Chapter 6.

SECTION II: JAVA SERVLETS

Introduction To Java Servlets

Over the past few years, there have been some incredible changes in the way computers are used. Large and cumbersome commercial applications, run on the desktop, are gradually losing their significance. With the rapid spread and adoption of the **W**orld **W**ide **W**eb as a network without geographical boundaries, information and business applications need to be accessed and run from virtually anywhere.

The web pages are not static anymore instead they are used for delivering services such as displaying bank account details, bank transactions, booking tickets, weather reports, news headlines and so on. Web pages have never been more capable of delivering dynamic content with the greatest of ease, than today.

In such a scenario, the demand for faster, lighter, robust, scaleable, commercial applications that can deliver across the Internet has risen exponentially.

Today there are several powerful tools to work with. In the past, if a database-driven application was required this was limited to writing **C**ommon **G**ateway **I**nterface [CGI] scripts to process form data and return results. CGI has served the purpose of adding interactive and dynamic content to a Web site in the past, it's most serious drawback being its lack of scalability and platform dependence.

Scripting Internet delivered, commercial applications using **A**ctive **S**erver **P**ages [ASP] and PHP have helped simplify Web application development. Such programming environments have provided developers with simple, tried and tested, methodologies to create robust, scalable, Web enabled commercial applications.

To address the specific requirements of building scalable, extensible, portable Server-side solutions for Internet based commercial applications, Sun Microsystems has developed a technology called **Servlets**, crafted using Java.

Java Servlets are efficient, due to an intuitive and automated threading model in which each request is handled by a new lightweight thread. Servlets are also platform [**i.e.** hardware] independent, they run using a set of standard interfaces [that make up the Servlet engine] and a **J**ava **V**irtual **M**achine [JVM]. It is the JVM that is built to run on a specific hardware platform. Thus if a JVM exists for a specific hardware platform any Servlet crafted and validated on any other hardware platform will run successfully.

Java Servlets actually provide an object-oriented and extensible middle tier for Web Server based applications. Servlets can access all of the enterprise Java APIs like JNDI, JDBC, RMI and Enterprise JavaBeans and so on.

This chapter will introduce and examine Java Servlets from their conceptual and code spec angles together with their related Server-side Java technologies. This should expose exactly how Server-side Java can act as a strong dependable code underlay in developing N-tier applications. Applications based on browser based client code accessing business logic and data services through Servlets run on a Java enabled Web server.

What Is A Servlet

A **Servlet** is a Server-side java program that is platform independent. Servlets, written in Java dynamically extend the functionality of any Java enabled Web server. Web servers generally cannot talk to databases. Hence, a Web server alone cannot dynamically create web page content using data held in a database table. Even if there is a large amount of legacy data in a database table, this information cannot be made accessible to public via the Web server, Internet and Browser combination.

This is where Servlets, [**i.e.** basically Java *.class files*] help extend the functionality of a Web server. Web servers use the Servlet's ability to access a database table, extract table data, and convert this data into a format acceptable to a Web server for delivery to a client's Browser via the Internet.

Servlets are written in Java. Java is a very portable language that obeys rules that permit Servlets to run on a standard framework [**i.e.** the JVM]. They provide a means to create sophisticated Web server extensions and are independent of operating system and hardware platform.

Even though Servlets are written in Java, their clients may not necessarily be written in Java. Servlet clients [**i.e.** an application that consumes the output of a Servlet] can be written in any desired language. This simply means that Servlets are used in the middle tiers of distributed application systems. Servlets can in turn be clients to other services, which are written in some other language.

Servlets are not tied to a specific client-server protocol but they are most commonly used with HTTP. The word **Servlet** is often used to represent **HTTP Servlet**.

Since Servlets run within a Web server's namespace they do not display a graphical interface to a user. A servlet's work is done behind the scene **i.e.** working with a Web server. Only the results of a servlet's processing [table data converted to suitably formatted HTML] are returned to a client's browser.

A Servlet Container [i.e. the Servlet Engine] provides the runtime environment in which a Servlet executes. The Servlet engine manages the life-cycle of Servlets from the moment they are spawned until they are destroyed. The Servlet engine exposes an API based on the version of the Servlet specification that it implements.

Servlets are supported by several Web servers such as Apache, Netscape, Microsoft IIS and others.

HINT

Servlets are the Web server side counterpart to Applets. Applets run only on the client side within a Java enabled Browser.

Applets are Java application components, which are downloaded, on demand, run in a Java enabled client Browser and thus participate in the commercial application, which needs them.

Why Servlets

- Are loaded into memory once and run from memory thereafter
- Are spawned as a thread, not as a process
- Are a powerful object-oriented abstraction of HTTP
- Are portable across multiple Web servers and platforms
- Are simple to design and implement
- Are tightly integrated with the Web server
- Service client requests efficiently
- Run within the secure and reliable scope of a **J**ava **V**irtual **M**achine [JVM]
- Obey the rules of a standard Servlet API
- Return vanilla HTML to the requesting Browser
- Are robust, scaleable, secure CGI Replacement
- Provide direct Database Access using native and ODBC based Db drivers
- Being on the Server-side provides code protection
- Are supported by several Web servers

What Can Servlets Do

A Servlet is a small, self-contained, portion of code that is executed at the Web Server in response to an HTTP request from a client's Browser. Servlets can do the following:

- Servlets can be used as a plug-in, that features a search engine or semi-customized applications such as web-based banking or inventory systems
- A Servlet can be used to process data POSTed over HTTP(S) using an HTML form and apply appropriate business logic
- A Servlet can handle multiple requests concurrently and these requests can be synchronized to support systems allowing collaboration between people
- Servlets can forward requests to other Servers and Servlets. This allows them to be used to balance load among several servers that mirror the same content
- Servlets can maintain and track sessions (either by URL rewriting or cookies). Servlets clean up the session precisely when a session is terminated, either by logging off or by a time-out
- Servlets can pass data between themselves
- Servlets can be used to serve as the HTTP speaking middle tier between any kind of thin client and large enterprise services being made available via EJB's

Servlets V/s CGI

- Java Servlets are more efficient, easier to use, more powerful, more portable and cheaper than traditional CGI and many alternative CGI-like technologies such as Perl
- With CGI, a new process is started for each HTTP request. The initialization process of the CGI program takes a longer time than its execution time

  With Servlets, the Java Virtual Machine stays up in the memory between requests and is handled by a lightweight Java thread, not a heavyweight operating system process
- If there are N simultaneous requests to the same CGI program, then the code for the CGI program is loaded into memory N times

  With Servlets, however, there are N threads but only a single copy of the Servlet class
- Servlets can be used for optimizations such as caching previous computations, keeping database connections open, than a regular CGI program
- Servlets have an infrastructure for automatically parsing and decoding HTML form data, reading and setting HTTP headers, handling cookies, tracking sessions and many other such utilities

  With these utilities, Java Servlets do several things that are difficult with regular CGI

- Servlets can talk directly to the Web server, whereas CGI programs cannot. Servlets can share data among each other, making database connection pools easy to implement
- Servlets are written in Java and follow a standardized API. Servlets written for I-Planet Enterprise Server can run virtually unchanged on Apache, Microsoft IIS or WebStar. This cannot happen with CGI
- Servlets are supported directly or via a plugin on almost every major Web server

How Are Servlets Created

Servlet code spec can be written using any ASCII editor. The code spec crafted is saved to the hard disk using <filename>**.java**. Notepad is an excellent example of a simple, no frills ACSII editor. <filename>**.java** is the Servlet's source file. This must be compiled using the Java compiler [**i.e.** javac], which is part of the **J**ava **D**evelopment **K**it [JDK].

After compilation, the source file will be converted to Java byte code and saved to the hard disk as <filename>**.class**. This is the Java **.**class file, which will run in a **J**ava **V**irtual **M**achine [JVM] anywhere.

Where Are Servlets Run

A Servlet runs behind the scene [without a Graphical Interface] on a Web server. Servlets always work in conjunction with the Web server software. Servlets provide a framework for creating applications that implement a **request / response** paradigm which is bound to the Web server.

For example, a browser sends a request to the Web server for HTML page content. The Web server will forward this ***request*** to a Servlet, via its extender software. At this point the Servlet will process the request [*access a database, table or run some other process*] construct an appropriate ***response*** [*usually formatted HTML*], which is returned to the requesting Browser via the Web server.

What Do Servlets Look Like

Servlets are based on request-response paradigm where requests are accepted and responses are generated. Servlets implement the Servlet interface either by extending a generic **or** an HTTP-specific Servlet interface implementation.

HINT

An interface is like a class with the following restrictions:

1. All its methods must be an abstract instance methods and no static methods are allowed
2. The variables defined in an interface must be a constant [i.e. static final]. The values of these variables need not be declared at compile time, but some computation can be done at class load time to compute the values. The variables need not be just of type int or string, they can be of any type
3. The static initializer methods cannot be defined in an interface. They should be defined outside the interface

The following is a Servlet code spec template:

```
import java.servlet.*;
public class MyFirstServlet extends HttpServlet
{
   public void service(ServletRequest request, ServletResponse response) throws
                              ServletException, IOException
   {
        . . .
   }
   . . .
}
```

The **service()** method has two parameters named Request and Response. These parameters encapsule the data sent by the client, provide access to parameters and allow Servlets to report status, including errors.

When MyFirstServlet.java is compiled using javac, the result of the compilation is a file MyFirstServlet.class.

How Are Servlets Run

In a production environment, Servlets are run using Web servers.

The Servlets API is a standard framework, used to write Servlet code spec. Servlet Engines provide Runtime Environment to Servlets. As long as the Servlet engine exposes the required Servlet API, the developer need not worry about how PORTABLE the Servlet could be.

It is the Servlet Engine that satisfies the project's requirements. They come as standalone and add-on.

Servlets are supported by nearly all Web servers from **A**pache to **Z**eus. Some Web servers support Servlets right out of the box. These kind of Web servers are called as a **Standalone Servlet Engine**. Other Web servers require a third-party plug-in to support Servlets. These kinds of Web servers are called an **Add-on Servlet Engine**.

Standalone Servlet Engines

A standalone Servlet engine is a server that includes built-in support for Servlets. Such an engine has the advantage that everything works right out of the box. One disadvantage, however, is that one has to wait for a new release of the Web server to get the latest Servlet support.

A Few Examples of Standalone Servlet Engines:

- Tomcat Server from Project Jakarta
- IBM's WebSphere Application Server
- BEA Weblogic Application Server
- Oracle Application Server
- Jetty Server
- iPlanet [Netscape] Web Server Enterprise Edition

Add-On Servlet Engines

An add-on Servlet engine functions as a plug-in to an existing server. It adds Servlet support to a Web server that was not originally designed with Servlets in mind.

Add-on Servlet engines have been written for many Web servers such as**:**

- Apache
- Netscape's FastTrack Server and Enterprise Server
- Microsoft's Internet Information Server and Personal Web Server
- Lotus Domino's Go Webserver
- StarNine's WebSTAR
- Apple's AppleShareIP

This type of engine is especially beneficial when a new release of the Servlet Engine is available. One does not need to wait for new release of the Web server.

A Few Examples of Add-on Servlet Engines:

- Tomcat Server from Project Jakarta

6. Introduction To Java Servlets

- Java-Apache Project's JServ Module
- Allaire's JRun Web Server
- New Atlanta's ServletExec
- Enhydra
- Servertec's iServer
- Unicom's Servlet CGI Development Kit
- Gefion Software's WAICoolRunner

Architecture Of A Servlet

Having answers to the following questions:

- What is a Servlet?
- Why Servlets?
- What Can Servlets Do?
- Servlets v/s CGI?
- How are Servlets created?
- Where are Servlets Run?
- What do Servlets look like?
- How are Servlets run?

It's time to actually answer the most important question. That is the **Architecture Of A Servlet**, which is explained, in the following chapter.