XML, AJAX AND WEB SERVICES

Introduction to xml

XML (Extensible Markup Language) is a markup language designed to store and transport data. It is both human-readable and machine-readable, making it a versatile tool for data representation.

Features of XML

Self-descriptive: XML uses tags to describe data, making it easy to understand the structure and meaning of the information.

Platform-independent: XML is text-based and can be used across different hardware and software environments.

Extensible: Users can define their own tags, making XML highly flexible.

Hierarchical Structure: XML documents have a tree-like structure, making it easy to represent complex data relationships.

Basic Syntax Rules

1. **Tags:** XML elements are defined using tags.

<tagname>Content</tagname>

2. **Root Element:** An XML document must have a single root element that contains all other elements.

<root>

<child>Content</child>

</root>

3. **Case Sensitivity:** XML tags are case-sensitive.

<Tag>Content</Tag> <!-- Valid -->

<tag>Content</tag> <!-- Different from above -->

4. **Attributes:** Elements can have attributes to provide additional information.

<element attribute="value">Content</element>

5. **Well-formed**: An XML document must follow proper nesting and closing of tags.

<root>

<child>Content</child>

</root>

Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<college>
  <department>
    <name>Computer Science</name>
    <course>
      <code>CS101</code>
      <title>Introduction to Programming</title>
      <credits>3</credits>
    </course>
  </department>
  <student>
    <id>S001</id>
    <name>Ram joshi</name>
    <major>Computer Science</major>
    <enrolledCourse>CS101</enrolledCourse>
  </student>
</college>
```

Introduction to XML Validation

XML Validation is the process of checking an XML document to ensure it conforms to a predefined structure and set of rules. This is crucial for ensuring data integrity and consistency, especially when exchanging data between systems or applications. XML validation can be performed using two primary methods:

- Document Type Definition (DTD)

- XML Schema (XSD)

Document Type Definition (DTD)

Document Type Definition (DTD) is a set of markup declarations that define the structure and the legal elements and attributes of an XML document. A DTD can be declared inside an XML document (internal DTD) or outside an XML document (external DTD).

Key Features of DTD:

Element Declarations: Define the elements in the XML document.

Attribute Declarations: Define the attributes of elements.

Entity Declarations: Define reusable pieces of text.

Notation Declarations: Define notations for non-XML data.

Advantages:

Simplicity and ease of use.

Widely supported by XML parsers.

Limitations:

Limited data types (only string and enumeration types).

No support for namespaces.

Less expressive than XML Schema.

Internal DTD

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE college [
  <!ELEMENT college (department, student)>
  <!ELEMENT department (name, course)>
  <!ELEMENT name (#PCDATA)>
  <!ELEMENT course (code, title, credits)>
  <!ELEMENT code (#PCDATA)>
```

```
  <!ELEMENT title (#PCDATA)>
  <!ELEMENT credits (#PCDATA)>
  <!ELEMENT student (id, name, major, enrolledCourse)>
  <!ELEMENT id (#PCDATA)>
  <!ELEMENT major (#PCDATA)>
  <!ELEMENT enrolledCourse (#PCDATA)>
]>
<college>
  <department>
    <name>Computer Science</name>
    <course>
      <code>CS101</code>
      <title>Introduction to Programming</title>
      <credits>3</credits>
    </course>
  </department>
  <student>
    <id>S001</id>
    <name>Ram Joshi</name>
    <major>Computer Science</major>
    <enrolledCourse>CS101</enrolledCourse>
  </student>
</college>
```

External DTD

An external DTD is a set of rules and structure for an XML document, defined in a separate file rather than within the XML document itself.

College.dtd (file name)

```
<!ELEMENT college (department, student)>
<!ELEMENT department (name, course)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT course (code, title, credits)>
<!ELEMENT code (#PCDATA)>
<!ELEMENT title (#PCDATA)>
<!ELEMENT credits (#PCDATA)>
<!ELEMENT student (id, name, major, enrolledCourse)>
<!ELEMENT id (#PCDATA)>
<!ELEMENT major (#PCDATA)>
<!ELEMENT enrolledCourse (#PCDATA)>
```

College.xml (file Name)

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE college SYSTEM "college.dtd">
<college>
 <department>
  <name>Computer Science</name>
  <course>
   <code>CS101</code>
   <title>Introduction to Programming</title>
   <credits>3</credits>
  </course>
 </department>
 <student>
  <id>S001</id>
  <name>Ram Joshi</name>
  <major>Computer Science</major>
```

```
    <enrolledCourse>CS101</enrolledCourse>
  </student>
</college>
```

Xml schema

An XML Schema is a blueprint for how an XML document should be structured. It tells you what elements and attributes are allowed, what data types they can have, and how they should be arranged, ensuring the document follows specific rules.

College.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">



<!-- Define the college element which contains department and student -->
  <xs:element name="college">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="department" type="departmentType"/>
        <xs:element name="student" type="studentType"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <!-- Define the departmentType complex type -->
  <xs:complexType name="departmentType">
    <xs:sequence>
      <xs:element name="name" type="xs:string"/>
      <xs:element name="course" type="courseType"/>
```

```
    </xs:sequence>
  </xs:complexType>

  <!-- Define the courseType complex type -->
  <xs:complexType name="courseType">
    <xs:sequence>
      <xs:element name="code" type="xs:string"/>
      <xs:element name="title" type="xs:string"/>
      <xs:element name="credits" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>

<!-- Define the studentType complex type -->
  <xs:complexType name="studentType">
    <xs:sequence>
      <xs:element name="id" type="xs:string"/>
      <xs:element name="name" type="xs:string"/>
      <xs:element name="major" type="xs:string"/>
      <xs:element name="enrolledCourse" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:schema>
```

XSL and XSLT

XSL (eXtensible Stylesheet Language) and XSLT (XSL Transformations) are used for transforming XML documents into different formats like HTML, plain text, or another XML structure. XSLT uses XSL to define how the transformation should occur.

XSLT (XSL Transformations) is a powerful language for transforming XML documents into other XML documents or different formats like HTML, plain text, or even into other XML structures.

XSLT uses XML-based syntax to define transformation rules. Here's a simple example of an XSLT stylesheet (college.xsl) that transforms a college.xml document into an HTML representation:

College.xsl(file Name)

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

 <!-- Match the college element -->
 <xsl:template match="college">
  <html>
   <body>
    <h2>College Information</h2>
    <xsl:apply-templates select="department"/>
    <xsl:apply-templates select="student"/>
   </body>
  </html>
 </xsl:template>

 <!-- Template for department -->
 <xsl:template match="department">
  <div>
   <h3>Department</h3>
   <p>Name: <xsl:value-of select="name"/></p>
   <h4>Courses</h4>
   <ul>
    <xsl:for-each select="course">
     <li>
      <p>Code: <xsl:value-of select="code"/></p>
      <p>Title: <xsl:value-of select="title"/></p>
      <p>Credits: <xsl:value-of select="credits"/></p>
```

```
      </li>
    </xsl:for-each>
   </ul>
  </div>
 </xsl:template>

 <!-- Template for student -->
 <xsl:template match="student">
  <div>
   <h3>Student</h3>
   <p>ID: <xsl:value-of select="id"/></p>
   <p>Name: <xsl:value-of select="name"/></p>
   <p>Major: <xsl:value-of select="major"/></p>
   <p>Enrolled Course: <xsl:value-of select="enrolledCourse"/></p>
  </div>
 </xsl:template>
</xsl:stylesheet>
```

XML processing with PHP

Technique of doing all the read, write and update operation using PHP in XML document is called XML processing with PHP. For processing the XML document we mainly use XML parser such as Tree-Based Parsers and Event-Based Parser.

What is an XML Parser?

To read and update, create and manipulate an XML document, you will need an XML parser. In PHP there are two major types of XML parsers:

- Tree-Based Parsers
- Event-Based Parsers

Tree-Based Parsers

Tree-based parsers holds the entire document in Memory and transforms the XML document into a Tree structure. It analyzes the whole document, and provides access to the Tree elements (DOM).

This type of parser is a better option for smaller XML documents, but not for large XML document as it causes major performance issues.
Example of tree-based parsers:

- SimpleXML
- DOM

Event-Based Parsers

Event-based parsers do not hold the entire document in Memory, instead, they read in one node at a time and allow you to interact with in real time. Once you move onto the next node, the old one is thrown away.
This type of parser is well suited for large XML documents. It parses faster and consumes less memory.
Example of event-based parsers:

- XMLReader
- XML Expat Parser

PHP SimpleXML Parser allows us to easily manipulate and get XML data.
SimpleXML is a tree-based parser. SimpleXML provides an easy way of getting an element's name, attributes and textual content if you know the XML document's structure or layou

SimpleXML turns an XML document into a data structure you can iterate through like a collection of arrays and objects.

PHP SimpleXML - Read From String

The PHP simplexml_load_string() function is used to read XML data from a string. Assume we have a variable that contains XML data, like this:

```
$myXMLData =
"<?xml version='1.0' encoding='UTF-8'?>
<note>
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend!</body>
</note>";
```

The example below shows how to use the simplexml_load_string() function to read XML data from a string:

Example

```
<?php
$myXMLData =
"<?xml version='1.0' encoding='UTF-8'?>
<note>
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend!</body>
</note>";

$xml=simplexml_load_string($myXMLData) or die("Error: Cannot create object");
print_r($xml);
?>
```

The output of the code above will be:

```
SimpleXMLElement Object ( [to] => Tove [from] => Jani [heading] => Reminder [body] =>
Don't forget me this weekend! )
```

note.xml

```
<note>
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend!</body>
</note>
```

Get the node values from the "note.xml" file:

```
<?php
$xml=simplexml_load_file("note.xml") or die("Error: Cannot create object");
```

```
echo $xml->to . "<br>";
echo $xml->from . "<br>";
echo $xml->heading . "<br>";
```

```
echo $xml->body;
?>
```

book.xml

```
<bookstore>
<book category="COOKING">
	<title lang="en">Everyday Italian</title>
	<author>Giada De Laurentiis</author>
	<year>2005</year>
	<price>30.00</price>
</book>
<book category="CHILDREN">
	<title lang="en">Harry Potter</title>
	<author>J K. Rowling</author>
	<year>2005</year>
	<price>29.99</price>
</book>
<book category="WEB">
	<title lang="en-us">XQuery Kick Start</title>
	<author>James McGovern</author>
	<year>2003</year>
	<price>49.99</price>
</book>
<book category="WEB">
<title lang="en-us">Learning XML</title>
<author>Erik T. Ray</author>
<year>2003</year>
<price>39.95</price></book></bookstore>
```

<!DOCTYPE html>

<html>

<body>

PHP SimpleXML - Get Node Values of Specific Elements

The following example gets the node value of the <title> element in the first and second <book> elements in

```
<?php
$xml=simplexml_load_file("books.xml") or die("Error: Cannot create object");
echo $xml->book[0]->title . "<br>";
echo $xml->book[1]->title;
?>
```

Everyday Italian

Harry Potter

```
</body>
</html>
```

PHP SimpleXML - Get Node Values - Loop

The following example loops through all the <book> elements in the "books.xml" file, and gets the node values of the <title>, <author>, <year>, and <price> elements:

```
<?php
$xml=simplexml_load_file("books.xml") or die("Error: Cannot create object");
foreach($xml->children() as $books) {
   echo $books->title . ", ";
   echo $books->author . ", ";
   echo $books->year . ", ";
   echo $books->price . "<br>";
}

?>
```

Everyday Italian, Giada De Laurentiis, 2005, 30.00
Harry Potter, J K. Rowling, 2005, 29.99
XQuery Kick Start, James McGovern, 2003, 49.99
Learning XML, Erik T. Ray, 2003, 39.95

PHP SimpleXML - Get Attribute Values

The following example gets the attribute value of the "category" attribute of the first <book> element and the attribute value of the "lang" attribute of the <title> element in the second <book> element:

```
<?php
$xml=simplexml_load_file("books.xml") or die("Error: Cannot create object");
echo $xml->book[0]['category'] . "<br>";
echo $xml->book[1]->title['lang'];
?>
```

XSL

XSL stands for E**X**tensible **S**tylesheet **L**anguage.The World Wide Web Consortium (W3C) started to develop XSL because there was a need for an XML-based Stylesheet Language.
CSS = Style Sheets for HTML

HTML uses predefined tags, and the meaning of each tag is **well understood**.

The <table> tag in HTML defines a table - and a browser knows **how to display it**.

Adding styles to HTML elements are simple. Telling a browser to display an element in a special font or color, is easy with CSS.

XSL = Style Sheets for XML

XML does not use predefined tags (we can use any tag-names we like), and therefore the meaning of each tag is **not well understood**.

A <table> tag could mean an HTML table, a piece of furniture, or something else - and a browser **does not know how to display it**.

XSL describes how the XML document should be displayed!

XSLT Introduction

XSLT is a language for transforming XML documents into XHTML documents or to other XML documents.

XPath is a language for navigating in XML documents.

- XSLT stands for XSL Transformations
- XSLT is the most important part of XSL
- XSLT transforms an XML document into another XML document
- XSLT uses XPath to navigate in XML documents
- XSLT is a W3C Recommendation

XSLT = XSL Transformations

XSLT is the most important part of XSL.XSLT is used to transform an XML document into another XML document, or another type of document that is recognized by a browser, like HTML and XHTML. Normally XSLT does this by transforming each XML element into an (X)HTML element.With XSLT you can add/remove elements and attributes to or from the output file. You can also rearrange and sort elements, perform tests and make decisions about which elements to hide and display, and a lot more.
A common way to describe the transformation process is to say that **XSLT transforms an XML source-tree into an XML result-tree**.

Style Sheet Declaration

The root element that declares the document to be an XSL style sheet is <xsl:stylesheet> or <xsl:transform>.
Note: <xsl:stylesheet> and <xsl:transform> are completely synonymous and either can be used!
The correct way to declare an XSL style sheet according to the W3C XSLT Recommendation is:

```
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
```

or:

```
<xsl:transform version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
```

To get access to the XSLT elements, attributes and features we must declare the XSLT namespace at the top of the document.
The xmlns:xsl="http://www.w3.org/1999/XSL/Transform" points to the official W3C XSLT namespace. If you use this namespace, you must also include the attribute version="1.0".

Start with a Raw XML Document

We want to **transform** the following XML document ("cdcatalog.xml") into XHTML:

```
<?xml version="1.0" encoding="UTF-8"?>
<catalog>
  <cd>
    <title>Empire Burlesque</title>
    <artist>Bob Dylan</artist>
    <country>USA</country>
    <company>Columbia</company>
    <price>10.90</price>
    <year>1985</year>
  </cd>
```

```
</catalog>
```

Viewing XML Files in IE, Chrome, Firefox, Safari, and Opera: Open the XML file (click on the link below) - The XML document will be displayed with color-coded root and child elements (except in Safari).

The <xsl:template> Element

The <xsl:template> element is used to build templates.
The **match** attribute is used to associate a template with an XML element. The match attribute can also be used to define a template for the entire XML document. The value of the match attribute is an XPath expression (i.e. match="/" defines the whole document).

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="/">
  <html>
  <body>
  <h2>My CD Collection</h2>
  <table border="1">
    <tr bgcolor="#9acd32">
      <th>Title</th>
      <th>Artist</th>
    </tr>
    <tr>
      <td>.</td>
      <td>.</td>
    </tr>
  </table>
  </body>
  </html>
</xsl:template>

</xsl:stylesheet>
```

Since an XSL style sheet is an XML document, it always begins with the XML declaration: **<?xml version="1.0" encoding="UTF-8"?>**.

The next element, **<xsl:stylesheet>**, defines that this document is an XSLT style sheet document (along with the version number and XSLT namespace attributes).
The **<xsl:template>** element defines a template. The **match="/"** attribute associates the template with the root of the XML source document.
The content inside the <xsl:template> element defines some HTML to write to the output.
The last two lines define the end of the template and the end of the style sheet.

The result from this example was a little disappointing, because no data was copied from the XML document to the output. In the next chapter you will learn how to use the **<xsl:value-of>** element to select values from the XML elements.

AJAX Introduction

AJAX = Asynchronous JavaScript and XML.
AJAX is not a new programming language, but a new way to use existing standards. AJAX is not a single technology but group of technologies such as: HTML / XHTML,CSS, JavaScript / DOM. AJAX is a technique for creating fast and dynamic web pages. AJAX allows web pages to be updated asynchronously by exchanging small amounts of data with the server behind the scenes. This means that it is possible to update parts of a web page, without reloading the whole page. Classic web pages, (which do not use AJAX) must reload the entire page if the content should change.

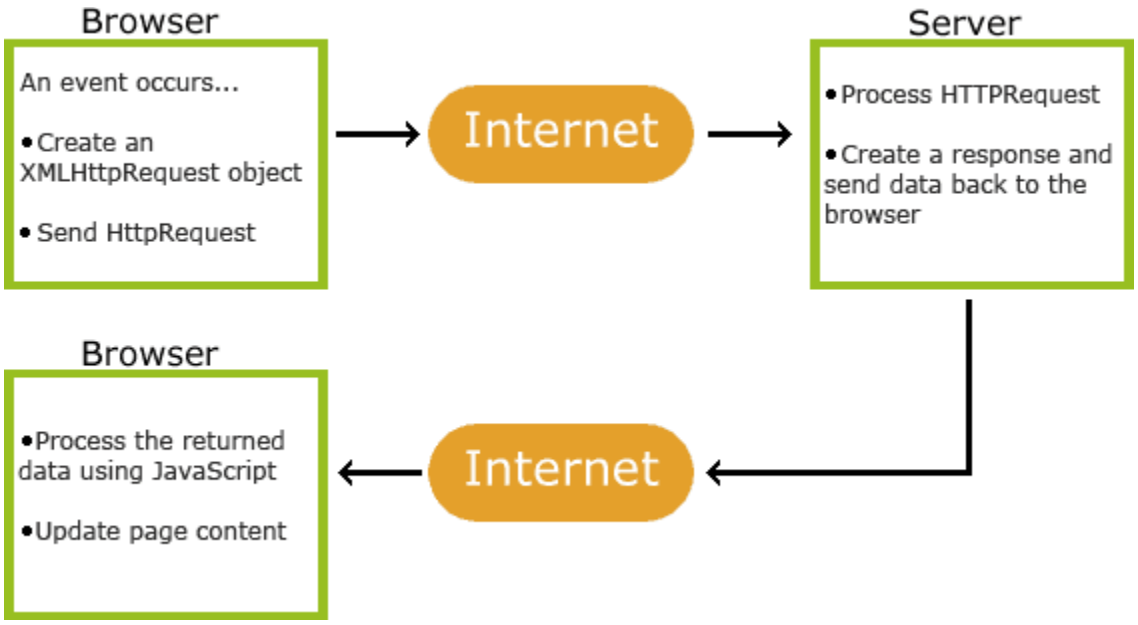
Examples of applications using AJAX: Google Maps, Gmail, Youtube, and Facebook tabs.

AJAX is Based on Internet Standards

AJAX is based on internet standards, and uses a combination of:

- XMLHttpRequest object (to exchange data asynchronously with a server)
- JavaScript/DOM (to display/interact with the information)
- CSS (to style the data)
- XML (often used as the format for transferring data)

Fig: AJAX working mechanism



The onreadystatechange event

When a request to a server is sent, we want to perform some actions based on the response.
The onreadystatechange event is triggered every time the readyState changes.
The readyState property holds the status of the XMLHttpRequest.
Three important properties of the XMLHttpRequest object:

Property	Description
Onreadystatechange	Stores a function (or the name of a function) to be called automatically each time the changes
readyState	Holds the status of the XMLHttpRequest. Changes from 0 to 4: 0: request not initialized 1: server connection established 2: request received 3: processing request 4: request finished and response is ready
Status	200: "OK" 404: Page not found

In the onreadystatechange event, we specify what will happen when the server response is ready to be processed.
When readyState is 4 and status is 200, the response is ready:

Example

```
xmlhttp.onreadystatechange=function()
 {

 if (xmlhttp.readyState==4 && xmlhttp.status==200)
  {

  document.getElementById("myDiv").innerHTML=xmlhttp.responseText;
  }

 }
```

Note: The onreadystatechange event is triggered five times (0-4), one time for each change in readyState.

AJAX PHP Example

AJAX is used to create more interactive applications.

AJAX PHP Example

The following example will demonstrate how a web page can communicate with a web server while a user type characters in an input field:
Start typing a name in the input field below:

First name:

Suggestions:

Example Explained

In the example above, when a user types a character in the input field, a function called "showHint()" is executed.
The function is triggered by the onkeyup event.
Here is the HTML code:

```
<html>
<head>
<script>
function showHint(str) {
   if (str.length == 0) {
      document.getElementById("txtHint").innerHTML = "";
      return;
   } else {
      var xmlhttp = new XMLHttpRequest();
      xmlhttp.onreadystatechange = function() {
         if (xmlhttp.readyState == 4 && xmlhttp.status == 200) {
            document.getElementById("txtHint").innerHTML = xmlhttp.responseText;
         }

      }

      xmlhttp.open("GET", "gethint.php?q=" + str, true);
      xmlhttp.send();
   }

}

</script>
</head>
<body>

<p><b>Start typing a name in the input field below:</b></p>
<form>
First name: <input type="text" onkeyup="showHint(this.value)">
</form>
<p>Suggestions: <span id="txtHint"></span></p>
</body>
</html>
```

Code explanation:
First, check if the input field is empty (str.length == 0). If it is, clear the content of the txtHint placeholder and exit the function.
However, if the input field is not empty, do the following:

- Create an XMLHttpRequest object
- Create the function to be executed when the server response is ready
- Send the request off to a PHP file (gethint.php) on the server
- Notice that q parameter is added gethint.php?q="+str
- The str variable holds the content of the input field

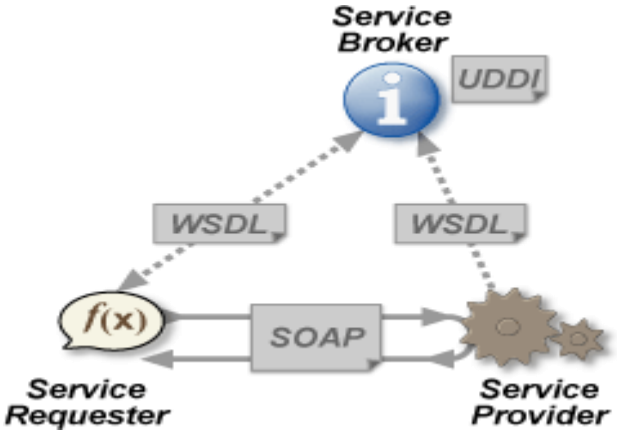
The PHP File - "gethint.php"

The PHP file checks an array of names, and returns the corresponding name(s) to the browser:

```
<?php
// Array with names
$a[] = "Anna";
$a[] = "Brittany";
$a[] = "Cinderella";
$a[] = "Diana";
$a[] = "Eva";
$a[] = "Fiona";
$a[] = "Gunda";
$a[] = "Hege";
$a[] = "Inga";
$a[] = "Johanna";
$a[] = "Kitty";
$a[] = "Linda";
$a[] = "Nina";
$a[] = "Ophelia";
$a[] = "Petunia";
$a[] = "Amanda";
$a[] = "Raquel";
$a[] = "Cindy";
$a[] = "Doris";
$a[] = "Eve";
$a[] = "Evita";
$a[] = "Sunniva";
$a[] = "Tove";
$a[] = "Unni";
$a[] = "Violet";
$a[] = "Liza";
$a[] = "Elizabeth";
$a[] = "Ellen";
$a[] = "Wenche";
$a[] = "Vicky";

// get the q parameter from URL
$q = $_REQUEST["q"];

$hint = "";

// lookup all hints from array if $q is different from ""
if ($q !== "") {
    $q = strtolower($q);
    $len=strlen($q);
```

```
    foreach($a as $name) {
        if (stristr($q, substr($name, 0, $len))) {
            if ($hint === "") {
                $hint = $name;
            } else {
                $hint .= ", $name";
            }
        }
    }
}

// Output "no suggestion" if no hint was found or output correct values
```

Web Services



A **Web service** is a method of communication between two electronic devices over a **network** in other words Web service is a software system designed to support **interoperable(**able to exchange information**)** machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP messages, typically conveyed using HTTP with an XML serialization

The companies pushing WSDL(**Web Services Description Language**), SOAP(Simple Object Access Protocol), and UDDI(Universal Description, Discovery and Integration) as the backbone of Web services are the same ones that have invested heavily in these technologies over the years.

Web Services in the Real World

It may be easier to come to some understanding of the term *Web services* by looking at a few places it is currently used on the Internet. Some big Internet companies, which you are probably already familiar with, offer Web services so you can tie your application into their systems.
A few of the services, which are Yahoo, Google, Amazon, and eBay.

Yahoo Web Services

The Yahoo Web service, which uses REST(Representational State Transfer), provides an application to use Yahoo's search engine to find images, businesses, news, and video on the Internet.

Google Web APIs

Google also offers a wide range of Web services, including searches as well as integration with many of their other services such as Ad Words and Blogger.

Amazon E-commerce Service (ECS)

Amazon provides access to its products and to its e-commerce functionality through its E-commerce Service (ECS). The service is accessible using either REST or SOAP, which offers

more flexibility to developers because they can use the technology they're most comfortable using.

eBay

eBay offers a developer program, at http://developer.ebay.com/, allowing an application to tap into its platform using eBay's XML API, REST, or SOAP.