**Java Servlets**

A Java servlet is a Java program that extends the capabilities of a server. Servlets can respond to any types of request. But the common implementation of these are applied on web servers. These Web Servlets are the java counterpart to other dynamic web content technologies (PHP, ASP.NET, etc.)

Servlets provide a component-based, platform-independent method for building Web-based applications, without the performance limitations of CGI programs. Servlets have access to the entire family of Java APIs, including the JDBC API to access enterprise databases.

Servlet life cycle can be defined as the entire process from its creation till the destruction. The following are the paths followed by a servlet

* The servlet is initialized by calling the init () method.
* The servlet calls service() method to process a client's request.
* The servlet is terminated by calling the destroy() method.
* Finally, servlet is garbage collected by the garbage collector of the JVM.

Life cycle methods

**Init():** First method to be called by the Applet. It is used for one-time initializations, just as with the init method of applets. **service():** main method to perform the actual task. The servlet container (i.e. web server) calls the service() method to handle requests coming from the client( browsers) and to write the formatted response back to the client.  
**doGet() :** A GET request results from a normal request for a URL or from an HTML form that has no METHOD specified and it should be handled by doGet() method.  
**doPost() :** A POST request results from an HTML form that specifically lists POST as the METHOD and it should be handled by doPost() method.  
**destroy():** Ends the life cycle of the servlet

## Handling cookies in servlets

## Cookies - text files stored on the client computer and they are kept for various information tracking purpose. Java Servlets transparently supports HTTP cookies. Cookies are usually set as an HTTP header.

Servlet Cookies Methods

**public void setDomain(String pattern) -** sets the domain to which cookie applies.  
**public String getDomain()-** gets the domain to which cookie applies  
**public void setMaxAge(int expiry) -**  sets how much time (in seconds) should elapse before the cookie expires. If not set, the cookie will last only for the current session.

**public int getMaxAge() -** returns the maximum age of the cookie, specified in seconds, By default, -1 indicating the cookie will persist until browser shutdown.  
**public String getName() -** returns the name of the cookie. The name cannot be changed after creation.  
**public void setValue(String newValue) -** sets the value associated with the cookie.  
**public String getValue() -** gets the value associated with the cookie.  
**public void setPath(String uri) -** sets the path to which this cookie applies (If you don't specify a path, the cookie is returned for all URLs in the same directory as the current page as well as all subdirectories).  
**public String getPath() -** gets the path to which this cookie applies.  
**public void setSecure(boolean flag) -** sets the boolean value indicating whether the cookie should only be sent over encrypted (i.e. SSL) connections.  
**public void setComment(String purpose) -** specifies a comment that describes a cookie's purpose. The comment is useful if the browser presents the cookie to the user.  
**public String getComment() -** returns the comment describing the purpose of this cookie, or null if the cookie has no comment.

## Setting Cookies with Servlet:

**Creating a Cookie object:** You call the Cookie constructor with a cookie name and a cookie value, both of which are strings.

Example: Cookie cookie = new Cookie("key","value");

## Sending the Cookie into the HTTP response headers:

## You use response.addCookie to add cookies in the HTTP response header.

Example: response.addCookie(cookie);

## Session Tracking with Servlets: HTTP is a "stateless" protocol which means each time a client retrieves a Web page, the client opens a separate connection to the Web server and the server automatically does not keep any record of previous client request.

## There are still ways that can maintain session between web client and web server. Those are by the use of cookies, hidden form fields, and URL rewriting

## The HttpSession Object

## - Servlet provides HttpSession Interface which provides a way to identify a user across more than one page request or visit to a Web site and to store information about that user.

**- You would get HttpSession object by calling the public method getSession() of HttpServletRequest using this syntax**HttpSession session = request.getSession();

## Methods available through HttpSession object:

## public Object getAttribute(String name) - returns the object bound with the specified name in this session, or null if no object is bound under the name. public Enumeration getAttributeNames() - returns an Enumeration of String objects containing the names of all the objects bound to this session.

## public long getCreationTime() - returns the time when this session was created. public String getId() - returns a string containing the unique identifier assigned to this session. public long getLastAccessedTime() - returns the last time the client sent a request associated with this session public int getMaxInactiveInterval() - returns the maximum time interval, in seconds, that the servlet container will keep this session open between client accesses. public void invalidate() - invalidates this session and unbinds any objects bound to it. public boolean isNew() - returns true if the client does not yet know about the session or if the client chooses not to join the session. public void removeAttribute(String name) - removes the object bound with the specified name from this session. public void setAttribute(String name, Object value) - binds an object to this session, using the name specified. public void setMaxInactiveInterval(int interval) - specifies the time in seconds, between client requests before the servlet container will invalidate this session.

# 