HTTP POST

Query Strings

Query String

- Allow users to send information in a URL

- Common Application:
  - User types a query in a search engine
  - Their query is sent in the URL as a query string

URL Recall

Protocol://host:port/path?query_string#fragment

- Query String - [Optional] Contains key-value pairs set by the client
- https://www.google.com/search?q=web+development
  - HTTPS request to Google search for the phrase "web development"
- https://duckduckgo.com/?q=web+development&ia=images
  - An HTTPS request to Duck Duck Go image search for the phrase "web development"

- Fragment - [Optional] Specifies a single value commonly used for navigation
- https://en.wikipedia.org/wiki/Uniform_Resource_Identifier
  - HTTPS Request for the URI Wikipedia page
- https://en.wikipedia.org/wiki/Uniform_Resource_Identifier#Definition
  - HTTPS Request for the URI Wikipedia page that will scroll to the definition of URI

Query String Format

https://duckduckgo.com/?q=web+development&ia=images

- Preceded by a question mark - ?
- Consists of key-values pairs
  - Key and value separated by =
  - Pairs separated by &
- Can only contain ASCII characters
- Cannot contain white space

Percent Encoding

- If a non-ASCII character is sent as part of a query string it must be percent encoded
- Specify byte values with a % followed by 2 hex bytes

- 한
  - %ed%95%9c

- " " <-- single space
  - %20

White Space

- URLs cannot contain spaces
- Spaces can be percent encoded as %20
- Can also replace spaces with +
  - The reserved character + indicates a key mapping to multiple values

Reserved Characters

- Some ASCII characters are reserved
  - Example: ? begins a query string
- Reserved characters must be % encoded

- Notable characters that are NOT reserved
  - Dash -
  - Dot .
  - Underscore _
  - Tilda ~

Reserved	
:	&
/	'
?	(
#	)
[	*
]	+
@	,
!	;
$	=

Dynamic Pages

- We've learned how to host static content from our servers
  - Content does not change
- For the rest of the semester we'll add dynamic features
  - Users can change content and interact with other users
- No longer making web sites
- Now we're developing **Web Applications**

HTML Forms

- Forms allow users to send information to your server
- Use the input elements inside a form element

```
<form action="/form-path" method="get">
    <label for="form-name">Enter your name: </label><br/>
    <input id="form-name" type="text" name="commenter"><br/>

    <label for="form-comment">Comment: </label><br/>
    <input id="form-comment" type="text" name="comment"><br/>

    <input type="submit" value="Submit">
</form>
```

HTML Forms

- When a user submits the form it will send an HTTP request to your server
  - Server responds with an HTTP response
  - Submitting a form will reload the page with the content of the response

```
<form action="/form-path" method="get">
    <label for="form-name">Enter your name: </label><br/>
    <input id="form-name" type="text" name="commenter"><br/>

    <label for="form-comment">Comment: </label><br/>
    <input id="form-comment" type="text" name="comment"><br/>

    <input type="submit" value="Submit">
</form>
```

HTML Forms

- The action attribute is the path for the form
- The method attribute is the type of HTTP request made
- When the form is submitted, an HTTP request is sent to the path using this method
  - This behaves similar to clicking a link

```
<form action="/form-path" method="get">
    <label for="form-name">Enter your name: </label><br/>
    <input id="form-name" type="text" name="commenter"><br/>

    <label for="form-comment">Comment: </label><br/>
    <input id="form-comment" type="text" name="comment"><br/>

    <input type="submit" value="Submit">
</form>
```

HTML Forms

- Use input elements for the user to interact with the form
- The type attribute specifies the type of input
  - This input is a text box
- The name attribute is used when the data is sent to the server

```
<form action="/form-path" method="get">
    <label for="form-name">Enter your name: </label><br/>
    <input id="form-name" type="text" name="commenter"><br/>

    <label for="form-comment">Comment: </label><br/>
    <input id="form-comment" type="text" name="comment"><br/>

    <input type="submit" value="Submit">
</form>
```

HTML Forms

- Should provide a label for each input
  - Helps with accessibility (ie. Screen readers)
  - Clicking the label focuses the input
- Use ids to associate labels with inputs

```
<form action="/form-path" method="get">
    <label for="form-name">Enter your name: </label><br/>
    <input id="form-name" type="text" name="commenter"><br/>

    <label for="form-comment">Comment: </label><br/>
    <input id="form-comment" type="text" name="comment"><br/>

    <input type="submit" value="Submit">
</form>
```

HTML Forms

- An input of type submit makes a button that will send the HTTP request when clicked
- The value attribute is the text on the button

```
<form action="/form-path" method="get">
    <label for="form-name">Enter your name: </label><br/>
    <input id="form-name" type="text" name="commenter"><br/>

    <label for="form-comment">Comment: </label><br/>
    <input id="form-comment" type="text" name="comment"><br/>

    <input type="submit" value="Submit">
</form>
```

HTML Forms

- This sends a GET request containing the form data in a query string
  - Page reloads with the content of the response

```
GET /form-path?commenter=Jesse&comment=Good+morning%21 HTTP/1.1
```

```
<form action="/form-path" method="get">
    <label for="form-name">Enter your name: </label><br/>
    <input id="form-name" type="text" name="commenter"><br/>

    <label for="form-comment">Comment: </label><br/>
    <input id="form-comment" type="text" name="comment"><br/>

    <input type="submit" value="Submit">
</form>
```

HTTP GET Limitations

- Sending form data in a query string can cause issues
  - Browsers and servers have limits on the length of a URL
  - Browsers and servers have limits on the the total length of a GET request, including headers
    - Typically a 4-16kB
  - How would we upload a file? URL must be ASCII. Entire file would be % encoded
- Enter **POST** requests

HTTP POST

- A POST request is used when the user is sending information to the server
  - As opposed to requesting (GETing) information
- A POST request will include a **body**
  - Follows same protocol (HTTP) as our responses
  - Will include Content-Length and Content-Type headers to know how to read the body

HTTP POST

- Process a POST request:
  - Find the blank line "\r\n\r\n" indicating the end of the headers
  - Read the Content-Length header
    - Read this many bytes after the blank line
  - Parse the body according to the Content-Type
- This is what browsers are doing to read your responses

HTML Forms - POST

- Change the method of a form to post to send the entered data in the body of a POST request

```
<form action="/form-path" method="post">
    <label for="form-name">Enter your name: </label><br/>
    <input id="form-name" type="text" name="commenter"><br/>

    <label for="form-comment">Comment: </label><br/>
    <input id="form-comment" type="text" name="comment"><br/>

    <input type="submit" value="Submit">
</form>
```

HTML Forms - POST

- A request is sent to the path from the action attribute without a query string
- Content-Type is a url encoded string containing the entered data
  - Same format as the query string
- Read the Content-Length to know how many bytes are in the body
  - Foreshadow: Very import when receiving more data than the size of your TCP buffer

```
POST /form-path HTTP/1.1
Content-Length: 27
Content-Type: application/x-www-form-urlencoded

commenter=Jesse&comment=Good+morning%21
```

HTML Forms - POST

- We can change the encoding type when we don't want url encoded data
- Specify multipart encoding to receive each input separately in the body

```
<form action="/form-path" method="post" enctype="multipart/form-data">
    <label for="form-name">Enter your name: </label><br/>
    <input id="form-name" type="text" name="commenter"><br/>

    <label for="form-comment">Comment: </label><br/>
    <input id="form-comment" type="text" name="comment"><br/>

    <input type="submit" value="Submit">
</form>
```

HTML Forms - POST

- Content-Type specifies a string that separates each input
- Each input has its own headers
- Great for submitting different types of data in the same form
  - Required for file uploads

```
POST /form-path HTTP/1.1
Content-Length: 252
Content-Type: multipart/form-data; boundary=----WebKitFormBoundaryfkz9sCA6fR3CAHN4

------WebKitFormBoundaryfkz9sCA6fR3CAHN4
Content-Disposition: form-data; name="commenter"

Jesse
------WebKitFormBoundaryfkz9sCA6fR3CAHN4
Content-Disposition: form-data; name="comment"

Good morning!
------WebKitFormBoundaryfkz9sCA6fR3CAHN4--
```

HTML Forms - POST

- Much more about parsing this format in the next lecture

```
POST /form-path HTTP/1.1
Content-Length: 252
Content-Type: multipart/form-data; boundary=----WebKitFormBoundaryfkz9sCA6fR3CAHN4

------WebKitFormBoundaryfkz9sCA6fR3CAHN4
Content-Disposition: form-data; name="commenter"

Jesse
------WebKitFormBoundaryfkz9sCA6fR3CAHN4
Content-Disposition: form-data; name="comment"

Good morning!
------WebKitFormBoundaryfkz9sCA6fR3CAHN4--
```

HTML Inputs

- Radio Buttons:
  - Provide multiple options with the same name
  - Only one option with the same name can be chosen
  - The value property is sent to the server with this name

```
<form action="/form-path" method="post" enctype="multipart/form-data">

    <input id="option1" type="radio" name="chooseOne" value="1">
    <label for="option1"> 1</label><br/>
    <input id="option2" type="radio" name="chooseOne" value="2">
    <label for="option2"> 2</label><br/>
    <input id="option3" type="radio" name="chooseOne" value="3">
    <label for="option3"> 3</label><br/>

    <input type="submit" value="Submit">
</form>
```

HTML Inputs

- Dropdown Menus:
  - Use the select element to create a dropdown
  - The name of the select is sent to the server with the value of the selected option

```
<form action="/form-path" method="post" enctype="multipart/form-data">

    <label for="dropdown">Select an option: </label><br/>
    <select id="dropdown" name="dropping">
        <option value="first">First</option>
        <option value="second">Second</option>
        <option value="third">Third</option>
        <option value="home">Home</option>
    </select>

    <input type="submit" value="Submit">
</form>
```

HTML Inputs

- As always
  - There are many more input types
  - Search the documentation for more that you can add to your sites