A

AJAX

`Asynchronous JavaScript and XML` (AJAX) is a new technique, which describes how other technologies, such as JavaScript, `Document Object Model` (`DOM`), and Extensible Markup Language (XML), can be used together to create interactive Web applications. In early days, a Web application created by using these technologies was not efficient and platform-independent, as compared to a desktop based application. To overcome this, Jesse James Garrett co-founder of Adaptive Path combined JavaScript, XML, and DOM to form a new technique, called `AJAX`.

In this technique, the request to the Web server is sent by using the `XMLHttpRequest` object. This object, a part of the JavaScript technology, helps in sending `asynchronous` requests to the server. With this request, Web applications can now interact with the Web server `asynchronously`**.** The time taken to refresh the page is also minimized, which enhances the efficiency of the Web application.

In this appendix, we trace the evolution of Web applications and the technologies used for developing them. We then discuss the problems associated with these technologies that were used to create Web applications in early days, and how these problems led to the development of the AJAX technique. Finally, we create a sample AJAX-based application.

Evolution of Web Applications

In earlier times, applications had their own client-based program that required to be configured on the client machine. Both the server and the client needed an upgrade when there were any changes made in the application. However, with the advent of Web-based applications, client-server applications changed a lot. A Web application provides a series of Web pages to the client, which is accessed by all types of clients using any browser of their choice. There is no need to install a separate client program on all client machines, as the Web browser interprets and displays all Web pages and works as a common client for a Web application. Therefore, we can now develop Web-based client-server application without spending time on creating separate client programs for different client machines.

In earlier times, we could access simple, static Hypertext Markup Language (HTML) pages using a Web browser, which sends a request to a Web server. The Web server then sends the requested web page, which is stored at the server using Hypertext Transfer Protocol (HTTP). These Web pages display static content, i.e. a constant state or a text file that does not change. The ever evolving technology lead to the requirement for designing a Web application that processes the data given by the client and presents dynamic content to the client arose, which was not possible through static Web pages. To overcome this problem, the evolution of Web application took place, which led to the development of technologies for processing the client request and generating response content dynamically. With the emergence of this new concept, new technologies also took birth. Therefore, the evolution of a Web application lead to the development of new technologies, which helped in creating Web applications. The following are few of the technologies, which can be used to create a Web application:

- Common Gateway Interface (CGI)

- Applet
- JavaScript
- Servlet
- JavaServer Pages (JSP)
- Active Server Pages (ASP)
- Hypertext Processor (PHP)
- Dynamic HTML (DHTML)
- XML

Common Gateway Interface

CGI is a standard protocol for interfacing the external application software with an information server, commonly known as Web server. Figure A.1 shows the processing of CGI:

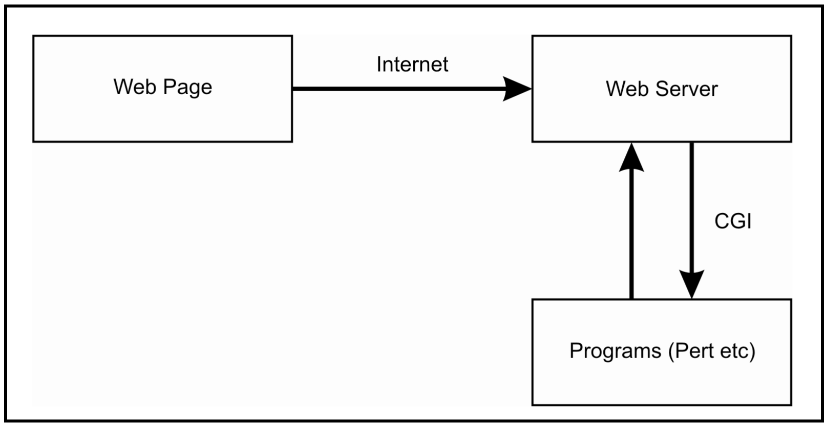


Figure A.1: Displaying How CGI works

CGI is not a programming language; rather it is a protocol that defines a set of rules on how the Web server communicates with the program. This functionality allows the server to pass the request from the client Web browser to the external application. In fact, CGI is a specification used to transfer information between the Web server and the CGI program. A CGI program can be written in any programming language, such as C, Perl, and Java, etc. CGI programs help the Web server to interact dynamically with Web users, e.g. a CGI program is used to process the form's data when it is submitted once. `CGI` also allows `HTML` pages to interact with applications rather than a static Web page. Some of the advantages of CGI are as follows:

- It is simple and quick to develop.
- It is rich in libraries. Therefore, there is no need to provide code to create the required class; instead, the developers need to provide code to implement the classes of the libraries provided by CGI.

The following are some of the limitations of CGI:

- CGI is slow; being slow is the flaw of CGI itself and not of the particular programming language used for scripting.
- Each time the process must be launched from the beginning and that takes a lot of time.
- The resources, such as the database connections, must be created and reloaded every time.
- The state is not persistent. On every request, a new state of a user is built.

Now let's move on to another technology called applets, related to Web applications.

Applet

An applet is a program written in Java programming language, which can be included in the HTML page and run within a Web browser. Java applets are normally used to include small, interactive components to a Web page. The applets are mainly used to provide dynamic user-interface and a number of graphical effects for the Web pages. When the Web client or the Web browser opens the Web page, the applet automatically is downloaded, similar to an image. However, with the applet, you cannot control the amount of space that the applet takes up on screen.

JavaScript

Initially, Netscape invented a simple scripting language known as LiveScript. LiveScript was a proprietary add-on to the HTML. When Sun's new programming language, Java, became popular, Netscape quickly switched over and came up with a new scripting language called JavaScript. The first four alphabets of the two technologies, Java and JavaScript, are quite similar; otherwise both are entirely different from each other.

JavaScript is the scripting language, which is used in many websites by the Web designer to create a client-side application with very less effort. The scripting language is interpreted at runtime and not compiled like other languages, such as C++, C#. Therefore, JavaScript is an interpreted language, which means that the scripts execute without compilation. JavaScript is also the client-side language, as it runs on the client browser. JavaScript can be used in almost all the Web browsers, such as Internet Explorer, Mozilla Firefox, Netscape and it can easily interact with the HTML elements.

Basically, JavaScript is designed to create interactivity with HTML pages. The following are the uses of JavaScript:

- Allows anyone to put the small snippets of the code into their HTML pages, as JavaScript is simpler to understand.
- Enables writing of dynamic text into HTML page. The variable text can also be written in HTML page, e.g. `document.write("<h1>" +name + "</h1>")`. This command writes the text of the name variable into the HTML page.
- Enables you to read and change the content of HTML controls. For example, the text inserted in the text field of an HTML page can be read with the help of JavaScript.
- Allows you to perform certain validations on the client-side. For example, ensuring that no text field is left blank, matching the passwords, and confirming the entries in the password fields while setting a new password, can be checked at client-side by using JavaScript as the scripting language.
- Allows you to create cookies that can be used to either store or retrieve relevant information on the client's computer.
- Enables you to load a specific page depending upon the client's request.
- Allows functions that are embedded in or included from HTML pages and interact with the DOM of the page.
- Allows you to change an image as the mouse cursor moves over it.
- Helps to call the new Web page, according to the client or user's action.

Till now, you were getting acquainted with the basic technologies related to Web application programming. Now, let's understand the technologies required at the time of Web application development.

Servlet

Java Servlet is an alternative to CGI programs. The major difference between CGI and servlets is that the servlets are persistent. In other words, once loaded, it stays to fulfill other subsequent requests. On the contrary, CGI disappears after fulfilling the request once. Moreover, similar to the applets that run on the browser, the servlets run on Java-enabled Web server.

The Java Servlet Application Programming Interface (API) allows the developer to add dynamic content to a Web server using Java platform. The servlets maintain the state across multiple requests by using HTTP cookies, session variables, Uniform Resource Locator (URL) rewriting, or the hidden fields. The Servlet API contains the `javax.servlet` package hierarchy. The Servlet API also specifies the expected interaction of the Web container and a servlet. The Web container is part of the Web server and performs numerous tasks. Some of these tasks include interacting with the servlets, managing the life cycle of the servlets, and mapping the URL to a particular servlet, if the URL requester has the correct access rights.

The main purpose of the servlets is to serve HTML to the client, mainly through HTTP protocol. Servlets are created, managed, and destroyed by the Web server, where they run. The servlets are recognized in the context of the Web server.

JavaServer Pages

JSP is an enhancement to the Java Servlet technology, offered by Sun Microsystems. The JSP technology provides a technique to dynamically generate Web pages. JSP also simplifies the process of creating or developing web-based applications.

The JSP technology allows HTML to be combined with Java on the same page. In response to the Web client's request, JSP allows software developers to dynamically generate HTML documents. The JSP technology allows Java code and certain pre-defined actions to be embedded into static content. The Java code is added with the use of the JSP directives, JSP scripting elements, and JSP actions. The JSP syntax adds additional XML-like tags, called JSP actions. The JSP compiler compiles JSP into servlets. The JSP compiler generates a servlet in Java code, which is then compiled by the Java compiler. The extension of the JSP files is `.jsp`.

Figure A.2 shows the different clients connecting to the Web server through the Internet:

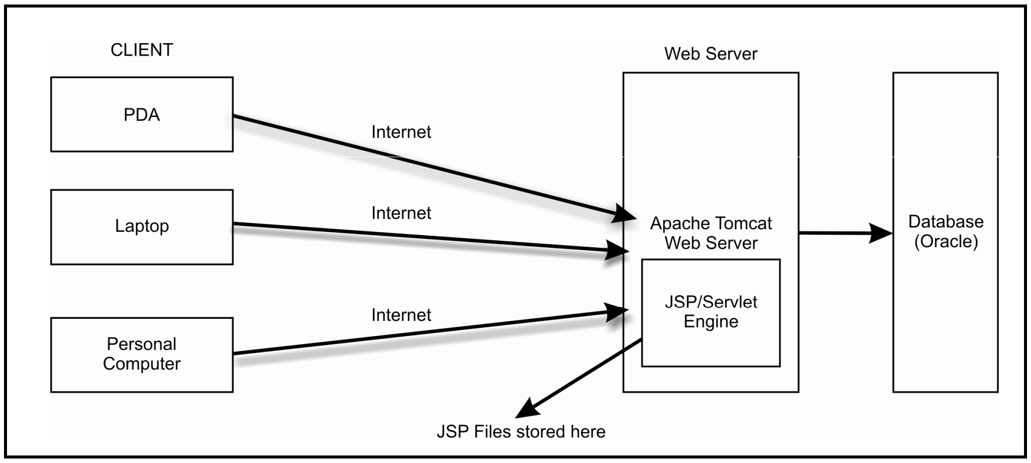


Figure A.2: Different Clients Connected to the Web Server

Figure A.2 shows the most popularly used Web server, Apache Tomcat Web server, running on Windows. As shown in Figure A.2, the JSP files run on the Web server in the JSP Servlet engine. The JSP Servlet engine dynamically generates the HTML output and sends it to the client's browser.

Active Server Pages

ASP is the Microsoft's server-side scripting engine for dynamically generating HTML or the Web pages for the Web browser. The default scripting language used to write ASP is VBScript. The simple approach to understand ASP is that ASP is a program that runs inside Internet Information Server (IIS). Similar to JSP, i.e. a server-side technology given by Sun Microsystems, ASP is the server-side technology provided by Microsoft.

The ASP page, though similar to the HTML page, also contains text, XML, and scripts. The scripts in an ASP file are executed on the server. The extension of the ASP files is `.asp`. When the Web browser sends the request for an ASP file, the IIS passes the request to the ASP engine, which then executes the scripts in the file after reading it line by line. Later, in the plain HTML format, the ASP file is returned to the Web browser.

The modification and additions to the contents of the Web page can be done dynamically with the help of ASP. The data from the database can also be accessed with the help of ASP. In addition, the result is returned to the Web browser. ASP, in comparison to CGI and Perl, is simpler and faster.

Hypertext Processor

Till now, you were aware of Sun Microsystems and Microsoft technologies, which are JSP and ASP, respectively. Let's move on to understand another server-side scripting technology called PHP, which is an alternative to ASP and JSP. Hypertext Processor, more commonly known as PHP, helps to create dynamic Web pages. You can embed PHP into HTML pages and generate a dynamic Web page. PHP programs can be deployed on a Web server. PHP uses the concept of CGI for server-side programming. It acts as a filter for displaying the dynamic content and can be used for extracting data from a database.

Dynamic HTML

Before studying DHTML in detail, let's first expand the term DHTML. DHTML stands for Dynamic Hypertext Markup Language and is the art of making the HTML pages dynamic. DHTML is a combination of technologies used to create dynamic and interactive websites and Web applications. In standard HTML, once the page is loaded from the server it will not change until another request is made to the server. On the contrary, dynamic HTML provides more control over HTML elements. It allows the HTML elements to change at any time without returning to the Web server; and uses the DOM, Cascading Style Sheet (CSS), HTML, and JavaScript to develop interactive Web applications.

Document Object Model

DOM allows changing any part of the Web page using DHTML. DOM is the API that serves as glue for binding a scripting language, such as JavaScript, with the markup language, such as HTML. HTML DOM defines the standard set of objects for HTML and allows access and manipulation of HTML objects in a standard way. It helps in specifying each part of the Web page and allows access by using naming conventions.

Cascading Style Sheets

CSS is used in DHTML to control the look and feel of a Web page. CSS is used to style the HTML elements. The font color, font size, background color, images, and the placements of objects on the Web page are defined in the CSS files. CSS allows the designer to control the style and layout of various Web pages. This can be done by defining the style for each HTML element and then can be applied to multiple Web pages at a time. This also enables a quick global change. In other words, if the same style is applied to all the HTML elements, all the elements will get automatically updated by simply making changes in the style. Therefore, DHTML helps in making the Web pages dynamic and provides a good look and feel of the Web page with the help of CSS.

The HTML elements or objects are placed in the Web page by using XHTML. With the help of DOM specified in the Web page, these objects, placed in the Web page, can be accessed or manipulated at any time. Now, let's understand XML.

XML

XML is the markup language used for the exchange of information between applications or organizations. XML allows the designers to create their own user-defined tags and enable the transmission, validation, and interpretation of data between applications or the data between the organizations. In simpler words, it allows the designer to provide data in tags by creating meaningful tags. Some of the XML related technologies are as follows:

- ❑ `XHTML`
- ❑ `XML DOM`
- ❑ Extensible Stylesheet Language Transformations (`XSLT)`
- ❑ `XML Parser`

Let's have an overlook to the listed XML technologies.

XHTML

As discussed earlier, XHTML is a cleaner but stricter version of HTML. XHTML is similar to HTML 4.x, and is defined in the form of an XML application. It consists of elements in HTML 4.01, combined with the syntax of XML. As previously stated, XML is the markup language that results in well-formatted documents. Therefore, XHTML provides the privilege of writing well-formed documents, which work in all the Web browsers. Certain rules to be followed while using XHTML are as follows:

- XHTML elements should be properly nested
- XHTML elements should always be closed
- XHTML elements should be in lowercase
- XHTML documents must have a single root element

XML DOM

DOM refers to the Document Object Model and presents the XML document as the tree-structure having the Root node as the parent element and the Elements, Attributes, and Text defined as the child nodes. Therefore, XML DOM defines the standard way for accessing and manipulating XML documents. The elements containing the text and the attributes, with the help of the DOM tree, can be manipulated and accessed. The contents of these elements can be modified, new elements can be created, or the unwanted elements can be removed from the DOM tree. The most important thing to be noted is that all the Elements, their Text, and their Attributes are known as the nodes.

In the DOM structure, the entire document is considered as the Document node; XML tag or the XML element is recognized as the Element node; the text in the XML elements are referred to as the Text node; attributes are considered the Attribute nodes; and the comments are considered the Comment node. In the DOM tree-structure, the nodes have a hierarchical relationship with each other. The terms parent and child are used to describe the relationships between the nodes.

Let's consider an example of a XML file and look at its DOM tree-structure. The code for the `product.xml` file containing the data related to various products is provided in Listing A.1:

Listing A.1: Displaying the Code for the products.xml File

```
<?xml version="1.0" encoding="UTF-8"?>
<PRODUCTDATA>
 <PRODUCT PRODID="P001">
        <PRODUCTNAME>Barbie Doll</PRODUCTNAME>
<DESCRIPTION>This is a toy for children in the age group Below 5 years  </DESCRIPTION>
        <PRICE>$24.00</PRICE>
        <QUANTITY>12</QUANTITY>
 </PRODUCT>
 <PRODUCT PRODID="P002">
        <PRODUCTNAME>Mini Bus</PRODUCTNAME>
<DESCRIPTION>This is a toy for children in the age group of 5-10 years  </DESCRIPTION>
        <PRICE>$42.00</PRICE>
        <QUANTITY>6</QUANTITY>
 </PRODUCT>
 <PRODUCT PRODID="P003">
        <PRODUCTNAME>Car</PRODUCTNAME>
<DESCRIPTION>This is a toy for children in the age group of 10-15 years </DESCRIPTION>
        <PRICE>$60.00</PRICE>
        <QUANTITY>21</QUANTITY>
 </PRODUCT>
</PRODUCTDATA>
```

In Listing A.1, `<PRODUCTDATA>` is the root element of the document. As all the other elements are within the `<PRODUCTDATA>` element, it is considered as the root element. The root element has three `<PRODUCT>` nodes and an Attribute node named, `PRODID`. Each of the Element nodes has a Text node as well.

Figure A.3 shows the DOM tree-structure for `products.xml`:

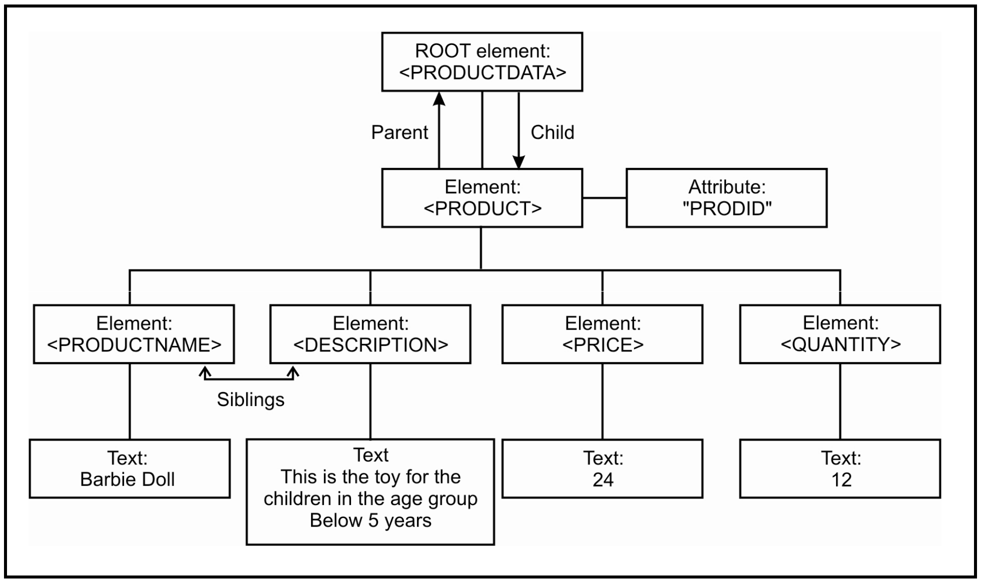


Figure A.3: Displaying the DOM Node Tree-structure

Figure A.3 shows only one child node, <PRODUCT>, of the parent node <PRODUCTDATA>. The `<PRODUCT>` child node has four Element nodes and an Attribute node. Each Element node has the respective Text node, as shown in Figure A.3.

XSLT

XSLT stands for Extensible Stylesheet Language Transformations, and represents a language used for transforming XML documents into XHTML or other XML documents. XSLT uses XPath for navigating through XML documents and finding information in it. In the transformation process, XSLT uses the XPath searches for those parts of the source document that match the pre-defined template. When a match is found, XSLT transforms the source document into the resultant document by applying the pre-defined template.

XML Parser

The XML parser is used to read, update, create, and manipulate XML documents. For manipulating the XML document, the XML parser loads the document into the computer's memory and then manipulates data by using the DOM node-tree-structure. The XML parser is the part of the software that reads the XML files and tests whether the XML document is well-formed against the given Document Type Definition (DTD) or the XML schema. Moreover, the XML parser also makes the XML files available to the application with the use of the DOM.

Till now, the appendix has dealt with the explanation of almost all the technologies used to create Web applications. All these technologies discussed earlier share a common problem, and AJAX proves to be the solution for these problems. Read on to know about them.

Problems of the Traditional Technologies

All the previously mentioned technologies use the classical or traditional Web application model. In the classical or traditional Web application model, the nature of interaction between the client and the server is of start-stop. In a traditional Web application model, the browser responds to the user action by discarding the current HTML page. Then, the request is sent back to the Web server and when the server completes the processing of request, it returns the response page to the Web browser. Finally, the browser refreshes the screen and displays the new HTML page. You will be surprised to note that the user is bound not to do anything on the Web pages until the entire process is completed.

In technical terms, the problem with the classical Web applications model was the synchronous request–response communication model. This can be explained with the help of Figure A.4:

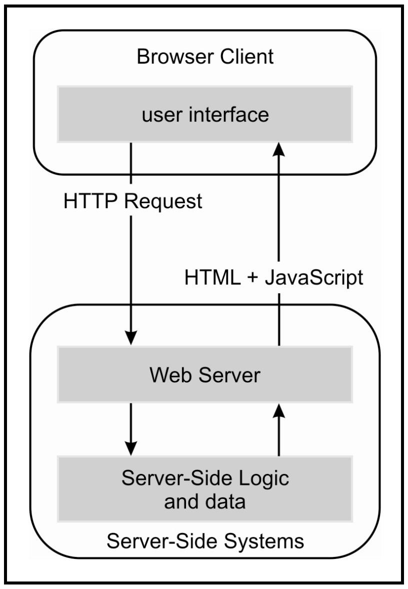


Figure A.4: Displaying the Classical Web Application Model

The classical Web application model, shown in Figure A.4, makes technical sense; however it does not provide the best user experience. As this classical application model keeps the user waiting, it does not provide the best user experience. To overcome this, the developer noticed a technical approach that Google used. After analyzing the facts of Google, on February 18, 2005 Jesse James Garrett, President, and founder of the Adaptive Path, came out with a new technique AJAX— that was based on the approach used by Google.

Let us move on further to learn about AJAX.

AJAX—the Solution

AJAX, a new approach to Web applications, is based on several technologies that help to develop applications with better user experience. It uses JavaScript and XML as the main technology for developing interactive Web applications. These applications are based on AJAX Web application model, which uses JavaScript and XMLHttpRequest object for asynchronous data exchange. The JavaScript uses XMLHttpRequest object to exchange data asynchronously over the client and server. Let's move further to have a detail study on the AJAX Web application Model.

AJAX Web Application Model

You already know that the major issue with regard to the classical Web application model was resolved through AJAX. The AJAX application eradicates the start-stop-start-stop nature or the click, wait, and refresh criteria of the client-server interaction. Figure A.5 shows how the intermediary layer is introduced between the user and the Web server:

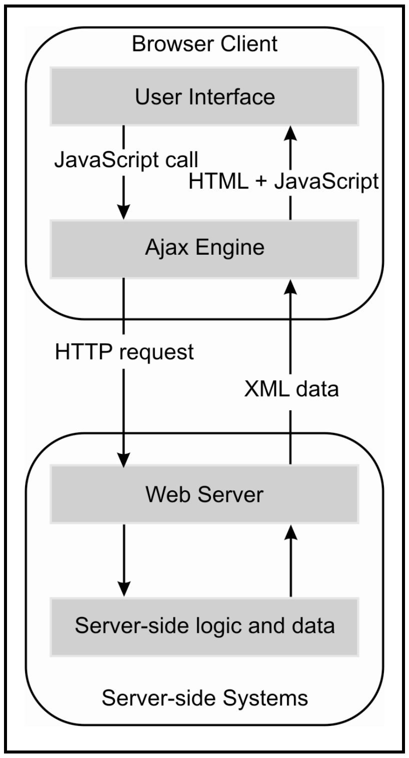


Figure A.5: Displaying the AJAX Web Application Model

Instead of loading the Web page during the beginning of the session, the browser loads the Ajax engine, written in JavaScript. As shown in Figure A.5, the Web page sends its requests using a JavaScript function. This JavaScript code makes a request to the server. The server response comprises of data and not the presentation, which implies that the data required by the page is provided by the server as the response, and the style or presentation is implemented on that data with the help of the markup language. Most of the page does not change. The parts of the page that need to change are updated. In other words, JavaScript dynamically updates the Web page, without redrawing everything. For the Web server, nothing has changed; it still responds to each request, just as it did earlier!

Though JavaScript makes a request to the server, you can still type in Web forms and even click buttons, while the Web server is still working in the background. Then, when the server completes its processing, your code updates just the part of the page that has changed. This way, you do not have to wait around for the entire cycle to be completed, which reflects the power of asynchronous requests.

AJAX engine, between the user and the application, irrespective of the server, does asynchronous communication. This prevents the user from waiting for the server to complete its processing. The AJAX engine takes care of displaying the user interface and the interaction with the server on the user’s behalf.

However, in traditional Web applications, the synchronous mode of communication existed between the client and the server, as shown in Figure A.6:

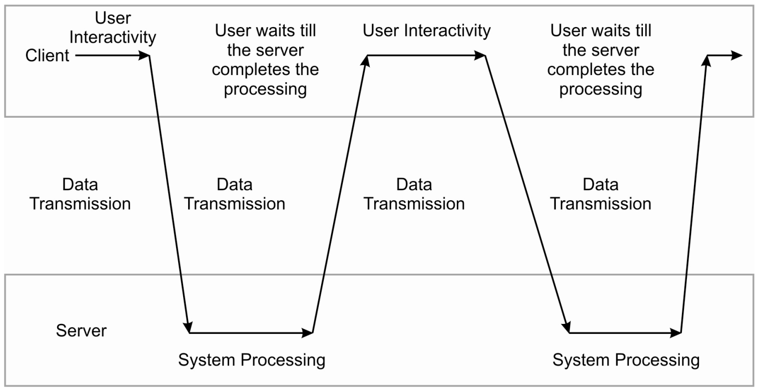


Figure A.6: Displaying the Synchronous Mode of Communication

As the essence of AJAX is a partial screen update and the asynchronous communication, the programming model, shown in Figure A.7, is not bound to a specific data exchange format or the specific programming language or the specific communication mechanism. Figure A.7 shows the asynchronous mode of communication:

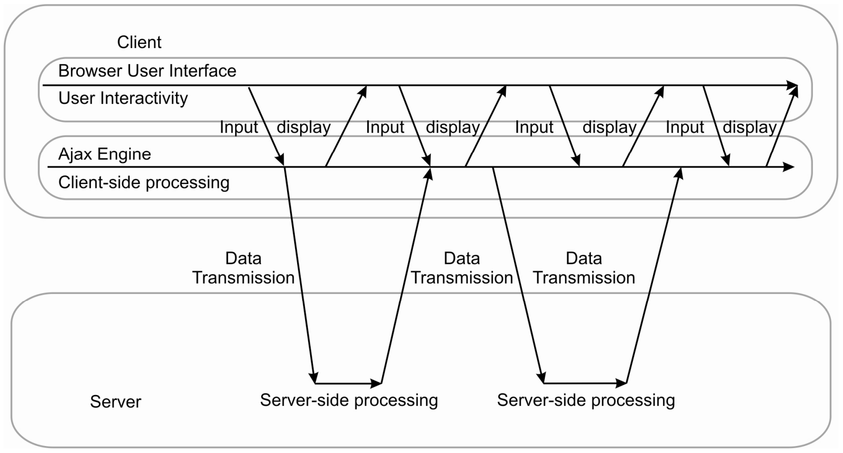


Figure A.7: Displaying Asynchronous Mode of Communication

As shown in Figure A.7, every user action generates an HTTP request that takes the form of a JavaScript to call the AJAX engine. Any response to the user action does not require the trip back to the server, unlike the classical Web application model. Rather, the AJAX engine handles on its own; i.e., the data validation, some navigational functions, editing data in memory, and so on, are handled by the AJAX engine.

If the AJAX engine needs to retrieve new data from the server or load additional interface code, the engine makes an asynchronous interaction with the server, using JavaScript and XMLHttpRequest object for asynchronous data exchange. The engine's interaction with the server does not interrupt the user's interaction with the application. In this way, asynchronous communication is done with the help of the AJAX engine.

Comparing Figures A.6 and A.7, it can be seen that in the asynchronous mode of interaction, there is no scope for the user to wait until the server-side processing gets over. The AJAX Web application model allows users to continue working, and simultaneously, if necessary, the AJAX engine interacts with the server without interrupting the user's interaction with the application.

After learning how AJAX works and how the problems or shortcomings of traditional technologies are overcome by using AJAX, let's create an AJAX application by using JavaScript.

Creating a Sample AJAX Application

Let's consider a scenario of a Jewelry showroom. The owner wants to design a Web page that would display the different jewelry items, along with some information, to its various customers. When the user points to an image whose information he wants to know, the details will be displayed on the Web page without the page being refreshed repeatedly. Let's create a jewelry application. In this application, we first need to create a Jewelery.html page, which shows information about the different jewelry in the showroom.

The code for the Jewelery.html page of the application is provided in Listing A.2 (save the Jewelery.html file in the \jewelry folder):

Listing A.2: Displaying the Code for the Jewelery.html File

```
<html>
<head>

<title>First AJAX Application</title>
<script language = "javascript">
  var XMLHttpRequestObj = false;
  if (window.XMLHttpRequest)
  {
        XMLHttpRequestObj = new XMLHttpRequest();
  }
  else if (window.ActiveXObject)
  {
        XMLHttpRequestObj = new
        ActiveXObject("Microsoft.XMLHTTP");
  }
  function getData(dataSource, divID)
  {
        if(XMLHttpRequestObj)
        {
                var obj = document.getElementById(divID);
                XMLHttpRequestObj.open("GET", dataSource);

                XMLHttpRequestObj.onreadystatechange = function()
                {
                        if (XMLHttpRequestObj.readyState == 4 &&
                         XMLHttpRequestObj.status == 200)
                        {
                                obj.innerHTML =
                                XMLHttpRequestObj.responseText;
                        }
                }

                XMLHttpRequestObj.send(null);
        }
  }
</script>
</head>
<body>
<H1>First Application using AJAX</H1>
<img src="images\image1.jpg" onmouseover = "getData('bangles.txt','targetDiv')">
<img src="images\image2.jpg" onmouseover = "getData('rings.txt','targetDiv')">
<img src="images\image3.jpg" onmouseover = "getData('necklaces.txt','targetDiv')">
<div id="targetDiv">
        <h1>Welcome to my Jewelery Showroom!</h1>
  </div>
</body>
</HTML>
```

In Listing A.2, an HTML page is designed, displaying the images of the various jewelry items available in the showroom. As soon as the user points the mouse pointer on any of the three images, the text saved in the respective text files are displayed on the Web browser. The three text files are as follows:

- `bangles.txt (Listing A.3)`
- `rings.txt(Listing A.4)`
- `necklaces.txt(Listing A.5)`

You also need to add a web.xml file in the WEB-INF folder of the application in which Jewelery.html page is mapped as the welcome page. Now, let's understand how the Jewelery.html page uses AJAX. When the mouse moves over any of the three images, the `onmouseover` event is generated and the JavaScript function, `getData()`, is called, as shown in the following code snippet:

```
<body>
   <H1>First Application using AJAX</H1>
   <img src="Image1.jpg" onmouseover="getData('bangles.txt','targetDiv')">
   <img src="Image2.jpg" onmouseover="getData('rings.txt','targetDiv')">
   <img src="Image3.jpg" onmouseover="getData('necklaces.txt','targetDiv')">
   <div id="targetDiv">
          <h1>Welcome to my Jewelery Showroom!</h1>
   </div>
</body>
```

The `getData()function` passes two text strings—first, the name of the text file, such as `bangles.txt`, `rings.txt`, or `necklaces.txt`, and secondly the name of the `<div>` element.

The text provided in the bangles.txt file is given in Listing A.3 (save the `bangles.txt file` in the \jewelry folder):

Listing A.3: Showing the Text Displayed on Hovering the Mouse on Bangles Image

```
We offer too many bangles to list!
Gold Bangles
Diamond bangles
```

The text provided in the rings.txt file is given in Listing A.4 (save the rings.txt file in the \jewelry folder):

Listing A.4: Showing the Text Displayed on Hovering the Mouse on Rings Image

```
Rings: Amazing rings with wonderful designing.
Heart-shaped rings
Diamond rings
gold rings
Gold plated rings
```

The text provided in the necklaces.txt file is given in Listing A.5 (save the `necklaces.txt file` in the \jewelry folder):

Listing A.5: Showing the Text Displayed on Hovering the Mouse on Necklace Image

```
The all kind of diamond and gold necklaces are also available.
Diamond necklace
Gold necklace
The necklaces are also designed according to your choice.
```

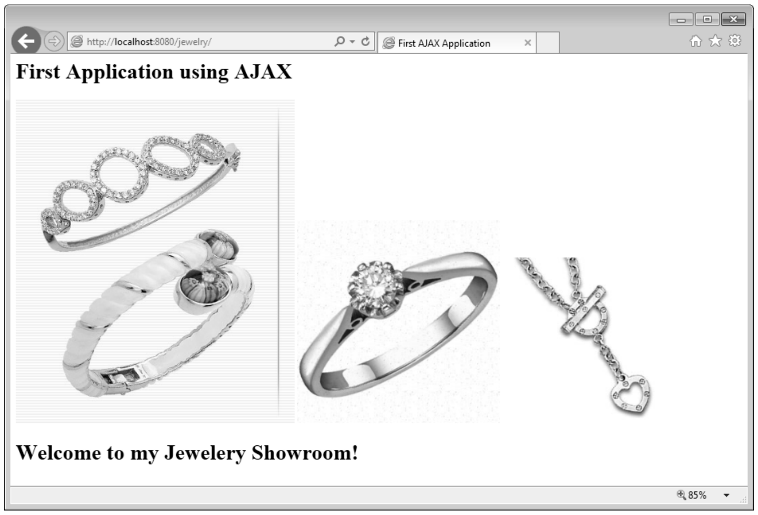
Apart from the preceding text files, three image files of the bangles, rings, and necklaces are also displayed on the Web page.

Any of these three text files is downloaded by the browser from the server, which is in the background, while the user is working with the rest of the Web page. Read on further to understand how this is done.

To run the jewelry application, ensure that a Web server is configured on your machine. Here, we are using Glassfish Server 4.0 as a Web server for running AJAX-based Web application. Package and deploy the jewelry application on Glassfish Server. Follow these steps to run the application:

- Open the Internet Explorer (IE).
- Type the following address (http://<IP address of Web Server>:8080/jewelry/) in the address bar of the IE and press Enter key to run the application (Figure A.8).

Figure A.8 shows the output of the jewelry application:

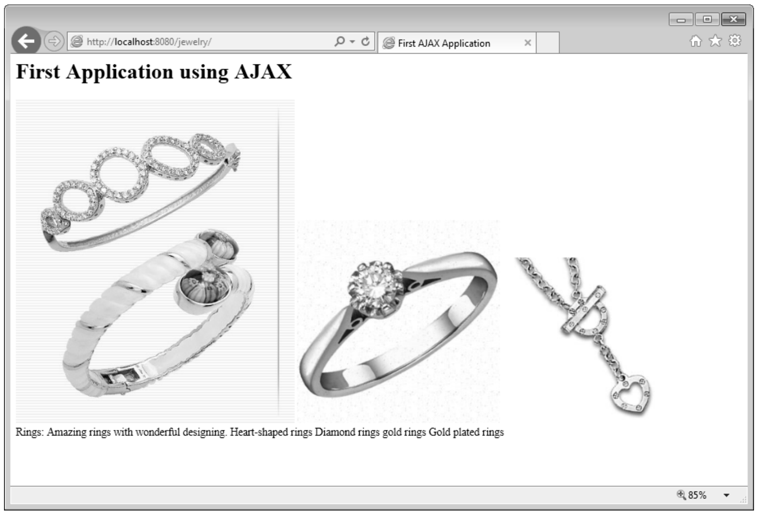


First Application using AJAX

Welcome to my Jewelery Showroom!

Figure A.8: Displaying the Output of the jewelry Application

- When you move the mouse pointer on any of the images, the text related to that image get displayed, as shown in the Figure A.9:



First Application using AJAX

Rings: Amazing rings with wonderful designing. Heart-shaped rings Diamond rings gold rings Gold plated rings

Figure A.9: Displaying a Simple AJAX Example

As the mouse moves from one image to another, the JavaScript in the page fetches some new text and replaces the older text, without even a screen flicker or page fetch or fuss.

Creating the **XMLHttpRequest** Object

The application created in this subsection requires the `XMLHttpRequest` object. In Listing A.2, towards the beginning of Jewelery.html code, locate the following code snippet:

```
<script language = "javascript">
var XMLHttpRequestObj = false;
.
.
.
```

The preceding code snippet declares a variable XMLHttpRequestObj. As this code snippet is outside any function, it runs immediately when a JSP page is loaded. This variable is set to false, so that the script can check later whether the variable is created or not. In case of browsers, such as Netscape, Firefox, and Opera, the XMLHttpRequest object is usually the part of the browser's window object and can be accessed as window.XMLHttpRequest. If the window.XMLHttpRequest returns true, an XMLHttpRequest object is created with the following code snippet:

```
if (window.XMLHttpRequest)
{
   XMLHttpRequestObj = new XMLHttpRequest();
}
```

On the contrary, a different perspective is required for the Internet Explorer Web browser. The ActiveXObject in the Internet Explorer (version 5 and above) is used to create the XMLHttpRequest object, as shown in the following code snippet:

```
if (window.ActiveXObject)
{
   XMLHttpRequestObj = new
   ActiveXObject ("Microsoft.XMLHTTP");
}
```

Therefore, depending upon the browser you are using, an XMLHttpRequest object is created.

When the user moves the mouse over the images, an "onmouseover" event is generated, which calls the getData() function. When the getData () function is called, the XMLHttpRequest object is first checked to see whether it is valid, and then further processing is done.

Opening the **XMLHttpRequest** Object for Asynchronous Downloads

When the valid object of the XMLHttpRequest is created, the object calls its open () function. You can configure the object to use the URL you want, by using the object's open() function. The syntax of the open () function is as follows:

```
req.open("GET",URL,true);
```

In the preceding code snippet, the first parameter indicates the type of HTTP method that is used for sending request; the second parameter is the URL of the requested resource and; the third parameter is optional, which shows whether the request is synchronous or asynchronous. The default value of third parameter is true, which indicates an asynchronous request.

In this application, the URL from which the data you want to fetch is passed from the getData() function as the dataSource argument. The URL can be opened with the standard HTTP techniques, such as GET, POST, or PUT. The following code snippet uses the GET method to request the respective text file on the server:

```
XMLHttpRequestObj.open("GET", dataSource);
```

When you open the XMLHttpRequest object, the XMLHttpRequest object contains the property named onreadystatechange, which allows handling the asynchronous loading operations. If this property is assigned to any JavaScript function, this function will be called each time the XMLHttpRequest object's state changes.

This JavaScript function is also known as "callback" function. When the server returns with the information, the callback function is invoked. In turn, the callback function can display the new information to the user. We have defined the callback function with the following JavaScript code snippet:

```
XMLHttpRequestObj.onreadystatechange = function ()
{
   if (XMLHttpRequestObj.readyState == 4 &&
    XMLHttpRequestObj.status == 200)
   {
         obj.innerHTML =
         XMLHttpRequestObj.responseText;
   }
}
```

Finally, when the `XMLHttpRequest` object is in its ready state and the status is equal to 200, then the data is fetched. The readyState value "0" indicates that the request is completed and the status 200 refers to the 'Ok' state of the `XMLHttpRequest` object, which means that the request resource is completely downloaded. The five states of the `XMLHttpRequest` object are as follows:

- 0 for uninitialized state
- 1 for loading state
- 2 for loaded state
- 3 for interactive state
- 4 for the complete state

The `status` property holds the status of the download. Table A.1 provides some possible values of the status property:

Table A.1: Possible Values for the status Property of XMLHttpRequest Object

Status	Possible values
Ok	200
Created	201
No Content	204
Reset Content	205
Partial Content	206
Bad Request	400
Unauthorized	401
Forbidden	403
Not Found	404
Method Not Allowed	405
Not Acceptable	406
Proxy Authentication Required	407
Request Timeout	408
Length Required	411
Requested Entity Too Large	413
Request URL Too Long	414
Unsupported Media Type	415
Internal Server Error	500
Not Implemented	501
Bad Gateway	502
Service Unavailable	503
Gateway Timeout	504
HTTP Version Not Supported	505

Therefore, to make sure that the data is completely downloaded, the value for the status property must be 200. Finally, when the data is completely downloaded, it is retrieved in either the standard HTML or the XML format. The `responseText` property is used to retrieve the data in standard HTML format. However, if your data is formatted as XML, then `responseXML` property is used.

After retrieving the data, you have to display the data on the Web page. The data is displayed by using the HTML <div> element, as shown in the following code snippet:

```
    <div id="targetDiv">
    <h1>Welcome to my Jewelery Showroom!</h1>
    </div>
```

The <div> element shows the location where you want to display the data. The id attributes is used to identify the <div> element and it is passed as an argument to the getData() function for bangles.txt file. The following code snippet shows the implementation of the `getData()` function passing `bangles.txt` and `targetDiv` as arguments:

```
    getData ('bangles.txt','targetDiv')
```