WEBD 236

Web Information Systems Programming

Week 13

(todd.whittaker@franklin.edu)

FRANKLIN UNIVERSITY

Agenda

- This week's expected outcomes
- This week's topics
- This week's homework
- Upcoming deadlines
- Solution to Homework 10
- Questions and answers

FRANKLIN UNIVERSITY

Week 13 Outcomes

- Send e-mail from web applications.
- Invoke web services from remote sites using cURL.
- Examine JSON as a data-interchange format.

Sending email from PHP

- SMTP (Simple Mail Transfer Protocol)
  - Must have a properly configured SMTP server.
  - Easy to configure it to work with Google's SMTP servers:
    - Create a GMail account, enable POP.
    - Communicate with Google over SSL.

Sending email from PHP

- Configuration issues with XAMPP
  - Must turn on SSL by adding the following line to `C:\xampp\php\php.ini`

```
; enable SSL for e-mail send/receive
extension=php_openssl.dll
```

  - Then restart Apache

Sending email from PHP

- Using the PEAR (PHP Extension and Application Repository) library for mail
  - Require the '`Mail.php`' file, pulled from `c:\xampp\php\PEAR`
  - Turn off strict error reporting (`Mail.php` is a library written for PHP4)

```
<?php
error_reporting(E_ALL & !E_STRICT);
require_once 'Mail.php';
```

Sending email from PHP

- Let's abstract this into a class!

```
class Email {
    private static $smtpHost = "ssl://smtp.gmail.com";
    private static $smtpPort = 465;
    private static $smtpAuth = true;
    private static $smtpUsername = 'someaccount@gmail.com';
    private static $smtpPassword = 'someAccountPassword';
    private $smtp;
    private $recipients;
    private $carbonCopy;
    private $blindCopy;
    private $subject;
    private $message;
    private $sender;
    private $body;
    private $contentType;
```


Sending email from PHP

- Let's abstract this into a class!

```
    public function __construct() {
        $options = array();
        $options['host'] = self::$smtpHost;
        $options['port'] = self::$smtpPort;
        $options['auth'] = self::$smtpAuth;
        $options['username'] = self::$smtpUsername;
        $options['password'] = self::$smtpPassword;

        $mail = new Mail();
        $this -> smtp = $mail -> factory('smtp', $options);

        if (is_a($this -> smtp, 'PEAR_Error')) {
            throw new Exception("Could not create mailer");
        }
        // ...continued...
```

Sending email from PHP

- Let's abstract this i

```
public function __co
    $options = array(
    $options['host']
    $options['port']
    $options['auth']
    $options['usernam
    $options['password

    $mail = new Mail();
    $this -> smtp = $mail -> factory('smtp', $options);

    if (is_a($this -> smtp, 'PEAR_Error')) {
        throw new Exception("Could not create mailer");
    }
    // ...contin
```

Your textbook uses `Mail::factory()`, but that triggers errors in `E_STRICT` mode since factory isn't declared to be `static`.

Same here where your book uses Your textbook uses `PEAR::isError()`.

Sending email from PHP

- Let's abstract this into a class!

```
        $this -> recipients = array();
        $this -> carbonCopy = array();
        $this -> blindCopy = array();
        $this -> sender = '"WEBD236 Team" <' .
             self::$smtpUsername . '>';
    }

    public function addRecipient($recipient) {
        $this -> recipients[] = $recipient;
    }

    public function setRecipient($recipient) {
        $this -> recipients = array();
        $this -> addRecipient($recipient);
    }
```

Sending email from PHP

- Let's abstract this into a class!

```
        $this
        $this
        $this
        $this
             se
    }

    public func
        $this -> recipi                precipient;
    }

    public function setRecipient($recipient) {
        $this -> recipients = array();
        $this -> addRecipient($recipient);
    }
```

The use here is that you can set many different options, and then from within a loop call `setRecipient()` followed by `send()` to deliver customized email messages. BCC is more efficient if the message is the same (one send vs. many).

Sending email from PHP

- Let's abstract this into a class!

```
    public function addCC($recipient) {
        $this -> carbonCopy[] = $recipient;
    }
    public function setCC($recipient) {
        $this -> carbonCopy = array();
        $this -> addCC($recipient);
    }
    public function addBcc($recipient) {
        $this -> blindCopy[] = $recipient;
    }
    public function setBcc($recipient) {
        $this -> blindCopy = array();
        $this -> addBcc($recipient);
    }
```

Sending email from PHP

- Let's abstract this into a class!

```
public function setSender($sender) {
    $this -> sender = $sender;
}

public function setSubject($subject) {
    $this -> subject = $subject;
}

public function setBody($body) {
    $this -> body = $body;
}

public function setContentType($contentType) {
    $this -> contentType = $contentType;
}
```


Sending email from PHP

- Let's abstract this into a class!

```
public function setSender($sender) {
    $this -> sender = $sender;
}

public function setSubject(
    $this -> subject = $sub
}

public function setBody($bo
    $this -> body = $body;
}

public function setContentType($contentType) {
    $this -> contentType = $contentType;
}
```

Defaults to plain text with no content type. Use `"text/html"` to send HTML-based mail.

Sending email from PHP

- Let's abstract this into a class!
  - Wouldn't it be nice to use the same template rendering engine we used for web-pages to generate email templates too?
    - A small, backwards compatible change to `renderTemplate` in `include/util.inc` makes this possible.

Sending email from PHP

- Modifying `include/util.inc`

```
function renderTemplate($view, $params = array(),
        $asString = false) {
    $useCaching = false; // turn off caching
    if (!file_exists($view)) {
        die("File $view doesn't exist.");
    }
    // do we have a cached version?
    clearstatcache();
    $cacheName = __cacheName($view);
    if ($useCaching && file_exists($cacheName) &&
            (filemtime($cacheName) >= filemtime($view))) {
        $contents = file_get_contents($cacheName);
    } else {

    // ...continued...
```

Sending email from PHP

- Modifying `include/util.inc`

```
        $contents = __importTemplate(array('unused', $view));
        $contents = preg_replace_callback('/@@\s*(.*)\s*@@/U',
            '__resolveRelativeUrls', $contents);
        $patterns = array(
            array('src' => '/{{/', 'dst' => '<?php echo('),
            array('src' => '/}}/', 'dst' => ');?>'),
            array('src' => '/\[\[/', 'dst' => '<?php '),
            array('src' => '/\]\]/', 'dst' => '?>')
        );
        foreach ($patterns as $pattern) {
            $contents = preg_replace($pattern['src'],
                $pattern['dst'], $contents);
        }
        file_put_contents($cacheName, $contents);
    } // ...continued...
```

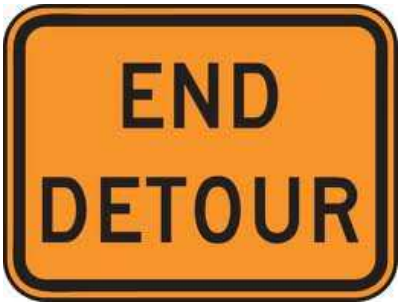

Sending email from PHP

- Modifying `include/util.inc`

```
    extract($params);
    ob_start();
    eval("?>" . $contents);
    $result = ob_get_contents();
    ob_end_clean();
    if (!$asString) {
        echo $result;
    }
    return $result;
}
```

Sending email from PHP

- Let's abstract this into a class!
  - Now, if we pass a third parameter (`true`), then it won't produce output, but will return the rendered string.



Sending email from PHP

- Meanwhile, back in `Email.inc`…

```
public function send($template=false, $variables=false) {
    if ($template) {
        $this -> body = renderTemplate($template,
            $variables, true);
    }
    $headers = array();
    $headers['From'] = $this -> sender;
    $headers['To'] = implode(", ", $this -> recipients);
    $headers['Cc'] = implode(", ", $this -> carbonCopy);
    $headers['Bcc'] = implode(", ", $this -> blindCopy);
    $headers['Subject'] = $this -> subject;
    if ($this -> contentType) {
        $headers['Content-type'] = $this -> contentType;
    }
    // ...continued...
```

Sending email from PHP

- Meanwhile, back in `Email.inc`…

```
        $allRecipients = array_merge($this -> recipients,
            $this -> blindCopy, $this -> carbonCopy);

        $result = $this -> smtp -> send(implode(", ",
            $allRecipients), $headers, $this -> body);

        if (is_a($result, 'PEAR_Error')) {
            throw new Exception($result -> getMessage());
        }
    }
} // end Email class
```


Sending email from PHP

- A simple `email_template.html`

```
<!DOCTYPEhtml PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html>
  <head><title>{{$title}}</title></head>
  <body>
    <p>Hello {{$firstName}},</p>
    <p>I'm writing to you today to tell you about our
    great product! It's not Spam, I assure you. It's an
    e-mail system that integrates easily with PHP and it's
    built on top of PEAR::Mail. Please consider buying it.</p>
    <p>Sincerely,</p>
    <p>WEBD236-V1WW</p>
  </body>
</html>
```

Sending email from PHP

- Using the `Email` class

```
$subject = "Testing an e-mail through PHP";
$recipient = '"Todd Whittaker" <todd.whittaker@franklin.edu>';
$email = new Email();
$email -> setRecipient($recipient);
$email -> setSubject($subject);
$email -> setContentType('text/html');
$email -> send(
    'email_template.html',
    array(
        'title' => $subject,
        'firstName' => 'Todd'
    )
);
```


Sending email from PHP

- Using th

```
$subject = "T
$recipient =
$email = new
$email -> set
$email -> set
$email -> set
$email -> sen
    'email_te
    array(
        'title' => $subject,
        'firstName' => 'Todd'
    )
);
```

Testing an e-mail through PHP March 28, 2012 10:53 AM

From: "WEBD 236" <webd236@gmail.com>

To: "Todd Whittaker" <todd.whittaker@franklin.edu>

Hello Todd,

I'm writing to you today to tell you about our great product! It's not Spam, I assure you. It's an e-mail system that integrates easily with PHP and it's built on top of PEAR::Mail. Please consider buying it.

Sincerely,

WEBD236-V1WW

Sending email from PHP

- Covered more in the textbook
  - Email address validation with RFC822

Accessing remote data

- Uses the cURL (Client URL) library
  - Configuration issues with XAMPP
    - Must turn on cURL by adding/uncommenting the following line to `C:\xampp\php\php.ini`

```
; enable cURL for web scraping
extension=php_curl.dll
```

    - Then restart Apache

Accessing remote data

- It is possible to pull data from any web site
  - But, it requires parsing the HTML
  - XML is easier to parse, but still cumbersome
  - JSON (JavaScript Object Notation) is designed for easy parsing.
    - Rapidly becoming the back-end of all web sites
    - "One page apps" like Google Mail send and receive JSON to communicate with the back end, and use JavaScript on the front end to display results.

Accessing remote data

- It is possible to pull data from any web site
  - But, it r
  - XML is
  - JSON (J
  easy pars
    - Rapidly be ing the back-end of all web sites
    - "One page apps" like Google Mail send and receive JSON to communicate with the back end, and use JavaScript on the front end to display results.

rsome

esigned for

AJAX (Asynchronous JavaScript and XML) has really become AJAJ (Asynchronous JavaScript and JSON).

Accessing remote data

- It is possible to pull data from any web site
  - But, it requires parsing the HTML
  - XML is easier to parse, but still cumbersome
  - JSON (JavaScript Object Notation) is designed for easy parsing
    - Rapidly
    - "One pag
      JSON to
      JavaScript on the front end to display results.

Our cURL example will interface with Twitter, extracting JSON data on trending topics.

Accessing remote data

- cURL, Twitter, and JSON

```
class Twitter {

    private static $curlOpts = array(
        CURLOPT_TIMEOUT        => 300,
        CURLOPT_CONNECTTIMEOUT => 60,
        CURLOPT_RETURNTRANSFER => true,
        CURLOPT_SSL_VERIFYHOST => false,
        CURLOPT_SSL_VERIFYPEER => false,
        CURLOPT_ENCODING       => 'gzip,deflate'
    );

    public function __construct() {
    }

    // ...continued...
```

Accessing remote data

- cURL, Twitter, and JSON

```
    public function getTrending($region = 1) {
        $curl = curl_init();
        curl_setopt_array($curl, self::$curlOpts);
        curl_setopt($curl, CURLOPT_URL,
            "https://api.twitter.com/1/trends/" . $region .
            ".json");
        $json = curl_exec($curl);
        $data = json_decode($json);
        return $data[0] -> trends;
    }

} // end Twitter class
```


Accessing remote data

- cURL, T

```
    public
        $cur
        curl
        curl


        $jso
        $dat
        retu
    }

} // end Twi
```


```
[{
    "trends" : [{
        "url" : "http:\/\/twitter.com\/search\/%23IfIMeetJustinBieber",
        "query" : "%23IfIMeetJustinBieber",
        "name" : "#IfIMeetJustinBieber",
        "events" : null,
        "promoted_content" : null
    }, {
        "url" : "http:\/\/twitter.com\/search\/%23YouKnowItsFriday",
        "query" : "%23YouKnowItsFriday",
        "name" : "#YouKnowItsFriday",
        "events" : null,
        "promoted_content" : null
    }, {
        "url" : "http:\/\/twitter.com\/search\/%23WeLoveKevinJonas",
        "query" : "%23WeLoveKevinJonas",
        "name" : "#WeLoveKevinJonas",
        "events" : null,
        "promoted_content" : null
    }, {
    // ...snip...
    }],
    "created_at" : "2012-03-28T17:37:59Z",
    "as_of" : "2012-03-28T17:38:26Z",
    "locations" : [{
        "name" : "Worldwide",
        "woeid" : 1
    }]
}]
```


Raw JSON returned from Twitter

UNIVERSITY

Ac

- cURL, Twit

```
    public func
        $curl =
        curl_set
        curl_set
            "htt
            ".js
        $json =
        $data =
        return $
    }

} // end Twitter
```

```
array(1) {
  [0]=>
  object(stdClass)#2 (4) {
    ["trends"]=>
    array(10) {
      [0]=>
      object(stdClass)#3 (5) {
        ["url"]=>
        string(48) "http://twitter.com/search/%23IfIMeetJustinBieber"
        ["query"]=>
        string(22) "%23IfIMeetJustinBieber"
        ["name"]=>
        string(20) "#IfIMeetJustinBieber"
        ["events"]=>
        NULL
        ["promoted_content"]=>
        NULL
      }
      [1]=>
      object(stdClass)#4 (5) {
        ["url"]=>
        string(45) "http://twitter.co
        ["query"]=>
        string(19) "%23YouKnowItsFriday"
        ["name"]=>
        string(17) "#YouKnowItsFriday"
        ["events"]=>
        NULL
        ["promoted_content"]=>
        NULL
      }
    // ...snip...
    }
}
```

JSON converted to PHP arrays/objects via `json_decode()`.

Accessing remote data

- Using the `Twitter` class

```
$twitter = new Twitter();
$trends = $twitter -> getTrending();
renderTemplate(
    'trending_template.inc',
    array(
        'trends' => $trends
    )
);
```

Accessing remote data

- Using the `Twitter` class

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html>
    <head><title>Twitter trending topics</title></head>
    <body>
        <h1>Twitter trending topics</h1>
        <ol>
            [[ foreach ($trends as $trend) : ]]
            <li><a href='{{$trend -> url}}'>
                {{$trend -> name}}</a></li>
            [[ endforeach; ]]
        </ol>
    </body>
</html>
```


Accessing remote data

- Using the `Twitter` class

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xht
<html>
    <head><title>Twitter
    <body>
        <h1>Twitter trend
        <ol>
            [[ foreach ($
            <li><a href='
                {{$trend
            [[ endforeach
        </ol>
    </body>
</html>
```

Twitter trending topics

1. #IfIMeetJustinBieber
2. #YouKnowItsFriday
3. #ReplaceATLSongsWithBoner
4. Un Gobierno DemocrÃ¡tico
5. We Love Kevin Jonas
6. Rise of the Guardians
7. MillÃ´r Fernandes
8. Rep. Bobby Rush
9. Today Is Gaga's Day
10. Sonisphere

Accessing remote data

- Considerations when building a web app
  - JSON is the data-interchange format of the web.
  - If you want to build something that others build on, then you'll need to expose services using JSON like Twitter does.
  - Page-refresh apps (like we've been building) are old technology. Ajax and single-page apps take less bandwidth, use REST more effectively, and are more responsive to users.

Show me the code!

- Email and Twitter classes are posted at http://cs.franklin.edu/~whittakt/WEBD236/.
  - Note that the email class needs to be configured with a working GMail address as a sender and a working address for a receiver.

Solution to HW 10

Next Week

- PHP Web Frameworks
- Final Exam Review

General Q & A

- Questions?
- Comments?
- Concerns?