COMP519 Web Programming
Lecture 29: REST (Part 3)
Handouts

Ullrich Hustadt

Department of Computer Science
School of Electrical Engineering, Electronics, and Computer Science
University of Liverpool

Contents

REST.php Motivation

- In the last lecture, we have decided that we will deal with HTTP requests to a web service

```
method https://api.liv.ac.uk/v1/resource?query
```

  by rewriting them to

```
method https://api.liv.ac.uk/v1/REST.php?resource=resource
    ↪&query
```

  and we have devised a rewrite rule to achieve that
- On the departmental systems,
  - the file REST.php will be in a directory ~*sgxyz*/public_html/v1
  - the HTTP requests take the form

```
method https://student.csc.liv.ac.uk/~sgxyz/v1/resource?
    ↪query
```

    and should be rewritten to

```
method https://student.csc.liv.ac.uk/~sgxyz/v1/REST.php?
    ↪resource=resource&query
```

REST.php Motivation

- We now have to implement REST.php
- To simply the implementation we ignore the *query* parameter
- REST.php has to 'translate' combinations of *method* and *resource* into PHP function calls
- Our implementation will make a decision on which PHP function to call solely based on *method* but this is design choice
- In a lot of situations more fine-grained decision making will be better

Useful PHP Functions

`Exception([string msg = ""[, int code = 0]])`

Creates an exception with exception message *msg* and exception code *cd*

```
throw new Exception('Method Not Supported', 405);
```

`set_exception_handler(exceptionHandler)`

- Sets the default exception handler if an exception is not caught within a try/catch block
- *exceptionHandler* should be a function that accepts an exception as an argument
- Execution will stop after the call of *exceptionHandler* is completed

```
function exceptionHandler($excpt) {
  echo "Uncaught exception: ", $excpt->getMessage(), "\n";
}
set_exception_handler('exceptionHandler');
throw new Exception('Spurious Exception');
echo "This code is not executed\n";
```

```
Uncaught exception: Spurious Exception
```

Useful PHP Functions

php://input

A read-only stream that allows to read raw data from the request body

```
$params = json_decode(file_get_contents('php://input'),
                      true);
```

Assuming that the request body contains JSON encoded data, read the whole of php://input and turn it into an associative array

explode(string *delimiter*, string *str*[, int *limit*])

Returns an array of strings, with a maximum of *limit* elements, each of which is a substring of *str* formed by splitting it on boundaries formed by the string *delimiter*

```
print_r(explode('/','this/is/a/filepath'));
```

```
Array
( [0] => 'this'  [1] => 'is'  [2] => 'a'  [3] => 'filepath')
```

Useful PHP Functions

```
header(string hStr[, bool repl = TRUE[, int httpRspCd]])
```

- Send a raw HTTP header including *hStr* and HTTP response code *httpRspCd*
- Replace a previous similar header if *repl* is TRUE, otherwise add

```
header('Location: http://www.example.com/');
```

Add a location header to the response

Together with a response code 302 (REDIRECT) would tell a browers to visit the URL indicated

```
header('Content-Type: application/json');
```

Add a header entry that indicates the request/response body contains JSON encoded data

Useful PHP Functions

`http_response_code([int` *`httpRspCd`*`])`

- Returns the previous HTTP response code and sets it to *`httpRspCd`* if that argument is provided
- Also sets the HTTP response text to a reason phrase provided
- *`httpRspCd`* is a standard HTTP response code

```
http_response_code(201)
```

Sets the HTTP response code to 201 (CREATED)

Destructuring Assignments

PHP has several ways in which array elements can be assigned to several variables in a single assignment

```
$ar1 = [       2,         3,         5];

list($x,$y,$z) = $ar1;                          // PHP 5.x or later
echo "\$x = $x  \$y = $y  \$z = $z\n";
$x = 2  $y = 3  $z = 5
[$u,$v,$w] = $ar1;                              // PHP 7.x or later
echo "\$u = $u  \$v = $v  \$w = $w\n";
$u = 2  $v = 3  $w = 5
[0 => $a ] = $ar1;
[ , $b, $c ] = $ar1;
echo "\$a = $a  \$b = $b  \$c = $c\n";
$a = 2  $b = 3  $c = 5
$ar2 = ['a' => 2, 'b' => 3, 'c' => 5];

['a' => $x, 'b' => $y, 'c' => $z] = $ar2; // PHP 7.x
echo "\$x = $x  \$y = $y  \$z = $z\n";
$x = 2  $y = 3  $z = 5
```

REST.php: Outline (1)

```php
<?php
require_once('Database.php');
require_once('Model.php');

set_exception_handler(function ($e) {
  $code = $e->getCode() ?: 400;
  header("Content-Type: application/json", FALSE, $code);
  echo json_encode(["error" => $e->getMessage()]);
  exit;
});

// Open database connection
$db        = new Database();
// Retrieve inputs
$method    = $_SERVER['REQUEST_METHOD'];
$resource  = explode('/', $_REQUEST['resource']);
$inputData = json_decode(file_get_contents('php://input'),
                         TRUE);
```

REST.php: Outline (2)

```
switch($method) {
  case 'GET':
    [$data,$status] = readData($db,$resource);
    break;
  case 'PUT':
  case 'POST':
    [$data,$status] = createData($db,$method,$resource,
                                 $inputData);
    break;
  case 'DELETE':
    [$data,$status] = deleteData($db,$resource);
    break;
  default:
    throw new Exception('Method Not Supported', 405);
}
header("Content-Type: application/json",TRUE,$status);
echo $data;
?>
```

REST.php: POST Requests

- We focus on POST HTTP requests that attempt to create a new student using our web service
- We allow such POST requests with minimum information consisting of first name, surname and programme

```
POST /~sgxyz/v1/students
Host: student.csc.liv.ac.uk

{"sname":"Clay","fname":"Cia","prog":"CSMS"}
```

or more complete information with one or two addresses

```
POST /~sgxyz/v1/students
Host: student.csc.liv.ac.uk

{"sname":"Ady", "fname":"Ada", "prog":"CSMS",
 "tAddr":{"streetHN":"1 Abby Road", "city":"Liverpool",
          "postCode":"L69 9AA", "country":"UK"},
 "pAddr":{"streetHN":"9 Mort Street", "city":"Wigan",
          "postCode":"WN2 4TU", "country":"UK"}}
```

- We assume that the web service will generate the student id

Database.php: Database Class

```
class Database {
  private $host     = "studdb.csc.liv.ac.uk";
  private $user     = "sgfsurn";
  private $passwd   = "-------";
  private $database = "sgfsurn";
  public  $conn;

  public function __construct() {
    // we use the same options as usual
    $opt = array( ... );
    $this->conn = null;
    try {
      $this->conn = new PDO('mysql:host=' . $this->host . ';
          ↪dbname=' . $this->database . ';charset=utf8mb4',
          ↪$this->user, $this->passwd, $opt);
    } catch (PDOException $e) {
      // If we can't get a database connection, we return
      // 503 Service Unavailable
      throw new Exception($e->getMessage(),503);
} } }
```

Database.php: Database (1)

```
CREATE TABLE `students` (
`id`        int(10)        NOT NULL,
`sname`     varchar(50)    DEFAULT NULL,
`fname`     varchar(100)   DEFAULT NULL,
`prog`      char(4)        DEFAULT NULL,

`tAddrId`   mediumint(9)   DEFAULT NULL,
`pAddrId`   mediumint(9)   DEFAULT NULL,

PRIMARY KEY (`id`),

KEY `termTime` (`tAddrId`),
KEY `permanent` (`pAddrId`),
CONSTRAINT `fk1` FOREIGN KEY (`tAddrId`)
                 REFERENCES   `addresses` (`id`),
CONSTRAINT `fk2` FOREIGN KEY (`pAddrId`)
                 REFERENCES   `addresses` (`id`)
) ENGINE=InnoDB;
```

Database.php: Database (2)

```
CREATE TABLE `addresses` (
`id`         mediumint(9) NOT NULL AUTO_INCREMENT,
`streetHN`   varchar(50)  DEFAULT NULL,
`city`       varchar(50)  DEFAULT NULL,
`postCode`   varchar(8)   DEFAULT NULL,
`country`    varchar(50)  DEFAULT NULL,

`studentId` int(10)       NOT NULL,

PRIMARY KEY (`id`),

CONSTRAINT `fk1` FOREIGN KEY (`studentId`)
                 REFERENCES   `students` (`id`)
                 ON DELETE CASCADE
) ENGINE=InnoDB;
```

Model.php: Address Class (1)

```
class Address {
  // Private properties do not appear in the JSON encoding
  // of an object.
  private $conn;

  private static $table = 'addresses';

  private $id, $studentId;
  private $parts = ['streetHN','city','postCode','country'];

  // Address properties
  public $streetHN, $city, $postCode, $country;

  // $_links will provide HATEOAS links
  public $_links;
```

Model.php: Address Class (2)

```
// The constructor only sets the database connection and
// the student id of the student to whom the address
// belongs.
public function __construct($db,$sid) {
  $this->conn      = $db->conn;
  $this->studentId = $sid;
}

// set() populates the public properties of an address,
// values can be provided as an array or another object.
public function set($source) {
  if (is_object($source))
    $source = (array)$source;
  foreach ($source as $key=>$value)
    if (in_array($key,$this->parts))
      $this->$key = $value;
    else
      throw new Exception("$key not an attribute of
          ↪address",400);
}
```

Model.php: Address Class (3)

```
// HATEOS links are not stored in the database, but
// generated using student Id $sid and address type
// $aType (one of 'tAddr' or 'pAddr').
// $this->_links is an array of objects. As PHP does
// not have literal objects, we cast arrays to create
// those objects.
public function setLinks($sid,$aType) {
  $this->_links     =
    [(object)['href' => "/students/$sid/addresses/$aType",
              'method' => 'GET',    'rel' => 'self'],
     (object)['href' => "/students/$sid/addresses/$aType",
              'method' => 'PATCH',  'rel' => 'edit'],
     (object)['href' => "/students/$sid/addresses/$aType",
              'method' => 'DELETE', 'rel' => 'delete']];
}
```

Model.php: Address Class (4)

```
// store() stores an address in the database.
// In the process a unique id is generated and returned.
public function store() {
  // An address belongs to a particular student
  $query = 'INSERT INTO ' . self::$table .
           '(studentId, streetHN, city, postCode,
             country) VALUES (?,?,?,?,?)');
  $stmt = $this->conn->prepare($query);
  $stmt->execute(array($this->studentId,$this->streetHN,
                       $this->city,$this->postCode,
                       $this->country));
  $this->id = $this->conn->lastInsertId();
  return $this->id;
}
```

Model.php: Address Class (5)

```
// read() retrieves an address from the database.
// $this->id must have been set when read() is called.
public function read($aType) {
  $query = 'SELECT * FROM ' . self::$table .
           ' WHERE id=:id';

  // Prepare and execute statement.
  $stmt  = $this->conn->prepare($query);
  $stmt->execute(array($this->id));
  // Fetch the single row that the query returns
  $row   = $stmt->fetch();
  // Transfer database data into properties.
  foreach($row as $key=>$value)
    $this->$key = $value;
  // Set HATEOS links
  $this->setLinks($this->studentId,$aType);
}
```

Model.php: Address Class (6)

```
// After we created a new Address object and filled its
// properties with user-provided values, none of the
// properties should still have a NULL value
// (though empty strings are allowed).
public function validate() {
  foreach ($this->parts as $key)
    if (is_null($this->$key))
      return FALSE;
  return TRUE;
}

// __toString() is called whenever we need a string
// representation of an Address object. We use its
// JSON representation for that purpose.
public function __toString() {
  return json_encode($this,
            JSON_UNESCAPED_UNICODE|JSON_UNESCAPED_SLASHES);
}
```

Model.php: Student Class (1)

```
class Student {
  private $conn;
  private static $table = 'students';
  private $parts= ['studentId','sname','fname','prog'];

  // Student Properties
  public  $studentId, $sname, $fname, $prog;

  // $_links will provide HATEOAS links and a link
  // to addresses
  public $_links;

  // $tAddrId: Unique id of the term time address
  // $pAddrId: Unique id of the permanent address
  // These are private so that they do not occur
  // in the JSON encoding of a Student object
  private $tAddrId, $pAddrId;

  // The constructor only sets the database connection
  public function __construct($db) { $this->conn = $db->conn
}
```

Model.php: Student Class (2)

```
// set() populates the public properties of a student,
// values can be provided as an array or another object.
public function set($source) {
  if (is_object($source))
    $source = (array)$source;
  foreach ($source as $key=>$value)
    if (in_array($key,$this->parts))
      $this->$key = $value;
    else
      throw new Exception("$key not an attribute of
          ↪student",400);
}

// We need a way to set the private properties
public function setPrivate($key,$value) {
  $this->$key = $value;
}
```

Model.php: Student Class (3)

```
// HATEOS links are not stored in the database, but
// generated using student Id $sid .
// $this->_links is an array of objects.
public function setLinks($sid = NULL) {
  if ($this->studentId)
    $sid = $this->studentId;
  $this->_links     =
    [(object)['href' => "/students/$sid/addresses",
              'method' => 'GET',    'rel' => 'addresses'],
     (object)['href' => "/students/$sid",
              'method' => 'GET',    'rel' => 'self'],
     (object)['href' => "/students/$sid",
              'method' => 'PATCH',  'rel' => 'edit'],
     (object)['href' => "/students/$sid",
              'method' => 'DELETE', 'rel' => 'delete']];
}
```

Model.php: Student Class (4)

```
// store() stores a student in the database
public function store() {
  $query = 'INSERT INTO ' . self::$table .
           '(studentId, sname, fname, prog, tAddrId,
             pAddrId) VALUES (?,?,?,?,?,?)';

  // Prepare statement
  $stmt = $this->conn->prepare($query);
  $stmt->execute(array($this->studentID,$this->sname,
                       $this->fname,$this->prog,
                       $this->tAddrId,$this->pAddrId));
  return $this->studentId;
}

// __toString() is called whenever we need a string
// representation of an Student object.
public function __toString() {
  return json_encode($this,
           JSON_UNESCAPED_UNICODE|JSON_UNESCAPED_SLASHES);
}
```

Model.php: Student Class (5)

```
// Class function that generates a new student ID
// If there already students in the database, take their
// highest studentID plus 1, otherwise use 2019000001
public static function generateID() {
  $maxIdArr = $this->conn->query("SELECT max(id) from
      ↪students")->fetch(PDO::FETCH_NUM);
  $newId    = min($maxIdArr[0]+1,2019000001);
  return $newId;
}

// After we created a new Student object and filled its
// properties with user-provided values, none of the
// properties should still have a NULL value
// (though empty strings are allowed).
public function validate() {
  foreach ($this->parts as $key)
    if (is_null($this->$key))
      return FALSE;
  return TRUE;
}
```

Model.php: Student Class (6)

```
// read() retrieves a student from the database.
// $this->studentId must have been set when read() is
// called.
public function read() {
  $query = 'SELECT * FROM ' . self::$table .
           ' WHERE studentId=?';

  // Prepare and execute statement
  $stmt  = $this->conn->prepare($query);
  $stmt->execute(array($this->studentId));
  // Fetch the single row that the query returns
  $row   = $stmt->fetch();
  // Transfer database data into properties.
  foreach($row as $key=>$value)
    $this->$key = $value;
  // Set HATEOS links
  $this->setLinks();
}
```

Model.php: createData Function (1)

```
function createData($db,$method,$resource,$data) {
  if (($method == 'POST')  &&
      (count($resource) == 1) &&
      ($resource[0] = 'students')) {
    return createStudent($db,$data);
  } elseif (($method == 'POST')  &&
            (count($resource) == 2) &&
            ($resource[0] = 'students')) {
    if (preg_match('/^\d{10}$/',$resource[1]))
      return createStudentWS($db,$resource[1],$data);
    else
      throw new Exception('Not a valid student Id', 400)
  } else {
    throw new Exception('Method Not Supported', 405);
  }
}
```

Model.php: createStudent Function (1)

```
function createStudent($db,&$data) {
  // We need to insert up to three database entries
  // Either all insertions should succeed or all should fail
  $db->conn->beginTransaction();

  $data['studentId'] = Student::generateId($db);
  $std1 = new Student($db->conn);

  // If the JSON data contains a term time address, then
  // try to create a corrsponding Address object, store it
  // in the database, remember its primary key value
  if (array_key_exists('tAddr',$data))
    $std1->setPrivate('tAddrId',
              createAddress($db,$data,'tAddr'));

  // Do the same for a permanent address
  if (array_key_exists('pAddr',$data))
    $std1->setPrivate('pAddrId',
              createAddress($db,$data,'pAddr'));
```

Model.php: createStudent Function (2)

```
  // Use the JSON data to set the public properties of
  // the Student object
  $std1->set($data);

  if ($std1->validate()) {
    // If sufficient data was provided, store the Student
    // object in the database
    $std1->store();
    $db->conn->commit();
    $std1->setLinks();
    // Return the student object and response code 201
    return [$std1,201];
  } else {
    // If insufficient data was provided, roll everything
    // back and create an error HTTP respone
    $db->conn->rollback();
    throw new Exception("Student data incomplete",400);
} }
```

Model.php: createAddress Function

```
// We want to use the createAddress function in two scenarios:
// - $data is student data including address information and
//   student Id
// - $data is just address data and a student Id is provided
//   as a separate argument
function createAddress($db,&$data,$aType,$sid = NULL) {
  if (array_key_exists('studentId',$data))
    $sid = $data['studentId'];
  if (array_key_exists($aType,$data))
    $aData = $data[$aType];
  else
    $aData = $data;
  $addr = new Address($db->conn,$sid);
  $addr->set($aData);
  if ($addr->validate())
    $addrId = $addr->store();
  else
    throw new Exception("Address $addrType incomplete",400);
  if (array_key_exists($aType,$data))
    unset($data[$aType]);
  return $addrId;
}
```

Model.php: To-do

- To complete the implementation of our web service we need to
  - extend `createData` with further cases that deal with the creating of addresses
  - we need to implement `readData` using the `read` methods we have already defined in the `Address` and `Student` classes
  - we need to implement `deleteData`
- We need to investigate how we can secure our web service
  - Cookie
  - Token (in query string or `Authorization` header)

Revision and Further Reading

- Digamber Rawat: Create Simple PHP 7|8 CRUD REST API with MySQL & PHP PDO. positronx.io, 27 Nov 2020. https://www.positronx.io/create-simple-php-crud-rest-api-with-mysql-php-pdo/, [accessed 25 Dec 2020]
- Mike Dalisay: How To Create A Simple REST API in PHP? Step By Step Guide! CodeOfaNinja, 29 Jun 2020. https://codeofaninja.com/2017/02/create-simple-rest-api-in-php.html [accessed 25 Dec 2020]
- Brad Traversy: PHP REST API From Scratch. Traversy Media, 26 Jul 2018. https://www.youtube.com/playlist?list=PLillGF-RfqbZ3_Xr8do7Q2R752xYrDRAo [accessed 25 Dec 2020]