Web Services

Chapter 19

Randy Connolly and Ricardo Hoar

OVERVIEW OF WEB SERVICES

Web Services

An overview

Web services are the most common example of a computing paradigm commonly referred to as **service-oriented computing** (SOC).

A **service** is a piece of software with a platform-independent interface that can be dynamically located and invoked.

Web services are a relatively standardized mechanism by which one software application can connect to and communicate with another software application using web protocols.

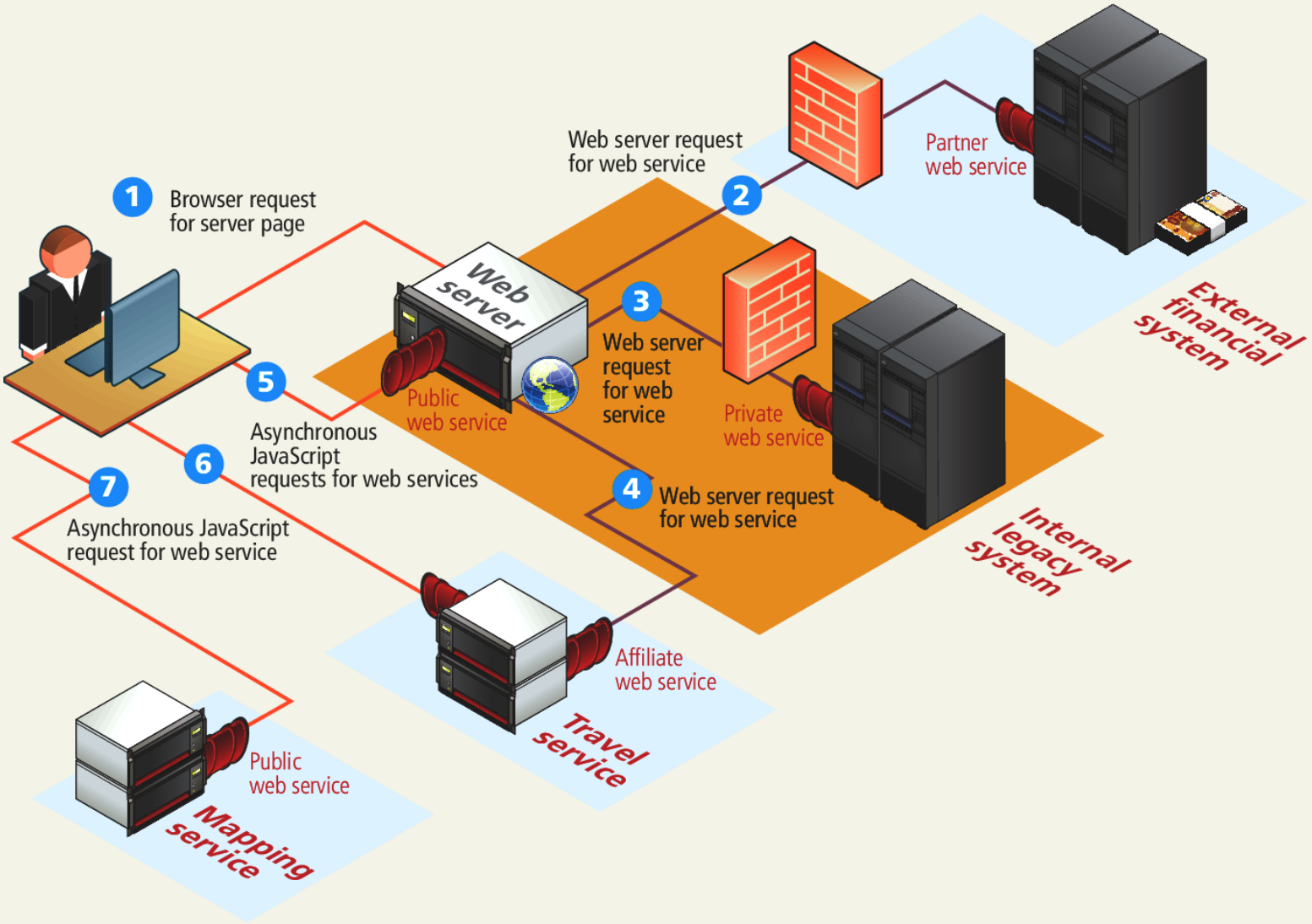
Web Services

Benefits

- they potentially provide interoperability between different software applications running on different platforms
- they can be used to implement something called a **service-oriented architecture** (SOA)
- they can be offered by different systems within an organization as well as by different organizations

Web Services

Visual Overview



Web Services

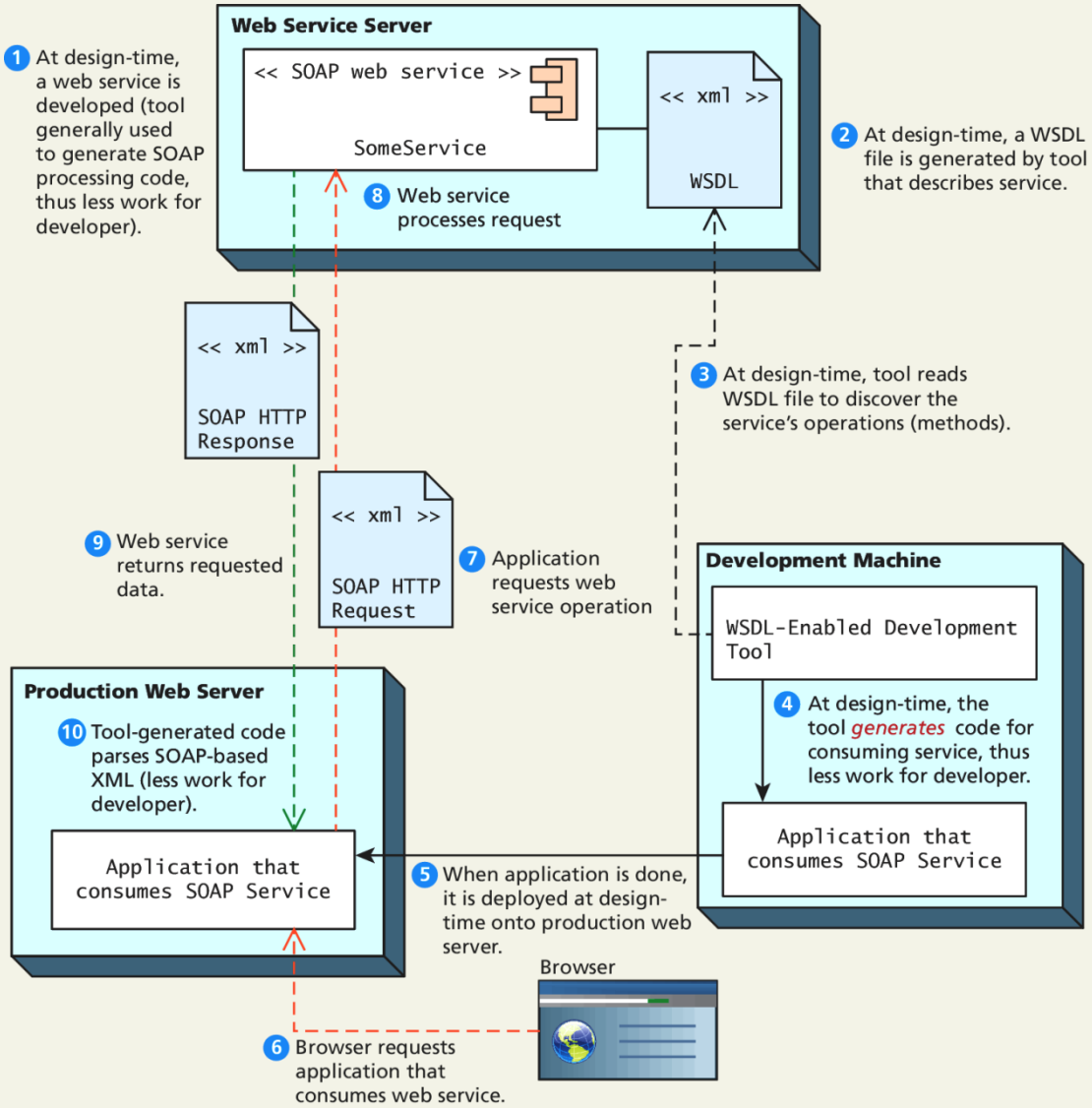
SOAP Services

SOAP is the message protocol used to encode the service invocations and their return values via XML within the HTTP header.

- SOAP and WSDL are complex XML schemas
- akin to using a compiler: its output may be complicated to understand
- the enthusiasm for SOAP-based web services had cooled.

Web Services

SOAP Services



Web Services

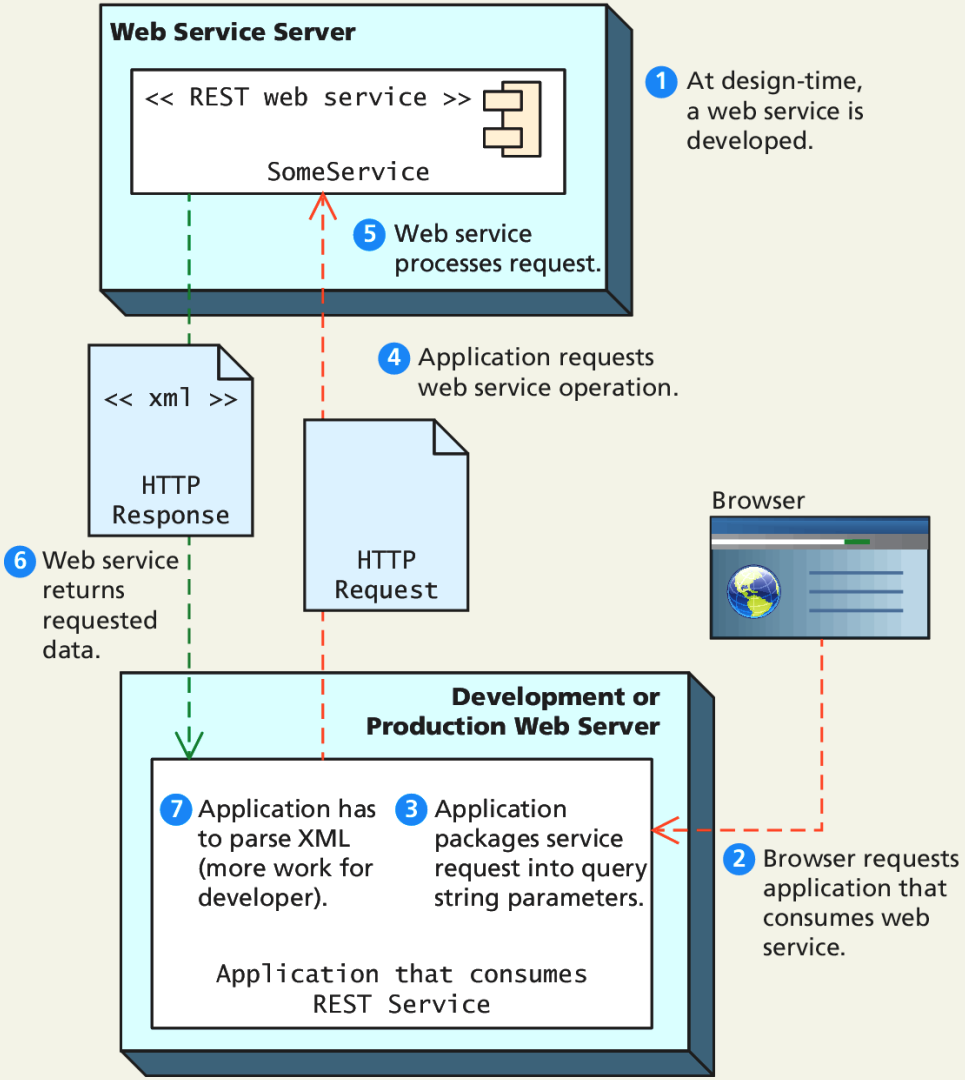
REST Services

REST stands for Representational State Transfer.

- RESTful web service does away with the service description layer, and needs no separate protocol for encoding message requests and responses.
- It simply uses HTTP URLs for requesting a resource/object (and for encoding input parameters).
- The serialized representation of this object, usually an XML or JSON stream, is then returned to the requestor as a normal HTTP response.
- REST appears to have almost completely displaced SOAP services.

Web Services

REST Services



An Example Web Service

We will only use REST from here on in

Consider the Google Geocoding API.

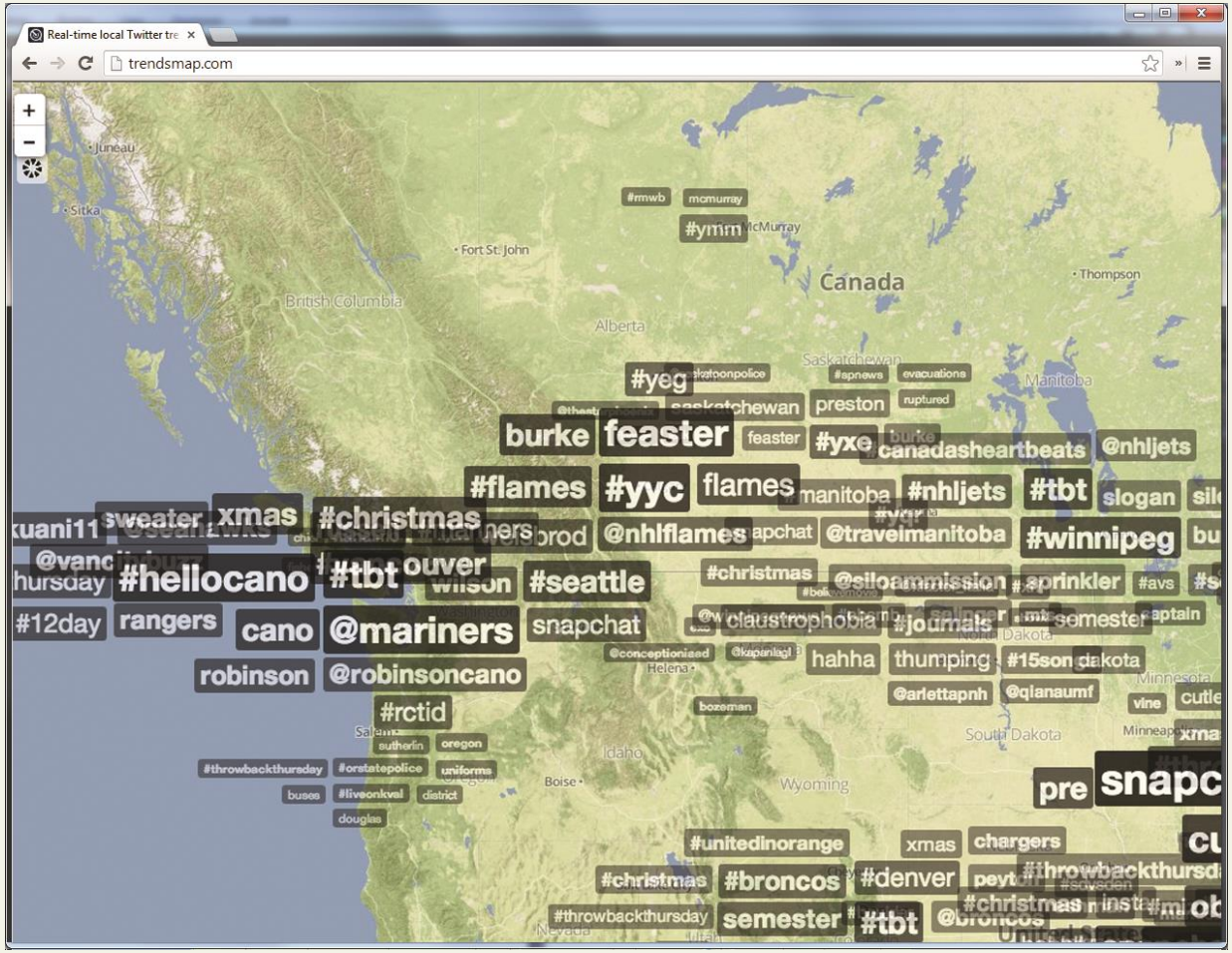
The Google Geocoding API provides a way to perform geocoding operations via an HTTP GET request, and thus is an especially useful example of a RESTful web service.

Geocoding typically refers to the process of turning a real-world address into geographic coordinates, which are usually latitude and longitude values

Reverse geocoding is the process of converting geographic coordinates into a human-readable address.

An Example Web Service

Mashups abound with web services



From trendsmap.com

An Example Web Service

More details

In this case the request will take the following form:

http://maps.googleapis.com/maps/api/geocode/xml?***address***

An example geocode request would look like the following:

http://maps.googleapis.com/maps/api/geocode/xml?address=British%20Museum,+Great+Russell+Street,+London,+WC1B+3DG&sensor=false

From trendsmap.com

An Example Web Service

The Response

```
HTTP/1.1 200 OK
Content-Type: application/xml; charset=UTF-8
Date: Fri, 19 Jul 2013 19:15:54 GMT
Expires: Sat, 20 Jul 2013 19:15:54 GMT
Cache-Control: public, max-age=86400
Vary: Accept-Language
Content-Encoding: gzip
Server: mafe
Content-Length: 512
X-XSS-Protection: 1; mode=block
X-Frame-Options: SAMEORIGIN

<?xml version="1.0" encoding="UTF-8"?>
<GeocodeResponse>
   <status>OK</status>
   <result>
      <type>route</type>
      <formatted_address>
         Great Russell Street, London Borough of Camden, London, UK
      </formatted_address>
      <address_component>
         <long_name>Great Russell Street</long_name>
         <short_name>Great Russell St</short_name>
         <type>route</type>
      </address_component>
      <address_component>
         <long_name>London</long_name>
         <short_name>London</short_name>
         <type>locality</type>
         <type>political</type>
      </address_component>
      ...
      <geometry>
         <location>
            <lat>51.5179231</lat>
            <lng>-0.1271022</lng>
         </location>
         <location_type>GEOMETRIC_CENTER</location_type>
         ...
      </geometry>
   </result>
</GeocodeResponse>
```

LISTING 17.13 HTTP response from web service

The response is a standard HTTP response with headers

This response is XML

The lat/lng is in there somewhere

Identifying and Authenticating Service Requests

Most web services are not open. Instead they typically employ one of the following techniques:

- **Identity**. Each web service request must identify who is making the request.
- **Authentication**. Each web service request must provide additional evidence that they are who they say they are.

Identity examples

Real World ways of limiting service

Web services that make use of an API key typically require the user (i.e., the developer) to register online with the service for an API key. This API key is then added to the GET request as a query string parameter.

For instance, to request to the Microsoft Bing Maps web service will look like the following :

http://dev.virtualearth.net/REST/v1/Locations?o=xml&query=British%20Museum,+Great+Russell+Street,+London,+WC1B+3DG,+UK&**key**=**[*BING API KEY HERE*]**

Identity examples

Real World ways of limiting service

Web services that make use of an API key typically require the user (i.e., the developer) to register online with the service for an API key. This API key is then added to the GET request as a query string parameter.

For instance, to request to the Microsoft Bing Maps web service will look like the following :

http://dev.virtualearth.net/REST/v1/Locations?o=xml&query=British%20Museum,+Great+Russell+Street,+London,+WC1B+3DG,+UK&**key**=**[*BING API KEY HERE*]**

Authentication

Real World ways of limiting service

Some web services are providing private/proprietary information or are involving financial transactions.

In this case, these services not only may require an API key, but they also require some type of user name and password in order to perform an authorization.

Many of the most well-known web services instead make use of the OAuth standard.

CONSUMING WEB SERVICES IN PHP

Consuming Web Services in PHP

There are three usual approaches in PHP for making a REST request:

- Using the file_get_contents() function.
- Using functions contained within the curl library.
- Using a custom library for the specific web service. Many of the most popular web services have free and proprietary PHP libraries available.

Consuming Web Services in PHP

The file_get_contents() function is simple but doesn't allow POST requests

Services that require authentication will have to use the curl extension library, which allows significantly more control over requests. You may need to configure your server to include curl support.

A Flickr Example

The Flickr web service provides a photo search service. The basic format for this service method is:

http://api.flickr.com/services/rest/method=flickr.photos.search&api_key=[*enter your flickr api key here*]&tags=[*search values here*]&format=rest

The service will return its standard XML photo list

A Flickr Example

Some Code using file_get_contents()

```
<?php

function constructFlickrSearchRequest($search)
{
   $serviceDomain = 'http://api.flickr.com/services/rest/?';
   $method = 'method=flickr.photos.search';
   $api_key = 'api_key=' . 'your Flickr api key here';
   $searchFor = 'tags=' . $search;
   $format = 'format=rest';
   // only 12 results for now
   $options = 'per_page=12';
   // due to copyright, we will use only the author's Flickr images
   $options .= '&user_id=31790027%40N04';

   return $serviceDomain . $method . '&' . $api_key .'&'
       . $searchFor . '&' . $format . '&' . $options;
}

?>
```

LISTING 17.14 Function to construct Flickr search request

```
$request = constructFlickrSearchRequest('Athens');
$response = file_get_contents($request);
```

A Flickr Example

Use file get contents

```
$request = constructFlickrSearchRequest('Athens');
$response = file_get_contents($request);

// Retrieve HTTP status code
$statusLine = explode(' ',$http_response_header[0], 3);
$status_code = $statusLine[1];

if ($status_code == 200) {

   // for debugging output response
   echo htmlspecialchars($response);
}
else {
   die("Your call to web service failed -- code=" . $status_code);
}
```

A Flickr Example

Use curl (and actually do something)

```
$request = constructFlickrSearchRequest('Athens');
echo '<p><small>' . $request . '</small></p>';

$http = curl_init($request);
// set curl options
curl_setopt($http, CURLOPT_HEADER, false);
curl_setopt($http, CURLOPT_RETURNTRANSFER, true);
// make the request
$response = curl_exec($http);
// get the status code
$status_code = curl_getinfo($http, CURLINFO_HTTP_CODE);
// close the curl session
curl_close($http);

if ($status_code == 200) {
   // create simpleXML object by loading string
   $xml = simplexml_load_string($response);
   // iterate through each <photo> element
   foreach ($xml->photos->photo as $p) {
      // construct URLs for image and for link
      $pageURL = "http://www.flickr.com/photos/" . $p['owner'] . "/"
                 .$p['id'];
      $imgURL = "http://farm" .$p["farm"] . ".staticflickr.com/"
        . $p["server"] . "/" . $p["id"] . "_" . $p["secret"] . "_q.jpg";
      // output links and image tags
      echo "<a href='" . $pageURL . "'>";
      echo "<img src='" . $imgURL . "' />";
      echo "</a>";
}
}
else {
   die("Your call to web service failed -- code=" . $status_code);
}
```

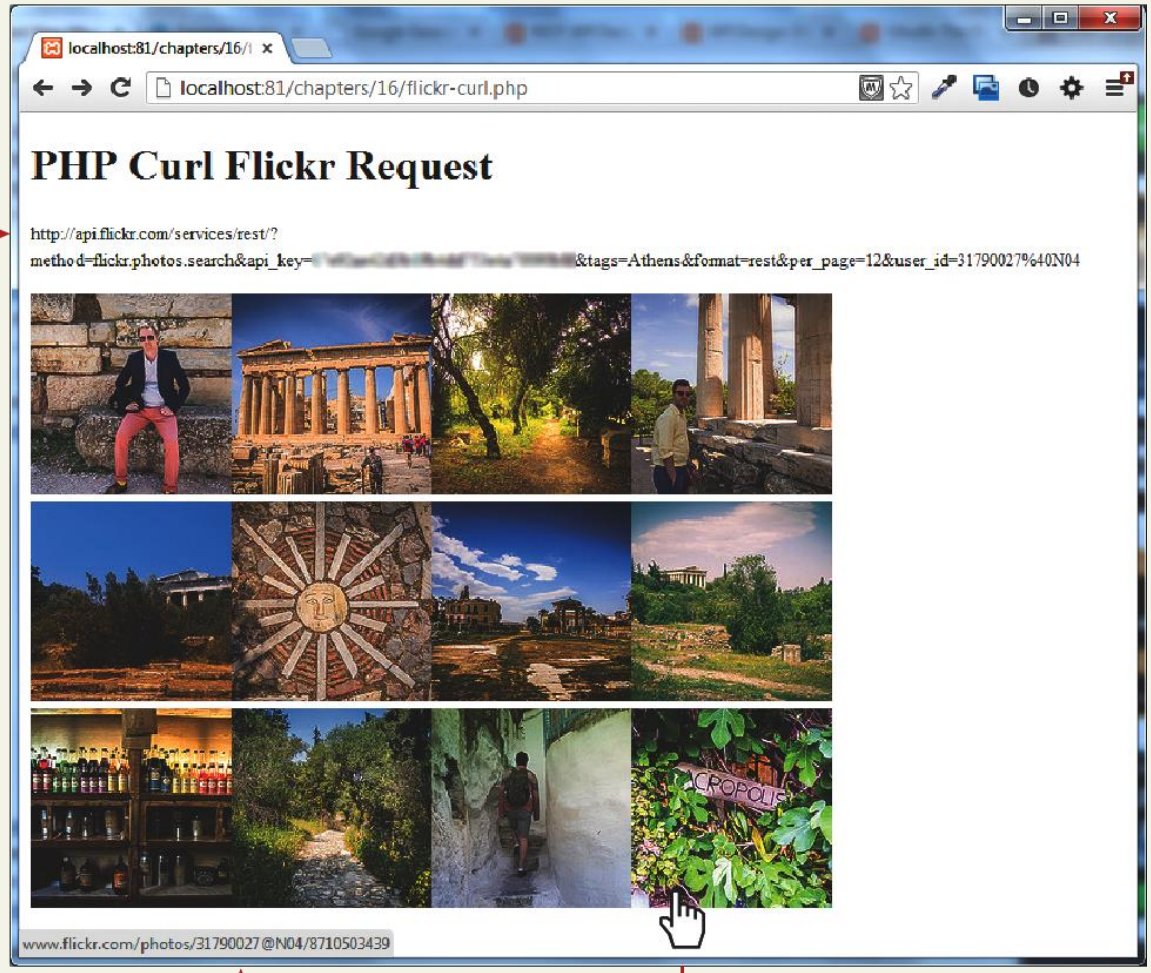
LISTING 17.15 Querying web service and processing the results

Make the request

Parse the XML

A Flickr Example

What that last code actually built



Web service request

URL of image link

Consuming JSON Services

Consuming a JSON web service requires almost the same type of PHP coding as consuming an XML web service.

But rather than using SimpleXML to extract the information one needs, one instead uses the json_decode() function.

Consuming JSON Services

Combine 2 services (using JSON)

To extract the latitude and longitude from the JSON string returned from the mapping web service, you would need code similar to the following:

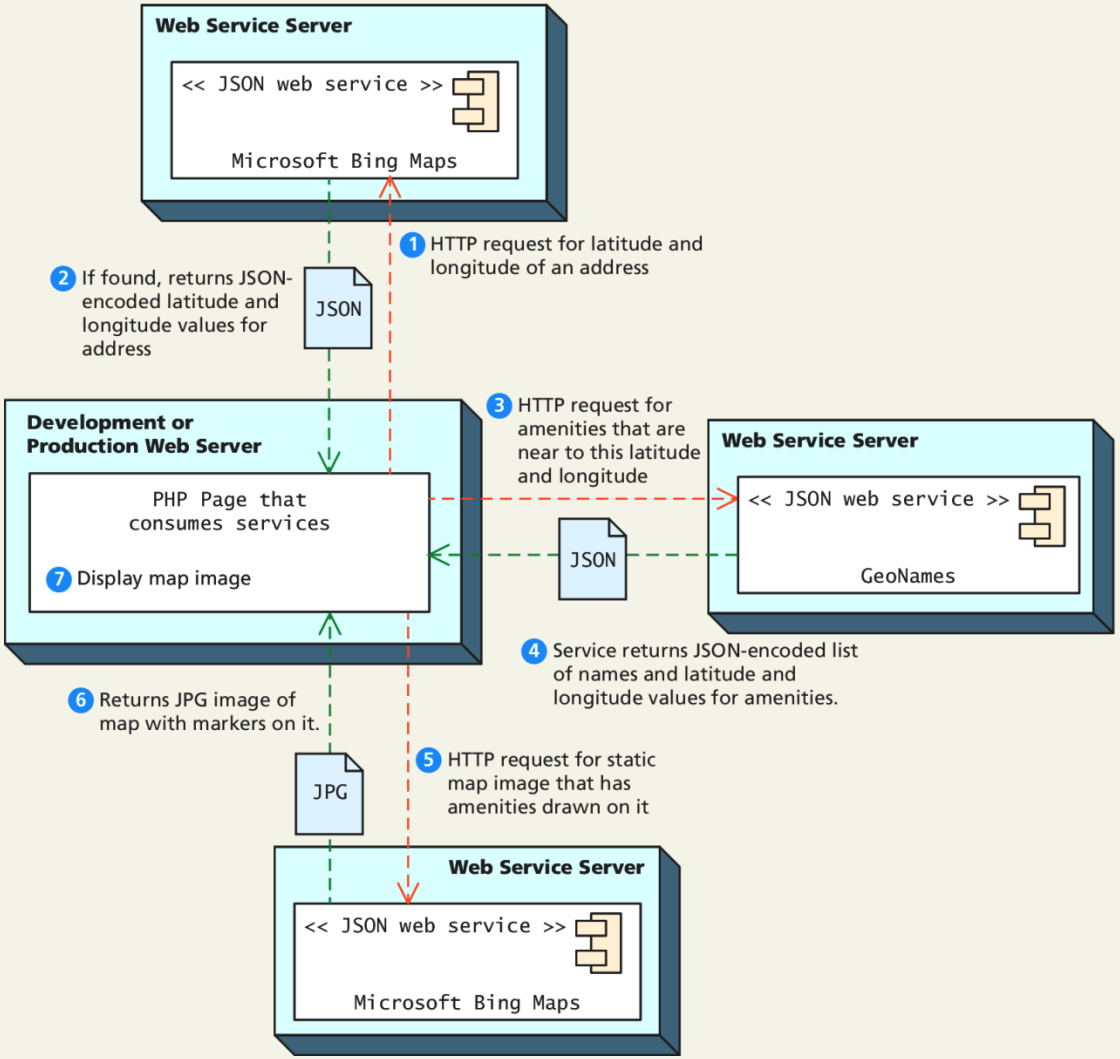
```
// decode JSON and extract latitude and longitude
$json = json_decode($response);
if (json_last_error() == JSON_ERROR_NONE) {
    $lat = $json->resourceSets[0]->resources[0]->point
         ->coordinates[0];
    $long = $json->resourceSets[0]->resources[0]->point
          ->coordinates[1];
}
```

Once our program has retrieved the latitude and longitude, the program then will use the GeoNames web service's. This request will take the following form:

http://api.geonames.org/findNearbyPOIsOSMJSON?lat=43.6520004&lng=-79.4082336&username=*your-username-here*

Consuming JSON Services

A complicated example with 2 services



Consuming JSON Services

More examples

URL of service request for static road map image

Zoom level (between 1 and 21)

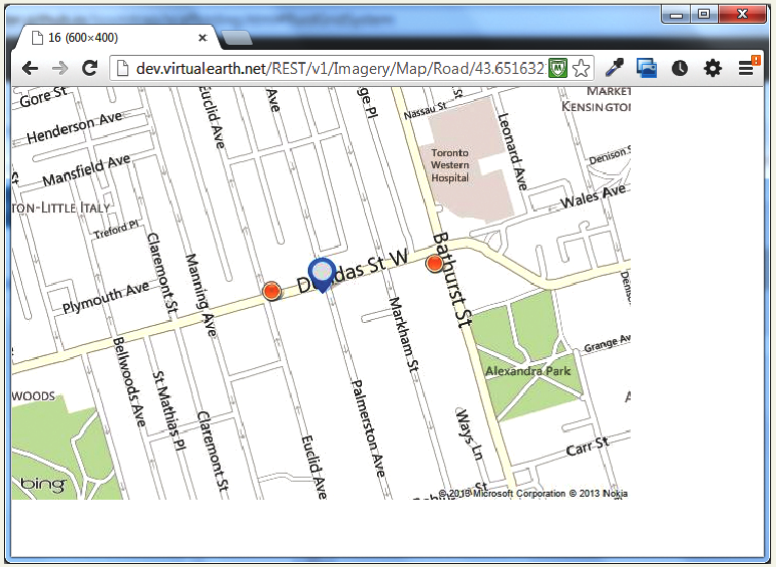
```
http://dev.virtualearth.net/REST/v1/Imagery/Map/Road/43.6516321,-79.4085317/16?
key=[your api key]
&mapSize=600,400
&pp=43.6516321,-79.4085317;66
&pp=43.6520854,-79.4061892;34
&pp=43.6516601,-79.4095859;34
```

Location (latitude and longitude) of center of map

Width and height of map in pixels

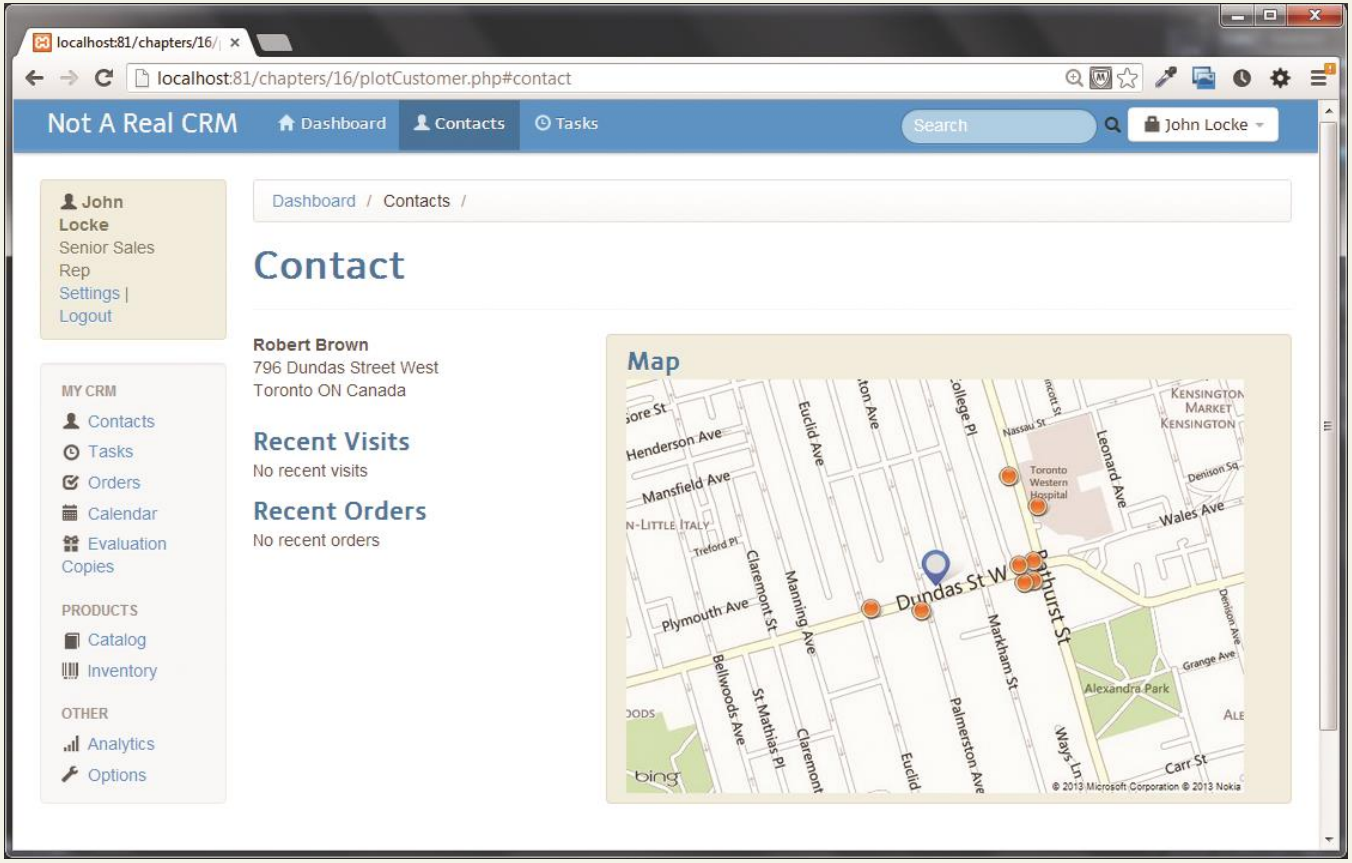
Location of marker (marker 66 = blue circle)

Location of other markers (amenities) with marker 34 = orange circle



Consuming JSON Services

More examples



CREATING WEB SERVICES

Creating Your Own Services

Web Services that is

Since REST services simply respond to HTTP requests, creating a PHP web service is only a matter of creating a page that responds to query string parameters and instead of returning HTML, it returns XML or JSON.

Our PHP page must also modify the Content-type header

Is important to recognize that not all web services are intended to be used by external clients. Many web services are intended to be consumed asynchronously by their own web pages via JavaScript

Creating an XML Service

Web Service, that is

The first service we will create will be one that returns data from our Book Customer Relations Management database.

To begin, we should determine the methods our service will support and the format of the requests.

crmServiceSearchBooks.php?**criteria**=yyy&**look**=zzz

- The **criteria** parameter will be used to specify what type of criteria we will use for the book search. This exercise will only support four values: imprint, category, look, and subcategory.
- The **look** parameter will be used to specify the actual value to search.

Creating an XML Service

Web Service, that is

For instance, if we had the following request:

crmServiceSearchBooks.php?criteria=subcategory&look=finance

It would be equivalent to the SQL search:

SELECT * FROM Books WHERE SubCategoryID=5

Creating an XML Service

Sample XML output

```
<?xml version="1.0" encoding="UTF-8"?>
<books>
  <book id="696">
    <isbns>
      <isbn10>0133140512</isbn10>
      <isbn13>9780133140514</isbn13>
    </isbns>
    <title>Entrepreneurial Finance</title>
    <authors>
      <author>
        <lastname>Adelman</lastname>
        <firstname>Philip</firstname>
        <institution>DeVry University</institution>
      </author>
      <author>
        <lastname>Marks</lastname>
        <firstname>Alan</firstname>
        <institution>DeVry University</institution>
      </author>
    </authors>
    <category>Business</category>
    <subcategory>Finance</subcategory>
    <year>2014</year>
    <imprint>Prentice Hall</imprint>
    <pagecount>448</pagecount>
    <description>For courses in ...</description>
  </book>
  <book>...</book>
  ...
</books>
```

LISTING 17.19 XML to be returned from crmServiceSearchBooks service

Creating an XML Service

Sample XML output

```php
<?php
require_once('includes/setup.inc.php');
require_once('includes/funcSearchBooks.inc.php');

// array to be used for query string validation and extraction
$acceptedCriteria = array('imprint','category','subcategory');
// parallel array to be used for constructing appropriate SQL
// criteria
$whereClause = array('Imprint=?','CategoryName=?','SubcategoryName=?');

// tell the browser to expect XML rather than HTML
// NOTE: comment this line out when debugging
header('Content-type: text/xml');
// check query string parameters and either output XML or error
// message (in XML)
if ( isCorrectQueryStringInfo($acceptedCriteria) ) {
   outputXML($dbAdapter, $acceptedCriteria, $whereClause);
}
else {
   echo '<errorResult>Error: incorrect query string values</errorResult>';
}
?>
```

LISTING 17.20 The crmServiceSearchBooks.php service

Creating an XML Service

Code details left as an exercise

There are different ways to output XML in PHP. One approach would be to simply echo XML within string literals to the response stream:

```
echo '<?xml version="1.0" encoding="UTF-8"?>’;
echo '<books>';

. . .
```

While this approach has the merit of familiarity, it will be up to the programmer to ensure that our page outputs well-formed and valid XML.

The alternate approach would be to use one of PHP’s XML extensions such as the XMLWriter object.

Creating an JSON Service

Web Service

Creating a JSON web service rather than an XML service is simply a matter of

- creating a JSON representation of an object
- setting the Content-type header to indicate the content will be JSON,
- and then outputting the JSON object

Since the built-in PHP json_encode() function does most of the work for us, our JSON service is simpler than the XML web service from the last section

Creating an JSON Service

Web Service

```php
<?php
require_once('includes/setup.inc.php');
require_once('includes/funcFindTitles.inc.php');

// Tell the browser to expect JSON rather than HTML
header('Content-type: application/json');

if ( isCorrectQueryStringInfo() ) {
   outputJSON($dbAdapter);
}
else {
   // put error message in JSON format
   echo '{"error": {"message":"Incorrect query string values"}}';
}

function outputJSON($dbAdapter) {
   // get query string values and set up search criteria
   $whereClause = 'Title Like ?';
   $look = $_GET['term'] . '%';

   // get the data from the database
   $bookGate = new BookTableGateway($dbAdapter);
   $results = $bookGate->findByFromJoins($whereClause, Array($look) );

   // output the JSON for the retrieved book data
   echo json_encode($results);

   $dbAdapter->closeConnection();
}
```

LISTING 17.24 JSON crmServiceFindTitleMatches service

Output headers

Function to create JSON From the Database, based on query

Creating an JSON Service

Web Service

For this function to work, the class of the custom object being converted must provide its own implementation of the JsonSerializable interface. T

his interface contains only the single method jsonSerialize().

In this web service, we are outputting JSON for objects of the Book class, so this class will need to implement this method

Creating an JSON Service

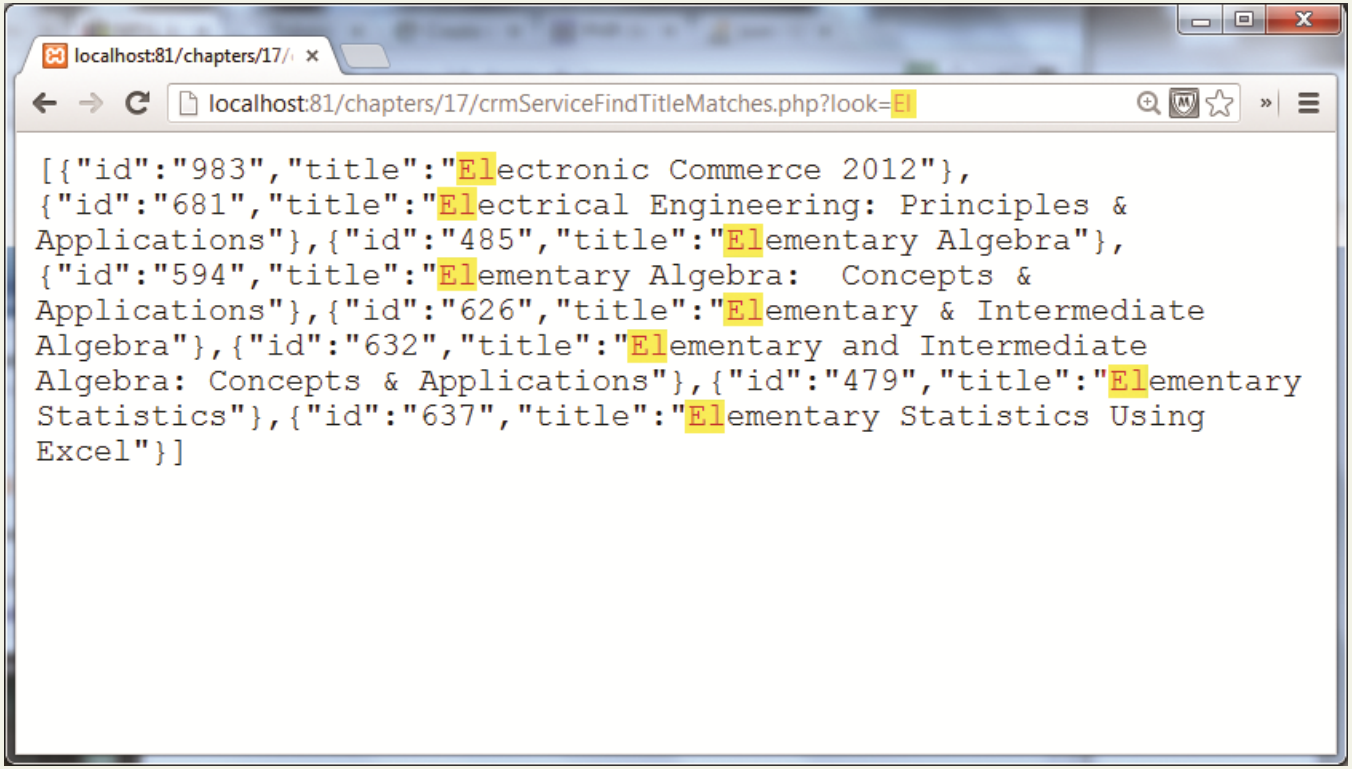
Web Service

```
class Book extends DomainObject implements JsonSerializable
{
    ...
    /*
     This method is called by the json_encode() function that is
     part of PHP
    */
    public function jsonSerialize() {
        return ['id' => $this->ID, 'value' => $this->Title];
    }
}
```

LISTING 17.25 Adding jsonSerializable() to Book class

Creating an JSON Service

Testing our service in the browser



INTERACTING ASYNCHRONOUSLY WITH WEB SERVICES

Tying it all together

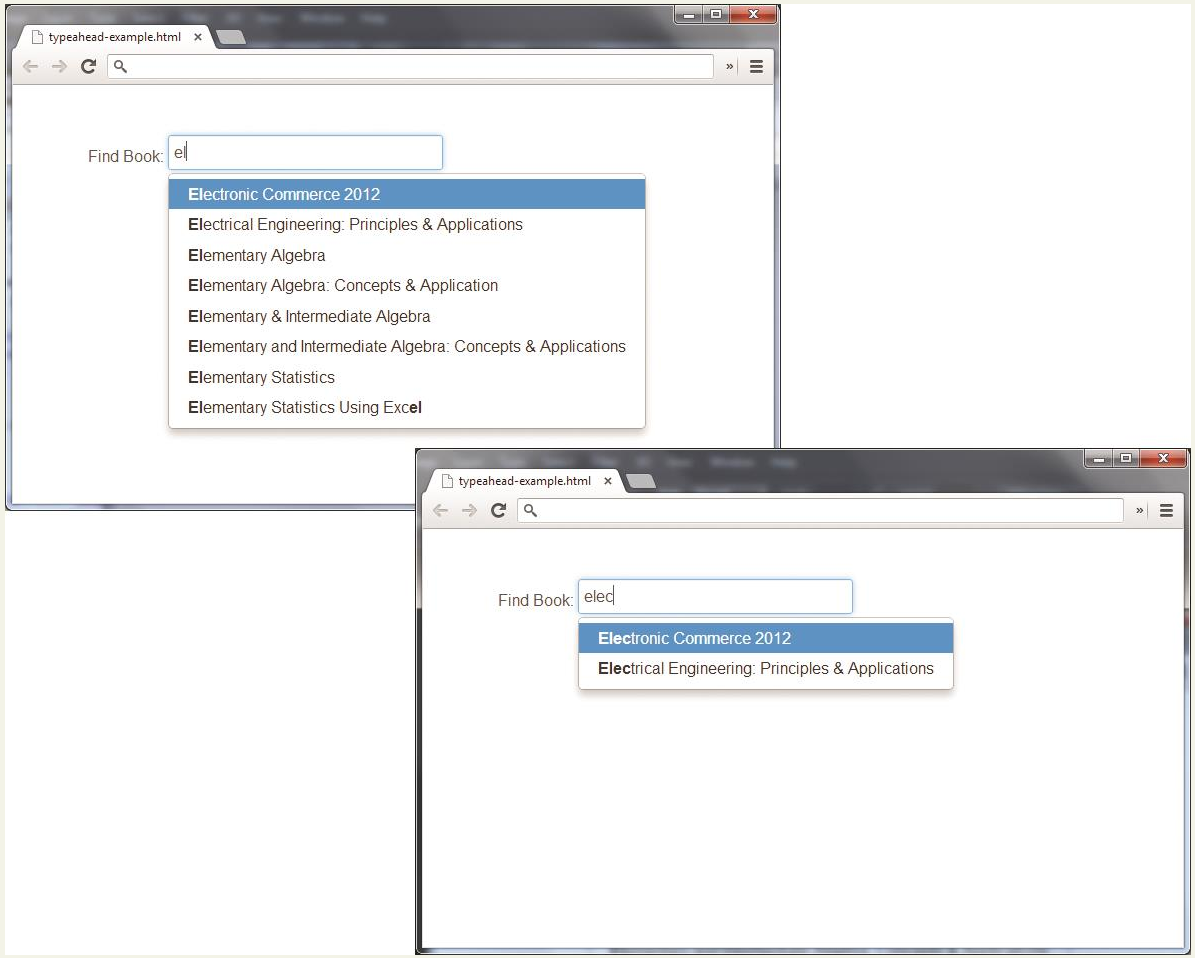
Consuming services asynchronously

Although it's possible to consume web services in PHP, it's far more common to consume those services asynchronously using JavaScript.

When using client-side requests for third-party services, there's also the advantage of distributing requests to each client rather then making all requests from your own server's IP address.

Consuming your own service

Autocomplete example



Consuming your own service

Autocomplete example

The code listens for changes to an input box with id *search*. With each change the code makes an asynchronous get request to the *source* URL, which in this case is the script in Listing 17.24 that returns JSON results. Those results are then used by autocomplete to display nicely underneath the input box.

```
$("#search").autocomplete({
    // URL, query string term=
    source:"crmServiceFindTitleMatches.php?",
    minlength:1,    //how many characters required before querying
    delay:1         //delay to prevent multiple events
});
```

LISTING 17.26 Autocomplete jQuery plug-in refreshes the list of suggestions to choose from

Using Google Maps

A popular mashup platform

Consider our photo-sharing website. We will show you how to build a map view that plots user photos onto a map using the location information associated with the image.

To begin using Google Maps, you must do three things

1. Include the Google Maps libraries in the <head> section of your page.
2. Define elements that will contain the maps.
3. Initialize instances of google.maps.Map (we will call it Map) in JavaScript and associate them with the elements.

Using Google Maps

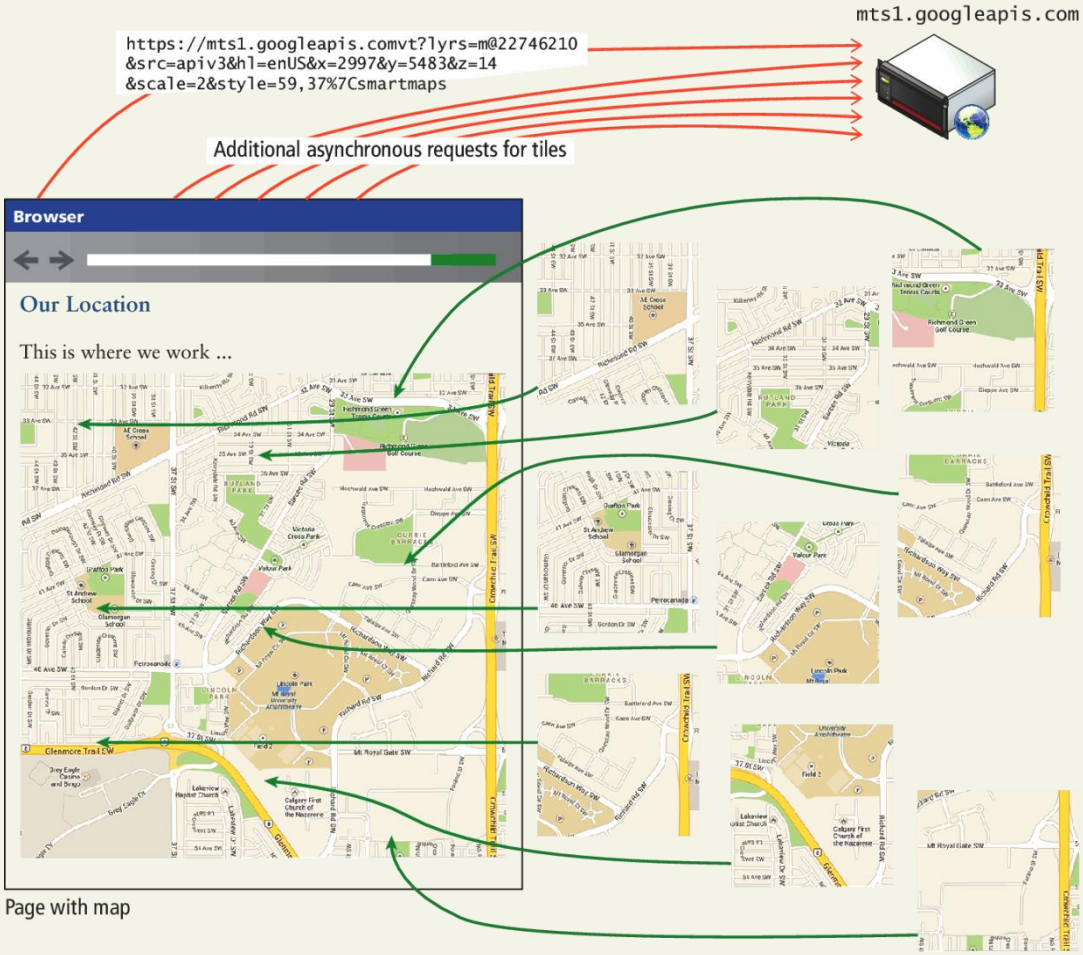
A popular mashup platform

```
<!DOCTYPE html>
<html>
<head>
   <script src="https://maps.googleapis.com/maps/api/js?v=3.
      exp&sensor=false"></script>
   <script src="http://code.jquery.com/jquery.js"></script>
</head>
<body>

<?php

function getGoogleMap($imageID, $latitude, $longitude) {
          return "<script>
          $(document).ready(function() {
             var map$imageID;
             var mapOptions = {
                 zoom:14,
                 center:new google.maps.LatLng($latitude,$longitude),
                 mapTypeId:google.maps.MapTypeId.ROADMAP
             };
             map$imageID = new google.maps.Map(
                          document.getElementById
                          ('map-canvas$imageID'),mapOptions);

          });
          </script>
         <div style='width: 400px; height: 400px;'
         class='map-canvas' id='map-canvas$imageID'></div>";
}

echo getGoogleMap(1, 51.011179,-114.132866);
?>

</body>
</html>
```

LISTING 17.27 Web page to output one map centered on Mount Royal University

Using Google Maps

Under the hood there are lots of asynchronous requests



Using Google Maps

A sample mashup