Topic 7 : Advanced features of internet-based programming

Learning outcomes

Upon completing this topic, you should be able to:

1. Understand cookies
2. Understand Sessions
3. Use of webservices in web programming
4. Importance of Cookies, sessions and web services in Internet Programming.

Introduction to Cookies and Sessions in PHP

1.1. Stateful client/server interaction

Many of our sites for example google.com knows who you are. How? How does a

client uniquely identify itself to a server, and how does the server provide specific

content to each client?

- HTTP is a stateless protocol; it simply allows a browser to request a single document from a web server.
- In this module, we'll learn about pieces of data called cookies used to work around this problem, which are used as the basis of higher-level sessions between clients and servers.

1.2 What is a cookie?

- Cookie: a small amount of information sent by a server to a browser, and then sent back by the browser on future page requests
- Cookies have many uses:
    1. authentication
    2. user tracking
    3. maintaining user preferences, shopping carts, etc.
- A cookie's data consists of a single name/value pair, sent in the header of the client's HTTP GET or POST request

How cookies are sent

- When the browser requests a page, the server may send back a cookie(s) with it.
- If your server has previously sent any cookies to the browser, the browser will send them back on subsequent requests
- Alternate model: client-side JavaScript code can set/get cookies

Facts and myths about cookies

- Cookies are like worms/viruses and can erase data from the user's hard disk.
- Cookies are a form of spyware and can steal your personal information.
- Cookies generate popups and spam.
- Cookies are only used for advertising.
- Cookies are only data, not program code.
- Cookies cannot erase or read information from the user's computer.
- Cookies are usually anonymous (do not contain personal information).
- Cookies CAN be used to track your viewing habits on a particular site.

How long does a cookie exist?

1. The default type; a temporary cookie that is stored only in the browser's memory

- when the browser is closed, temporary cookies will be erased
- cannot be used for tracking long-term information
- safer, because no programs other than the browser can access them

2. One that is stored in a file on the browser's computer

- can track long-term information
- potentially less secure, because users (or programs they run) can open cookie files, see/change the cookie values, etc.

Where to find cookies in the PC ?

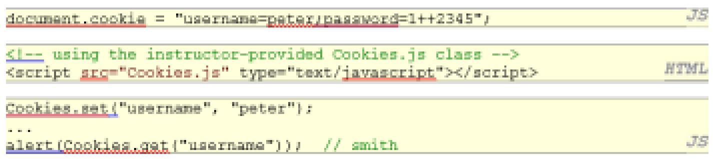
This depends on the different browsers that a given user is using for example.

- Internet Explorer : HomeDirectory\Cookies e.g. C:\Documents and Settings\jsmith\Cookies each is stored as a .txt file similar to the site's domain name
- Firefox: HomeDirectory\.mozilla\firefox\???.
  default\cookies.txt view cookies in Firefox preferences:

- Privacy, Show Cookies... or from page info,view cookies.

Cookies in JavaScript

- JS has a global document.cookie field (a string)



```
document.cookie = "username=peter;password=1++2345";
```

```
<!-- using the instructor-provided Cookies.js class -->
<script src="Cookies.js" type="text/javascript"></script>
```

```
Cookies.set("username", "peter");
...
alert(Cookies.get("username"));  // smith
```

- You can manually set/get cookie data from this field (sep. by ;), and it will be saved in the browser we have written a Cookies.js helper class with methods set, get, exists, remove

1.3 Setting a cookie in PHP

```
setcookie("name", "value");
```

- setcookie("username", "martay"); setcookie("favoritecolor", "blue");
- setcookie causes your script to send a cookie to the user’s browser setcookie must be called before any output statements (HTML blocks, print, or echo)
- you can set multiple cookies (20-50) per user, each up to 3-4K bytes
- technically, a cookie is just part of an HTTP header, and it could be set using PHP’s header function (but this is less convenient, so you would not want to do this):

```
header("Set-Cookie: username=martay; path=/; secure");
```

Retrieving information from a cookie

```
$variable = $_COOKIE["name"]; # retrieve value of the cookie
```

Example of cookies code.

```
if (isset($_COOKIE["username"])) {

$username = $_COOKIE["username"];

print("Welcome back, $username.\n");

}

else

{

print("Never heard of you.\n");

}

print("All cookies received:\n");

print_r($_COOKIE);
```

- any cookies sent by client are stored in $_COOKIES associative array
- use isset function to see whether a given cookie name exists
- unset function deletes a cookie

Setting a persistent cookie in PHP

setcookie("name", "value", timeout);

$expireTime = time() + 60*60*24*7;

# 1 week from now

setcookie("CouponNumber", "389752",

$expireTime);

setcookie("CouponValue", "100.00",

$expireTime);

• to set a persistent cookie, pass a third parameter for its timeout in second

• time function returns the current time in seconds

– date function can convert a time in seconds to a readable date

Removing a persistent cookie

setcookie("name", "", time() - 1);

setcookie("CouponNumber", "", time() - 1);

• if the server wants to remove a persistent cookie, it should set it again, passing

a timeout that is prior to the present time

SESSION

2.1 What is a session?

Session: an abstract concept to represent a series of HTTP requests and responses between a specific Web browser and server.

- HTTP doesn't support the notion of a session, but PHP does

Sessions Vs. Cookies:

- a cookie is data stored on the client
- a session's data is stored on the server (only 1 session per client)
- sessions are often built on top of cookies:
- The only data the client stores is a cookie holding a unique session ID
- On each page request, the client sends its session ID cookie, and the server uses this to find and retrieve the client's session data

2.2 How sessions are established

- Client’s browser makes an initial request to the server
- Server notes client’s IP address/browser, stores some local session data, and sends a session ID back to client
- Client sends that same session ID back to server on future requests
- Server uses session ID to retrieve the data for the client’s session later, like a ticket given at a coat check room

Sessions in PHP

```
session_start();
```

- session_start signifies your script wants a session with the user
  - must be called at the top of your script, before any HTML output is produced

- When to call session_start:
- if the server hasn’t seen this user before, a new session is created
- otherwise, existing session data is loaded into $_SESSION associative array
- you can store data in $_SESSION and retrieve it on future pages

Accessing session data

```
$_SESSION["name"] = value; - # store session data


$variable = $_SESSION["name"]; - # read session data

if (isset($_SESSION["name"]))

{ # check for session data

if (isset($_SESSION["points"])) {

$points = $_SESSION["points"];

print("You’ve earned $points points.\n");

} else {

$_SESSION["points"] = 0; # default

}
```

• the $_SESSION associative array reads/stores all session data

• use isset function to see whether a given value is in the session

Where is session data stored

• on the client, the session ID is stored as a cookie with the name PHPSESSID

• on the server, session data are stored as temporary files such as /tmp/sess_fcc17f071...

• you can find out (or change) the folder where session data is saved using the

session_save_path function

• for very large applications, session data can be stored into a SQL database (or

other destination) instead using the session_set_save_handler function

Browsers that don’t support cookies

session_start(); # same as usual

# Generate a URL to link to one of our site’s pages]

# (you probably won’t ever need to do this)

$orderUrl = "/order.php?PHPSESSID=" . session_id();

• if a client’s browser doesn’t support cookies, it can still send a session ID as a query string parameter named PHPSESSID

– this is done automatically; session_start detects whether the browser

supports cookies and chooses the right method

• if necessary (such as to build a URL for a link on the page), the server can

find out the client’s session ID by calling the session_id function

Session timeout

• because HTTP is stateless, it is hard for the server to know when a user has finished a session

• ideally, user explicitly logs out, but many users don’t

• client deletes session cookies when browser closes

• server automatically cleans up old sessions after a period of time

– old session data consumes resources and may present a security risk

– adjustable in PHP server settings or with session_cache_expire function

– you can explicitly delete a session by calling session_destroy

Introduction to Web Services

- Web Services are self-contained, modular, distributed, dynamic applications that can be described, published, located, or invoked over the network to create products, processes, and supply chains. These applications can be local, distributed, or Web based. Web services are built on top of open standards such as TCP/IP, HTTP, Java, HTML, and XML. • Web services are XML-based information exchange systems that use the Internet for direct application-to-application interaction. These systems can include programs, objects, messages, or documents.
- A complete web service is, therefore, is any service that:
    1. Is available over the Internet or private (intranet) networks
    2. Uses a standardized XML messaging system.
    3. Is not tied to any one operating system or programming language.
    4. Is self-describing via a common XML grammar.
    5. Is discoverable via a simple find mechanism

3.1 Components of Web Services.

The basic Web services platform is XML + HTTP.

All the standard Web Services works using following components

• SOAP (Simple Object Access Protocol)

• UDDI (Universal Description, Discovery and Integration)

• WSDL (Web Services Description Language)

3.2. How Does it Work?

- You can build a Java-based Web Service on Solaris that is accessible from your Visual Basic program that runs on Windows. You can also use C# to build new Web Services on Windows that can be invoked from your Web application that is based on JavaServer Pages (JSP) and runs on Linux.

Example.

- Consider a simple account-management and order -processing system. The accounting personnel use a client application built with Visual Basic or JSP to create new accounts and enter new customer orders. The processing logic for this system is written in Java and resides on a Solaris machine, which also interacts with a database to store the information.

The steps illustrated above are as follows:

1. The client program bundles the account registration information into a SOAP message.
2. This SOAP message is sent to the Web Service as the body of an HTTP POST request.

3. The Web Service unpacks the SOAP request and converts it into a command that the application can understand. The application processes the information as required and responds with a new unique account number for that customer.
4. Next, the Web Service packages up the response into another SOAP message, which it sends back to the client program in response to its HTTP request.
5. The client program unpacks the SOAP message to obtain the results of the account registration process.

3.3. Benefits of using Web Services

1. Exposing the existing function on to network: A Web service is a unit of managed code that can be remotely invoked using HTTP, that is, it can be activated using HTTP requests. So, Web Services allows you to expose the functionality of your existing code over the network. Once it is exposed on the network, other application can use the functionality of your program.
2. Connecting Different Applications ie Interoperability: Web Services allows different applications to talk to each other and share data and services among themselves.
3. Standardized Protocol: Web Services uses standardized industry standard protocol for the communication. All the four layers (Service Transport, XML Messaging, Service Description and Service Discovery layers) uses the well-defined protocol in the Web Services protocol stack.
4. Low Cost of communication: Web Services uses SOAP over HTTP protocol for the communication, so you can use your existing low-cost internet for implementing Web Services.

3.4. Web Services Behavioral Characteristics

1. XML-based-Web Services uses XML at data representation and data transportation layers.
2. Loosely coupled-A consumer of a web service is not tied to that web service directly. The web service interface can change over time without compromising the client's ability to interact with the service.
3. Coarse-grained-Object-oriented technologies such as Java expose their services through individual methods. An individual method is too fine an operation to provide any useful capability at a corporate level.
4. Ability to be synchronous or asynchronous-Synchronicity refers to the binding of the client to the execution of the service. In synchronous invocations, the client blocks and waits for the service to complete its operation before continuing.
5. Supports Remote Procedure Calls (RPCs)-Web services allow clients to invoke procedures, functions, and methods on remote objects using an XML based protocol. Remote procedures expose input and output parameters that a web service must support. Component development through Enterprise JavaBeans (EJBs) and .NET Components has increasingly become a part of architectures and enterprise deployments.
6. Supports document exchange One of the key advantages of XML is its generic way of representing not only data, but also complex documents.

3.5. Web Services Architecture

There are three major roles within the web service architecture:

1. Service provider: -This is the provider of the web service. The service provider implements the service and makes it available on the Internet.
2. Service requestor-This is any consumer of the web service. The requestor utilizes an existing web service by opening a network connection and sending an XML request.
3. Service registry-This is a logically centralized directory of services. The registry provides a central place where developers can publish new services or find existing ones.

3.6. Web Service Protocol Stack

The stack is still evolving, but currently has four main layers.

1. Service transport-This layer is responsible for transporting messages between applications. Currently, this layer includes hypertext transfer protocol (HTTP), Simple Mail Transfer Protocol (SMTP), file transfer protocol (FTP), and newer protocols, such as Blocks Extensible Exchange Protocol (BEEP).
2. XML messaging-This layer is responsible for encoding messages in a common XML format so that messages can be understood at either end. Currently, this layer includes XML-RPC and SOAP.
3. Service description-This layer is responsible for describing the public interface to a specific web service. Currently, service description is handled via the Web Service Description Language (WSDL).
4. Service discovery-This layer is responsible for centralizing services into a common registry, and providing easy publish/find functionality. Currently, service discovery is handled via Universal Description, Discovery, and Integration (UDDI).

Web Services - Examples

Based on the Web Service Architecture we will create following two components as a part of Web Services implementation.

- Service provider or publisher: This is the provider of the web service. The service provider implements the service and makes it available on the Internet or intranet. We will write a simple web Service using NET SDK.
- Service requestor or consumer This is any consumer of the web service. The requestor utilizes an existing web service by opening a network connection and sending an XML request.

The following is a standard template for a Web Service.

.NET Web Services use the .asmx extension.

Note that a method exposed as a Web Service has the WebMethod attribute. Save this file as FirstService.asmx in the IIS virtual directory (as explained in configuring IIS; for example, c:\Mysite)

FirstService.asmx

```
<%@ WebService language="C" class="FirstService" %>
using System;
using System.Web.Services;
using System.Xml.Serialization;
[WebService (Namespace ="http://localhost/MyWebServices/")]
public class FirstService : WebService
{
[WebMethod]
public int Add(int a, int b)
{
return a + b;
}
[WebMethod]
public String SayHello()
{
return "Hello World";
}
}
```

To test a Web Service, it must be published. A Web Service can be published either on an intranet or the Internet. We will publish this Web Service on IIS running on a local machine.

Start with configuring the IIS.

- Open Start->Settings->Control Panel->Administrative tools->Internet Services Manager.
- Expand and right-click on [Default Web Site]; select New ->Virtual Directory.
- The Virtual Directory Creation Wizard opens. Click Next.
- The "Virtual Directory Alias" screen opens. Type the virtual directory name—for example, MyWebsite—and click Next.
- The "Web Site Content Directory" screen opens. Here, enter the directory path name for the virtual directory—for example, c:\MyWebSite—and click Next.
- The "Access Permission" screen opens. Change the settings as per your requirements. Let's keep the default settings for this exercise. Click the Next button. It completes the IIS configuration. Click Finish to complete the configuration.

To test our Web Service, copy FirstSer.asmx in the IIS virtual directory created above (C:\MyWebServices).

Open the Web Service in Internet Explorer (http://localhost/MyWebServices/FirstSer.asmx).

It should open your Web Service page. The page should have links to two methods exposed as Web Services by our application. however, it is a bit more involved.

remember, we will write two types of service consumers, one Web- and another

Windows application-based consumer. Should start with Web Service consumer

first.

Summary

- HTTP is a stateless protocol; it simply allows a browser to request a single document from a web server.
- Cookie: a small amount of information sent by a server to a browser, and then sent back by the browser on future page requests.
- Session: an abstract concept to represent a series of HTTP requests and responses between aspecific Web browser and server
- HTTP doesn't support the notion of a session, but PHP does
- Web Services are self-contained, modular, distributed, dynamic applications that can be described, published, located, or invoked over the network to create products, processes, and supply chains.

Benefits of using Web Services

• Exposing the existing function on to network:

• Connecting Different Applications ie Interoperability:

• Standardized Protocol:

• Low Cost of communication:

Revision Questions

1. Name three uses of cookies in web-based applications.
2. Write a simple php code to show how to set a persistent cookie in PHP
3. Discuss web services and implement a proxy to consume the above example of a web-based service consumer.
4. Explain how a session is established in web applications
5. Write a php code that can be used to setup a session.
6. Write a php application such as the Cosmestic.php cosmetics search script so that it remembers the current user's last query (if any), and offers the user a chance to search for it again, such as:
7. Welcome back! Would you like to repeat your recent search for My Cosmetics?
8. Pretend that the cosmestic-search program is running on a system that wants to limit repeated usage by particular users. Add code so that a given user can only conduct one session per day