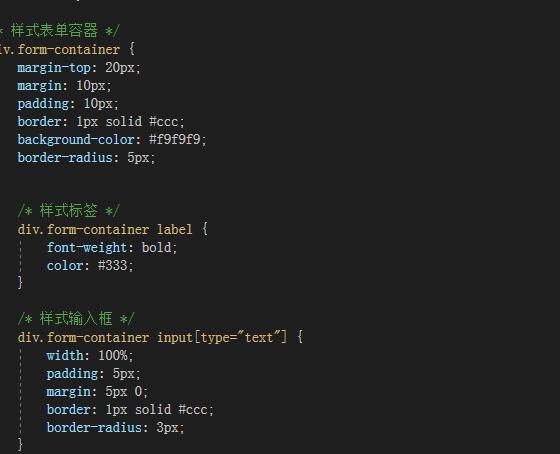
The Edit(attractions) method accepts a attractions object as an argument. Inside the method, first check that ModelState is valid, and if not, return a ContentResult with an error message.

If ModelState is valid, the attractions object is appended to the data context (model1.Attractions) and its state is set to modified. Next, call model1.SaveChanges() to save the changes to the database.

Finally, a ContentResult with an "OK" text is returned, indicating that the editing operation was successful.

**Add page**







Form container style (div.form-container) :

The upper margin is 20 pixels. The uniform margin is 10 pixels. The inside margin is 10 pixels.

The border is a solid #ccc (gray) line 1 pixel wide.

The background color is #f9f9f9. The border radius is 5 pixels to achieve rounded corners.

label style (div.form-container label) :

font weight: bold. The text color is #333 (dark gray).

input box style (div.form-container input[type="text"]) :

The width is 100%. The inside margin is 5 pixels. The upper and lower margins are 5 pixels. The border is a solid #ccc (gray) line 1 pixel wide. The border radius is 3 pixels to achieve rounded corners.

submit button style (div.form-container input[type="submit"]) :

The background color is #007bff (dark blue). The text color is white. The inner margin is 10 pixels above and below, 20 pixels around. Border None. The border radius is 5 pixels to achieve rounded corners.

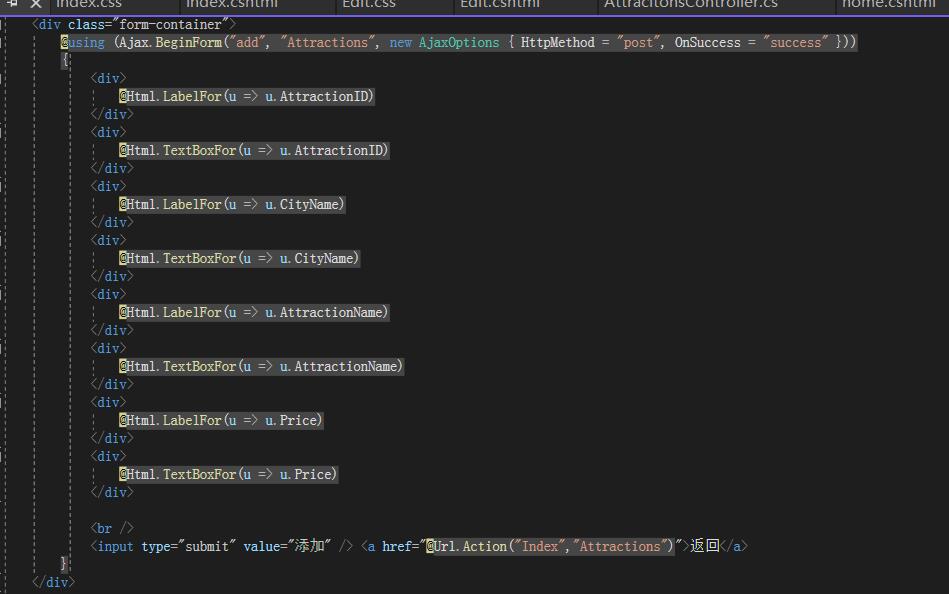
The cursor style is cursor: pointer.

Return the link style (div.form-container a) :

The text color is #007bff (dark blue). Text is not underlined. The left margin is 10 pixels.

Return the link hover style (div.form-container a:hover) :

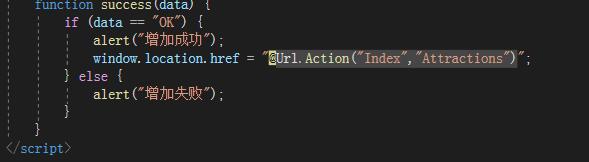
Underline effect when mouse hover.



This code creates a form container and uses the style of a style sheet container (class="form-container"). Inside the form container, the Ajax.beginform method with AJAX is used to create a POST request form, specifying the JavaScript function success to call upon successful submission.

Next, a series of <div> elements are generated as containers for form fields using ASP.NET's Razor syntax, and the @Html.LabelFor and @Html.TextBoxFor methods are used to generate corresponding tags and input fields that are automatically generated based on model properties.

Finally, we use... input type="submit"&gt; Create a submit button and return link with the <a> tag, and use the @Url.Action method to generate the URL to return the link.



function success(data) : Defines a JavaScript function called success that takes a single parameter, data.

Conditional judgment: Use if (data == "OK") to determine whether the incoming data is equal to the string "OK".

Success tip: If the condition is true, a pop-up dialog box says "Add success".

Page redirection: The page is redirected to the specified URL through window.location.href. @Url.Action("Index","Attractions") is used to generate the URL of the target page.

Failure message: If the condition is not true, a dialog box is displayed saying "Add failed".



This action method is used to display a form page and return a View. Here, assume that there is a view file called "Add.cshtml" that is used to render the form page.

This action method uses an HTTP POST request and accepts a parameter of type Attractions, which is used to receive data submitted by the form. Inside the method, first check the validation state of the model (ModelState.IsValid). If model validation fails, "error" is returned, indicating that the form data is incorrect.

If the model validates, add the form data to the database (assuming model1 is the database context), and save the changes. Finally, "OK" is returned, indicating that the form submission was successful.

These two action methods together realize the function of displaying and submitting a form page. When the form is submitted, the data is asynchronously sent to the server via AJAX, which processes it on the server side and returns the corresponding results, which are processed by the front-end through defined JavaScript functions.

**Delete function**

1



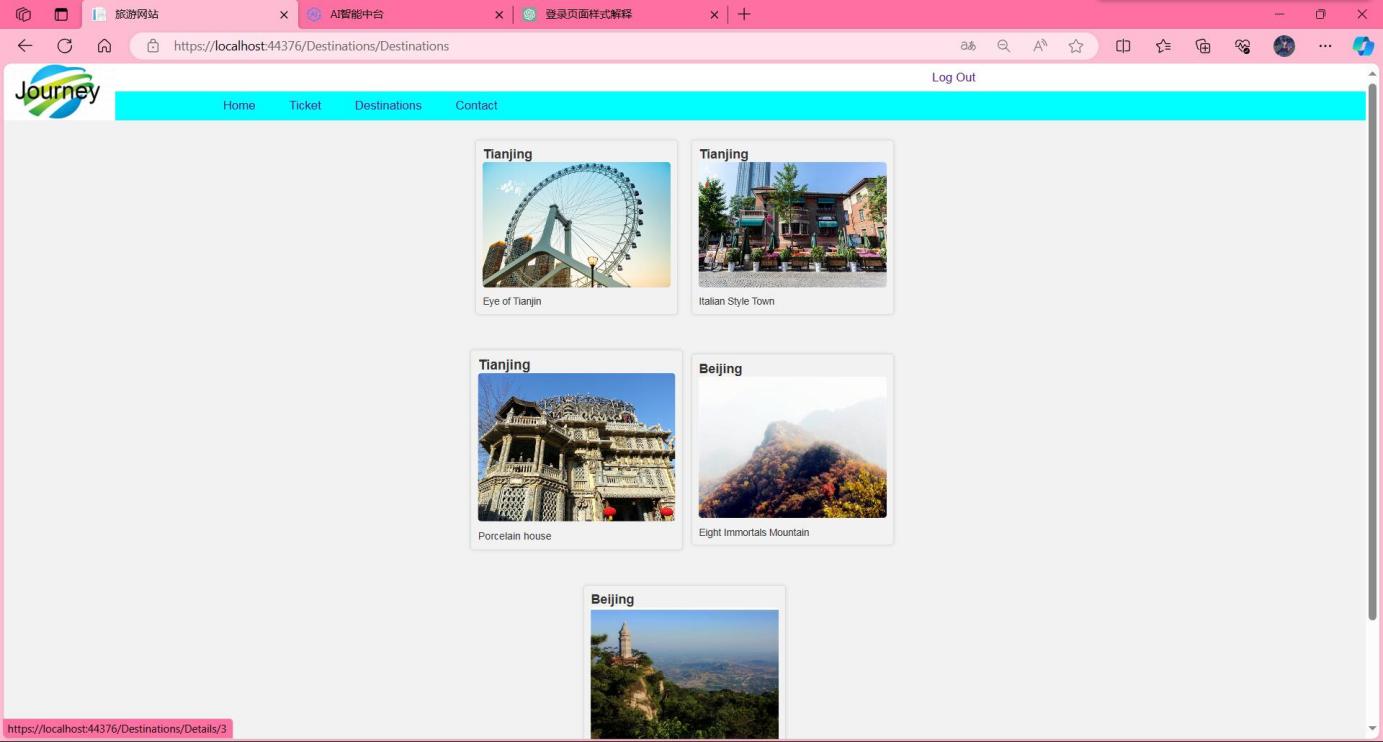
function del(id) : Defines a JavaScript function called del that takes a parameter id.

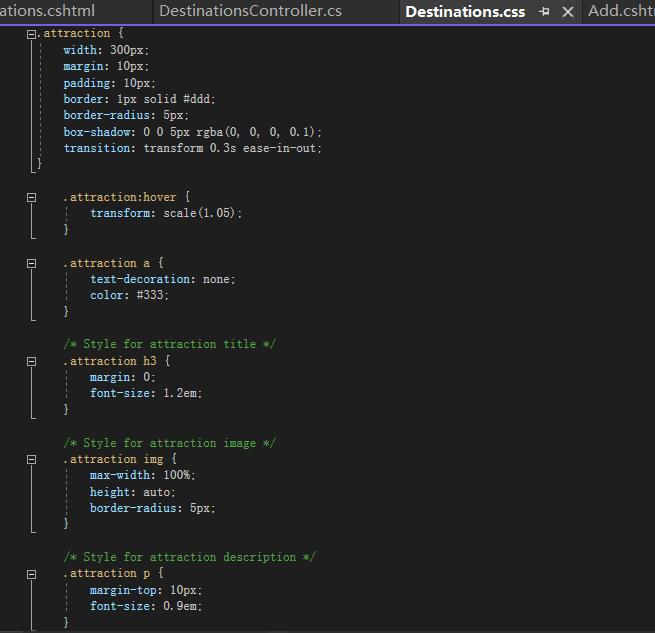
confirm dialog: Use the Confirm function to bring up a confirmation dialog, asking the user if they are sure to delete.

Send a delete request: If the user clicks the Confirm button, use jQuery's $.POST function to send a POST request to the specified delete action method. The requested address is generated by @Url.Action("Delete", "Attractions"), and the id parameter is passed in the URL by concatenating.

Handling the deletion result: When the request is returned, a callback function is executed, receiving a data parameter representing the returned data. If the returned data is equal to the string "ok", a prompt box will pop up indicating "successful deletion" and redirect to the "Index" action method of the "Attractions" controller. If the returned data is not equal to "ok", a pop-up box says "Delete failed".

**Destinations page**





width: Set the width of the spot block to 300px.

margin: Set the margin to 10px to control the spacing between the attraction blocks.

padding: Sets the padding to 10px to increase the distance between the content and the border.

border: Set the border to a 1px wide gray border.

Border-radius: Set the rounded corner of the border to 5px to make the corner of the attraction block more rounded.

box-shadow: Adds a slight shadow effect to enhance the depth of the block.

transition: Adds a 0.3 second transition effect to make scaling smoother.

When the mouse hovers over the attraction block, the attraction block is zoomed and transformed by the :hover pseudo-class selector so that it appears slightly enlarged (1.05 times) to provide visual feedback.

Set the link style within the attraction block, remove the underline of the link and set the color to dark gray.

Set the style of the title inside the attraction block, set the margin of the title to 0, and the font size to 1.2em.

Style the image inside the attraction block, make sure the maximum width of the image is 100%, the height is adaptive, and add a 5px border rounded corner.

Set the style of the description inside the attraction block, set the upper margin to 10px and the font size to 0.9em.



<div> : Define a container containing the attractions information, using the CSS class name "attractions".

@foreach (var attraction in Model) : Use Razor syntax to iterate through the set of attraction models passed to the view, executing the following code block foreach attraction.

<div> : Define a container that contains information about a single attraction, using the CSS class name "attraction".

<a href> : Creates a link to the "Details" action method of the Destinations controller and passes the ID of the destination as an argument. This link will be used to view the details of the attraction.

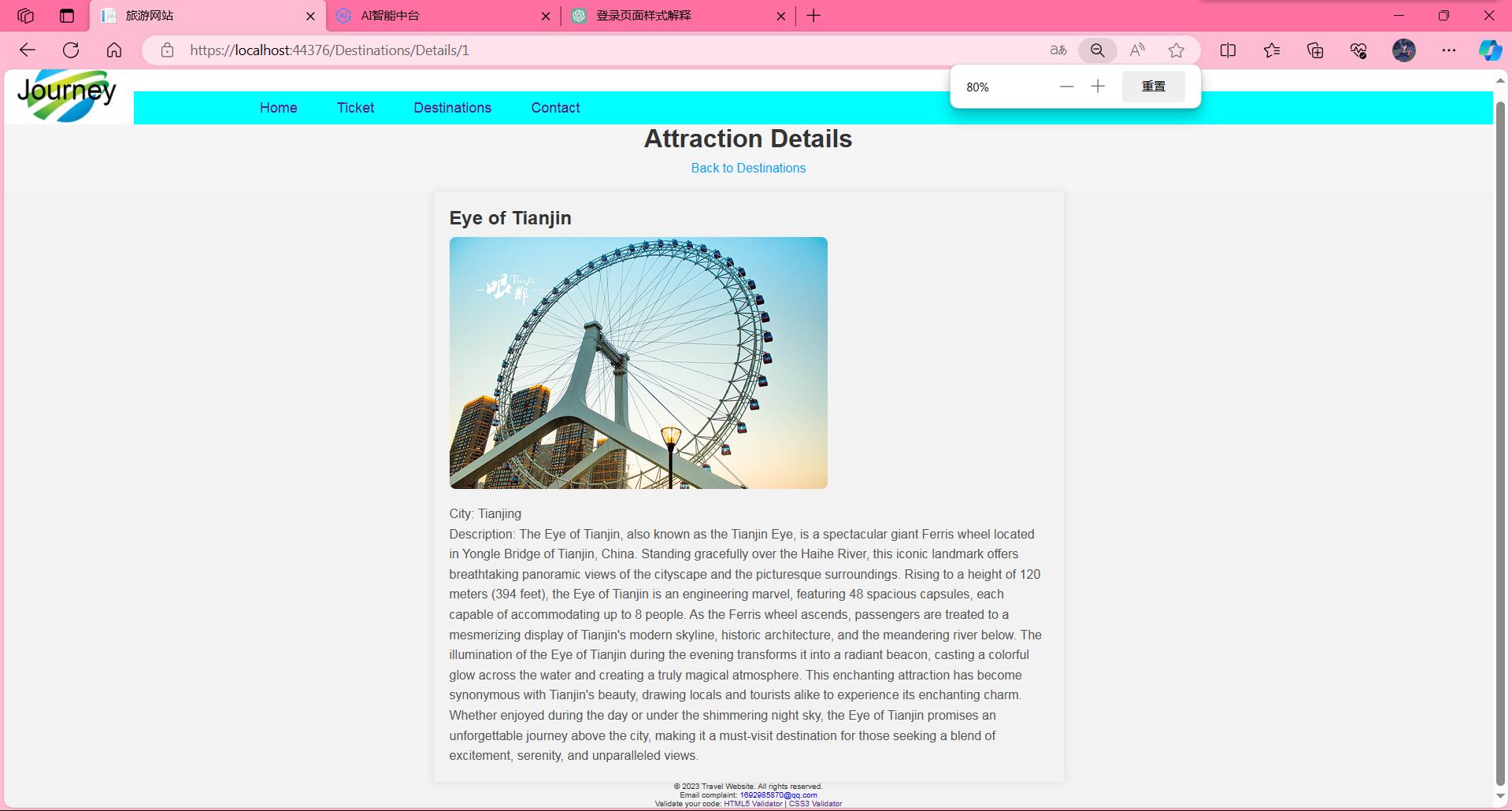
<h3> @Attract. CityName</h3> : Displays the name of the city where the attraction is located.

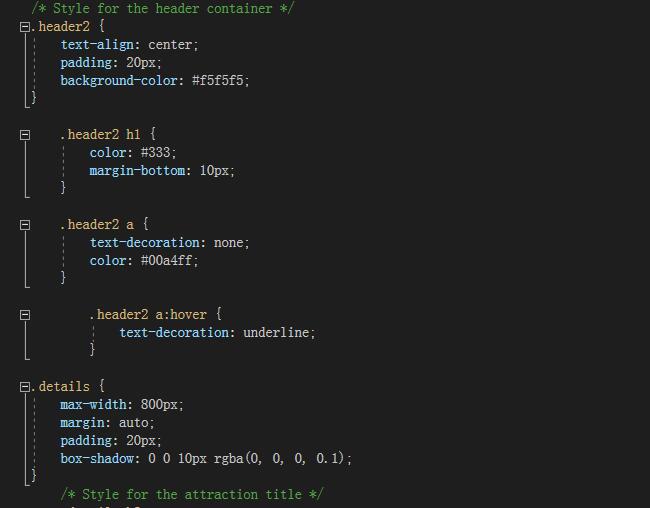
<img src> : Displays a picture of the attraction, the image path specified by the ImageUrl attribute in the attraction model.

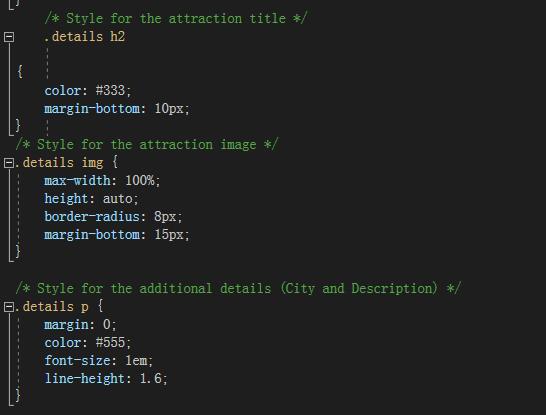
< p > @ attraction. AttractionName < / p > : show the name of the scenic spots.

The whole code block will generate a corresponding number of attraction display blocks according to the attraction information cycle in the model, and each block contains relevant information about the attraction. Users can click on links on each attraction block to see details.

**Detail page**







Common style (.attraction) :

width: Set the width of the spot block to 300px.

margin: Set the margin to 10px to control the spacing between the attraction blocks.

padding: Sets the padding to 10px to increase the distance between the content and the border.

border: Set the border to a 1px wide gray border.

Border-radius: Set the rounded corner of the border to 5px to make the corner of the attraction block more rounded.

box-shadow: Adds a slight shadow effect to enhance the depth of the block.

transition: Adds a 0.3 second transition effect to make scaling smoother.

attraction:hover:

When the mouse hovers over the attraction block, the attraction block is zoomed and transformed by the :hover pseudo-class selector so that it appears slightly enlarged (1.05 times) to provide visual feedback.

Link style (.attraction a) :

text-decoration: removes underscores from links.

color: Sets the link text color to dark gray.

Title style (.attraction h3) :

margin: Sets the margin of the title to 0.

Font-size: Sets the font size of the title to 1.2em.

Image style (.attraction img) :

max-width: Sets the maximum width of the image to 100%.

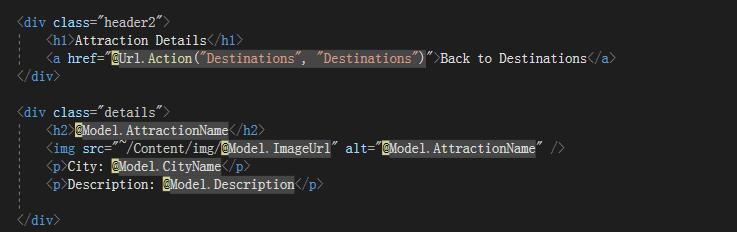
height: Sets the height adaptive of the image.

Border-radius: Add 5px border corners.

Description style (.attraction p) :

Margin-top: Sets the upper margin of the description to 10px.

Font-size: Sets the description font size to 0.9em.



<div> : Define a container to display the page title and return links, and apply the CSS class name "header2".

<h1>Attraction Details</h1> : Displays the title of the page "Attraction Details".

<a href>Back to Destinations</a> : Creates a link to the Destinations action method of the Destinations controller to return to the destination page.

<div> : Defines a container that displays the details of the attraction and applies the CSS class name "details".

<h2>@Model.AttractionName</h2> : Displays the name of the attraction, which is obtained from the view's Model.

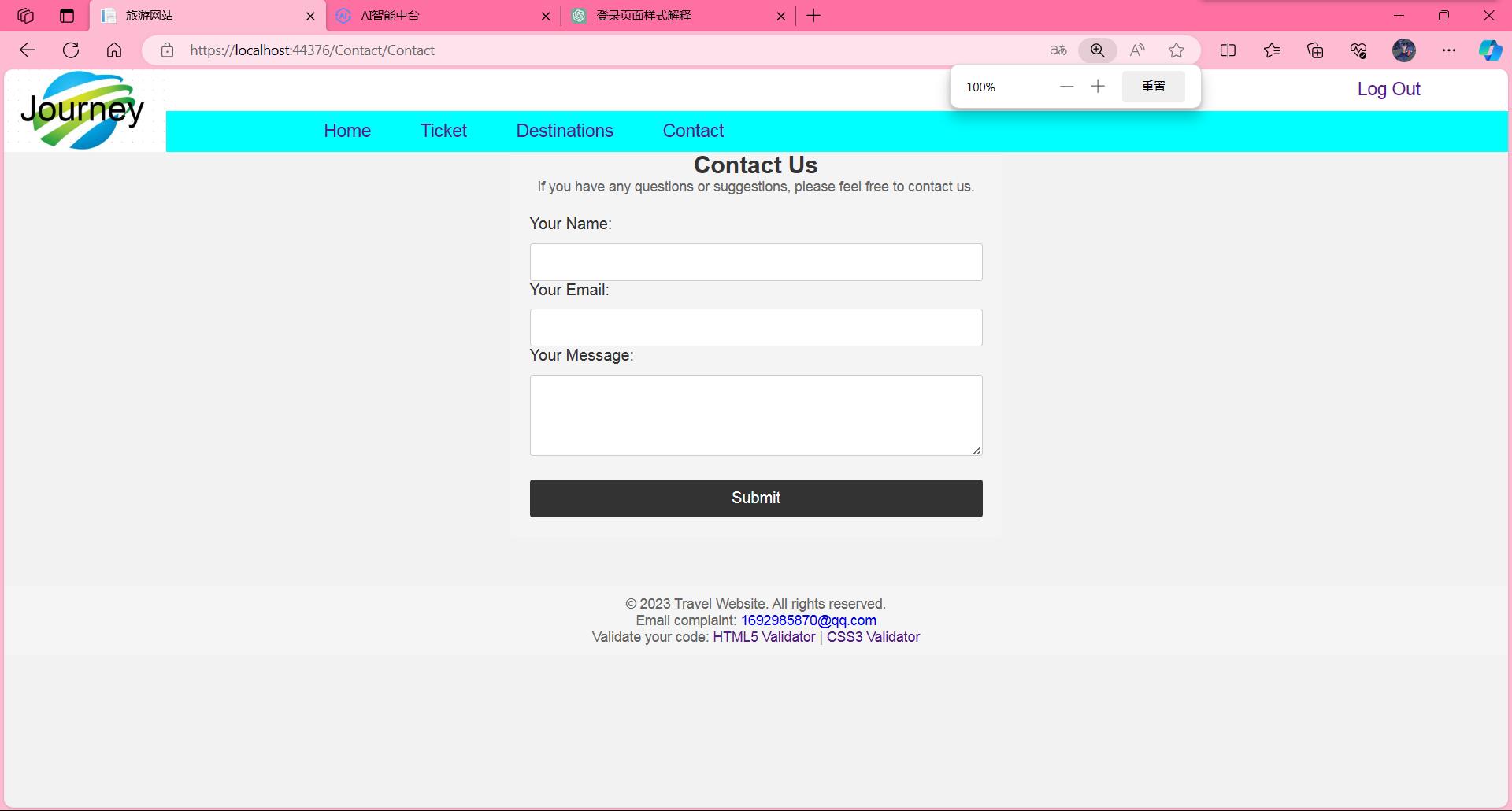
<img src alt="@Model.AttractionName" /> : Displays an image of the attraction. The image path is specified by the "ImageUrl" attribute in the view's model, with the "~/Content/img/" prefix indicating the path relative to the site root.

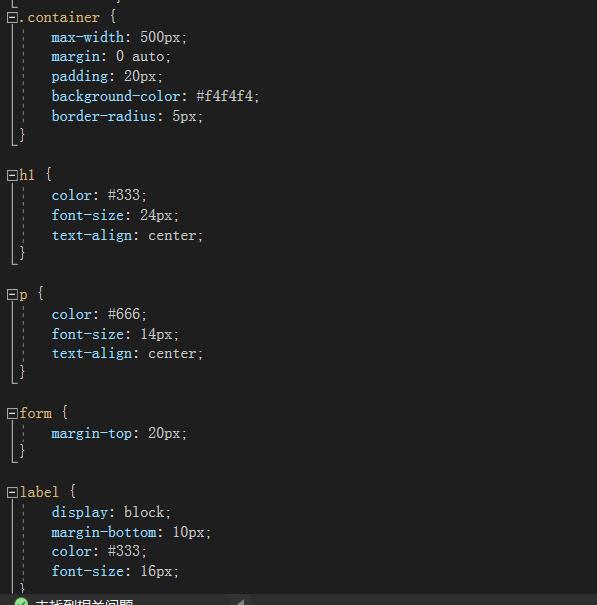
<p>City: @Model.CityName</p> : Displays the name of the city to which the attraction belongs, which is obtained from the view's Model.

<p>Description: @Model.Description</p> : Displays the description of the attraction, which is obtained from the view's Model.

This code dynamically displays information such as the name, picture, city, and description of the attraction based on the data in the view's Model, and provides a return link to return to the destination page.

**Contact page**







Container style (.container) :

max-width: Set the maximum width of the container to 500px to ensure that it remains the right width on larger screens. margin: Sets the left and right margins of the container to auto so that they are centered horizontally. padding: Sets the inside margin of the container to 20px to increase the distance between the contents and the border. Background-color: Sets the background color of the container to #f4f4f4. Border-radius: Add 5px border corners.

Title style (h1) :

color: Set the font color of the title to #333. Font-size: Sets the font size of the title to 24px. Text-align: Sets the horizontal center of the title text.

Paragraph style (p) :

color: Set the paragraph font color to #666. Font-size: Sets the font size of the paragraph to 14px. Text-align: Sets the horizontal centering of the paragraph text.

form Style:

Margin-top: Set the top margin of the form to 20px, with a certain distance from the content above.

label style:

display: Displays the label as a block-level element in a single line. Margin-bottom: Set the bottom margin of the label to 10px, with a certain distance from the input box below. color: Set the font color of the label to #333. Font-size: Sets the font size of the label to 16px.

input field style (input[type="text"], input[type="email"], textarea) :

padding: Sets the inside margin of the input box to 10px to increase the distance between the content and the border. border: Set the input box to a 1px wide gray border. Border-radius: Add 3px border corners. Font-size: Sets the font size of the input box to 14px.

submit button style (button[type="submit"]) :

display: Displays the button as a block-level element in a single line. width: Set the button width to 100%. padding: Sets the inside margin of the button 10px to increase the distance between the content and the border. Margin-top: Set the upper margin of the button to 20px with a certain distance from the upper input box.

Background-color: Sets the button's background color to #333. color: Set the font color of the button to #fff. Font-size: Sets the button font size to 16px. border: Deletes the border style of the button. Border-radius: Add 3px border corners. cursor: Sets the cursor style of the mouse over the button to hand.



<div> : Defines a container that applies the previously mentioned CSS style class "container".

<h1>Contact Us</h1> : Displays the main title of the page "Contact Us".

<p>If you have any questions or suggestions, please feel free to contact us.</p> : If you have any questions or suggestions, please feel free to contact us.

Form section:

&lt; form action="#" method="post"&gt; : Defines a form where action="#" means that the form data will be submitted to the current page, and method="post" means that a POST request will be used.

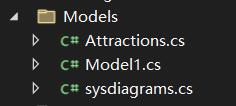
Name input box: &lt; label for="name"&gt; Your Name:&lt; /label&gt; &lt; input type="text" id="name" name="name" required&gt; , add labels and input fields for the name field, and set them as required fields.

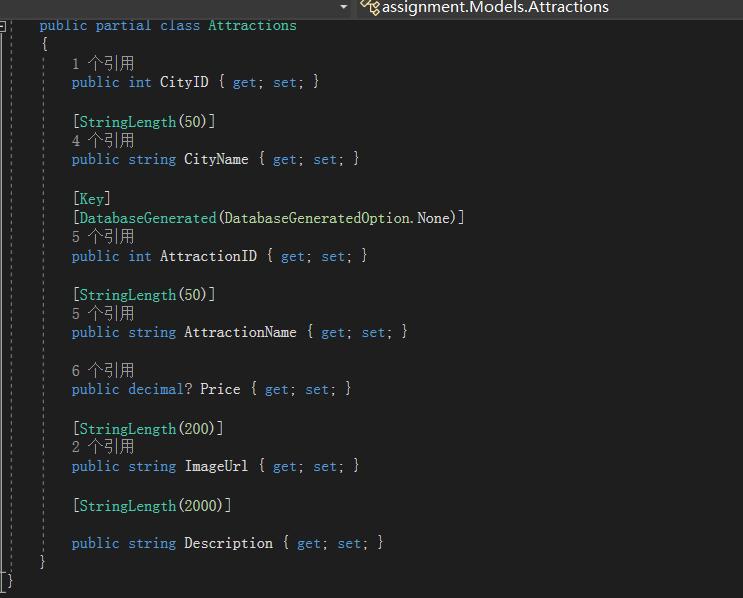
Email input box: &lt; label for="email"&gt; Your Email:&lt; /label&gt; &lt; input type="email" id="email" name="email" required&gt; , add labels and input boxes for the email fields and set them as required fields.

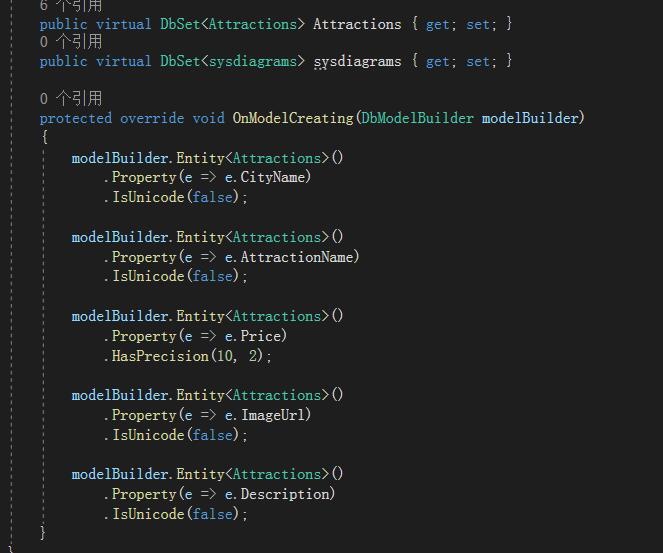
Message text box: &lt; label for="message"&gt; Your Message:&lt; /label&gt; &lt; textarea id="message" name="message" rows="4" required&gt; &lt; /textarea&gt; , add a label and multi-line text input field to the message field, and set it as a required field.

Submit button: &lt; button type="submit"&gt; Submit&lt; /button&gt; , defines a submit button.

**Models**







My database model.