Programming Applications

Spring 2017-2018

Asst. Prof. Dr. Pelin GÖRGEL
(paras@istanbul.edu.tr)

9.03.2018

Client-side Advantages

- ***Server Resources*** –

  Each client uses its own resources to execute the code on the webpage, hence saving resources for the provider.

  This is inherently the disadvantage with server-side scripting, where the server must use valuable resources to parse each page it sends out, possibly slowing down the web site.

Client-side Advantages (Continued)

- Scripts in the category of client side scripts are executed by, and run directly in, the user's browser itself.
- They are easily embedded into your web site using easy to use HTML code and because client side scripts are being ran in the browser they can easily and quickly respond to your user's actions which can make for a more interactive experience for your user when they visit your web site.
- By including code within a web page, a number of features can be added to a Web page without the need to send information to the Web Server which takes time.
- Tasks on the client side include data validation, special formatting and more.

Server Side Programming

- Server-side scripts run on the server. This reduces the amount of bugs or compatibility problems since the code runs on one server using one language and hosting software.
- Server-side programming can also be encrypted when users send form variables, protecting users against any hack attempts.
- Some examples of server-side programming languages are C#, VB.NET, and PHP.

Server-side Advantages

- ***Code Protection*** - Because the server side code is executed before the HTML is sent to the browser, your code is hidden.
- ***Browser Independent*** - Server-side code is browser independent. Because the HTML code returned by your browser is simple HTML, you do not need to worry about the version of browser, javascript etc. that the client is using.

Server-side Advantages (Continued)

- ***Application Updates*** - By operating on the server-side updating to new versions, fixing bugs, implementing code patches become a simple process of updating the server machine.
- Consider in opposition to this, a windows application running on the client - in order to update or make fixes, the client must download the latest version each time.
- Uses less of your visitors resources.

Server-side Advantages (Continued)

- Since server side scripts run on the server they generally have more flexibility than client side scripts since you have a greater range of files that they can access and more variety of scripting languages to choose from.
- Some examples of scripting languages that you can use with server side scripts include Java, PHP, Perl, Python, and VBScript among others.

Server-side Advantages (Continued)

- **Security**

  Server-side scripts are more secure than client-side.

  For instance, when a user accesses a bank account online, the server side-script communicates with the client using encryption.

  A client-side script is plain text and run on the client's browser. Any hacker can view the code and steal private information from the user's computer.

How do we choose the scripting type?

- Heavier, more complex applications or applications requiring more security are typically server side scripts.
- Lighter applications which do not need the information in the server are typically client side scripts.

Server-side Script

- <script language=vbscript RUNAT=Server>
- .....
- </script>

Client-side Script

- <script language=vbscript RUNAT=Client>
- .....
- </script>

Request-Response

- A client sends a request to a server and the server sends a response to it.
- HTTP response indicates the necessary time to remain the content fresh.
- The browser avoids an other request to the server in this duration.

Protocols

- A **protocol** is the method of exchanging data over a computer network such as local area network, Internet, Intranet, etc.
- Each protocol has its own method of how data is formatted when sent and what to do with it once received, how that data is compressed or how to check for errors in data.
- Protocols are often text, and simply describe how the client and server will have their conversation.
- Some well known protocols using TCP/IP are:
  SMTP, Telnet, HTTP, FTP.
- Every Web server on the Internet is compatible with the HTTP protocol.

HTTP
(Hyper Text Transfer Protocol)

- It provides the communication between a web browser and web server.
- The request methods are **GET** and **POST**.
- A request is made by;
  - Submitting a form,
  - Typing a URL,
  - Clicking a hyperlink

GET Method

- GET: It sends data in the URL.
  - http://yandex.com.tr/yandsearch?lr=11508&text=computer
  - http://yandex.com.tr/yandsearch?lr=11508&text=computer+java
  - www.google.com/search?q=deitel
- Limits data length

POST Method

- POST: It sends the data as a part of HTTP message, not part of a URL.
  - Much more secure
  - Higher data limit

Get vs Post

1. We can use GET method to GET information from the Server	1. We can use POST method to POST information to the Server
2.usually GET requests are READ-ONLY	2.usually POST Requests are WRITE or UPDATE operations
3.End-user provided information will be append to the URL as the part of Query String and send to the Server	3.End-user provided information will be encapsulated in the request body and send to the Server
4.By using GET Request we can send only Character Data(ASCII) and we cannot send binary data like images..	4. By using POST Request we can send both Binary and Character data to the server

Get vs Post

5.By using GET request we can send only limited amount of information, which is varied from browser to browser.	5.By using POST request we can send huge amount of information to the server
6.Security is less and hence we cannot send sensitive information like username, password etc.	6.Security is more and hence we can send sensitive information like username, password etc.
7.Bookmarking of GET Request is possible	7.Bookmarking of POST Request is not possible
8.Caching of GET Request is possible	8.Caching of POST Request is not possible

Security

- Web server can be a simple piece of software.
- It takes the file name sent in with the GET command, retrieves that file and sends it down the wire to the browser.
- Most servers add some level of **security** to the serving process. For example, if you have ever gone to a Web page and had the browser pop up a dialog box asking for your name and password, you have encountered a password-protected page. The server lets the owner of the page maintain a list of names and passwords for those people who are allowed to access the page; the server lets only those people who know the proper password see the page.
- More advanced servers add further security to allow an encrypted connection between server and browser, so that sensitive information like credit card numbers can be sent on the Internet.
- Network sniffing

Cookie Basics

- Cookies are not programs, and they cannot run like programs do. Therefore, they cannot gather any information on their own. Nor can they collect any personal information about you from your machine.
- **A cookie** is a piece of text that a Web server can store on a user's hard disk. Cookies allow a Web site to store information on a user's machine and later retrieve it. The pieces of information are stored as **name-value pairs**.

- For example, a Web site might generate a unique ID number for each visitor and store the ID number on each user's machine using a cookie file.
- If you visit **www.google.com**, and the site may place a cookie on your machine as following:
- UserID A9A3BECE0563982D www.google.com/

Cookie Basics (continued)

- The Web site stores the data, and later it receives it back. A Web site can only receive the data it has stored on your machine. It cannot look at any other cookie, nor anything else on your machine.
- The data moves in the following manner:
- If you type the URL of a Web site into your browser, your browser sends a request to the Web site for the page. For example, if you type the URL **http://www.amazon.com** into your browser, your browser will contact Amazon's server and request its home page.
- When the browser does this, it will look on your machine for a cookie file that Amazon has set. If it finds an Amazon cookie file, your browser will send all of the name-value pairs in the file to Amazon's server along with the URL. If it finds no cookie file, it will send no cookie data.
- Amazon's Web server receives the cookie data and the request for a page. If name-value pairs are received, Amazon can use them.
- If no name-value pairs are received, Amazon knows that you have not visited before. The server creates a new ID for you in Amazon's database and then sends name-value pairs to your machine in the header for the Web page it sends. Your machine stores the name-value pairs on your hard disk.
- The Web server can change name-value pairs or add new pairs whenever you visit the site and request a page.

How do Web sites use cookies?

- Web sites use cookies in many different ways:

- Sites can **accurately determine how many people actually visit the site.**
- The only way for a site to accurately count visitors is to set a cookie with a unique ID for each visitor. Using cookies, sites can determine how many visitors arrive, how many are new versus repeat visitors and how often a visitor has visited.
- Sites can **store user preferences** so that the site can look different for each visitor (often referred to as **customization**). For example, if you visit **msn.com**, it offers you the ability to "change content/layout/color."
- It also allows you to enter your zip code and get customized weather information. When you enter your zip code, the following name-value pair gets added to MSN's cookie file:

  WEAT CC=NC%5FRaleigh%2DDurham®ION=
  www.msn.com/

How do Web sites use cookies? (continued)

- E-commerce sites can implement things like **shopping carts** and **"quick checkout" options**. The cookie contains an ID and lets the site keep track of you as you add different things to your cart. Each item you add to your shopping cart is stored in the site's database along with your ID value. When you check out, the site knows what is in your cart by retrieving all of your selections from the database. It would be impossible to implement a convenient shopping mechanism without cookies or something like them.

Java Server Faces

Some Differences Between JSP and JSF

- JSF is a framework for web applications.
- JSF uses the MVC architecture.
- In JSP, all Java codes are embedded into the HTML codes.
- JSF is faster and for more complex applications.

Process of a JSF Execution

1. You invoke the application from a server
2. Browser requests the app.'s JSF page
3. Web server receives this request
4. GlassFish application server or any JSF 2.0 compliant container such as Apache TomCat behaves like a web server.

Process of a JSF Execution (Continued)

5. The framework includes Faces servlet (a software component running on the server) that processes each requested page.
6. The framework returns a response to the client.
7. You need GlassFish to send the related data to the browser.

Setup GlassFish Server

- Go;

https://glassfish.java.net/download.html

- Choose;

Java EE 7 Full Platform

- Download;

glassfish-4.1.1.zip

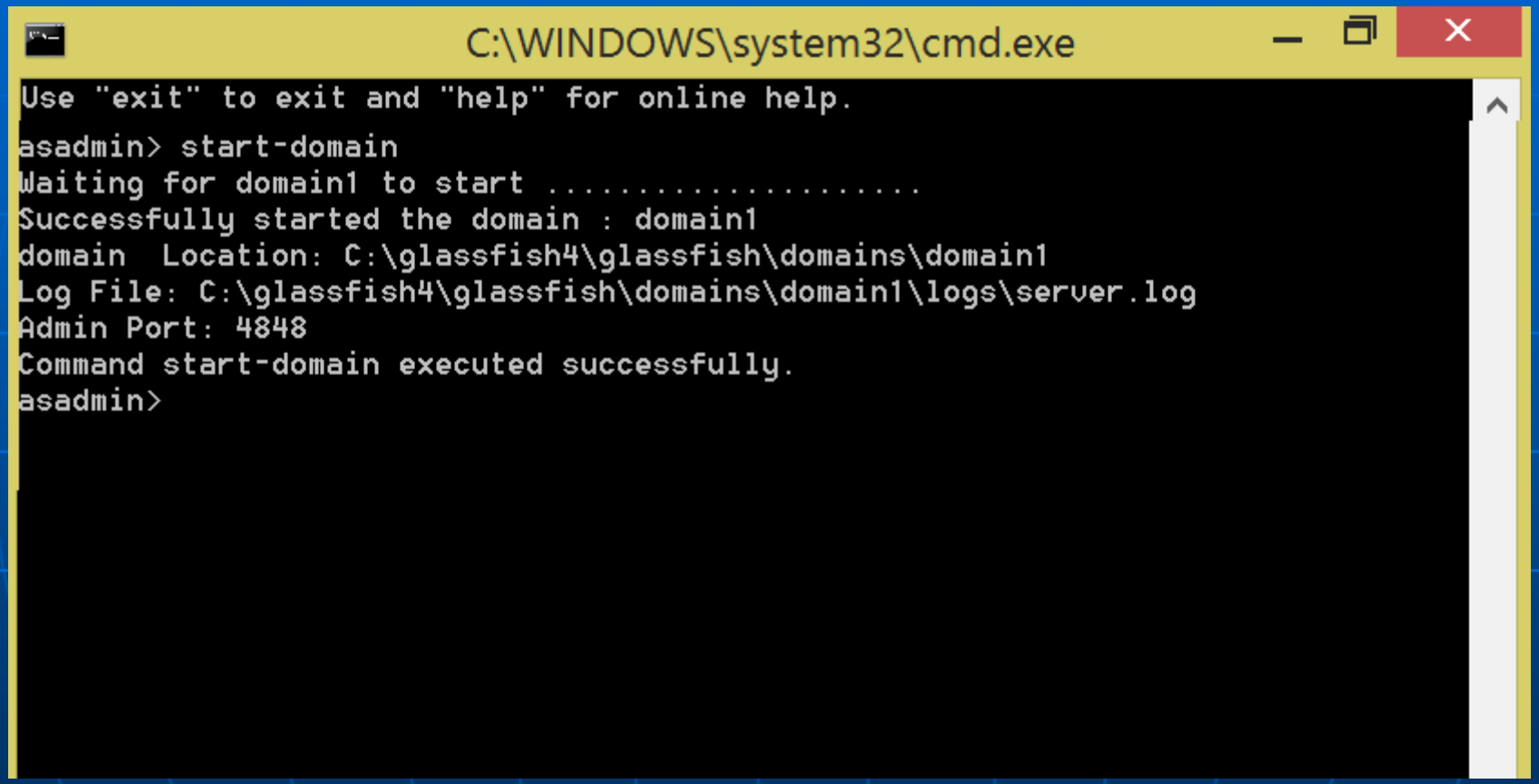
- Extract the zip file (keeping the folder name) under C:\ driver directly.

Start GlassFish Server

- Go to the GlassFish folder under C driver.
- Open «bin» folder.
- Click «asadmin.bat» file.
- In the console:

asadmin> start-domain

- Do not close the console during the execution.


C:\WINDOWS\system32\cmd.exe
Use "exit" to exit and "help" for online help.
asadmin> start-domain
Waiting for domain1 to start .....
Successfully started the domain : domain1
domain Location: C:\glassfish4\glassfish\domains\domain1
Log File: C:\glassfish4\glassfish\domains\domain1\logs\server.log
Admin Port: 4848
Command start-domain executed successfully.
asadmin>

09:58

JSF Components

JSF component	Description
`h:form`	Inserts an XHTML `form` element into a page.
`h:commandButton`	Displays a button that triggers an event when clicked. Typically, such a button is used to submit a form's user input to the server for processing.
`h:graphicImage`	Displays an image (e.g., GIF and JPG).
`h:inputText`	Displays a text box in which the user can enter input.
`h:outputLink`	Displays a hyperlink.
`h:panelGrid`	Displays an XHTML `table` element.
`h:selectOneMenu`	Displays a drop-down list of choices from which the user can make a selection.
`h:selectOneRadio`	Displays a set of radio buttons.
`f:selectItem`	Specifies an item in an `h:selectOneMenu` or `h:selectOneRadio` (and other similar components).

JSF App. 1

- *index.xhtml (JSF file)*

```
<?xml version='1.0' encoding='UTF-8' ?>

<!-- index.xhtml -->
<!-- JSF page that displays the current time on the web server -->
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
   xmlns:h="http://java.sun.com/jsf/html">
   <h:head>
      <title>WebTime: A Simple Example</title>
      <meta http-equiv="refresh" content="60" />
   </h:head>
   <h:body>
      <h1>Current time on the web server: #{webTimeBean.time}</h1>
   </h:body>
</html>
```

JSF App. 1

- *WebTimeBean.java (Bean file)*

```
import java.text.DateFormat;
import java.util.Date;
import javax.faces.bean.ManagedBean;

@ManagedBean( name="webTimeBean" )
public class WebTimeBean
{
   // return the time on the server at which the request was received
   public String getTime()
   {
      return DateFormat.getTimeInstance( DateFormat.LONG ).format(
         new Date() );
   } // end method getTime
} // end class WebTimeBean
```