

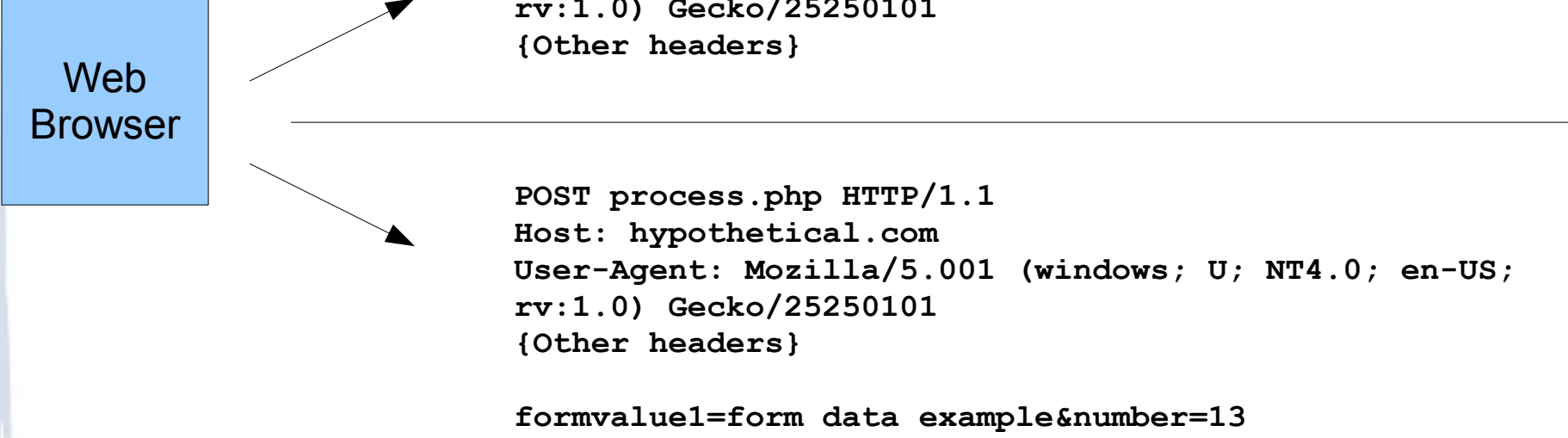
More PHP Details

In This Lecture

- Built in variables.
- Form processing using PHP.

Overview

Web
Browser



```
graph LR; WB[Web Browser] --> GET[GET process.php?data1=2 HTTP/1.1  
Host: hypothetical.com  
User-Agent: Mozilla/5.001 (windows; U; NT4.0; en-US;  
rv:1.0) Gecko/25250101  
{Other headers}]; WB --> POST[POST process.php HTTP/1.1  
Host: hypothetical.com  
User-Agent: Mozilla/5.001 (windows; U; NT4.0; en-US;  
rv:1.0) Gecko/25250101  
{Other headers}  
  
formvalue1=form data example&number=13];
```

GET process.php?data1=2 HTTP/1.1

Host: hypothetical.com

User-Agent: Mozilla/5.001 (windows; U; NT4.0; en-US;
rv:1.0) Gecko/25250101
{Other headers}

POST process.php HTTP/1.1

Host: hypothetical.com

User-Agent: Mozilla/5.001 (windows; U; NT4.0; en-US;
rv:1.0) Gecko/25250101
{Other headers}

formvalue1=form data example&number=13

Overview

- The HTTP Requests generated by the browser can be submitted in two methods (generally): GET and POST.
- Data(from forms*) can be submitted by
 - Encoding it with the URL of the request (GET)
 - Writing it down on the message body (POST)

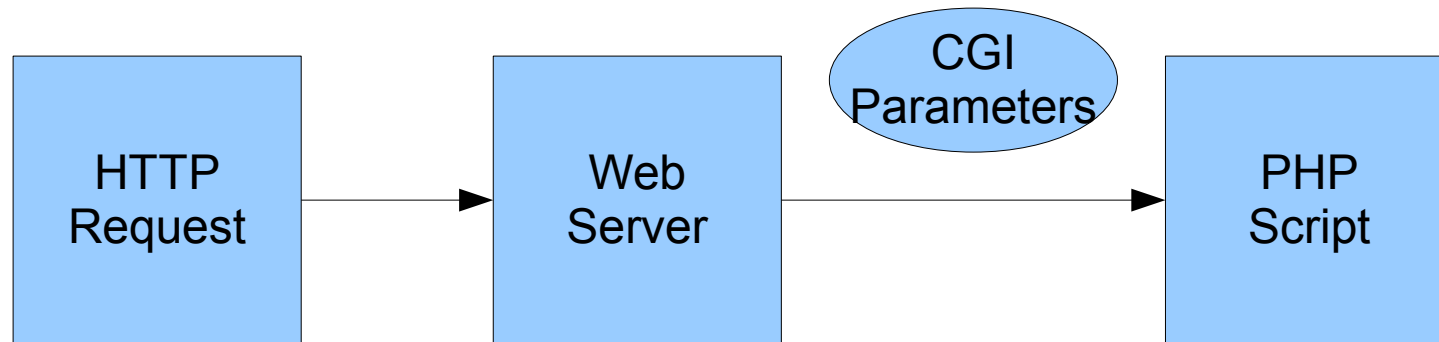
Overview

- Data from submitted forms:
 - The **value** of the form element is associated with the form element's **name**.
 - i.e. `<input type='text' name='str' value='Hello'/>` becoms **str=Hello**
 - Spaces and other special characters are **percent-encoded** (see URL).

Overview:

- Data in HTTP Requests:
 - Request line data (i.e. HTTP method used)
 - Headers: Request headers, general headers, entity headers
 - Form data or user-defined data

In PHP



- The Web server passes HTTP request info as parameters to be used by the PHP script.
- These CGI parameters are automatically “stored” in built-in variables for ready use in PHP.

Predefined/Built-in Vars in PHP

- **Superglobals** – global variables that are always available in all scopes.

- \$GLOBALS
- \$_SERVER
- \$_GET
- \$_POST
- \$_FILES

- \$_REQUEST
- \$_SESSION
- \$_ENV
- \$_COOKIE
- \$HTTP_RAW_POST_DATA
- \$php_errormsg, \$http_response_header, \$argc, \$argv

`$_SERVER`

(or `$HTTP_SERVER_VARS` [dep])

- Associative array containing values described by the **CGI 1.1 specification** – i.e. headers, paths, script locations, etc.
 - i.e. `$_SERVER['HTTP_USER_AGENT']` is the user-agent value specified by the User-Agent header of the request.

`$_GET` and `$_POST`

(formerly `$HTTP_GET_VARS*`, `$HTTP_POST_VARS*`)

- Associative array containing data submitted to the script using GET and POST method respectively.
 - `$_GET['name']` is the value of data associated with '**name**' say in
 - GET `process.php?name=pedro` HTTP/1.1
 - `$_POST['name']` is the value associated with '**name**' say in

POST `process.php` HTTP/1.1

{Other headers}

name=pedro&other=juan

`$_FILES`

- Associative array of items uploaded to current script via POST.
- Multi-dimensional array.

\$_REQUEST, \$_COOKIE, \$_SESSION, \$_ENV

- \$_REQUEST - Associative array contains \$_GET, \$_POST, \$_COOKIE.
- \$_COOKIE and \$_SESSION – will be discussed under **Application State Management**
- \$_ENV – gives you access to environment settings (runtime environment)
 - Values found in the php.ini config file for PHP.

Form Processing in PHP

- On the browser: everything that has value is submitted
 - Textfield value, textarea value, submit button used, etc.
 - **Name** attribute is particularly useful for easy reference to form data.

Processing textfields and other Single Valued Data items.

- Straightforward: use the name as used in the form to refer to the data item.

```
<form  
  action='process.php'  
  method='get'>  
  
  <input type='text'  
        name='fname' />  
  
  <input type='submit' />  
  
</form>
```

```
//In process.php  
  
$fname= $_GET['fname'];
```

Forms with Multiple Submit buttons

```
<form action='process.php'  
method='get'>
```

```
<input type='submit'  
      name='open'  
      value='Open' />
```

```
<input type='submit'  
      name='close'  
      value='Close' />
```

```
</form>
```

```
//in process.php
```

```
if (isset($_GET['open']))  
{
```

```
    //if the Open submit
```

```
    //was used used
```

```
}
```

```
//isset function checks
```

```
//for existence of var.
```

Submitting Multi-valued data items

(check box groups and multi-select boxes)

- For PHP to effectively process multi-valued data items, the name in the form should contain “[]” so values are automatically saved as arrays.

```
<form action="..." >  
  <input type='checkbox'  
        name='choices['  
        value='red' /> Red  
  <input type='checkbox'  
        name='choices['  
        value='blue' />Blue  
  ...  
</form>
```


Submitting Multi-valued data items

(check box groups and multi-select boxes)

```
//in process.php
```

```
$c = $_GET['choices'];
```

```
//$c is an array
```

```
//so $_GET['choices'][0] is possible
```

```
foreach ($c as $index=>$value) {
```

```
    echo "Choice No. $index is  
    $value";
```

```
}
```

File uploads

```
<form
```

```
  enctype="multipart/form-data"
```

```
  method="post"
```

```
  action="processupload.php">
```

```
    <input type='hidden' name='MAX_FILE_SIZE'
      value='3000' />
```

PHP-specific requirement
Must precede file input field



```
    Send file: <input type='file' name='ufile' />
```

```
    <input type='submit' />
```

```
</form>
```

File uploads

```
//To get the file details
echo "Orig name of file " . $_FILES['ufile']['name'] .
    "<br/>";
echo "File Type: " . $_FILES['ufile']['type'] . "<br/>";
echo "Size: " . $_FILES['ufile']['size'] . " bytes <br/>";
echo "File name in server is " .
    $_FILES['ufile']['tmp_name'] . "<br/>";
echo "Error : " . $_FILES['ufile']['error'] . "<br/>";
//then you can use file functions in PHP
//such as move_upload_file( );
```

Multiple File Uploads

```
<form action="file-upload.php" method="post"  
enctype="multipart/form-data">
```

```
  Send these files:<br />
```

```
  <input name="userfile[]" type="file" /><br />
```

```
  <input name="userfile[]" type="file" /><br />
```

```
  <input type="submit" value="Send files" />
```

```
</form>
```

JSP vs PHP Predefined Vars

- Some JSP Predefined equivalents
 - **request** -- equivalent to `$_REQUEST` and `$_SERVER`
 - i.e. `request.getParameter("name");` gets single valued data item
 - `request.getParameters("choices");` for multiple valued data item
 - `request.getHeader("User-Agent");` getting header info.
 - **response**
 - **Session**, etc.