Homework 4: Server-side Scripting

1. Objectives

- Get experience with the PHP programming language
- Get experience with the Markit on Demand APIs
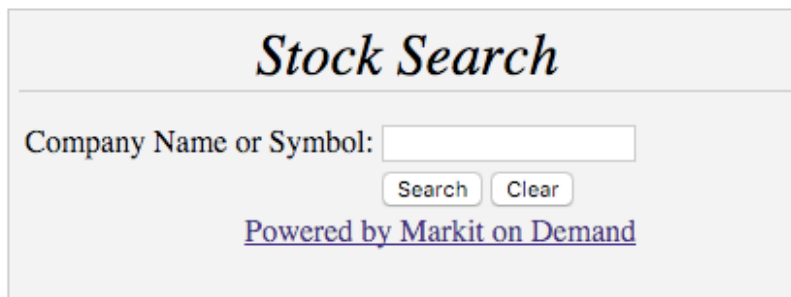- Get experience using XML and JSON parsers in PHP.

2. Description

In this exercise, you are asked to create a webpage that allows you to search for stock information using the *Markit on Demand* APIs, and the results will be displayed in tabular format.

2.1. Description of the Search Form

A user first opens a page, called **stock.php (or any valid web page name)**, where he/she can enter a company name or symbol. An example is shown in Figure 1. Providing a value for the "*Company Name or Symbol*" field is mandatory (the validation should be done using one of the attributes provided by HTML5). Also, the form should include a *Markit on Demand* disclaimer "e.g., Powered by Markit on Demand", linking to the web page:
'http://www.markit.com/product/markit-on-demand'.

Figure 1: Initial Search Screen

The search form has two buttons:
- **Search** button: If the user clicks on the search button without providing a value in the "Company Name or Symbol" field, HTML5 should validate it (an example is shown in Figure 2). An example of valid input is shown in Figure 3. Once the user has provided valid data, your client script should send a request to your web server for **stock.php** with the form data. 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 *Markit on Demand* restful Web Service.
- **Clear** button: This button must clear the result area and the text field. The Clear operation is done using a JavaScript function.

Figure 2: An Error Message when there is no input
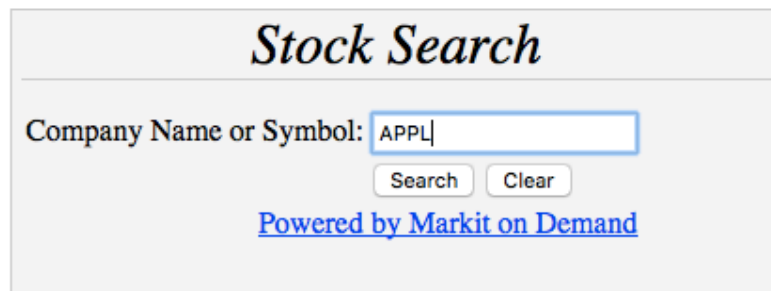
Figure 3: An Example of Valid Input

2.2. Displaying Results

In this section, we outline how to use the form data to construct the calls to restful web services from *Markit on Demand* APIs and display the result in the web page. Mainly we use two web services: Lookup and Quote. The Lookup web service returns an array of company matches for a given input, which contains the name, symbol and the trading exchange of the company while the Quote web service returns the current value of a corresponding company stock when user inputs a symbol of a company stock

The search operation is based on two main steps:

1. Use the Lookup web service in the *Markit on Demand* API to find the stock symbol corresponding to the user input.
2. Use the symbol returned from the Lookup web service to call the Quote web service to get the company stock information.

The PHP script (i.e., **stock.php**) uses the input information (company name or symbol) to construct a restful web service URL to retrieve all companies matching the query input:

http://dev.markitondemand.com/MODApis/Api/v2/**Lookup**/xml?input=**APPL**

The Lookup web service URL has a parameter called *input*. The value of the *input* parameter should be the text entered in the "Company Name or Symbol" edit box. The above link is an example of calling lookup service with the input "AAPL". The response of this URL is an XML-formatted object. Figure 4 shows an example of the returned response of the lookup service.

```
▼<LookupResultList>
   ▼<LookupResult>
       <Symbol>AAPL</Symbol>
       <Name>Apple Inc</Name>
       <Exchange>NASDAQ</Exchange>
   </LookupResult>
   ▶<LookupResult>...</LookupResult>
   ▶<LookupResult>...</LookupResult>
   ▶<LookupResult>...</LookupResult>
   ▶<LookupResult>...</LookupResult>
   ▶<LookupResult>...</LookupResult>
   ▶<LookupResult>...</LookupResult>
   ▶<LookupResult>...</LookupResult>
   ▶<LookupResult>...</LookupResult>
   ▶<LookupResult>...</LookupResult>
</LookupResultList>
```

Figure 4: A Sample Result of the lookup service

The PHP script (i.e., **stock.php**) should parse the returned XML-formatted object and extract the necessary fields. After extracting the data, the PHP script should display the data in a tabular format below the search form. A sample output is shown in Figure 5. All returned data is used to render a result table. The symbol, name and exchange values should be displayed in the result table. Please note that the symbol value will be used for the next step to retrieve the quote information.

Figure 5: An Example of Search Result

If the Lookup service returns an empty result set, the page should return "*No Record has been found*". Figure 6 shows an example when searching for a company name "*BLAHBLAH*".

Figure 6: Search Result when there is no matching result

When the search result has at least one record, you need to map the data extracted from the lookup service result to render the result table as follows.

Table Column	Lookup service response
Name	<Name> tag under LookupResult

Symbol	<Symbol> tag under LookupResult
Exchange	<Exchange> tag under LookupResult

The details column contains a "More Info" link. When users click on the "More Info" link of a certain company, the PHP script (i.e., stock.php) should use the **corresponding** symbol value to construct another restful web service URL to query the quote service to get the company quote information for the given symbol such as in:

http://dev.markitondemand.com/MODApis/Api/v2/Quote/json?symbol=**SYMBOLVALUE**

The response from a query of the quote method is a JSON-formatted object. An example of a returned JSON-formatted object from this method is available at:

http://dev.markitondemand.com/MODApis/Api/v2/Quote/json?symbol=*AAPL*

Figure 7 shows an example of the returned JSON file. The PHP script should parse the returned JSON-formatted object and extract some fields. After extracting the data, the PHP script should display the data in a tabular format below the search form. A sample output is shown in Figure 8.

```
{
    "Status":"SUCCESS",
    "Name":"Apple Inc",
    "Symbol":"AAPL",
    "LastPrice":95,
    "Change":0.980000000000004,
    "ChangePercent":1.04233141884706,
    "Timestamp":"Mon Feb 8 15:59:00 UTC-05:00 2016",
    "MSDate":42408.6659722222,
    "MarketCap":526735385000,
    "Volume":5125894,
    "ChangeYTD":105.26,
    "ChangePercentYTD":-9.74729241877257,
    "High":95.69,
    "Low":93.05,
    "Open":93.24
}
```

Figure 7: A Sample JSON-formatted object returned from *Markit on Demand* quote service

Figure 8: Search Result When clicking the More Info Link

When the status value is "SUCCESS", the PHP script should map the data retrieved from the quote service response to render the company stock table using the following mapping:

Table Column	Quote Service Response
Name	The value of *Name*
Symbol	The value of *Symbol*
Last Price	The value of *LastPrice*
Change	The value of *Change* rounded to two decimal points followed by a marker representing the price change trend of the stock .
Change Percent	The value of *ChangePercent* rounded to two decimal points followed by a percentage "%" character and a marker representing the price change trend of the stock

Timestamp	The value of *Timestamp* displayed in YYYY-MM-DD HH:MM AM/PM format.
Market Cap	The value of *MarketCap should be divided by a billion (1,000,000,000) and rounded* to two decimal points. The unit is B (stands for billion).
Volume	The value of *Volume* in Integer format. The value should be rendered with a thousand operator format (e.g., 9,876,543)
Change YTD	The value of *LastPrice* minus *ChangeYTD*, in parentheses if a negative value. The output value should be rounded to two decimal points followed by a mark representing the YTD change trend of the stock.
Change Percent YTD	The value of *ChangePercentYTD*. The value should be rounded to two decimal points followed by a percentage "%" character and a mark representing the YTD change trend of the stock
High	The value of *High*. Display the original data, no round is needed.
Low	The value of *Low*. Display the original data, no round is needed.
Open	The value of *Open*. Display the original data, no round is needed.

Regarding the marker icon rendered beside the values of *Change*, *Change Percent*, *Change YTD*, and *Change Percent YTD*, a red-down arrow icon is displayed if the value is negative while a green-up arrow icon is displayed if the value is positive. The icons can be found on Canvas under HW4 folder.

If the returned result of the quote service has the "FAILURE" status value, the PHP script should display a message "*There is no stock information available*". For example, searching for stock information using the symbol "GRNREG", it returns the JSON output shown in Figure 9. Figure 10 shows how the PHP script handles this case.

```
{ ⊟
    "Status":"Failure|APP_SPECIFIC_ERROR",
    "Name":"NASD REN ENER GE",
    "Symbol":"GRNREG",
    "LastPrice":0,
    "Change":0,
    "ChangePercent":0,
    "Timestamp":null,
    "MSDate":0,
    "MarketCap":0,
    "Volume":0,
    "ChangeYTD":0,
    "ChangePercentYTD":0,
    "High":0,
    "Low":0,
    "Open":0
}
```

Figure 9: An Example of the JSON response of Quote Service when the status is "FAILURE"

Figure 10: An Example of the search result when the Quote Service returns a response with "FAILURE" status

In summary, the search mechanism to be implemented behaves as follows:

- Based on the input data in the search form, construct a web service URL to retrieve the output from the lookup service.
- Parse the returned XML and extract the symbol values.
- Call the quote service and retrieve the JSON-formatted output.
- Parse the returned JSON-formatted output and extract the quote.
- Display the stock information in tabular format.

Important Note: It is mandatory to retrieve the result from Lookup service using XML format and the result from Quote service using JSON format.

2.3. Saving Previous Inputs

In addition to displaying the results, the PHP page should maintain the provided values to display the current result. For example, if a user searches for "*Company Name or symbol: GE*", the user should see what was provided in the search form when displaying the results. Specifically, when clicking on the "Search" button, the page should display the result retrieved from the Lookup web service and keep the value provided in the search form. In addition, when clicking on the "More Info" link, the page should display the result retrieved from the Quote web service and keep the value provided in the search form. It follows that you need to keep the whole search box/input fields and buttons even while displaying results/errors.

3. Hints

3.1. Markit on demand API Documentation

For information about the Markit on Demand API, please go to:

http://dev.markitondemand.com/MODApis/

3.2. Parsing XML files in PHP

You are free to choose any XML parsing library, but we recommend the SimpleXML library. The SimpleXML library provides 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 go to:

- 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

PHP 5 SimpleXML Functions

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

PHP 5 SimpleXML Iteration Functions

Function	Description
current()	Returns the current element
getChildren()	Returns the child elements of the current element
hasChildren()	Cheks 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

3.3 Parsing JSON-formatted data in PHP

In PHP 5, you can parse JSON-formatted data using the "*json_decode*" function. For more information, please go to http://php.net/manual/en/function.json-decode.php.

To read the contents of a JSON-formatted object, you can use the "*file_get_contents*" function.

4. Files to Submit

Submit your files (likely only a single .php file) to Canvas account so that they can be graded. Included a readme file with your submission that includes the URL to your running site.