eBookStore Service

Due: Thu of Week 24 (10th April) at **23:59pm**
Submission folder: \\raptor.kent.ac.uk\files\proj\co639\assessment2*USERNAME*\public_html [*]

You will develop a REST-like web service for purchasing books in digital format. The service functionalities are offered by defining entry points (APIs) with specific GET and POST parameters and return JSON results so that different front-ends could be developed independently and other services could interface with it.

The service provides the following functionalities:

a. Books can be created by administrators of the service (admins) using the *create_book* API. Read access to information about books (title, authors, description, cover image, price and reviews) is open (*image*, *books*, *book*, *review* APIs).
b. Users can register on the service by providing a username, password and a contact email address (*user_create*). A *login* API allows to both admins and registered users to log in the service and associate either admin or user authority with the session, so that access to the other APIs can be controlled. *Logout* discards the current authority associated with the session.
c. Registered users can create reviews for a book (*review_create*), which contain a rating (integer from 0 to 5). Users can modify or delete their own reviews, while admins can delete any review (*review_update*, *review_delete*).
d. When a user wants to purchase a book the service contacts Paypal to create a payment and redirects the user to Paypal (*purchase_create*). Once the user approves the payment Paypal redirects the user back to the activation API (*purchase_activate*). During activation the service executes the payment on the Paypal service and if successful records the purchase of the book. If the user cancels the payment she will be redirected to the *purchase_cancel* API. The content of the book can be accessed at any time by users who purchased it (*book_download*).
e. The service will maintain a secure audit log about purchase requests, activations and downloads. An admin user can get access to log entries using the *log* API.

A summary of the API, with the parameters and the expected return values is presented in Table 1.

[*] You can access scripts on raptor from a web browser by specifying the path without "files" and "public_html" dirs, e.g. to access \\raptor.kent.ac.uk\files\proj\co639\assessment2\USERNAME\public_html\script.php visit http://raptor.kent.ac.uk/proj/co639/assessment2/USERNAME/script.php. If you cannot execute it, log into raptor with putty and do a "chmod 644 /proj/co639/assessment2/USERNAME/public_html/script.php" (no files at start but with public_html). You might need to change also directory permissions, e.g., "chmod 711 /proj/co639/assessment2/USERNAME/public_html/".

Hints & Marking

Database access

To provide the service functionalities you need to store data in database tables. You have been allocated an account on the dragon mysql server. If you didn't had a dragon account previously you should have received an email with the password to access dragon, if you already had an account and you fogot the password you can reset the password following the instructions on http://www.cs.kent.ac.uk/systems/faq/databases/ . The name of the database you have access to is the same name as your UoK login, to access the database from PHP and from the command line please refer to the material presented in the slides. In your code do not use string literals for DB username, password and database name, but make use of string constants defined in config.php (see Deployment section below). You need to follow a specific structure for the user table (see user_create API) but you are free to decide the table name and structure for the other tables.

Paypal Accounts & Sandbox

To test your code you will need (i) to create a Paypal account, (ii) to create an application for the assignment, and (iii) to create at least one SANDBOX personal account to test purchases. To get these, follow this procedure:

1. Create a new Paypal personal account for the project. To create this go to https://www.paypal.com/uk and create a new account, you do not need to enter payment information, just confirm the email address you provided.
2. Go to http://developer.paypal.com and log in using the account you just created:
    a. Click on the Applications tab and then on "My Apps" to the left and then on the "Create App" button on the right. After creation you are presented with the Client ID and Secret that you need to use to access Paypal from your code, you can access this at any time from the app page.
    b. By clicking on "Sandbox account" on the left you can see a list of sandbox accounts. When signing up the system already created a sandbox business paypal account that will be used as the merchant for testing your application. You need to create at least one personal account to act as a buyer.

Paypal documentation & REST API

General documentation on Paypal can be accessed from http://developer.paypal.com. Paypal provides different ways for integration. For this project you need to use the REST APIs, which allow to receive payments from Paypal without dealing with credit card information (https://developer.paypal.com/docs/integration/web/accept-paypal-payment/). The complete reference to the Paypal REST API is at https://developer.paypal.com/docs/api/ which also contains summary examples on using the SDKs for several programming languages.

Paypal PHP SDK

For the assignment you must use the Paypal PHP SDK installed on raptor in /courses/co639/ .[†]

To use it in your project follow the following procedure:

1. First log into raptor via putty and change the current directory to your submission directory:

   ```
   cd /proj/co639/assessment2/USERNAME/public_html
   ```

2. copy the sdk_config.ini from the vendor/paypal/rest-api-sdk-php/sample folder to the current directory.

   ```
   cp /courses/co639/vendor/paypal/rest-api-sdk-php/sample/sdk_config.ini .
   ```

   You need then to edit the sdk_config.ini with your application Client ID and secret.

3. to use the SDK API in your scripts add the following line:

   ```
   require_once PAYPAL_PHP_SDK . '/vendor/autoload.php';
   ```

   you need to define the constant `PAYPAL_PHP_SDK` in your config.php file (see 'Deployment' section). You also would want to import the following API classes, which should be sufficient for your project:

   ```
   use PayPal\Api\Amount;
   use PayPal\Api\Payer;
   use PayPal\Api\Payment;
   use PayPal\Api\RedirectUrls;
   use PayPal\Api\Transaction;
   use PayPal\Api\PaymentExecution;
   ```

4. have a look at `CreatePaymentUsingPayPal.php` and `ExecutePayment.php` in `/courses/co639/vendor/paypal/rest-api-sdk-php/sample/payments/` for examples on how to use the API. As mentioned earlier, you should also have a look at https://developer.paypal.com/webapps/developer/docs/api/ for reference on the Paypal API and more PHP samples.

NOTES

At the moment the paypal sandbox account does not work properly with non US currency. Please use USD as the currency for payments.

[†] If you want to develop on your own installation you can install composer (https://getcomposer.org/doc/00-intro.md) then follow the instructions at https://developer.paypal.com/docs/api/ (click the PHP tab) but you need to switch to the version in /courses/co639 before submission (i.e. make sure you do not have a vendor dir in your submission directory).

Some examples you can find on developer.paypal.com create OAuthTokenCredential hardcoding Client ID and secret in the code and use it to create an ApiContext. You should instead configure Client ID and secret in the sdk_config.ini and invoke Payment::create and Payment::execute without passing an APIContext as shown in the slides. (if no APIContext is specified one is created automatically using information from sdk_config.ini).

When constructing the return and cancel URL do not hardcode the absolute url but make use of the constant SERVICE_URL in your config.php file (see 'Deployment' section).

Deployment

Please make sure that your submission works from the submission directory, testing the deployment on raptor well before the deadline. All APIs should be accessible from the root of the submission folder, without any additional directory.

Data folder. Write access to the submission folder will be prevented after the deadline, so to allow the application to work after submission, **if you decide to store any data on the file-system (e.g., cover image or book content) it need to be placed in /proj/co639/assessment2_data/ USERNAME (*DATA_FOLDER*)**. This includes Paypal SDK logs, so be sure to either disable logging in the sdk_config.ini file or set the log filename to point to the data folder. *Please be sure not to have any PHP code running from the data folder, as it could not be considered for marking.*

config.php. To ease marking, please create a config.php file in the root of the submission directory and define (see define()) the following constants:

```
define('DB_USERNAME', <your username>);
define('DB_PASSWORD', ...);
define('DB_HOST', 'dragon.kent.ac.uk');
define('DB_DATABASE', <your username>);
define('DATA_FOLDER', '/proj/co639/assessment2_data/USERNAME/');
define('SERVICE_URL','http://raptor.kent.ac.uk/proj/co639/assessment2/USERN
AME/');
define('AUDIT_LOG_START_KEY', ...);
define('PAYPAL_PHP_SDK', '/courses/co639/');
define('PP_CONFIG_PATH', __DIR__);
```

Use the require_once statement to **include this file in any other php file you define** and use the defined constants in your code instead of using hardcoded values (e.g., when connecting to the database, accessing the file system or constructing links to other URI of the service). The constant `AUDIT_LOG_START_KEY` is the initial key used to sign entries in the audit log. The last two defines are used to define the SDK location and point it to the sdk_config.ini file in the root of your project.

README.txt. Please also provide in a README.txt the credentials for admin and the registered users. You also need to provide the credentials (username and password) for the sandbox buyer accounts. You may provide username and password of the Paypal account you used. Do not use your personal Paypal account, if you already have one, but create a new one for the assignment, see below for more information.

Front-end and PHP frameworks

You are not required to develop a front-end for the eBook Service. To test the service you can use the html forms on raptor under /courses/co639/test. Simply copy the directory under your submission folder and point your browser to http://raptor.kent.ac.uk/proj/co639/assessment2/USERNAME/test/index.html.

You are not allowed to use any PHP framework, just plain PHP.

Marking scheme:

- User registration, password management [6 marks]
- Login and session management [4 marks]
- Book and reviews functionalities [6 marks]
- Selling functionalities and Paypal integration [6 marks]
- Secure Audit Log [4 marks]
- Design, code layout, readability/comments and conciseness [4 marks]

Code conciseness/ commenting /etc

To achieve these marks we're looking for you to have thought carefully about how you structure your assignment to minimise repetition and maximise reuse. We're also looking for you to maximise readability by formatting you code and commenting it appropriately.

Plagiarism

You must tackle this assignment individually. You must not share code with others and you must be diligent in preventing others from seeing or copying your own code. The department considers allowing others to copy your code as being complicit in plagiarism and this often results in students receiving a penalty.

I personally take plagiarism extremely seriously and will investigate and report any suspicion I have. I will also use automated detection software to improve my detection rate.

For detailed guidelines on plagiarism please go to:

http://www.cs.kent.ac.uk/teaching/student/assessment/plagiarism.local

Endpoint	method	parameters	return value	accessible by	notes
/user_create.php	POST	username, password, email	{ "success" : Boolean, "message" : String }	Anyone	Allows to register on the website with a chosen username if it is available. Users are stored in a table called "users" on the database, which is composed of three fields "username", "password","email" and "type" (set to either "admin" or "user"), which is used to distinguish admins and regular users. Note that admins cannot be registered using the API and we assume that are manually configured in the database.
/login.php	POST	username, password	{ "success" : Boolean, "message" : String }	Anyone	Allows users and admins to log into the service, this api causes a session cookie to be set in the caller, checks username and password and associates user or admin authority to the session.
/logout.php	GET		{ "success" : Boolean, "message" : String }	User/Admin	Destroys the authorisation authority associated with the current session. Returns success if there was user or admin authority associated with the session.
/book_create.php	POST	title, authors, description, price, image, content	{ "success" : Boolean, "message" : String, "book_id": String }	Admin	Admins can create books passing in the details of the book. Image and content are two files (an image/jpeg and an application/pdf) sent as form input elements of type "file" (see /courses/co639/test/book_create.html)
/image.php	GET	book_id	Returns the image associated with the book. The response should be an image/jpeg.	Anyone	Returns the image associated to a book or returns HTTP 404 error code.
/books.php	GET	title (optional), authors (optional), start (optional), length (optional)	[Book, ...] where Book = {"book_id": String, "title": String, "authors": String, "description": String, "price" : Number}	Anyone	Returns the list of books. The query can specify a substring filter on title/author. Returns up to *length* items starting from the *start* offset.
/book.php	GET	book_id	{ "success": false, "message": String} or {"success": true, "book": Book, "reviews": [review_id1, review_id2, ...] }, Book as before, review ids as strings	Anyone	Returns details of a specific book.
/review_create.php	POST	book_id, user, review, rating	{ "success" : Boolean, "message" : String, "review_id": String }	User	Adds a review by the current user (specified by username). Users can only add a single review per book.
/review_update.php	POST	book_id, user, review, rating	{ "success" : Boolean, "message" : String }	User who authored review	Modifies the review by the current user, old values for review and rating are overwritten.
/review.php	GET	review_id	{"success": true, "book_id": String, "user": String, "review": String, "rating": Number } or { "success": false, "message": String }	Anyone	Returns a review.
/review_delete.php	GET	review_id	{ "success" : Boolean, "message" : String }	Admin, user who authored review	Deletes a review.
/purchase_create.php	GET	book_id,user	Redirects user or returns {"success": false, "message": String }	User	Issues a new Paypal payment for the book, storing the paymentID received from Paypal in the DB, and redirecting the user to paypal using header("Location: ...") If a payment for the purchase was already accepted by the user returns error, if was created but not accepted redirects the user to the payment page. Logs calls and outcome to the audit log.
/purchase_activate.php	POST	book_id,user,token,PayerID	{ "success" : Boolean, "message" : String }	User	Tries to execute the payment on Paypal, and if successful flags the book as purchased by the user. Logs call and outcome to the audit log.
/book_download.php	GET	book_id, user	Returns the book content. The response should be a application/pdf.	User who purchased book	If the user purchased the book, it provides access to the book, returning the book file, and logs the download. If a user downloads a purchased book more than 100 times or didn't purchased the book it returns HTTP code 403 (Forbidden).
/purchase_cancel.php	GET	book_id, user, token		User	If the user has a pending payment forgets about the payment. The call is audited.
/purchases.php	GET	user(optional), book_id(optional), start(optional), length(optional)	[{"book_id": String, "user": String }, ...]	Admin/Users	Users can check their purchases, admin can check all purchases.
/log.php	GET	start(optional), length(optional)	[Entry, ...], where Entry = {"index": Number, "timestamp": String, "cleartext_message": String, "hash": String, "signature": String }	Admin	Return log entries to the admin.
Notes	All returned parameters should be safe to display in HTML pages.				
	All returned data needs to be valid JSON, unless noted otherwise. Note that the success field in responses should be a boolean not a string.				