Server-side Web Applications

CS-370: Software Design and Development

Robert Bruce

Server-side web applications

- The World Wide Web is comprised of static and dynamic web-content.
- Static web content is a text file written in HTML. This content does not change and can be sent to a web-browser directly from the server.
- Server-side dynamic web content is generated on-the-fly by a web server.
- One method for generating dynamic web-content is via server-side applications and the Common Gateway Interface (CGI).
- The Common Gateway Interface (CGI) acts as a communications layer between a client (web browser) and a program running on a web server.

CGI example: user login authentication

- One example which uses the Common Gateway Interface (CGI) is a login-in web page.
- A user enters their “username” and “password” through a form which is sent to a server-side application (via CGI) for authentication processing.
- The authentication process involves verifying that the given username exists and the password matches what the user entered.
- If authentication is successful, the server side application creates dynamic web page content welcoming the user to a landing page (or some members-only area of a website).
- It is important to note, HTTP is a stateless protocol. This means HTTP doesn’t remember the authentication which just occurred. Therefore we will need some sort of mechanism to remember the user (and not endlessly prompt the user to authenticate over-and-over again).
- One mechanism to “remember” users is called an HTTP cookie.
- Another mechanism to “remember” users is amending the URL with a parameter while the user is logged-in. This is called session management.

Parameter passing in the Common Gateway Interface

- A web server communicates with an application in the Common Gateway Interface in one of two ways:
  - GET protocol
  - POST protocol
- The GET protocol embeds variable and value pairs in the URL. The web server sends these variable value pairs to server-side applications via the environment variable “QUERY_STRING”.
- The POST protocol sends a bunch of variable-value pairs through the environment variable “CONTENT_LENGTH”. The POST method is transparent to the user (i.e. not visible in the URL).

GET protocol

- The GET protocol embeds all variable-value pairs inside the URL. For example:
- `process_form_data.cgi?first_name=Robert&middle_name=James&last_name=Bruce&favorite_programming_language=C&operating_system_used1=GNU%2FLinux`
- We can break apart the above URL into variable-value pairs as follows (the ampersand serves as a delimiter to separate variable-value pairs):

1) `first_name=Robert`
2) `middle_name=James`
3) `last_name=Bruce`
4) `favorite_programming_language=C`
5) `operating_system_used1=GNU%2FLinux`

GET protocol

- We can further separate each variable-value pairs using the equals as a delimiter:
  - VARIABLE: `first_name`
  - VALUE: `Robert`
  - VARIABLE: `middle_name`
  - VALUE: `James`
  - VARIABLE: `last_name`
  - VALUE: `Bruce`
  - VARIABLE: `favorite_programming_language`
  - VALUE: `C`
  - VARIABLE: `operating_system_used1`
  - VALUE: `GNU%2FLinux`

GET protocol

- You may have noticed the last variable-value pair was as follows:
  - VARIABLE: `operating_system_used1`
  - VALUE: `GNU%2FLinux`
- The “%2F” is HTML code for a slash symbol “/” (ASCII hex code 2F).
- We have to convert to convert the “%2F” back into a slash ourselves. After the conversion, the value (above) then would be “GNU/Linux”.

HTML forms

- The GET and POST methods are utilized in HTML forms.
- HTML forms retrieve input from a user's web browser via an HTML web page.
- HTML forms may contain the following types of user-input:
  - Radio buttons: only one button is selectable
  - Check boxes: one or more buttons are selectable
  - Text entry boxes: fill in a string of text
  - Pull-down menus: select an option from a predefined menu
  - A "Submit" button
  - A "Cancel" button
  - A user-labeled button

HTML forms

- When defining each HTML form input type (e.g. text box, radio button, push button, etc.) we should define the variable (which will store the user's input). This variable is ultimately what we will be looking for in our variable-value pairs on the server-side.
- For text entry boxes, the value is whatever the user entered in the text box.
- For radio buttons, check boxes, push buttons, etc. We define what the value will be when the user presses that button.

HTML forms

- When defining an HTML form, it is vital you define the “action” element.
- The “action” element defines the URL which will process the HTML form data when the user presses the “submit” button.
- The “method” element defines how the data in the HTML form will be sent to the server and ultimately our programs via the Common Gateway Interface. There are two possible values: GET or POST.

An example HTML form

```
<form action="https://blue.cs.sonoma.edu/~rbruce/cgi/process_form_data.cgi" enctype="application/x-
www-form-urlencoded" method="post">
  <p>Please select one of the following options then press the submit button:</p>
  <input type="radio" id="DISPLAY_GET_FORM" name="use_form_protocol" value="get"> 
  <label for="DISPLAY_GET_FORM">Display an HTML form using GET protocol</label><br>
  <input type="radio" id="DISPLAY_POST_FORM" name="use_form_protocol" value="post"> 
  <label for="DISPLAY_POST_FORM">Display an HTML form using the POST protocol</label><br>
  <br><br>
  <input type="submit" value="Submit">
</form>
```

The example HTML form

```
<form action="https://blue.cs.sonoma.edu/~rbruce/cgi/process_form_data.cgi" enctype="application/x-
www-form-urlencoded" method="post">
  <p>Please select one of the following options then press the submit button:</p>
  <input type="radio" id="DISPLAY_GET_FORM" name="use_form_protocol" value="get"> 
  <label for="DISPLAY_GET_FORM">Display an HTML form using GET protocol</label><br>
  <input type="radio" id="DISPLAY_POST_FORM" name="use_form_protocol" value="post"> 
  <label for="DISPLAY_POST_FORM">Display an HTML form using the POST protocol</label><br>
  <br><br>
  <input type="submit" value="Submit">
</form>
```

Dissecting the example HTML form

```
action="https://blue.cs.sonoma.edu/~rbruce/cgi/process_form_data.cgi"
```

- The action command (above) species the URL to submit data to when the user presses the “submit” button.
- *Tip: notice the server-side application above is called “process_form_data.cgi”. I purposefully labeled this server-side application with a generic “dot cgi” extension for security reasons. I don’t reveal to the client (user) what programming language I used to write the server-side application.*

The example HTML form

```
<form action="https://blue.cs.sonoma.edu/~rbruce/cgi/process_form_data.cgi" enctype="application/x-
www-form-urlencoded" method="post">
  <p>Please select one of the following options then press the submit button:</p>
  <input type="radio" id="DISPLAY_GET_FORM" name="use_form_protocol" value="get"> 
  <label for="DISPLAY_GET_FORM">Display an HTML form using GET protocol</label><br>
  <input type="radio" id="DISPLAY_POST_FORM" name="use_form_protocol" value="post"> 
  <label for="DISPLAY_POST_FORM">Display an HTML form using the POST protocol</label><br>
  <br><br>
  <input type="submit" value="Submit">
</form>
```

The example HTML form

```
<form action="https://blue.cs.sonoma.edu/~rbruce/cgi/process_form_data.cgi" enctype="application/x-
www-form-urlencoded" method="post">
  <p>Please select one of the following options then press the submit button:</p>
  <input type="radio" id="DISPLAY_GET_FORM" name="use_form_protocol" value="get"> 
  <label for="DISPLAY_GET_FORM">Display an HTML form using GET protocol</label><br>
  <input type="radio" id="DISPLAY_POST_FORM" name="use_form_protocol" value="post"> 
  <label for="DISPLAY_POST_FORM">Display an HTML form using the POST protocol</label><br>
  <br><br>
  <input type="submit" value="Submit">
</form>
```

Dissecting the example HTML form

```
method="post"
```

- All data specified in the HTML form will be sent to the server (and ultimately to our server-side application via the POST protocol.

The example HTML form

```
<form action="https://blue.cs.sonoma.edu/~rbruce/cgi/process_form_data.cgi" enctype="application/x-
www-form-urlencoded" method="post">
  <p>Please select one of the following options then press the submit button:</p>
  <input type="radio" id="DISPLAY_GET_FORM" name="use_form_protocol" value="get"> 
  <label for="DISPLAY_GET_FORM">Display an HTML form using GET protocol</label><br>
  <input type="radio" id="DISPLAY_POST_FORM" name="use_form_protocol" value="post"> 
  <label for="DISPLAY_POST_FORM">Display an HTML form using the POST protocol</label><br>
  <br><br>
  <input type="submit" value="Submit">
</form>
```

The example HTML form

```
<form action="https://blue.cs.sonoma.edu/~rbruce/cgi/process_form_data.cgi" enctype="application/x-
www-form-urlencoded" method="post">
  <p>Please select one of the following options then press the submit button:</p>
  <input type="radio" id="DISPLAY_GET_FORM" name="use_form_protocol" value="get"> 
  <label for="DISPLAY_GET_FORM">Display an HTML form using GET protocol</label><br>
  <input type="radio" id="DISPLAY_POST_FORM" name="use_form_protocol" value="post"> 
  <label for="DISPLAY_POST_FORM">Display an HTML form using the POST protocol</label><br>
  <br><br>
  <input type="submit" value="Submit">
</form>
```

Dissecting the example HTML form

```
<input type="radio" id="DISPLAY_GET_FORM" name="use_form_protocol" value="get">
```

- This radio button's variable is "use_form_protocol".
- If selected by the user, this variable's value will be "get".
- The "id" field is used for screen-readers to make the web form easy to fill-out for visually impaired users.

The example HTML form

```
<form action="https://blue.cs.sonoma.edu/~rbruce/cgi/process_form_data.cgi" enctype="application/x-
www-form-urlencoded" method="post">
  <p>Please select one of the following options then press the submit button:</p>
  <input type="radio" id="DISPLAY_GET_FORM" name="use_form_protocol" value="get"> 
  <label for="DISPLAY_GET_FORM">Display an HTML form using GET protocol</label><br>
  <input type="radio" id="DISPLAY_POST_FORM" name="use_form_protocol" value="post"> 
  <label for="DISPLAY_POST_FORM">Display an HTML form using the POST protocol</label><br>
  <br><br>
  <input type="submit" value="Submit">
</form>
```

The example HTML form

```
<form action="https://blue.cs.sonoma.edu/~rbruce/cgi/process_form_data.cgi" enctype="application/x-
www-form-urlencoded" method="post">
  <p>Please select one of the following options then press the submit button:</p>
  <input type="radio" id="DISPLAY_GET_FORM" name="use_form_protocol" value="get"> 
  <label for="DISPLAY_GET_FORM">Display an HTML form using GET protocol</label><br>
  <input type="radio" id="DISPLAY_POST_FORM" name="use_form_protocol" value="post"> 
  <label for="DISPLAY_POST_FORM">Display an HTML form using the POST protocol</label><br>
  <br><br>
  <input type="submit" value="Submit">
</form>
```

Dissecting the example HTML form

```
<input type="radio" id="DISPLAY_POST_FORM" name="use_form_protocol" value="post">
```

- This radio button's variable is "use_form_protocol".
- If selected by the user, this variable's value will be "post".
- The "id" field is used for screen-readers to make the web form easy to fill-out for visually impaired users.

The example HTML form

```
<form action="https://blue.cs.sonoma.edu/~rbruce/cgi/process_form_data.cgi" enctype="application/x-
www-form-urlencoded" method="post">
  <p>Please select one of the following options then press the submit button:</p>
  <input type="radio" id="DISPLAY_GET_FORM" name="use_form_protocol" value="get"> 
  <label for="DISPLAY_GET_FORM">Display an HTML form using GET protocol</label><br>
  <input type="radio" id="DISPLAY_POST_FORM" name="use_form_protocol" value="post"> 
  <label for="DISPLAY_POST_FORM">Display an HTML form using the POST protocol</label><br>
  <br><br>
  <input type="submit" value="Submit">
</form>
```

An example HTML form

```
<form action="https://blue.cs.sonoma.edu/~rbruce/cgi/process_form_data.cgi" enctype="application/x-
www-form-urlencoded" method="post">
  <p>Please select one of the following options then press the submit button:</p>
  <input type="radio" id="DISPLAY_GET_FORM" name="use_form_protocol" value="get"> 
  <label for="DISPLAY_GET_FORM">Display an HTML form using GET protocol</label><br>
  <input type="radio" id="DISPLAY_POST_FORM" name="use_form_protocol" value="post"> 
  <label for="DISPLAY_POST_FORM">Display an HTML form using the POST protocol</label><br>
  <br><br>
  <input type="submit" value="Submit">
</form>
```

Dissecting the example HTML form

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

- This push button's variable is "submit".
- This push button's value is "Submit". That is also the label which will appear on the button.
- If the user presses this button, the user's browser will send the user's form data to the URL specified in the ACTION tag.
- The "submit" button variable value pair will also be submitted to the server along with any other variable-value pairs specified in the HTML form.

Example: server-side CGI web application

- I’ve created an example server-side CGI application in the C programming language for you.
- I’ve created a Makefile to build this application on Blue.
- To download this program, please see the Files folder under Canvas. Then click “Example programs”. Then click “Common Gateway Interface”. Then click “c”. You should see two files in this folder: “Makefile” and “process_form_data.c”
- You may also run this application under Blue: https://blue.cs.sonoma.edu/~rbruce/cgi/process_form_data.cgi

Retaining state in HTTP

- Maintaining state on a website is critical for any functionality that is members-only or requires some sort of authentication before being able to view or access content.
- Typically one creates two web-based forms on their website for such functionality:
  - A registration page to sign-up and create a new account to access the website.
  - A user-login / password page for existing users who have already signed-up for an account.

Retaining state in HTTP

- How do you maintain state within HTTP since it is a stateless protocol?
  - HTTP cookie
  - Defining a session variable.
- The HTTP cookie method is not reliable since users may disable cookies. HTTP cookies are created by your web browser (client). They are small files the browser creates and stores on the user's computer. This small file contains a variable-value pair. A server-side application can ask your browser (client) to read the cookie (if it exists and you allow cookies to be read).
- I don't rely on cookies at all for website development. I use sessions. Sessions are transparent to the user and always work.
- A session is simply a variable (session) and a value that I embed in an HTML form as a hidden variable or I embed as a variable=value pair within a URL (if using the GET method).
- PHP has built-in session management. C and C++ do not have such functionality built-in.
- *TIP: I encrypt session values on the server. The client never sees the unencrypted session value. This keeps clients from tampering with session values and makes it exceptionally difficult to compromise my website.*

For further reading...

If you are interested in all various types of environment variables the web-server could send to your server-side application via the Common Gateway Interface, please review the document:

RFC-3875: The Common Gateway Interface (version 1.1)

https://www.ietf.org/rfc/rfc3875