

XML Processing and Web Services

Chapter 19

Chapter 19

1

XML Overview

2

XML Processing

3

JSON

4

Overview of
Web Services

5

Consuming Web
Services in PHP

6

Creating Web
Services

7

Interacting
Asynchronously
with Web Services

8

Summary

Chapter 19

1

XML Overview

2

XML Processing

3

JSON

4

Overview of
Web Services

5

Consuming Web
Services in PHP

6

Creating Web
Services

7

Interacting
Asynchronously
with Web Services

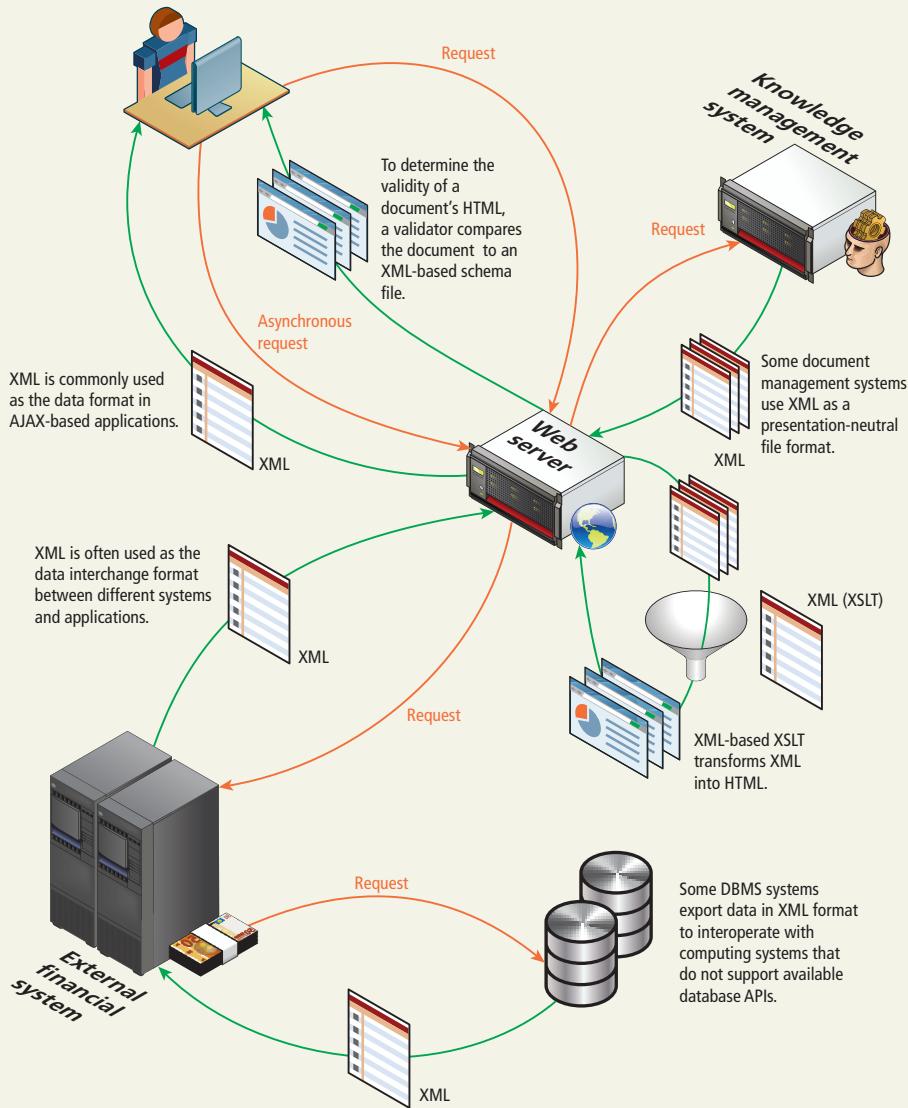
8

Summary

XML Overview

- Recall XML is a markup language, but unlike HTML, XML can be used to mark up any type of data
- One key benefit of XML data is that as plain text, it can be read and transferred between applications and different operating systems
- XML is used on the web server to communicate asynchronously with the browser
- used as a data interchange format for moving information between systems

XML Overview



XML Overview

Well-Formed XML

- Element names are composed of any of the valid characters
- Element names can't start with a number.
- There must be a single-root element.
- All elements must have a closing element (or be self-closing).
- Elements must be properly nested.
- Elements can contain attributes.
- Attribute values must always be within quotes.
- Element and attribute names are case sensitive.

XML Overview

Well-Formed XML Simplified Example

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<art>
    <painting id="290">
        <title>Balcony</title>
        <artist>
            <name>Manet</name>
            <nationality>France</nationality>
        </artist>
        <year>1868</year>
        <medium>Oil on canvas</medium>
    </painting>
</art>
```

XML Overview

Valid XML

A valid XML document is one that is well formed and whose element and content conform to the rules of either its document type definition (DTD) or its schema

- DTDs tell the XML parser which elements and attributes to expect in the document as well as the order and nesting of those elements
- A DTD can be defined within an XML document or within an external file.

XML Overview

Example Document Type Definition (DTD)

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE art [
<!ELEMENT art (painting*)>
<!ELEMENT painting (title,artist,year,medium)>
<!ATTLIST painting id CDATA #REQUIRED>
<!ELEMENT title (#PCDATA)>
<!ELEMENT artist (name,nationality)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT nationality (#PCDATA)>
<!ELEMENT year (#PCDATA)>
<!ELEMENT medium (#PCDATA)>
]>
<art>
...
</art>
```

Chapter 19

1

XML Overview

2

XML Processing

3

JSON

4

Overview of
Web Services

5

Consuming Web
Services in PHP

6

Creating Web
Services

7

Interacting
Asynchronously
with Web Services

8

Summary

XML Processing

XML Processing in JavaScript

- The in-memory approach , which involves reading the entire XML file into memory
- The event or pull approach , which lets you pull in just a few elements or lines at a time

XML Processing

XML Processing in JavaScript

```
if (window.XMLHttpRequest) {  
    // code for IE7+, Firefox, Chrome, Opera, Safari  
    var xmlhttp = new XMLHttpRequest()  
}  
else {  
    // code for old versions of IE (optional)  
    var xmlhttp = new ActiveXObject("Microsoft.XMLHTTP");  
}  
  
// load the external XML file  
xmlhttp.open("GET","art.xml",false);  
xmlhttp.send();  
var xmlDoc = xmlhttp.responseXML;  
// now extract a node list of all <painting> elements  
var paintings = xmlDoc.getElementsByTagName("painting");
```

XML Processing

XML Processing in jQuery

```
var art = '<?xml version="1.0" encoding="ISO-8859-1"?>';  
  
art += '<art><painting id="290"><title>Balcony ... </art>';  
  
// use jQuery parseXML() function to create the DOM object  
  
var xmlDoc = $.parseXML(art);  
  
// convert DOM object to jQuery object  
  
var xml = $(xmlDoc);  
  
// find all the painting elements  
  
var paintings = xml.find("painting"); //...
```

XML Processing

XML Processing in PHP

- The DOM extension
- SimpleXML extension,
- XML parse
- XMLReader
- Combining XMLReader and SimpleXML

XML Processing

SimpleXML

```
<?php  
  
$filename = 'art.xml';  
  
if (file_exists($filename)) {  
  
    $art = simplexml_load_file($filename);  
  
    // access a single element  
  
    $painting = $art->painting[0];  
  
    echo '<h2>' . $painting->title . '</h2>';
```

XML Processing

XPath with SimpleXML

```
$art = simplexml_load_file($filename);  
  
$titles = $art->xpath('/art/painting/title');  
  
foreach ($titles as $t) {  
  
    echo $t . '<br/>';  
  
}  
  
$names = $art->xpath('/art/painting[year>1800]/artist/name');  
  
foreach ($names as $n) {  
  
    echo $n . '<br/>';  
  
}
```

XML Processing

XMLReader

```
$filename = 'art.xml';

if (file_exists($filename)) {

    // create and open the reader

$reader = new XMLReader();

$reader->open($filename);

    // loop through the XML file

    while ( $reader->read() ) {

        //...
```

XML Processing

Combining XMLReader with SimpleXML

```
//...
while($reader->read()) {
    $nodeName = $reader->name;
    if ($reader->nodeType == XMLREADER::ELEMENT
        && $nodeName =='painting') {
        // create a SimpleXML object from the current painting node
        $doc = new DOMDocument('1.0', 'UTF-8');
        $painting = simplexml_import_dom($doc->importNode(
            $reader->expand(),true));
        // now have a single painting
        echo '<h2>' . $painting->title . '</h2>';
        echo '<p>By ' . $painting->artist->name . '</p>';
    }
}
```

Chapter 19

1

XML Overview

2

XML Processing

3

JSON

4

Overview of
Web Services

5

Consuming Web
Services in PHP

6

Creating Web
Services

7

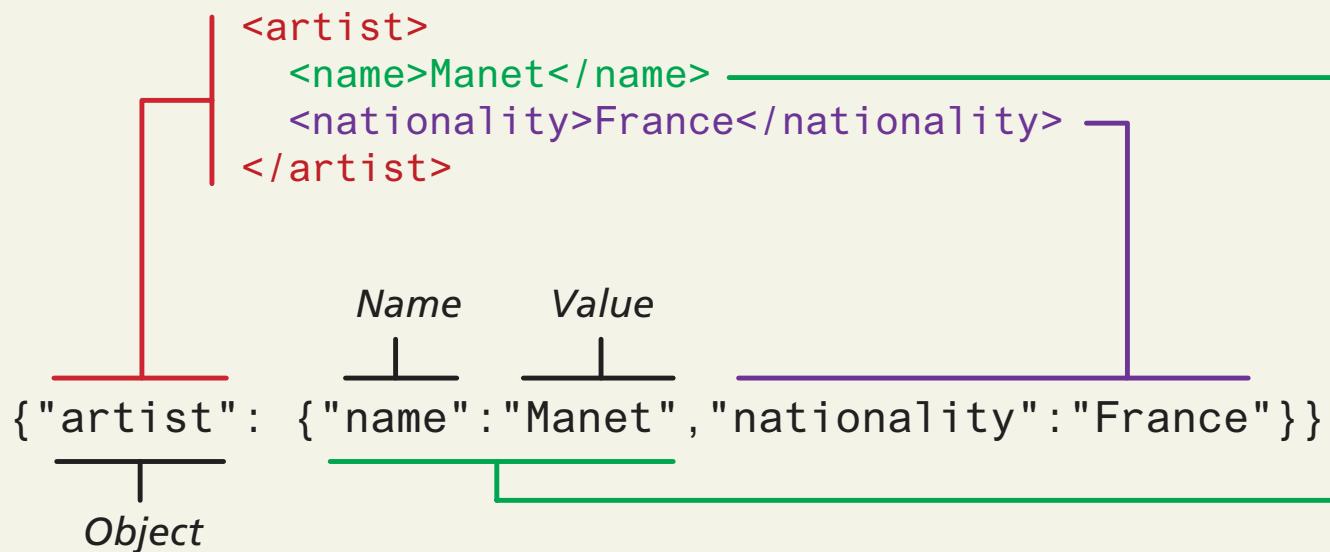
Interacting
Asynchronously
with Web Services

8

Summary

JSON

Sample JSON



JSON

Using JSON in Javascript

```
var text = '{"artist": {"name":"Manet","nationality":"France"}}';  
var a = JSON.parse(text);  
alert(a.artist.nationality);
```

JSON

Using JSON in PHP

```
<?php  
  
// convert JSON string into PHP object  
  
$text = '{"artist": {"name":"Manet","nationality":"France"} }';  
  
$anObject = json_decode($text);  
  
// check for parse errors  
  
if (json_last_error() == JSON_ERROR_NONE) {  
  
    echo $anObject->artist->nationality;  
  
}  
  
?>
```

Chapter 19

1

XML Overview

2

XML Processing

3

JSON

4

Overview of
Web Services

5

Consuming Web
Services in PHP

6

Creating Web
Services

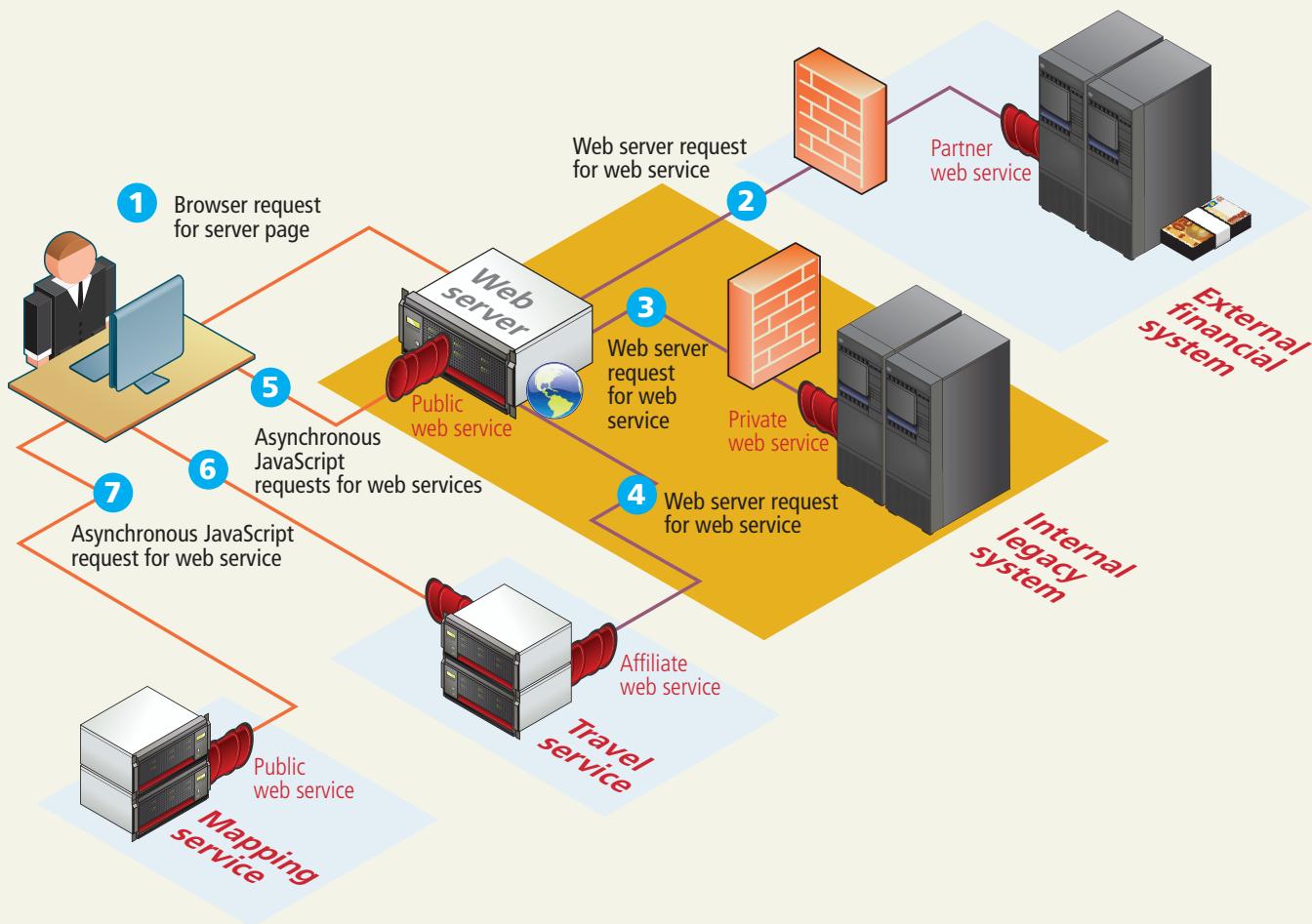
7

Interacting
Asynchronously
with Web Services

8

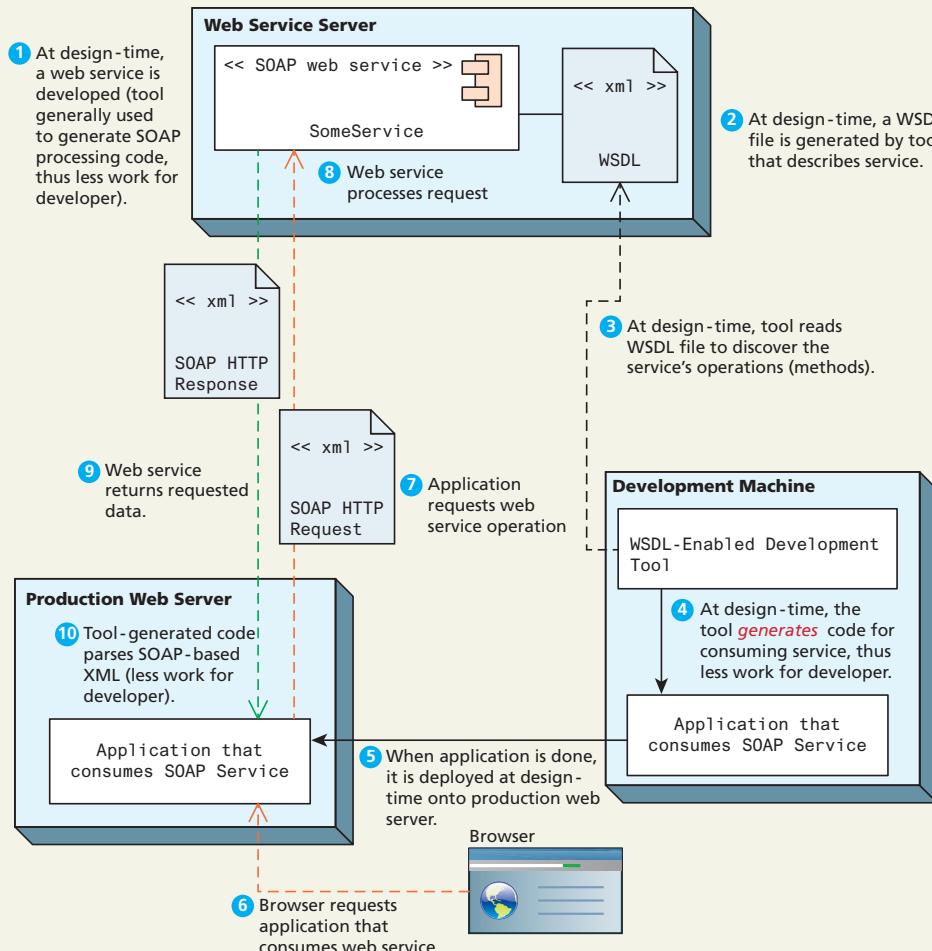
Summary

Overview of Web Services



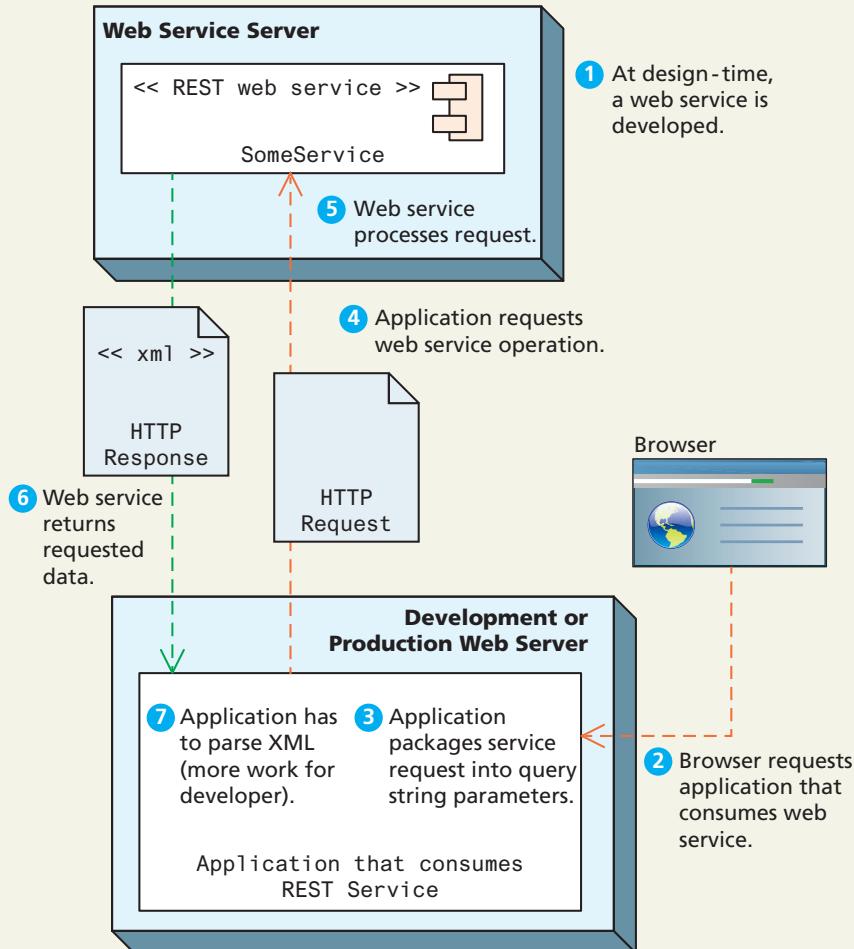
Overview of Web Services

SOAP Services



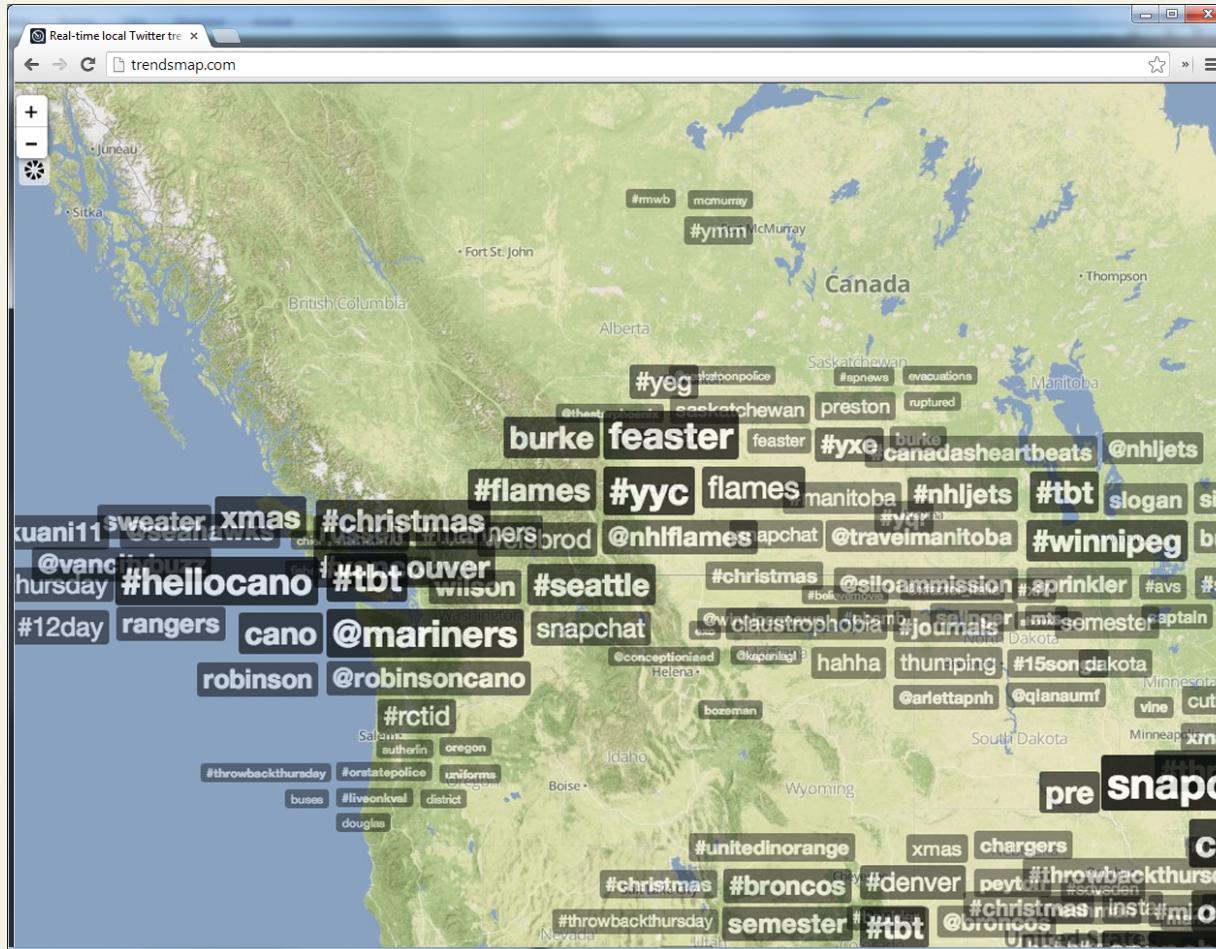
Overview of Web Services

REST Services



Overview of Web Services

An Example Web Service



Overview of Web Services

Identifying and Authenticating Service Requests

- 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.

API Keys

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

Chapter 19

1

XML Overview

2

XML Processing

3

JSON

4

Overview of
Web Services

5

Consuming Web
Services in PHP

6

Creating Web
Services

7

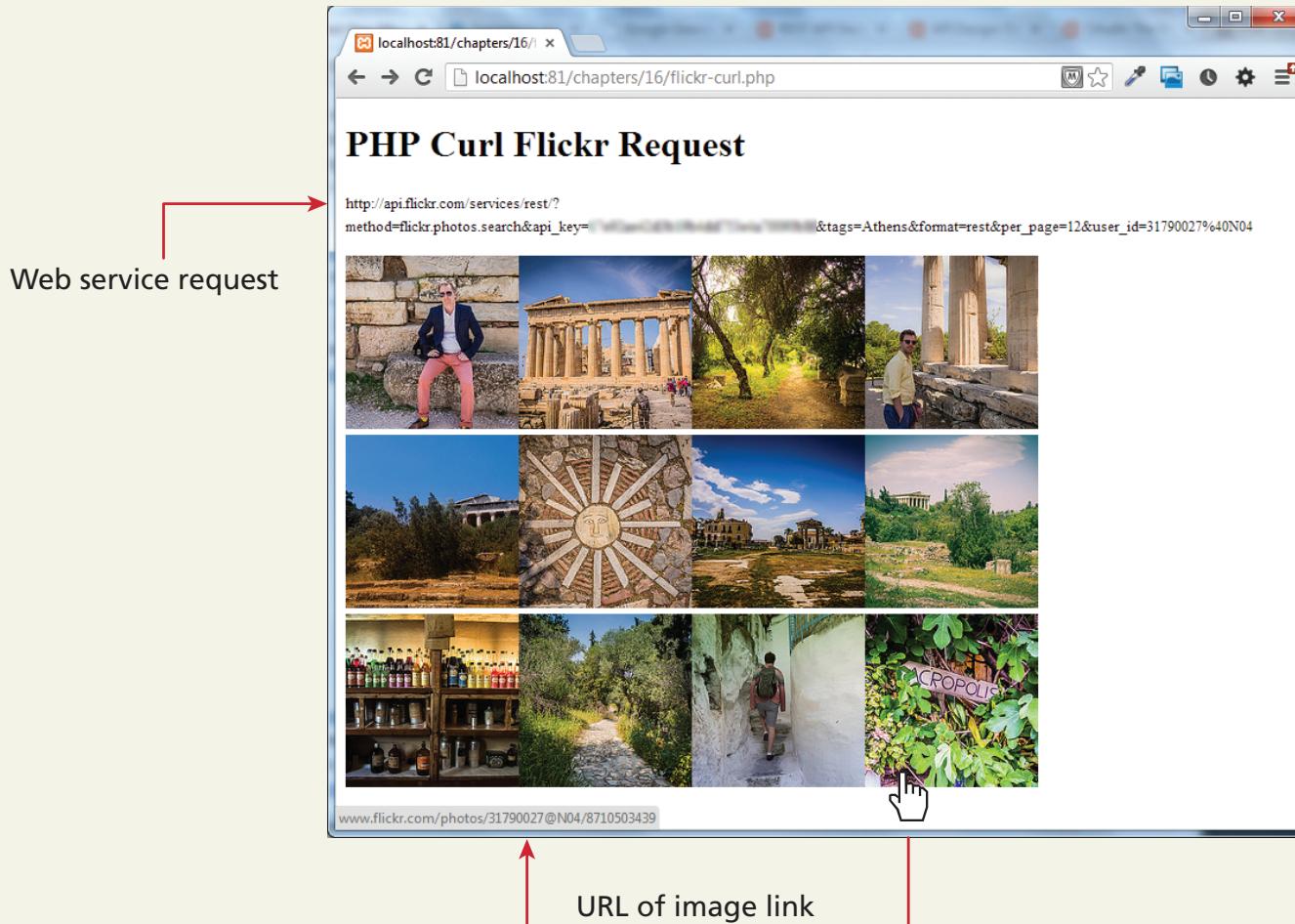
Interacting
Asynchronously
with Web Services

8

Summary

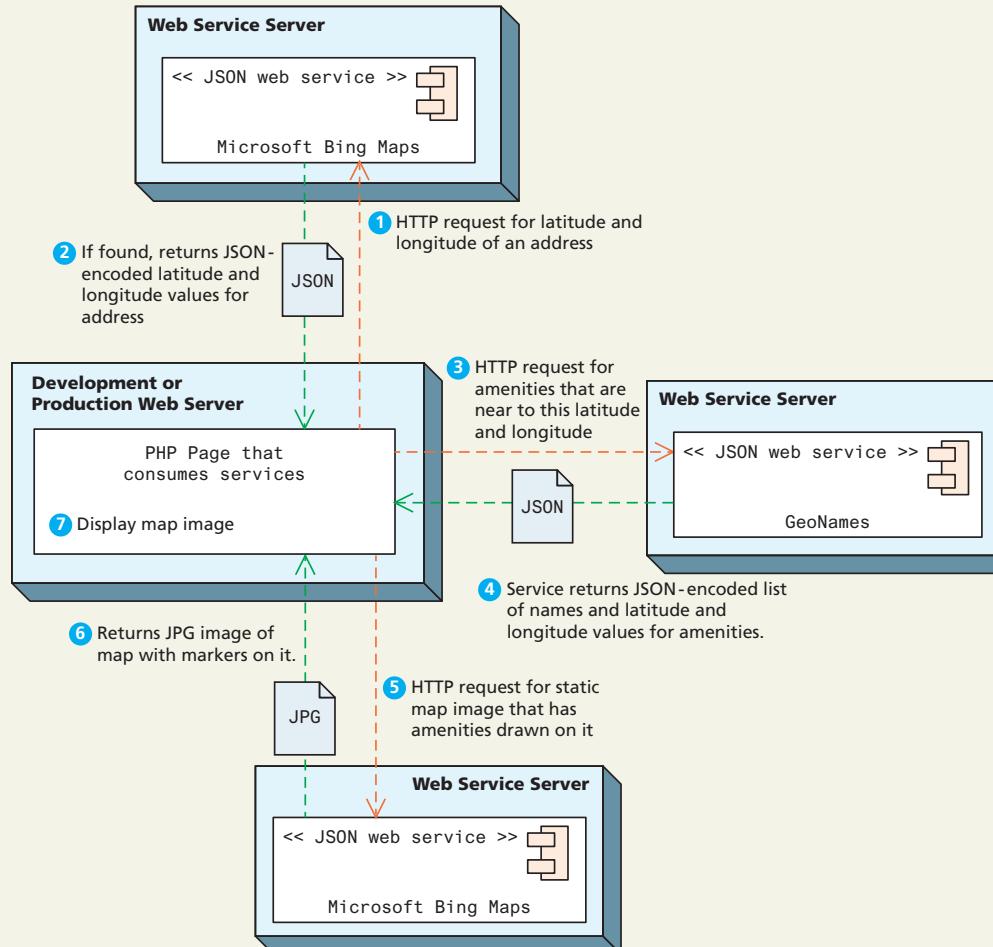
Consuming Web Services in PHP

Consuming an XML Web Service



Consuming Web Services in PHP

Consuming a JSON Web Service



Consuming Web Services in PHP

Consuming a JSON Web Service

URL of service request for static road map image
[&mapSize=600,400](http://dev.virtualearth.net/REST/v1/Imagery/Map/Road/43.65163,-79.40853/16?key=[your api key])&key=[your api key]

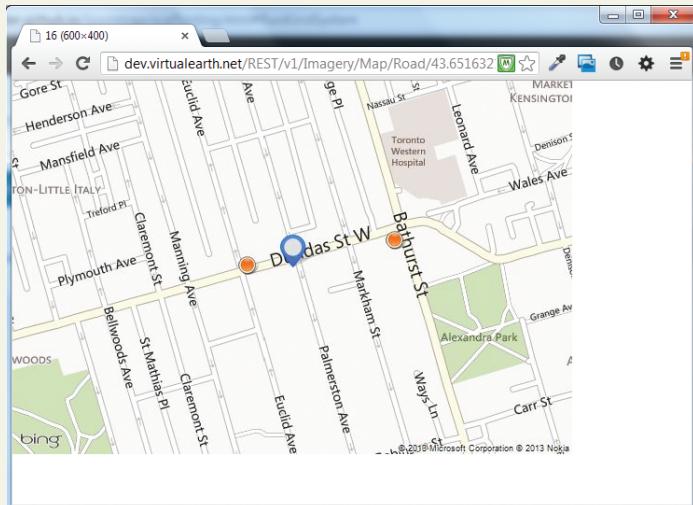
Zoom level (between 1 and 21)
16

Width and height of map in pixels
600,400

Location (latitude and longitude) of center of map
43.65163, -79.40853

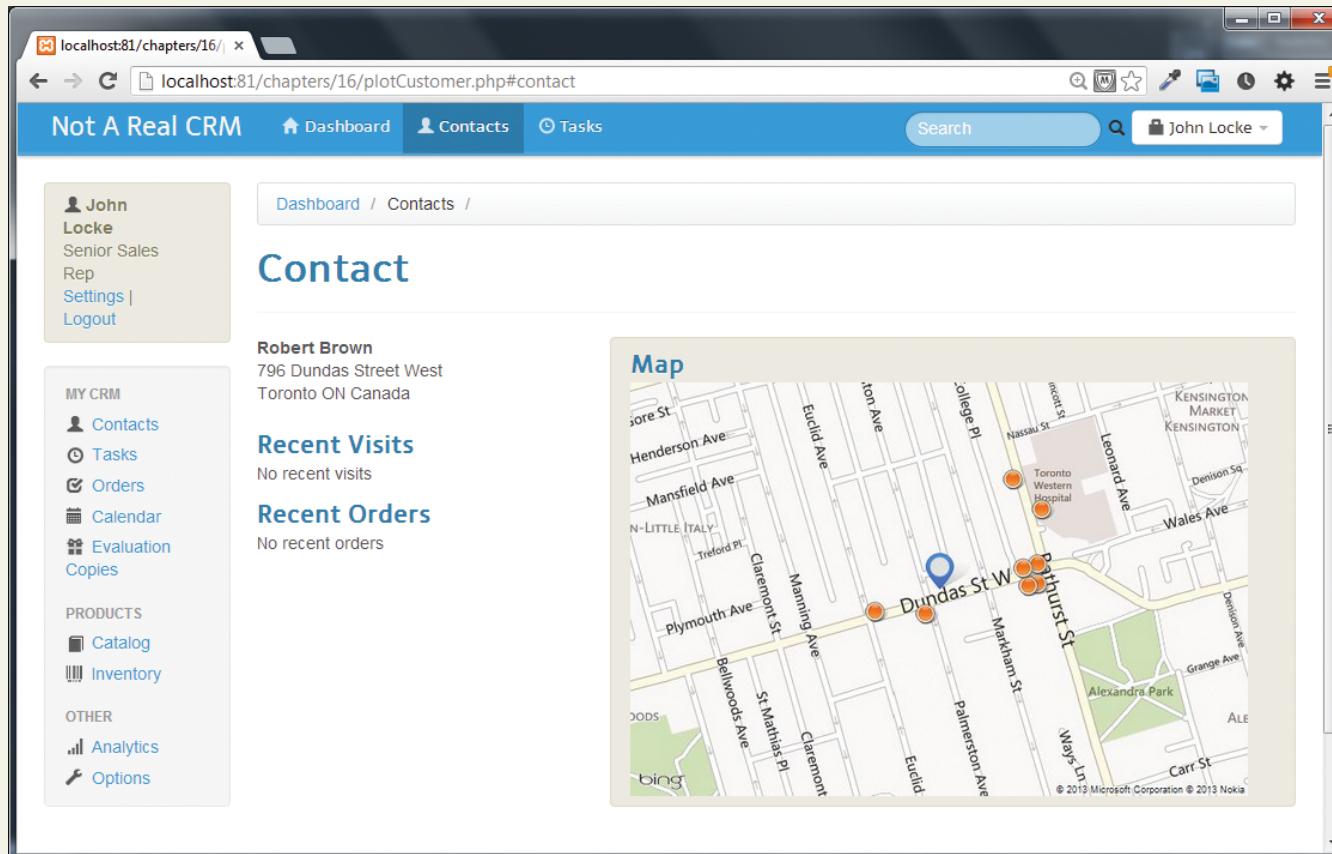
Location of marker (marker 66 = blue circle)
43.65163, -79.40853;66

Location of other markers (amenities) with marker 34 = orange circle
43.65208, -79.40618;34
43.65166, -79.40958;34



Consuming Web Services in PHP

Consuming a JSON Web Service



Chapter 19

1

XML Overview

2

XML Processing

3

JSON

4

Overview of
Web Services

5

Consuming Web
Services in PHP

6

Creating Web
Services

7

Interacting
Asynchronously
with Web Services

8

Summary

Creating Web Services

Creating a JSON Web Service

- Consider the URL and format of requests
- Tell the browser to expect JSON rather than HTML
 - `Header('Content-Type: application/json');`
- Use `json_encode()` to format.
- Implement `JsonSerializable`

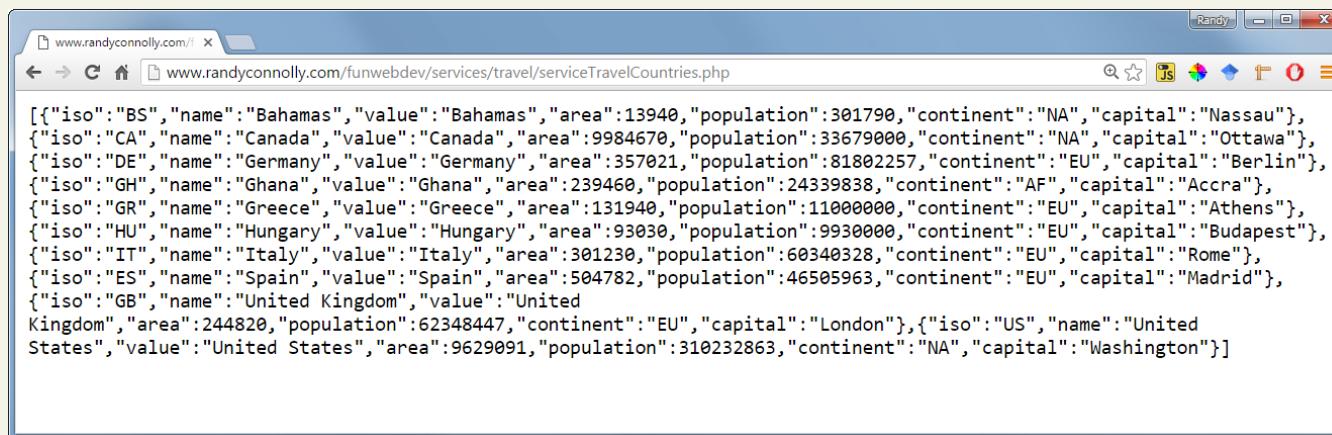
Creating Web Services

Creating a JSON Web Service

```
class Country extends DomainObject implements JsonSerializable
{
    ...
    /*
        This method is called by the json_encode() function that is part of PHP
    */
    public function jsonSerialize() {
        return ['iso' => $this->ISO,
                'name' => $this->CountryName,
                'value' => $this->CountryName,
                'area' => $this->Area,
                'population' => $this->Population,
                'continent' => $this->Continent,
                'capital' => $this->Capital
            ];
    }
}
```

Creating Web Services

Creating a JSON Web Service



The screenshot shows a Microsoft Internet Explorer window with the title bar "www.randyconnolly.com/" and the address bar "www.randyconnolly.com/funwebdev/services/travel/serviceTravelCountries.php". The main content area displays a JSON array of country data:

```
[{"iso": "BS", "name": "Bahamas", "value": "Bahamas", "area": 13940, "population": 301790, "continent": "NA", "capital": "Nassau"}, {"iso": "CA", "name": "Canada", "value": "Canada", "area": 9984670, "population": 33679000, "continent": "NA", "capital": "Ottawa"}, {"iso": "DE", "name": "Germany", "value": "Germany", "area": 357021, "population": 81802257, "continent": "EU", "capital": "Berlin"}, {"iso": "GH", "name": "Ghana", "value": "Ghana", "area": 239460, "population": 24339838, "continent": "AF", "capital": "Accra"}, {"iso": "GR", "name": "Greece", "value": "Greece", "area": 131940, "population": 11000000, "continent": "EU", "capital": "Athens"}, {"iso": "HU", "name": "Hungary", "value": "Hungary", "area": 93030, "population": 9930000, "continent": "EU", "capital": "Budapest"}, {"iso": "IT", "name": "Italy", "value": "Italy", "area": 301230, "population": 60340328, "continent": "EU", "capital": "Rome"}, {"iso": "ES", "name": "Spain", "value": "Spain", "area": 504782, "population": 46505963, "continent": "EU", "capital": "Madrid"}, {"iso": "GB", "name": "United Kingdom", "value": "United Kingdom", "area": 244820, "population": 62348447, "continent": "EU", "capital": "London"}, {"iso": "US", "name": "United States", "value": "United States", "area": 9629091, "population": 310232863, "continent": "NA", "capital": "Washington"}]
```

Chapter 19

1

XML Overview

2

XML Processing

3

JSON

4

Overview of
Web Services

5

Consuming Web
Services in PHP

6

Creating Web
Services

7

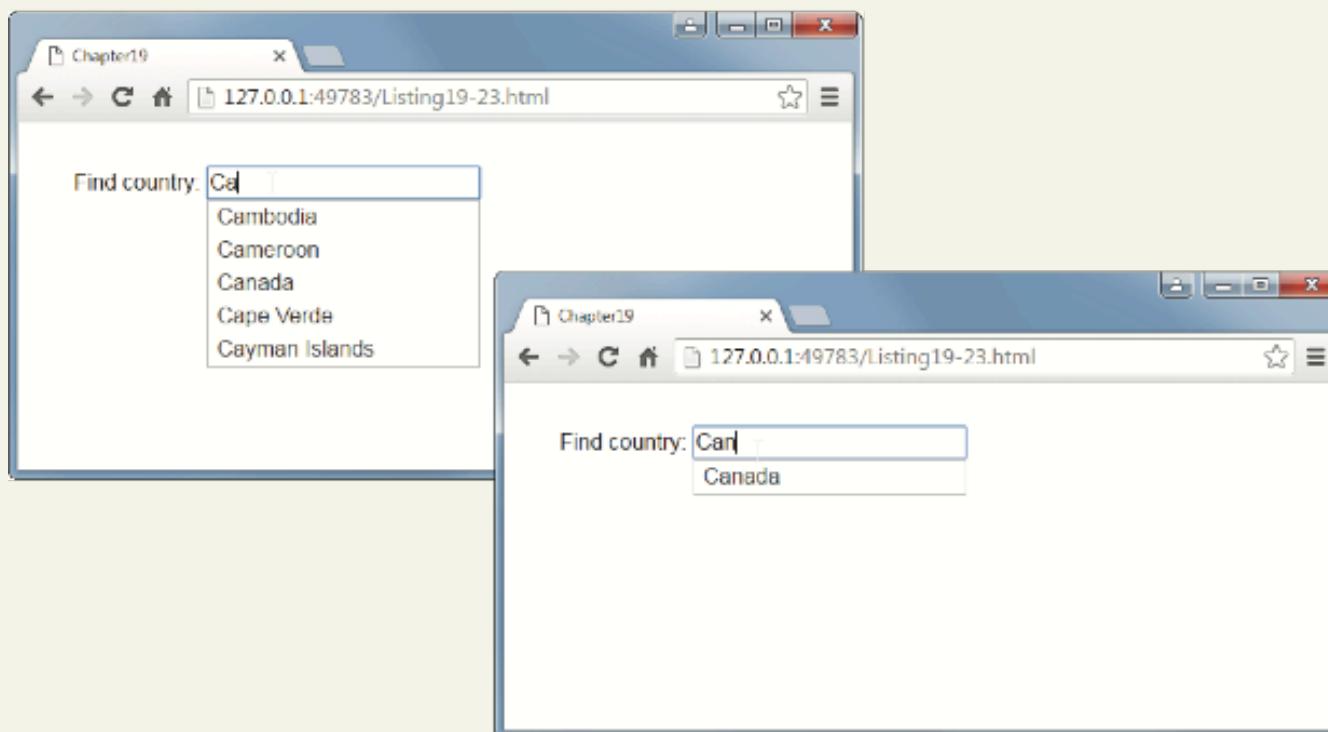
Interacting
Asynchronously
with Web Services

8

Summary

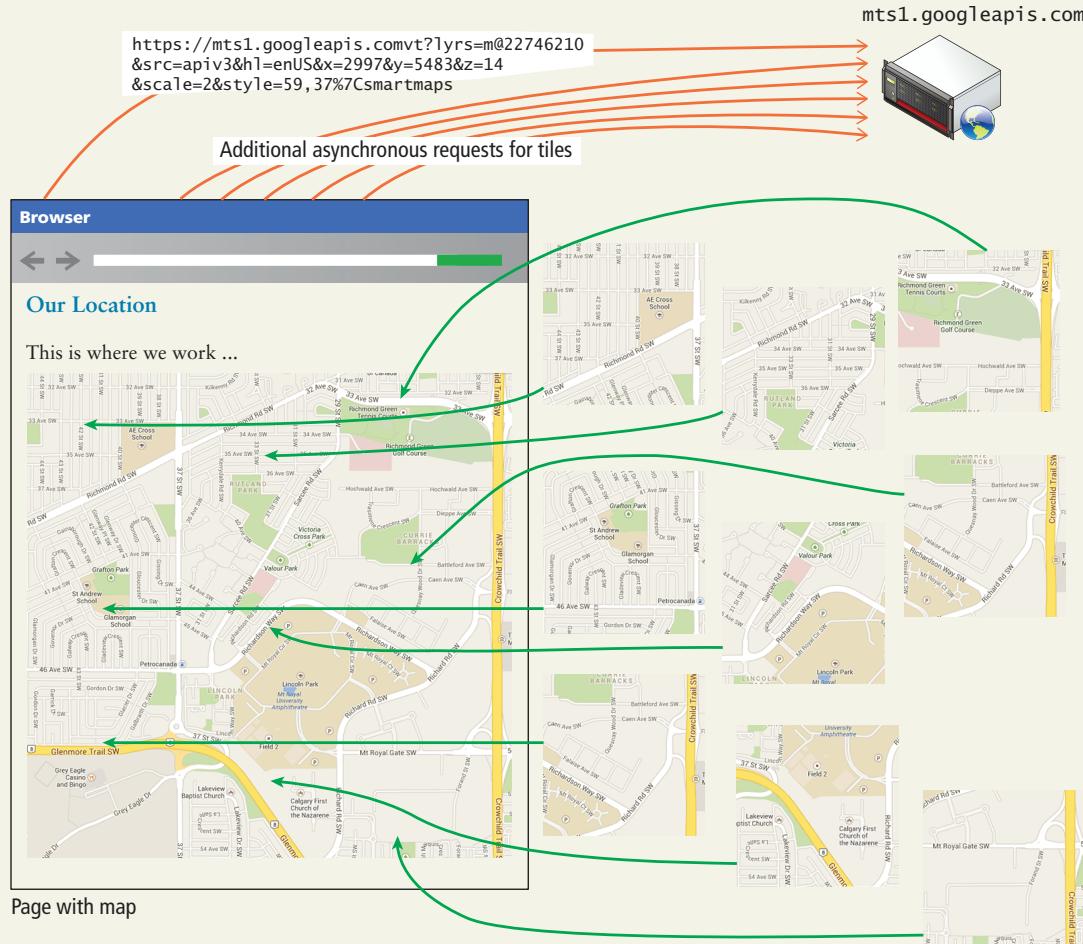
Interacting Asynchronously with Web Services

Consuming Your Own Service



Interacting Asynchronously with Web Services

Using Google Maps



Interacting Asynchronously with Web Services

Using Google Maps

```
<script type='text/javascript'  
src='https://maps.googleapis.com/maps/api/js?key=yourkey'></script>
```

```
<style>  
/* map element needs a styled size otherwise it doesn't appear at all */  
#map {  
height: 500px;  
width: 600px  
}  
</style>
```

Interacting Asynchronously with Web Services

Using Google Maps

```
<script>
$(function() {
    // hard-coded latitude and longitude for demonstration purposes
    var ourLatLong = {lat: 51.011179 , lng: -114.132866 };
    var ourMap = new google.maps.Map(document.getElementById('map'), {
        center: ourLatLong,
        scrollwheel: false,
        zoom: 14
    });
});
</script>
</head>
<body>
    <h2>Our Location</h2>
    <h3>This is where we work ... </h3>
    <div id="map"></div>
</body>
</html>
```

Chapter 19

1

XML Overview

2

XML Processing

3

JSON

4

Overview of
Web Services

5

Consuming Web
Services in PHP

6

Creating Web
Services

7

Interacting
Asynchronously
with Web Services

8

Summary

Summary

Key Terms

authentication	REST	valid XML
DOM extension	reverse geocoding	web services
event or pull approach	root element	well-formed XML
geocoding	service	XML declaration
identity	service-oriented	XML parser
in-memory approach	architecture	XMLReader
JSON	service-oriented	XPath
mashup	computing	XSLT
Node	SimpleXML	

Summary

Questions?