Query Strings
and
HTML Templates

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

HTML Templates

The Problem

- We want to serve custom HTML
- In HW2-Obj4 you're sending different content based on the query string
  - There are 8 different images that can be requested
  - There are 256 unique combinations of images that can be requested
  - Are you going to write 256 different HTML files?
  - What about the user's name? Write 256 HTML files for each possible name? Write and serve millions of HTML files and still miss countless names

HTML Templates

- Instead of writing complete HTML files
  - Write an HTML template
- An HTML template is an "incomplete" HTML file that is used to generate complete pages
- Use additional markup to add placeholders in the HTML
- Replace the placeholders with data at runtime

HTML Templates

- Example template with 3 placeholders
- The title, description, and image_filname will be replaced later
  - Provide values for these 3 placeholders to serve a response

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>This page was generated from a template</title>
</head>
<body>

<h1>{{title}}</h1>

<p>{{description}}</p>

<img src="{{image_filename}}"/>

</body>
</html>
```

HTML Templates

- To substitute the placeholders
  - Use any string manipulation that gets the job done
  - Find/replace is the simplest solution
  - May want more advanced approaches if you want to add more functionality

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>This page was generated from a template</title>
</head>
<body>

<h1>{{title}}</h1>

<p>{{description}}</p>

<img src="{{image_filename}}"/>

</body>
</html>
```

Common Template Features

- Loops
- To add loops to your templates
  - Choose syntax for the start and end of the loop

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>This page was generated from a template</title>
</head>
<body>

{{loop}}
<h6>{{content_from_data_structure}}</h6>
{{end_loop}}

</body>
</html>
```

Common Template Features

- Use string manipulation to find the start and end tags
- Iterate over your data
  - Add the contained HTML with the placeholder replaced for each value of your data

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>This page was generated from a template</title>
</head>
<body>

{{loop}}
<h6>{{content_from_data_structure}}</h6>
{{end_loop}}

</body>
</html>
```

Common Template Features

- Conditionals
- Can use similar approach as loops
- Choose syntax for the start and end of each block in the conditional

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>This page was generated from a template</title>
</head>
<body>

{{if cookie_set}}
<h6>Welcome back!</h6>
{{else}}
<h6>Welcome!</h6>
{{end_if}}

</body>
</html>
```

Common Template Features

- Search for your tags
- Extract and evaluate the conditional
  - Choose how this will be evaluated
- Add the appropriate block of HTML to the page

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>This page was generated from a template</title>
</head>
<body>

{{if cookie_set}}
<h6>Welcome back!</h6>
{{else}}
<h6>Welcome!</h6>
{{end_if}}

</body>
</html>
```

HW2 Template

- You need a page that will display the user's name and their requested images
- Add each image as the src in its own img element
  - The browser will request these images in separate HTTP requests using your objective 3 paths
- General strategy:
  - Write an HTML template with a loop containing an img element with a variable for the src attribute
  - Iterate over all the requested image names and use the filenames as the content for the loop
- The syntax for your template is up to you

Examples