Homework 6: Server-side Scripting using PHP, XML and eBay API
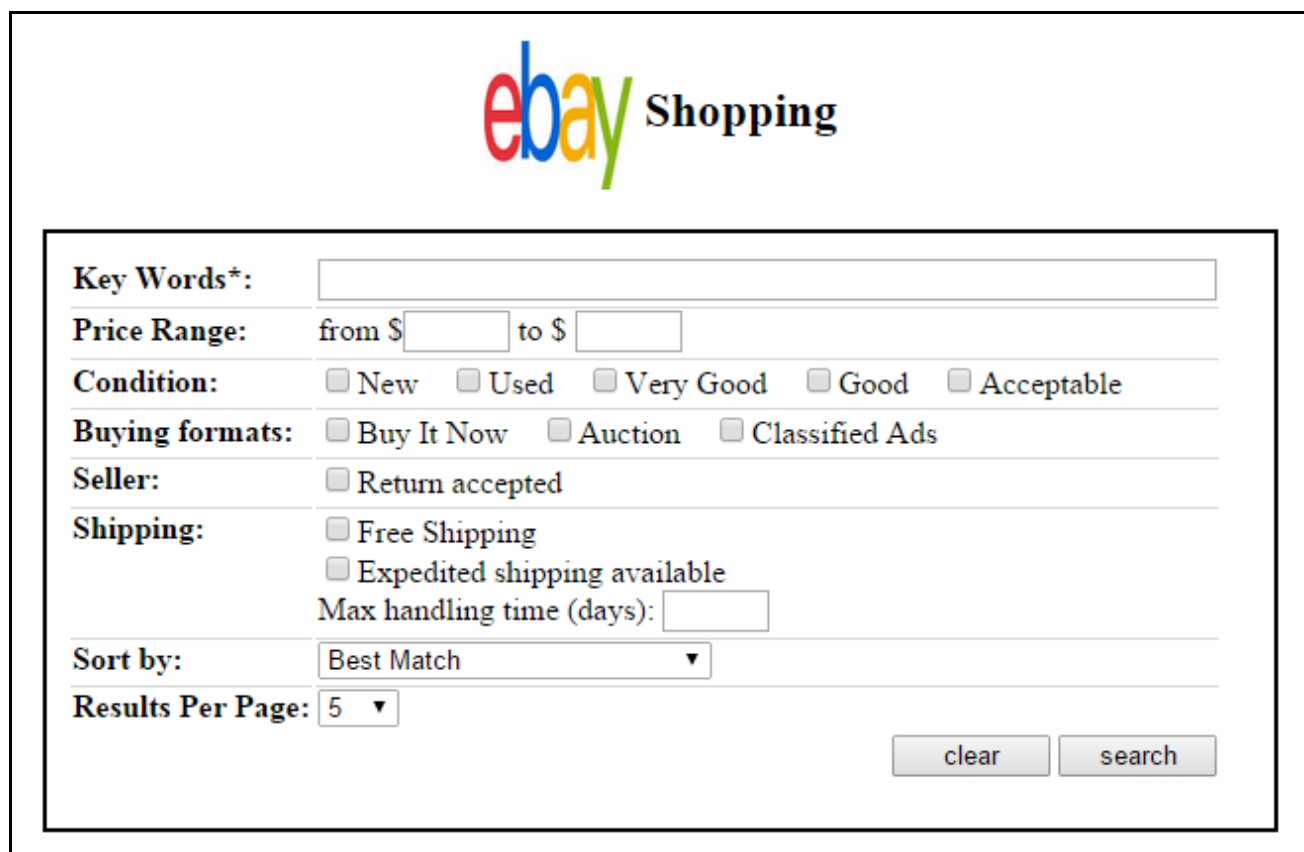
1. Objectives
- Get experience with PHP programming language.
- Get experience with eBay Finding API.
- Get experience using an XML parser in PHP.

2. Description
In this exercise, you are asked to create a webpage that allows users to search items for sale on eBay.com using their API, and the results will be displayed in a tabular format. Instructions on how to use the API are given in section 3.1.

The user first opens a page as shown below in Figure 1, where he/she can enter a query in the keywords textbox. The eBay logo can be found at: http://cs-server.usc.edu:45678/hw/hw6/ebay.jpg.

Figure 1: Initial search page

Once the user has provided valid data (a keyword is required), your script will make a request to your web server providing it with the form data that was entered. You can use either **GET or POST** to transfer the form data to the web server. A PHP script will retrieve the data and send it to the eBay API service "***findItemsAdvanced***". The API call to eBay is done simply by a URL REST

request. For example, if you are just searching for "harry potter" and you retrieve the first 5 results sorted by ascending price (i.e., from low to high price including shipping cost) in XML format, the URL will be:

- http://svcs.eBay.com/services/search/FindingService/v1?siteid=0&**SECURITY-APPNAME=**YourEBayAppKey&**OPERATION-NAME=findItemsAdvanced**&SERVICE-VERSION=1.0.0&**RESPONSE-DATA-FORMAT=**XML&**keywords=**harry%20potter&**paginationInput.entriesPerPage=**5&**sortOrder=** PricePlusShippingLowest

A sample result snippet of 2 results is shown below in Figure 2:

Figure 2. Results page

2.1. DESCRIPTION OF THE SEARCH FORM

The following is a brief description of the fields in the search form (see Figure 1) which you need to implement in your homework.

1. **Key Words**: This is a text box, which enables the user to search for matching items by inputting keywords. This must be a non-empty string. If the **search** button is clicked while the text box is empty, an alert should inform the user about this mandatory field (see Figure 3), and no query should be performed. While building the API URL, you **must** make this string html-safe (i.e., make it url-encoded).

Figure 3. Empty query

2. **Price Range:** These are two text boxes, which enable the user to assign a price range for the matching items. These can be positive integers or decimal numbers [0, ∞). You must ensure that **Minimum Price <= Maximum Price. For specifying the price range,** it is possible to assign only the lower price range or the higher price range or both of them.
3. **Condition**: These are five check boxes "*New*, *Used*, *Very Good*, *Good*, and *Acceptable*" which enable the user to control the item condition retrieved in the results. The user can select one of the options or a combination of them to search for items in specific conditions.
4. **Buying formats**: These are three check boxes "*Buy It Now*, *Auction* or *Classified Ad*". The user can select one of these options or a combination of them to control the buying format of items retrieved in the results.
5. **Seller**: Specifies if the user wants only items sold by a seller who accept returns.
6. **Shipping**: Specify if only "free shipping" items should be returned; if only items available with expedited shipping should be returned, and/or maximum handling time, whose minimum is 1 day (time taken by the seller to deliver the purchased item to the carrier, NOT the shipment time). Default is "all" not specified.
7. **Sort By**: This specifies the ordering of the results table, which can be one of:
   a. Best Match   **- THIS IS THE DEFAULT**
   b. Price: highest first
   c. Price + Shipping: highest first
   d. Price + Shipping: lowest first
8. **Results Per Page**: This specifies the number of items to be retrieved per API call. **Default is 5**, but it can take the values of **5, 10, 15**, or **20** ONLY.

The search form has two buttons:
1. **SEARCH button:** This button validates whether the user provided the mandatory field (i.e., key words) and verifies the correctness of the provided data in the fields (e.g., the price range is valid, the value of handling time is an integer value >= 1). The validation should be implemented in a JavaScript function. Then, an HTTP request is made to your web server

providing it with the form data that was entered. You can use either **GET or POST** to transfer the form data to the Web server.

2. **CLEAR** button: This button **must** clear the result area, all text fields, uncheck all checkboxes and reset "sort by" and "results per page" fields to their default values mentioned above. The clear operation is done using a JavaScript function.

2.2. DISPLAYING RESULTS

The list of results should be displayed as shown in Figure 2. Items are to be vertically stacked (1 item per row). To the left is the item's display image. On the top right, a hyperlink with the item description linking to that item's product page on eBay is displayed. Below it, the item's "condition" and "Buying Format" (Buy it Now, Auction or Classified Ad) is dislayed. Following this, if applicable, you must also indicate whether the item is top rated using the image *itemTopRated.jpg* (http://cs-server.usc.edu:45678/hw/hw6/ebay.jpg). Then, you must indicate, if applicable, if the seller accepts returns.

And then, there's shipping information to be displayed:

1. Whether it is a "Free Shipping" item;
2. Whether expedited shipping is available for that item;
3. The handling time for the item (Ships in 1, 2, … days)

In the last info bar, the price of the item (+ \$x for shipping, if applicable) is displayed in **bold**, followed by the location the item ships from in *italics.*

An example for an item which is new, top rated, offered for selling immediately (i.e., Buy It Now), and its seller accepts returns is shown in Figure 4. The shipping for this item is not free and the expedited shipment is available. The seller delivers the item to the carrier for shipment in 1 day. Because the shipment is not free, the shipping cost is mentioned beside the item price.
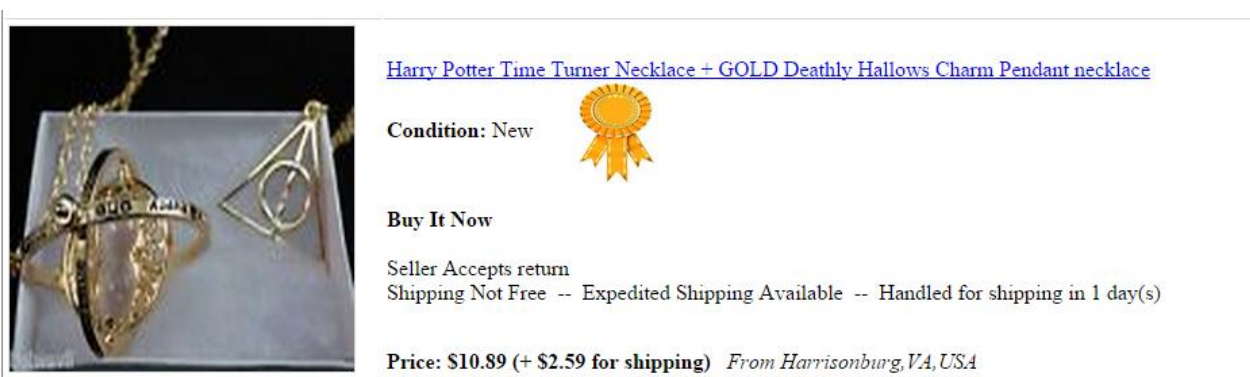

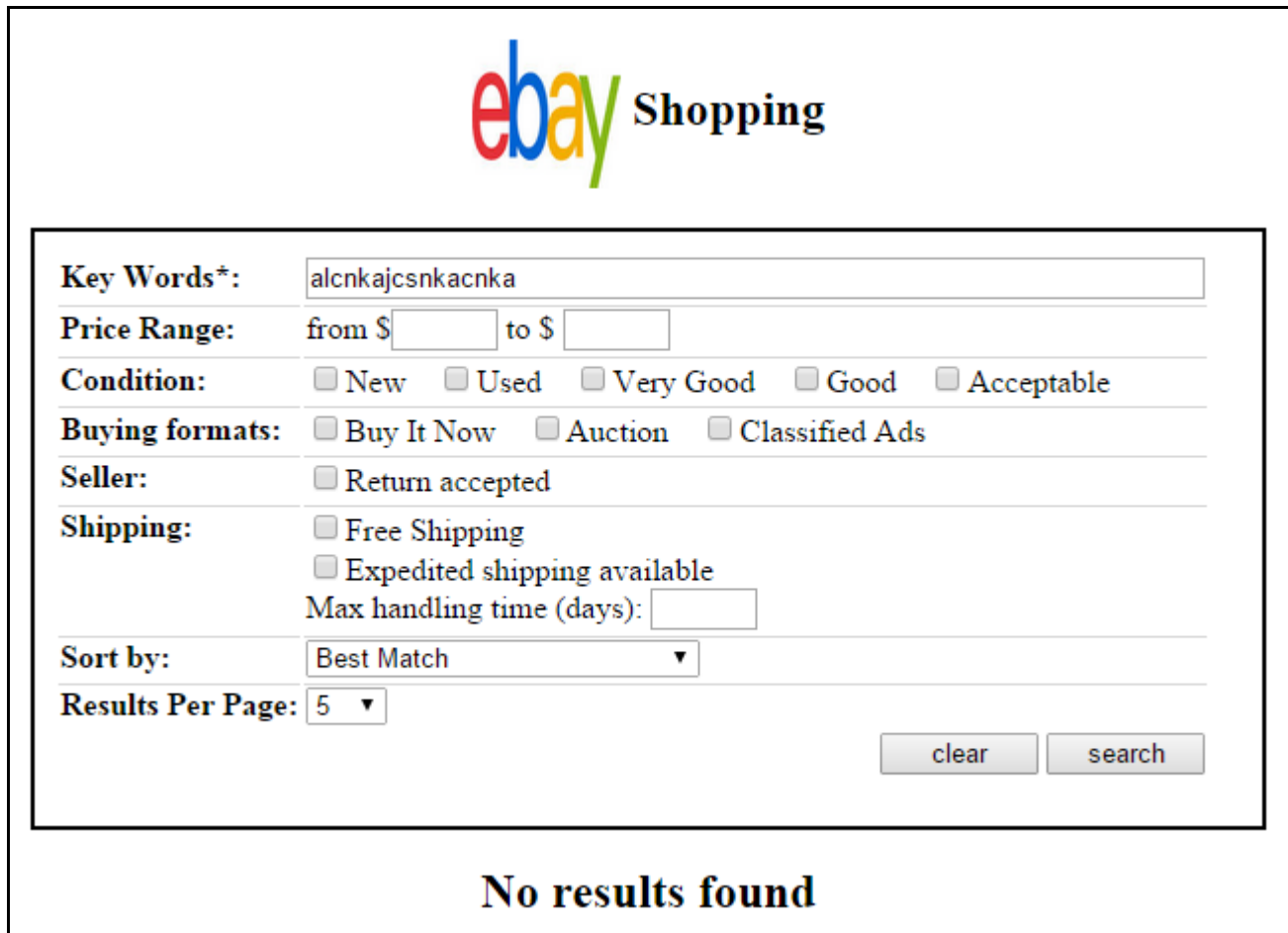**Figure 4: An example of Item in the result table**

At the top of the result table, you should mention the total of the entries matching the search criteria and the key words used in the search. An example of the result title is "*101928* Results for *harry potter*" as shown in Figure 2.

2.3. <mark>SAVING PREVIOUS INPUTS</mark>

In addition to displaying the results, your page should maintain the provided values to display the current result. For example, if one searches for "Free Shipping" items for "harry potter", one should see what was provided in the search form and the corresponding results. Same goes for **all** fields and input types. It follows that you need to keep the whole search box/input fields and buttons even while displaying results/errors.

2.4. NO RESULTS

In case where the eBay API returns no items for a specific keyword such as "*alcnkajcsnkacnka*", you should display "No results found" or any similar message as shown in Figure 5.

Figure 5: An example when No Results is found

3. How to Use the eBay "findItemsAdvanced" API

3.1. HOW TO CREATE THE eBay App ID

To use any of the eBay APIs you should first register for membership in the eBay developer program at https://go.developer.eBay.com/. After registration you will be able to create your own eBay application ID.

When visiting the eBay developer portal, click on "Join" on the top of the page, and fill out the application. For "Company", you may give USC. Then, create a "Production Keys". Do not create more than one Key Set (set of 3 keys, DEVID, AppID, CertID). When you have created your Key Set, your account page may look something like Figure 6 (ignore the "Sandbox Keys")

Figure 6: How to Create an eBay App ID

Under the "Application Keys" section, "Production Keys" sub-section, you'll find your AppID. In Figure 6, the application ID is marked in red box. When constructing the URL to call the eBay APIs, your Application ID should be provided as a value for the parameter SECURITY-APPNAME.

3.2 CONSTRUCTING URL FOR eBay API CALLS.

In this homework, we will use the eBay *"findItemsAdvanced"* API. A comprehensive reference about this API is available at:

http://developer.eBay.com/DevZone/finding/CallRef/findItemsAdvanced.html.

Table 1 shows the mapping between the search fields and the URL parameters to call the eBay API. Some of the search fields are handled through ItemFilter. The item filter is specified using two parameters: *itemFilterName* and *itemFilterValue.*

Search Field	URL parameter
Key words	*Keywords.* (the value of this parameter should be encoded). *Hint:* you can use the PHP function *urlencode.*
Sort by (A drop-down list displaying: 1. Best Match 2. Price: highest first 3. Price + Shipping: highest first 4. Price + Shipping: lowest first)	*sortOrder.* The possible values are: 1. BestMatch 2. CurrentPriceHighest 3. PricePlusShippingHighest 4. PricePlusShippingLowest)
Results Per Page (A drop-down list) 1. 5 2. 10 3. 15 4. 20	*paginationInput.entriesPerPage.* The possible values are: 1. 5 2. 10 3. 15 4. 20
Price Range From	The name of the item filter is *MinPrice.* The value of the filter is the value of "Price Range From" field.
Price Range To	The name of the item filter is *MaxPrice.* The value of the filter is the value of "Price Range To" field.
Condition (a set of check boxes: New, Used, Very Good, Good, and Acceptable)	The name of the item filter is *Condition.* The possible values are: 1. 1000 (means New) 2. 3000 (means Used) 3. 4000 (means Very Good) 4. 5000 (means Good) 5. 6000 (means Acceptable)
Buying Format (a set of check boxes: Buy It Now, Auction, Classified Ads)	The name of the item filter is *ListingType.* The possible values are: 1. FixedPrice (means Buy it Now) 2. Auction 3. Classified (means Classified Ads)
Seller - Returns Accepted	The name of the item filter is *ReturnsAcceptedOnly.* The possible values are true or false.
Shipping - Free Shipping	The name of the item filter is *FreeShippingOnly.*

	The possible values are true or false.
Shipping - Expedited shipping available	The name of the item filter is *ExpeditedShippingType*. One possible value is *Expedited*.
Shipping - Max handling time (days)	The name of the item filter is *MaxHandlingTime*. The value of the filter is the value of "Max handling time" field.

Table 1: Mapping between the search fields and the URL parameters

In addition to the parameters mentioned in the above table, there are five parameters which should be included in every call. The name of these parameters and their values are listed below:

- siteid=0
- SECURITY-APPNAME=YourEBayAppKey
- OPERATION-NAME=findItemsAdvanced
- SERVICE-VERSION=1.0.0
- RESPONSE-DATA-FORMAT=XML

Every filter should have two parameters (name and value). When listing the filters, they should be indexed starting from ZERO. An Example of listing three parameters:

itemFilter**[0]**.name=*filter1NAME***&**itemFilter**[0]**.value=*filter1Value*&itemFilter**[1]**.name=*filter2NAME*&
&itemFilter**[1]**.value=*filter2Value*&itemFilter**[2]**.name=*filter1NAME*&itemFilter**[2]**.value=*filter3Value*

If the filter is assigned to multiple values, the values should be mentioned in a list of parameters and the list of the values should be indexed from ZERO. For example, To filter items based on their condition to be *NEW* or *USED* or *Very Good* can be written as:

itemFilter**[X]**.name=Condition**&**itemFilter**[X]**.value**[0]**=1000**&**itemFilter**[X]**.value**[1]**=3000**&**itemFilter**[X]**.value**[2]**=4000

Another example is to filter items based on their buying format to be *Buy It Now*, *Auction*, or *Classified Ads*:

itemFilter**[X]**.name=ListingType**&**itemFilter**[X]**.value**[0]**=FixedPrice**&**itemFilter**[X]**.value**[1]**=Auction**&**
itemFilter**[X]**.value**[2]**=Classified

For more information about filters, you can read this page:

http://developer.ebay.com/DevZone/finding/CallRef/types/ItemFilterType.html**.**

3.3 PARSING XML RESULTS

To view a sample XML response from the *findItemsAdvanced* API, just copy and paste the URL below and hit Enter inn your **Firefox** browser.

http://svcs.eBay.com/services/search/FindingService/v1?**siteid**=0&**SECURITY-APPNAME=**YourEBayAppKey&**OPERATION-NAME=findItemsAdvanced**&**SERVICE-**

VERSION=1.0.0**&RESPONSE-DATA-FORMAT**=XML**&keywords**=harry%20potter**&paginationInput.entriesPerPage=5&sortOrder**=PricePlusShippingLowest

Please note that the SECURITY-APPNAME parameter is developer-specific as described in section 3.2. Figure 7 below is an example of the high level XML response of the findItemsAdvanced API.

```
- <findItemsAdvancedResponse>
    <ack>Success</ack>
    <version>1.13.0</version>
    <timestamp>2015-02-14T01:32:48.621Z</timestamp>
  - <searchResult count="5">
    + <item></item>
    + <item></item>
    + <item></item>
    + <item></item>
    + <item></item>
    </searchResult>
  - <paginationOutput>
      <pageNumber>1</pageNumber>
      <entriesPerPage>5</entriesPerPage>
      <totalPages>20630</totalPages>
      <totalEntries>103146</totalEntries>
    </paginationOutput>
  - <itemSearchURL>
      http://www.ebay.com/sch/i.html?_nkw=harry+potter&_ddo=1&_ipg=5&_pgn=1
    </itemSearchURL>
  </findItemsAdvancedResponse>
```

Figure 7: High-level successful response xml

In Figure 7, the item tags have been minimized. Expanding the fourth item element (since it has eBay top seller and accepts returns fields) gives us the following (Figure 8).

```
+ <item></item>
- <item>
    <itemId>301324332021</itemId>
    - <title>
        Harry Potter Time Turner Necklace + GOLD Deathly Hallows Charm Pendant necklace
    </title>
    <globalId>EBAY-US</globalId>
    - <primaryCategory>
        <categoryId>29798</categoryId>
        <categoryName>Harry Potter</categoryName>
    </primaryCategory>
    - <secondaryCategory>
        <categoryId>155101</categoryId>
        <categoryName>Necklaces & Pendants</categoryName>
    </secondaryCategory>
    - <galleryURL>
        http://thumbs2.ebaystatic.com/m/mshZMkXZAiwr9qJFyqyVaXg/140.jpg
    </galleryURL>
    - <viewItemURL>
        http://www.ebay.com/itm/Harry-Potter-Time-Turner-Necklace-GOLD-Deathly-Hallows-Charm-Pendant-necklace-/301324332021?pt=LH_DefaultDomain_0
    </viewItemURL>
    <paymentMethod>PayPal</paymentMethod>
    <autoPay>true</autoPay>
    <postalCode>22801</postalCode>
    <location>Harrisonburg,VA,USA</location>
    <country>US</country>
    - <shippingInfo>
        <shippingServiceCost currencyId="USD">2.59</shippingServiceCost>
        <shippingType>FlatDomesticCalculatedInternational</shippingType>
        <shipToLocations>US</shipToLocations>
        <shipToLocations>CA</shipToLocations>
        <expeditedShipping>true</expeditedShipping>
        <oneDayShippingAvailable>false</oneDayShippingAvailable>
        <handlingTime>1</handlingTime>
    </shippingInfo>
    - <sellingStatus>
        <currentPrice currencyId="USD">10.89</currentPrice>
        <convertedCurrentPrice currencyId="USD">10.89</convertedCurrentPrice>
        <sellingState>Active</sellingState>
        <timeLeft>P5DT20H51M15S</timeLeft>
    </sellingStatus>
    - <listingInfo>
        <bestOfferEnabled>false</bestOfferEnabled>
        <buyItNowAvailable>false</buyItNowAvailable>
        <startTime>2014-09-22T22:25:26.000Z</startTime>
        <endTime>2015-02-19T22:30:26.000Z</endTime>
        <listingType>StoreInventory</listingType>
        <gift>false</gift>
    </listingInfo>
    <returnsAccepted>true</returnsAccepted>
    - <galleryPlusPictureURL>
        http://galleryplus.ebayimg.com/ws/web/301324332021_1_1_1.jpg
    </galleryPlusPictureURL>
    - <condition>
        <conditionId>1000</conditionId>
        <conditionDisplayName>New</conditionDisplayName>
    </condition>
    <isMultiVariationListing>false</isMultiVariationListing>
    <topRatedListing>true</topRatedListing>
</item>
+ <item></item>
</searchResult>
+ <paginationOutput></paginationOutput>
+ <itemSearchURL></itemSearchURL>
```

Figure 8: Fourth <item> Detailed

The case where no results are found is shown in Figure 9. You can then query the value of *paginationOutput->totalEntries. If* the value of *totalEntries* is zero, display "no results were found" in the results area.

```
− <findItemsAdvancedResponse>
      <ack>Success</ack>
      <version>1.13.0</version>
      <timestamp>2015-02-14T02:23:07.293Z</timestamp>
      <searchResult count="0"/>
   − <paginationOutput>
      <pageNumber>0</pageNumber>
      <entriesPerPage>5</entriesPerPage>
      <totalPages>0</totalPages>
      <totalEntries>0</totalEntries>
   </paginationOutput>
   − <itemSearchURL>
      http://www.ebay.com/sch/i.html?_nkw=sodcnkisbdcjkshbdcjkr&_ddo=1&_ipg=5&_pgn=1
   </itemSearchURL>
</findItemsAdvancedResponse>
```

Figure 9: No Results XML

The result tags from which you should extract information for display are given in Table 2. ==Your PHP program should parse the resulting XML and extract the information for all items==. All information specific to an item will be within its <item> element as shown above. The root <item> tag for the fourth element shown here will be findItemsAdvancedResponse->searchResult->item. Since there are multiple item elements, the item will be the 4th element in item[] array for this example.

Item Information in the Result Table	Tags in eBay XML response
Total Results Found	*paginationOutput->totalEntries*
Item's image URL	*searchResult->item->galleryURL*
Item description	*searchResult->item->title*
Item's eBay link for description	*searchResult->item->viewItemURL*
Condition	*searchResult->item->condition->conditionDisplayName*
Item Top Rated Image	if the value of *searchResult-->item->topRatedListing* is true, display the image itemTopRated.jp. You can find the image at http://cs-server.usc.edu:45678/hw/hw6/itemTopRated.jpg
Buying Format/Listing Type	*searchResult->item->listingInfo->listingType* ● if the value of listingType is "FixedPrice" or "StoreInventory", print "Buy It Now" ● If the value is "Auction" print "Auction"

	● If the value is "Classified" print "Classified Ad"
Seller Accepts return?	*searchResult->item->returnsAccepted* is either true or false
Free Shipping?	If the value of the tag *searchResult-->item->shippingInfo->shippingServiceCost* is equal to 0.0, print "FREE Shipping". Otherwise, print "Shipping Not FREE".
Expedited Shipping available?	*searchResult->item->shippingInfo->expeditedShippin*g is either true or false
Handling Time/Time to Ship	*searchResult->item->shippingInfo->handlingTime*
Price	the price is displayed in the format: *Price: $20 (+ $3 for shipping)* The price value is read from *searchResult->item->sellingStatus-> convertedCurrentPrice* The shipping cost is read from *searchResult->item->shippingInfo->shippingServiceCost* The shipping cost phrase (*+ $$$ for shipping*) is mentioned if and only if the value of *shippingServiceCost* is greater than ZERO
Ships from Location	searchResult->item->location

Table 2: XML Tags information

4. Hints

4.1. PARSING XML FILES IN PHP

You are free to choose any XML parsing library, but we recommend the *SimpleXML* library. The SimpleXML library is a simple way of getting an XML element's name, attributes, and text. As of PHP 5, the SimpleXML library functions are part of the PHP core. No installation is required to use these functions. The following two tables show a set of functions which you may use. For more detailed information, please read:

- http://www.w3schools.com/php/php_xml_simplexml.asp
- http://php.net/manual/en/book.simplexml.php
- http://www.w3schools.com/php/php_ref_simplexml.asp

Function	Description
__construct()	Creates a new SimpleXMLElement object
addAttribute()	Adds an attribute to the SimpleXML element

addChild()	Adds a child element the SimpleXML element
asXML()	Formats the SimpleXML object's data in XML (version 1.0)
attributes()	Returns attributes and values within an XML tag
children()	Finds the children of a specified node
count()	Counts the children of a specified node
getDocNamespaces()	Returns the namespaces DECLARED in document
getName()	Returns the name of the XML tag referenced by the SimpleXML element
getNamespaces()	Returns the namespaces USED in document
registerXPathNamespace()	Creates a namespace context for the next XPath query
saveXML()	Alias of asXML()
simplexml_import_dom()	Returns a SimpleXMLElement object from a DOM node
simplexml_load_file()	Converts an XML file into a SimpleXMLElement object
simplexml_load_string()	Converts an XML string into a SimpleXMLElement object
xpath()	Runs an XPath query on XML data

Table 3: PHP 5 SimpleXML Functions

Function	Description
current()	Returns the current element
getChildren()	Returns the child elements of the current element
hasChildren()	Checks whether the current element has children
key()	Return the current key
next()	Moves to the next element
rewind()	Rewind to the first element
valid()	Check whether the current element is valid

Table 4: PHP 5 SimpleXML Iteration Functions

5. Files to Submit

On your course homework page, you should update the **HW6 link** to refer to your new initial web page. Also, Submit your files (likely only a single .php file) electronically to the csci571 account so

that they can be graded and compared to all other students' code via the MOSS code comparison tool.

IMPORTANT:
All discussions and explanations in Piazza related to this homework are part of the homework description. So please review all Piazza threads before finishing the assignment.