Realistic practical applications with PHP

workshop

PHP in the lab

1. Initial setup - these steps only need to be done once:
Open terminal and set up public_html directory

```
$ umask 077
$ cd
$ chmod go+x .
$ mkdir public_html
$ chmod 755 public_html
```

2. Create file **test.php** with the following content and save it in public_html:

```
<?php
        phpinfo();
?>
```

3. You can now access **test.php** from the browser in the lab with the following url:

http://cscidb.csci.viu.ca/~USERNAME/test.php

Part I

ALIEN ABDUCTION REPORT

Application goal

We want to collect information from all people around the globe who were abducted by aliens

Question: can we do this without any server programming?

- Create folder aliens in your public_html folder
- Set folder permissions to 0755
- Put content of folder 01.01.aliens into aliens folder: index.html and style.css
- Access it in a browser with

http://cscidb.csci.viu.ca/~USERNAME/aliens

File named index is a default file served from a given directory

What is in HTML file: form

"mailto" is a protocol that allows form data to be delivered via email

```
<form method="post" action="mailto:owner@aliensabductedme.com">
  <p><label for="firstname">First name:</label>
    <input type="text" id="firstname" name="firstname" /></p>
  <p><label for="lastname">Last name:</label>
    <input type="text" id="lastname" name="lastname" /></p>
  <p><label for="email">What is your email address?</label>
    <input type="text" id="email" name="email" /></p>
  <p><label for="other">Anything else you want to add?</label>
    <textarea id="other" name="other"></textarea></p>
  <input type="submit" value="Report Abduction" name="submit" />
</form>
```

Edit this to your own e-mail address

The submit button tells the form to execute the form action.

Delivering the report

- Demo: test it link

- The HTML form doesn't know how to actually send an email message, so it delegates the task to the user's own email program.
- Submitting the form results in the form data getting emailed...sort of.
- The HTML form code is fine, but mailto isn't a good way to deliver form data.
- You need a way to take control of the delivery of the web form. More specifically, you need PHP to package the form data into a readable email message, and then make sure it gets sent automatically.

How to call PHP program on server

- A form element's action attribute is what connects a form to a PHP script, causing the script to run when the form is submitted.

```
<form action = "report.php" method = "post">
```

Change form action attribute to report.php

- When the user clicks the Report Abduction button in the form, the form action causes the *report.php* script to be run **on the server** to process the form data.

- From folder 01.02.aliens add report.php to your working aliens folder

PHP script report.php

```
<html>
 <head>
  <title>Aliens Abducted Me - Report an Abduction</title>
 </head>
 <body>
  <h2>Aliens Abducted Me - Report an Abduction</h2>
<?php
 $when_it_happened = $_POST['whenithappened'];
 $how_long = $_POST['howlong'];
 $alien_description = $_POST['description'];
 echo 'Thanks for submitting the form.<br />';
 echo 'You were abducted ' . $when_it_happened;
 echo ' and were gone for ' . $how_long . '<br />';
 echo 'Describe them: ' . $alien_description . '<br />';
?>
 </body>
</html>
```

Static HTML

PHP code: produces different HTML code every time somebody submits form data

Generating automatic e-mail in php script

```
$to = ‘user@gmail.com';
$subject = 'Aliens Abducted Me - Abduction Report';
$msg = "$name was abducted .
 $when_it_happened and was gone for $how_long.\n" .
 "Number of aliens: $how_many\n" .
 "Alien description: $alien_description\n" .
 "What they did: $what_they_did\n" .
 "Other comments: $other";
mail($to, $subject, $msg, 'From:' . $email);
```

link

Testing PHP script after uploading it to the server

- The end result of the PHP script is a pure HTML web page that was dynamically generated on the server.

PHP syntax: tags

```
<?php
  $when_it_happened = $_POST['whenithappened'];
  $how_long = $_POST['howlong'];
  $alien_description = $_POST['description'];
  echo 'Thanks for submitting the form.<br />';
  echo 'You were abducted ' . $when_it_happened;
  echo ' and were gone for ' . $how_long . '<br />';
  echo 'Describe them: ' . $alien_description . '<br />';
?>
```

PHP syntax: variables

Names are case-sensitive

```
<?php
  $when_it_happened = $_POST['whenithappened'];
  $how_long = $_POST['howlong'];
  $alien_description = $_POST['description'];
  echo 'Thanks for submitting the form.<br />';
  echo 'You were abducted ' . $when_it_happened;
  echo ' and were gone for ' . $how_long . '<br />';
  echo 'Describe them: ' . $alien_description . '<br />';
?>
```

- implicitly declared (weekly typed language)
- always start with $
- naming agreement: lower case letters with underscore
- after $ - underscore or letter, after may contain numbers

PHP syntax: statement

```
<?php
  $when_it_happened = $_POST['whenithappened'];
  $how_long = $_POST['howlong'];
  $alien_description = $_POST['description'];
  echo 'Thanks for submitting the form.<br />';
  echo 'You were abducted ' . $when_it_happened;
  echo ' and were gone for ' . $how_long . '<br />';
  echo 'Describe them: ' . $alien_description . '<br />';
?>
```

PHP syntax: form variables

```
<?php
  $when_it_happened = $_POST['whenithappened'];
  $how_long = $_POST['howlong'];
  $alien_description = $_POST['description'];
  echo 'Thanks for submitting the form.<br />';
  echo 'You were abducted ' . $when_it_happened;
  echo ' and were gone for ' . $how_long . '<br />';
  echo 'Describe them: ' . $alien_description . '<br />';
?>
```

Access form variables of the submitted page by **name** attribute of the input element

PHP syntax: string concatenation

```
<?php
  $when_it_happened = $_POST['whenithappened'];
  $how_long = $_POST['howlong'];
  $alien_description = $_POST['description'];
  echo 'Thanks for submitting the form.<br />';

  echo 'You were abducted ' . $when_it_happened;

  echo ' and were gone for ' . $how_long . '<br />';

  echo 'Describe them: ' . $alien_description . '<br />';
?>
```

PHP syntax: $_POST

```
<?php
  $when_it_happened = $_POST['whenithappened'];
  $how_long = $_POST['howlong'];
  $alien_description = $_POST['description'];
  echo 'Thanks for submitting the form.<br />';
  echo 'You were abducted ' . $when_it_happened;
  echo ' and were gone for ' . $how_long . '<br />';
  echo 'Describe them: ' . $alien_description . '<br />';
?>
```

- **$_POST** transports form data to your script

Back to aliens abduction

- Problem: too many e-mails
- We need a better way to store the alien abduction data: be able to get to the data when we want and keep data in a safe place other than an e-mail box

- What we need is a **database**, a fancy, ultra-organized electronic file cabinet.

Databases

- Databases are collection of files managed by a special program called a **database management system**
- You communicate with a database server in a language it can understand, which is (in most cases) **SQL**.

SQLite database in the lab

//To create (or access) database *aliens*

sqlite3 aliens;

//execute sql script or manually execute each command in it

.read aliens_abduction.sql

Creating SQLite database

- First, copy sql script aliens_abduction.sql from 01.03.aliens into your public_html/aliens folder

- To simplify things, we put all application-related files into the same directory

- In a terminal, cd to “aliens” folder, and create (and open) database *aliens*

```
sqlite3 aliens;
```

- Execute sql script which you put into this directory:

```
.read aliens_abduction.sql
```

Single-table database

```
CREATE TABLE  aliens_abduction  (
        abduction_id  INT PRIMARY KEY,
        first_name  VARCHAR(30),
        last_name  VARCHAR(30),
        when_it_happened  DATE,
        how_long  VARCHAR(30),
        how_many  VARCHAR(30),
        alien_description  VARCHAR(100),
        what_they_did  VARCHAR(100),
        other  VARCHAR(100),
        email  VARCHAR(50)
);
```

How to create an AUTOINCREMENT field in SQLite.

Short answer: A column declared INTEGER PRIMARY KEY will autoincrement.

Long answer:

If you declare a column of a table to be INTEGER PRIMARY KEY, then whenever you insert a NULL into that column of the table, the NULL is automatically converted into an integer which is one greater than the largest value of that column over all other rows in the table, or 1 if the table is empty.

Check that the table has been created

```
SELECT name FROM sqlite_master WHERE type='table‘;
```

//info about the table

```
pragma table_info(aliens_abduction);
```

SQL: INSERT data

INSERT INTO aliens_abduction (first_name, last_name, when_it_happened, how_long, how_many, alien_description,what_they_did, other, email)

VALUES ('Sally', 'Jones', '2008-10-12', '1 day', 'four', 'green with six tentacles', 'We just talked and played with a dog', 'no comments', 'sally@gregs-list.net');

SQL: view SELECTed data

```
SELECT * FROM aliens_abduction;
SELECT first_name FROM aliens_abduction;
```

How to exit SQLite shell

.quit

.q

Ctrl +D

Ctrl + C (Windows shell)

Let PHP handle the insertion

1. Connect to the database
2. Insert data with PHP script
3. Close connection

- Copy (override) file report.php from 01.03.aliens to your **public_html/aliens** working directory
- Copy also report.html to the same directory (for future use)

Connecting to the database server with PHP

```
// Create (connect to) SQLite database in file
$file_db = new PDO('sqlite:aliens');

// Set errormode to exceptions
$file_db->setAttribute(PDO::ATTR_ERRMODE,
                       PDO::ERRMODE_EXCEPTION);
```

Preparing INSERT statement

```
// Prepare INSERT statement with data collected from the user form
$insert = "INSERT INTO aliens_abduction " .
          "(first_name, last_name, when_it_happened, how_long) " .
" VALUES ('{$_POST['firstname']}', '{$_POST['lastname']}', " .
          " '{$when_it_happened}', '{$how_long}');";
```

Executing INSERT

```
$stmt = $file_db->prepare($insert);
$stmt->execute();
```

Close connection

```
// Close file db connection
$file_db = null;
```

Trap errors in try/catch block

```
try {
        //all the previous code goes here
}  catch(PDOException $e) {
        // Print PDOException message
        echo $e->getMessage();
}
```

New report demo

link

Display data in HTML page using PHP

- Open connection
- Select data into $result, which is a special PDO object, that reflects the underlying table of data
- Populate data using PHP embedding into HTML
- Close connection

- Remove ~~index.html~~
- Copy index.php from 01.03.aliens to your working aliens folder
- Now default file served from this folder will be index.php, not index.html

- What’s in index.php?

Displaying the results from the database in a dynamically generated html page

Open connection

```
// Connect to SQLite database in file
$file_db = new PDO('sqlite:aliens');
```

SELECTing the data

```
//prepare SELECT statement to SQLite3 file_db

$query = "SELECT * FROM aliens_abduction
ORDER BY when_it_happened DESC LIMIT 5";

//execute query
$result = $file_db->query($query);
```

Writing data into HTML table

```
<table>
<?php
foreach($result as $row) {
// Display each row as a table row
        echo '<tr class="heading"><td colspan="2">' .
        $row['when_it_happened'] . ' : ' .
                $row['first_name'] . ' ' .
                $row['last_name'] . '</td></tr>';
        echo '<tr><td><strong>Abducted for:</strong><br /> ' .
                $row['how_long'];
        echo '</td><td><strong>Alien description:</strong><br /> ' .
                $row['alien_description'] . '</td></tr>';
}
?>
</table>
```

Close connection

```
// Close file db connection
$file_db = null;
```

Complete code demo

link

Test in your directory:

http://cscidb.csci.viu.ca/~USERNAME/aliens

Our first realistic PHP application is now complete

Possible extensions:

- Admin page (new): to delete some reports
- Index page: search functionality to the index page, or the Report page: possibility to upload images or videos

Part II

USER EXPERIENCE SURVEY

Application goal

- Provide web interface to collect user experience data for client-side board games developed by CSCI 311 team

- In public_html folder create a new directory: “games”
- First, unzip file games.zip and copy all folders with game submissions into games folder
- Then, from folder 02.01.games copy index.html, showgame.php, and css.css into a games folder
- Now you can access the application from your browser:

http://cscidb.csci.viu.ca/~USERNAME/games

User form for game selection

index.html at http://owl.csci.viu.ca/~barskym/games/index0.html

```
<form method="GET" action="showgame.php">
                <select name="author">
                        <option>ABDULLAH-GURLEEN</option>
                        <option>ANDREW</option>
                        <option>BRANDON</option>
                        …
                        <option>MATTHIAS</option>
                        <option>ZAC-LOCHLIN</option>
                        <option>ZIJIAN</option>
                </select>
                <input type="submit" value="Show"/>
 </form>
```

Routing pages dynamically with showgame.php

index.html at http://owl.csci.viu.ca/~barskym/games/index0.html

```
<?php
header('Location: http://owl.csci.viu.ca/~barskym/games/'.
                                                $_GET['author']);
die;
?>
```

HTTP response: header

- Using PHP, you can control the headers sent by the server to the browser
- The *header()* function immediately sends a header from the server to the browser and must be called before any actual content is sent to the browser. This is a very strict requirement—if even a single character or space is put before <?php tag, the browser will reject request with an error.
- Parameters consist of key-value pairs in the following format:

```
header('key: value');
```

Setting response content-type

```
header("Content-type: type/subtype");
```

```
<?php
  header('Content-Type: text/plain');
  echo 'This <strong>text</strong> won't actually be bold.';
?>
```

- by default, a PHP script's output is assumed to be HTML
- use the *header* function to specify non-HTML output
- repeat: must appear before any other output generated by the script

header example

```
<?php
  header('Refresh: 5;
          url=http://www.mysite.net/about.php');
  echo 'In 5 seconds you'll be taken to the About page.';
?>
```

- this header is called a **refresh header** since it refreshes a page after a period of time has elapsed.
- you often see the URL in such headers reference the current page so that it refreshes itself.

Content ("MIME") types

MIME type	related file extension
text/plain	.txt
text/html	.html, .htm, ...
text/css	.css
text/javascript	.js
text/xml	.xml
image/gif	.gif
image/jpeg	.jpg, .jpeg
video/quicktime	.mov
application/octet-stream	.exe
application/json	.json

If you are serving .html file (not generated by php), changing its content-type does not have an effect: browser knows how to interpret html files

Survey form: list item for each game: 14 times

```
<ul>
  <li>
    <p><strong>Game of 15 </strong>: <a href="http://server.com/AUTHOR_FOLDER">
                                                    link</a>.
      <select name="AUTHOR_FOLDER">
        <option value="0">--------</option>
        <option value="1">1st place</option>
        <option value="2">2nd place</option>
        …
      </select>
    </p>
    <p>   <textarea name="AUTHOR_FOLDERcomments" cols="25" rows="5">
            Your comments here
          </textarea>
    </p>
  </li>
```

Survey form: list item for each game: the only difference

```
<ul>
  <li>
    <p><strong>Game of 15 </strong>: <a href="http://server.com/AUTHOR_FOLDER">
                                                                    link</a>.
      <select name="AUTHOR_FOLDER">
        <option value="0">--------</option>
        <option value="1">1st place</option>
        <option value="2">2nd place</option>
        …
      </select>
    </p>
    <p>   <textarea name="AUTHOR_FOLDERcomments" cols="25" rows="5">
            Your comments here
          </textarea>
    </p>
  </li>
```

Dynamically generating survey form with PHP

- Open folder 02.02.games and copy index.php into games directory
- Delete ~~index.html~~ to be able to run index.php instead
- Copy files **games.txt** and **names.txt** into the same directory. These files contain names of the games (1 name per line) and name of developers (1 name per line)

First read two files

```
<?php
        $names = file("names.txt", FILE_IGNORE_NEW_LINES);
        $games = file("games.txt", FILE_IGNORE_NEW_LINES);
?>
```

Based on these arrays, dynamically generate 14 list items

```
<ul>
<?php
        for ($i = 0; $i < count($names); $i++)  {
                $name = $names[$i];
                $game = $games[$i];
?>
  <li>
        <strong><?= $game ?> </strong>:
                <a href="http://server.com/games/<?= $name ?>">link</a>.
        <select name="<?= $name ?>">
          <option value="0">--------</option>
          <option value="1">1st place</option> ...
        </select>
  </li>
<?php
        }
?>
</ul>
```

User experience survey **form**

```
<form method="POST" action="survey.php">
        …
        <input type="submit" value="Submit opinion"/>
</form>
```

What is in survey.php?

- From folder 02.03.games copy survey.php into games directory

Generating a single-table database in php code

```
// Create table surveyWP2013
$file_db->exec("CREATE TABLE IF NOT EXISTS surveyWP2013 (
                    id INTEGER PRIMARY KEY,
                    reviewer TEXT,
                    comments TEXT,
                    developer TEXT,
                    place INTEGER
                )");
```

Loop through names and record results into a table

```
$names = file("names.txt", FILE_IGNORE_NEW_LINES);
foreach ($names as $name)  {
        $comments=$_POST[$name.'comments'];
        $developer=$name;
        $place=$_POST[$name];

        $file_db->query("INSERT INTO surveyWP2013 ".
        "VALUES (NULL,'".$_POST['reviewer']."','".$comments."','"
        .$developer."',".$place.")");
}
```

Courtesy: display submitted ranks to the user

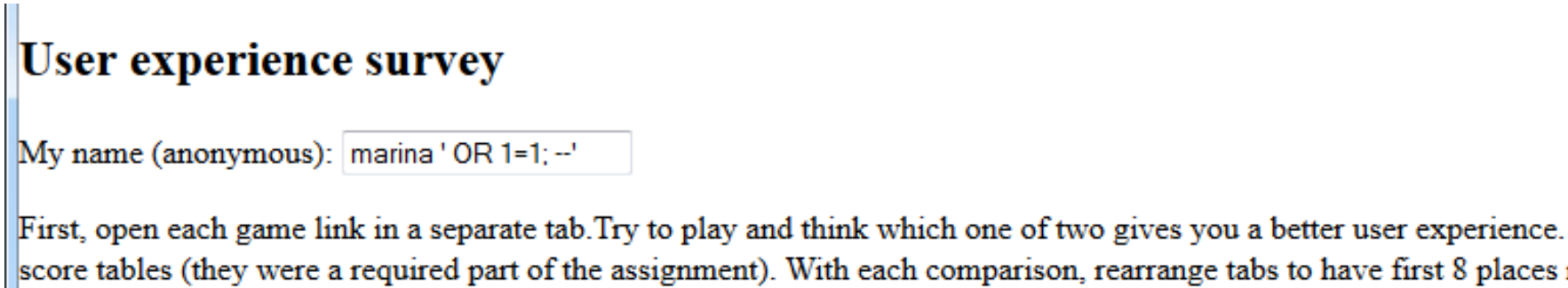
```
$reviewer = $_POST['reviewer'];
$results = $file_db->query("SELECT * FROM surveyWP2013 WHERE reviewer='".$reviewer."'");

foreach ($results as $row) {
    echo ' For: '.$row['developer'].
            '  place:'.$row['place'].
            ' comment:'.$row['comments'].'<br>';
}
```

User can see results submitted by him only

- Generate several reports by different users: X, Y, Z
- Next time, type in the reviewer field:

yourName' OR 1=1; --'

- What do we see? How did it happen?



- Link to test:

link

SQL injection attack

- The idea is to convince the application to run a different SQL query.
- If the application is creating SQL strings naively on the fly and then running them, it's straightforward to create some real surprises.

- More examples: link

Viewing secret data

("**SELECT * FROM surveyWP2013 WHERE reviewer='**".$reviewer."**'**");

$reviewer = "marina' OR 1=1; --'"

Produces:

SELECT * FROM surveyWP2013 WHERE reviewer='marina' OR 1=1; --'

Commented out the rest of PHP code to avoid error message

Overriding survey data

"INSERT INTO surveyWP2013 VALUES (NULL,'".$_POST['reviewer']."','".$comments."','".$developer."',".$place.") "

- +

$comments = "the best','marina',1); --')";

- gives:

INSERT INTO surveyWP2013 VALUES (NULL,'marina','the best','marina',1)

SQL injection examples

```
SELECT email, passwd, login_id, full_name
FROM members
WHERE email = 'x' OR full_name LIKE '%Bob%'; --'
```

```
SELECT email, passwd, login_id, full_name
FROM members
WHERE email = 'x'; DROP TABLE members; --'
```

```
SELECT email, passwd, login_id, full_name
FROM members WHERE email = 'x';
INSERT INTO members
VALUES ('sneeky@unixwiz.net','hello','world','New memeber');--'
```

Better (more secure) php-SQL code

- From folder 02.04. games copy index.php and survey.php – the final versions

- Demo: link

Declare user-filled parameters as SQL variables

```
$insert = "INSERT INTO surveyWP2013
(reviewer, developer, place, comments)
VALUES (:reviewer, :developer, :place, :comments)";
$stmt = $file_db->prepare($insert);
```

Bind SQL parameters to PHP variables

```
// Bind parameters to statement variables
$stmt->bindParam(':reviewer', $reviewer);
$stmt->bindParam(':developer', $developer);
$stmt->bindParam(':place', $place);
$stmt->bindParam(':comments', $comment);
```

And only then execute:

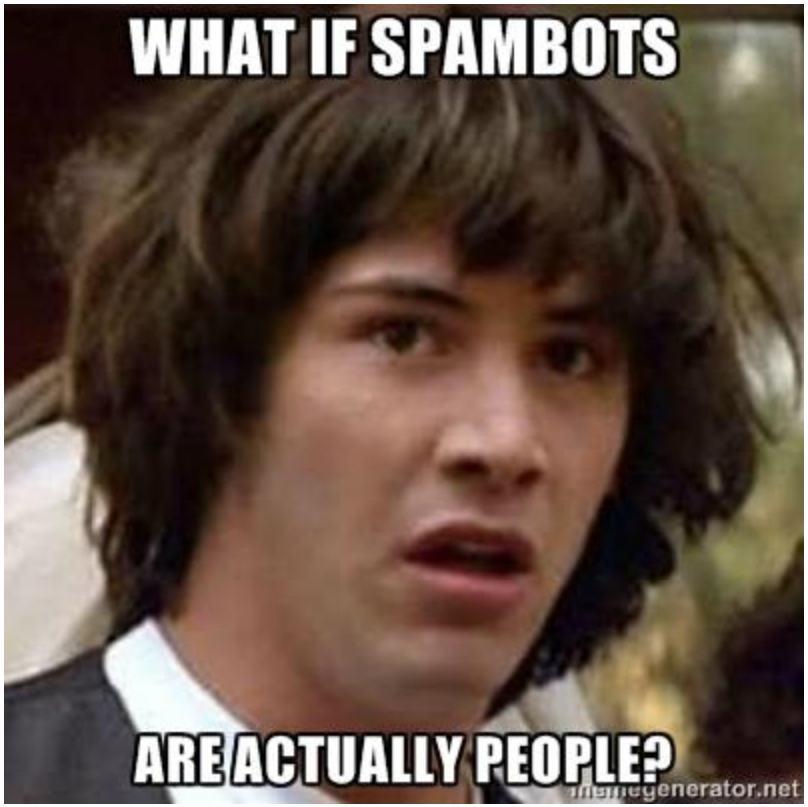
```
$stmt->execute();
```

Possibility for new attacks

- The problem with our survey form is that it does nothing to prevent **automated** submissions, meaning that any reasonably crafty bot programmer can create a bot that repeatedly fills the form with preferred data and submits it
- This will soon fill database with bogus ranks and crush VIU server

- The form needs a verification field that must be entered successfully in order for the score to submit.
- And the specific verification of this field needs to be something that is easy for a real **human** but difficult for a **machine**.

Robot attack



- **What questions you would ask to distinguish between a real human and the artificial brain of a robot?**

Are you a robot? ◯ Yes ◯ No

What was Elvis' favorite food? [ ]

Retinal scan: [Look into your web cam and click]

Enter the letters displayed: [ ] kdyqmc

What is the result of 7 + 5? [ ]

What kind of animal is this? [ ]

Enter the letters displayed: [ ] 

Thumbprint scan: [Press your thumb down and click]

CAPTCHA test

Completely

Automated

Public

Turing test to tell

Computers and

Humans

Apart

Generating image on the server with PHP

- Code is in 02.04.games captcha.php

- Graphic packages are not installed in the lab:

link

rand()

- This built-in function returns a random integer number, either within a specified range or between 0 and the built-in constant RAND_MAX (server dependent).
- To obtain a random number within a certain range, just pass the lower and upper limits of the range as two arguments to rand()

rand(2,20)

chr()

- This built-in function converts a number to its ASCII character equivalent.

  As an example, the number 97 is the ASCII code for the lowercase letter 'a':

  chr(97) returns the single character 'a'.

Do actual survey

- Please take time to share your user experience
- Top 8 assignments are for a real bonus

- You can submit your opinion only once

http://owl.csci.viu.ca/~barskym/survey/

Part III

WEB SERVICES WITH PHP

Web services

web service: software functionality that can be invoked through the internet using common protocols

- like a remote function(s) you can call by contacting a program on a web server
- many web services accept parameters and produce results
- service's output can be text, HTML, XML, JSON or other content types

RESTful web services

- **Re**presentational **S**tate **T**ransfer :

  presented with a network of Web pages (a virtual state-machine), the user progresses through an application by selecting links (state transitions), resulting in the next page (representing the next state of the application) being transferred to the user and rendered for his use.
- **REST** is initially described in the context of HTTP.
- RESTful applications maximize the use of the existing, well-defined HTTP interface, its built-in capabilities, and minimize the addition of new application-specific features on top of it.

Central principles of REST

- The existence of *resources* (sources of specific information), each of which is referenced with a global identifier (e.g., a URI in HTTP).
- In order to manipulate these resources, *components* of the network (user agents and origin servers) communicate via a standardized interface (e.g., HTTP) and exchange *representations* of these resources (the actual documents conveying the information).
- In addition to URIs, Internet media types, request and response codes HTTP has a rich vocabulary of operations:

  GET POST PUT DELETE etc.

  REST uses these operations and existing features of the well-defined HTTP protocol.

REST vs. SOAP

- SOAP RPC encourages each application designer to define new, application- specific operations that supplant HTTP operations:

  getUsers()

  getNewUsersSince(date SinceDate)

  savePurchaseOrder(string CustomerID, string PurchaseOrderID) etc.

- This additive, "re-invention of the wheel" vocabulary — defined on the spot and subject to individual judgment or preference — disregards many of HTTP's existing capabilities, such as authentication, caching, and content-type negotiation.
- The advantage of SOAP over REST comes from this same limitation: Since it does not take advantage of HTTP conventions, SOAP works equally well over raw TCP, named pipes, message queues etc.

Web services with PHP

- As we already know, RESTful web services can be written in PHP and contacted by the browser through Ajax or JSONP

- The content provided by the service can be of many different types:

Content ("MIME") types

Example: Exponent web service

- Write a web service that accepts a base and exponent and outputs base raised to the exponent power.
- For example, the following "query" should output 81 :

http://example.com/exponent.php?base=3&exponent=4

- Solution:

```
header("Content-type: text/plain");
$base = $_GET["base"];
$exp = $_GET["exponent"];
$result = pow($base, $exp);
print $result;
```

Lab 8. Web services with PHP

- Create a small personal single-table database of books. Each book in a database has the following attributes: title, author, year, edition, category
- Write a PHP RESTful web service that accepts such parameters as book title, or book category and displays an HTML page with the corresponding list of books from your custom database, depending on the query
- Write a consumer application (web page) that will use your web service to perform a search from an HTML form

Example: book sales web service (from JSONP lecture)

```
<?php
$filename = "../saleswithdates.json";
$data = file_get_contents($filename);// json string

if(array_key_exists('callback', $_GET)){

   header('Content-Type: text/javascript; charset=utf8');

   $callback = $_GET['callback'];
   echo $callback.'('.$data.');';

}else{
   // normal JSON string
   header('Content-Type: application/json; charset=utf8');
   echo $data;
}
?>
```

Example of a PHP-based project

Perfect mismatch: link

Source code: link