

Introduction to Servlet Request

True job of a Servlet is to handle client request. Servlet API provides two important interfaces **javax.servlet.ServletRequest** and **javax.servlet.http.HttpServletRequest** to encapsulate client request. Implementation of these interfaces provide important information about client request to a servlet.

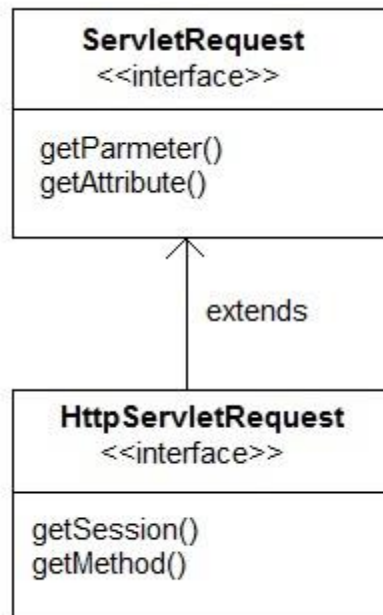
Some Important Methods of ServletRequest

Methods	Description
Object <code>getAttribute(String name)</code>	return attribute set on request object by name
Enumeration <code>getAttributeNames()</code>	return an Enumeration containing the names of the attributes available in this request
int <code>getContentLength()</code>	return size of request body
int <code>getContentType()</code>	return media type of request content
ServletInputStream <code>getInputStream()</code>	returns a input stream for reading binary data
String <code>getParameter(String name)</code>	returns value of parameter by name
String <code>getLocalAddr()</code>	returns the Internet Protocol(IP) address of the interface on which the request was received
Enumeration <code>getParameterNames()</code>	returns an enumeration of all parameter names

String[] <code>getParameterValues(String name)</code>	returns an array of String objects containing all of the values the given request parameter has, or null if the parameter does not exist
ServletContext <code>getServletContext()</code>	return the servlet context of current request.
String <code>getServerName()</code>	returns the host name of the server to which the request was sent
int <code>getServerPort()</code>	returns the port number to which the request was sent
boolean <code>isSecure()</code>	returns a boolean indicating whether this request was made using a secure channel, such as HTTPS.
void <code>removeAttribute(String name)</code>	removes an attribute from this request
void <code>setAttribute(String name, Object o)</code>	stores an attribute in this request.

HttpServletRequest interface

HttpServletRequest interface adds the methods that relates to the **HTTP** protocol.



Some important methods of HttpServletRequest

Methods	Description
String <code>getContextPath()</code>	returns the portion of the request URI that indicates the context of the request
Cookies <code>getCookies()</code>	returns an array containing all of the Cookie objects the client sent with this request
String <code>getQueryString()</code>	returns the query string that is contained in the request URL after the path
HttpSession <code>getSession()</code>	returns the current HttpSession associated with this request or, if there is no current session and create is true, returns a new session

String <code>getMethod()</code>	Returns the name of the HTTP method with which this request was made, for example, GET, POST, or PUT.
Part <code>getPart(String name)</code>	gets the Part with the given name
String <code>getPathInfo()</code>	returns any extra path information associated with the URL the client sent when it made this request.
String <code>getServletPath()</code>	returns the part of this request's URL that calls the servlet

Example demonstrating Servlet Request

In this example, we will show how a parameter is passed to a Servlet in a request object from HTML page.

index.html

```
<form method="post" action="check">
Name <input type="text" name="user" >
<input type="submit" value="submit">
</form>
```

web.xml

```
<servlet>
    <servlet-name>check</servlet-name>
    <servlet-class>MyServlet</servlet-class>
</servlet>
<servlet-mapping>
    <servlet-name>check</servlet-name>
    <url-pattern>/check</url-pattern>
</servlet-mapping>
```

MyServlet.java

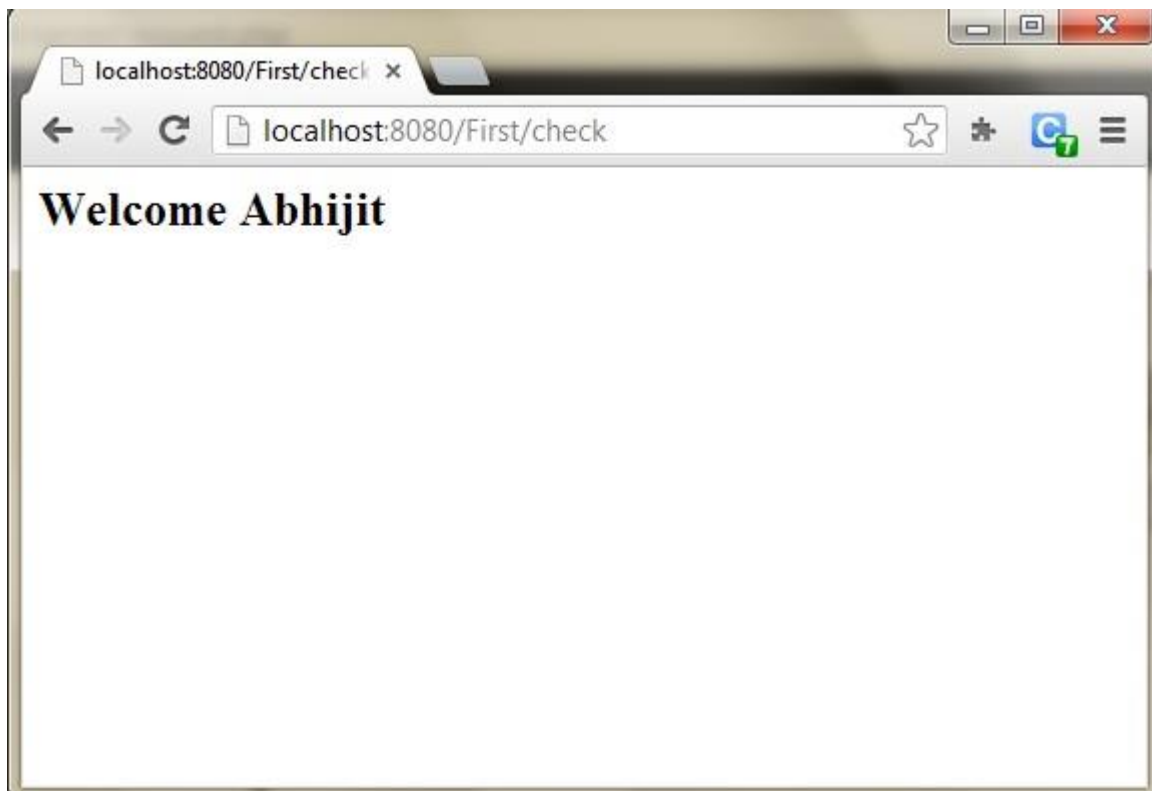
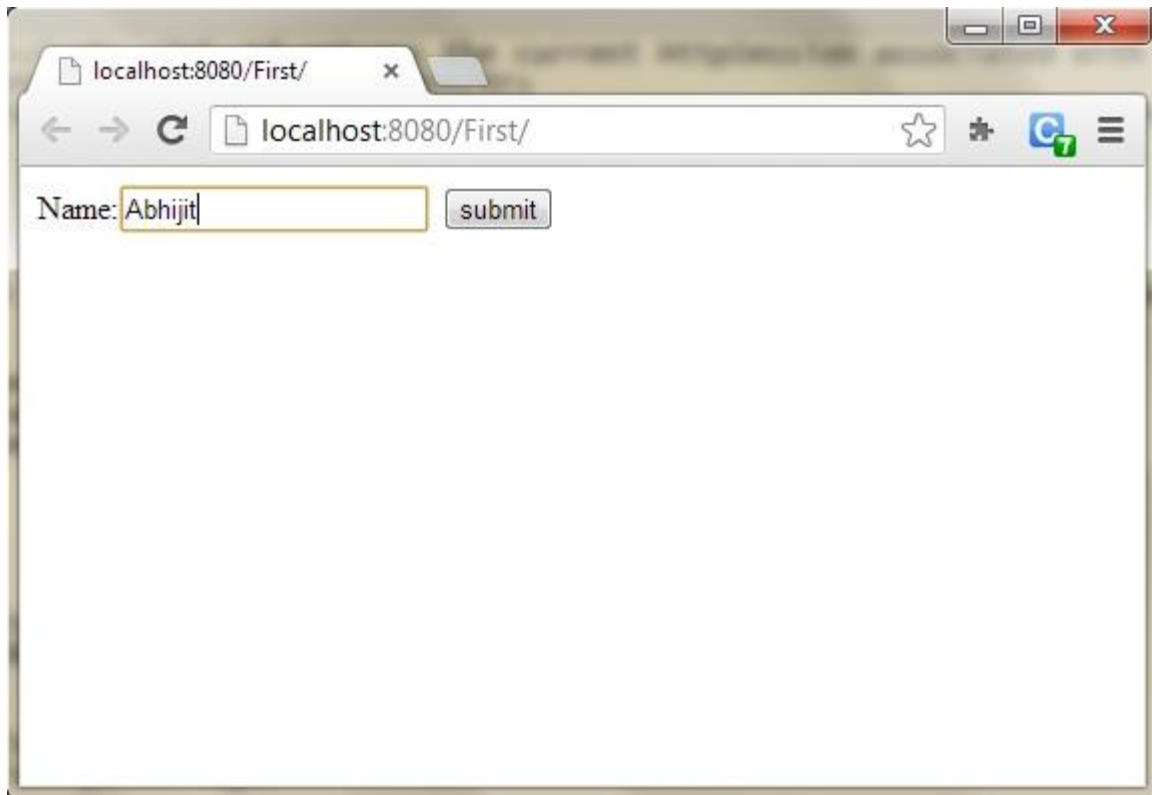
```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class MyServlet extends HttpServlet {

    protected void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        PrintWriter out = response.getWriter();
        try {

            String user=request.getParameter("user");
            out.println("<h2> Welcome "+user+"</h2>");
        } finally {
            out.close();
        }
    }
}
```

Output :



Introduction to Servlet Response

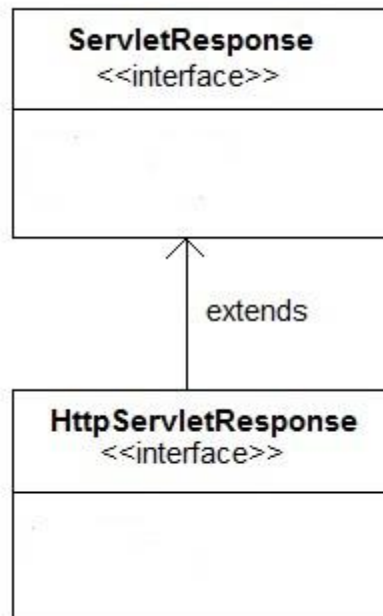
Servlet API provides two important interfaces **ServletResponse** and **HttpServletResponse** to assist in sending response to client.

Some Important Methods of ServletResponse

Methods	Description
PrintWriter <code>getWriter()</code>	returns a PrintWriter object that can send character text to the client.
void <code>setBufferSize(int size)</code>	Sets the preferred buffer size for the body of the response
void <code>setContentLength(int len)</code>	Sets the length of the content body in the response In HTTP servlets, this method sets the HTTP Content-Length header
void <code>setContentType(String type)</code>	sets the content type of the response being sent to the client before sending the respond.
void <code>setBufferSize(int size)</code>	sets the preferred buffer size for the body of the response.
boolean <code>isCommitted()</code>	returns a boolean indicating if the response has been committed
void <code>setLocale(Locale loc)</code>	sets the locale of the response, if the response has not been committed yet.

HttpServletResponse Interface

HttpServletResponse interface adds the methods that relates to the **HTTP** response.



Some Important Methods of HttpServletResponse

Methods	Description
void <code>addCookie(Cookie cookie)</code>	adds the specified cookie to the response.
void <code>sendRedirect(String location)</code>	Sends a temporary redirect response to the client using the specified redirect location URL and clears the buffer
int <code>getStatus()</code>	gets the current status code of this response
String <code>getHeader(String name)</code>	gets the value of the response header with the given name.

void <code>setHeader(String name, String value)</code>	sets a response header with the given name and value
void <code>setStatus(int sc)</code>	sets the status code for this response
void <code>sendError(int sc, String msg)</code>	sends an error response to the client using the specified status and clears the buffer

Introduction to Request Dispatcher

RequestDispatcher is an interface, implementation of which defines an object which can dispatch request to any resources(such as HTML, Image, JSP, Servlet) on the server.

Methods of RequestDispatcher

RequestDispatcher interface provides two important methods

Methods	Description
void <code>forward(ServletRequest request, ServletResponse response)</code>	forwards a request from a servlet to another resource (servlet, JSP file, or HTML file) on the server
void <code>include(ServletRequest request, ServletResponse response)</code>	includes the content of a resource (servlet, JSP page, HTML file) in the response

How to get an Object of RequestDispatcher

`getRequestDispatcher()` method of **ServletRequest** returns the object of **RequestDispatcher**.

```
RequestDispatcher rs = request.getRequestDispatcher("hello.html");  
rs.forward(request,response);
```

ServletRequest object **resource name**

```
RequestDispatcher rs = request.getRequestDispatcher("hello.html");  
  
rs.forward(request,response);
```

forward the request and response to "hello.html" page

OR

```
RequestDispatcher rs = request.getRequestDispatcher("hello.html");  
rs.include(request,response);
```

ServletRequest object **Resource name**

```
RequestDispatcher rs = request.getRequestDispatcher("first.html");  
  
rs.include(request,response);
```

include the response of "first.html" page in current servlet response

Example demonstrating usage of RequestDispatcher

In this example, we will show you how RequestDispatcher is used to **forward** or **include** response of a resource in a Servlet. Here we are using **index.html** to get username and password from the user, **ValidateServlet** will validate the password entered by the user, if the user has entered

"studytonight" as password, then he will be forwarded to **Welcome** Servlet else the user will stay on the index.html page and an error message will be displayed.

Files to be created :

- **index.html** will have form fields to get user information.
- **Validate.java** will validate the data entered by the user.
- **Welcome.java** will be the welcome page.
- **web.xml** , the deployment descriptor.

index.html

```
<form method="post" action="Validate">
Name:<input type="text" name="user" /><br/>
Password:<input type="password" name="pass" /><br/>
<input type="submit" value="submit">
</form>
```

Validate.java

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class Validate extends HttpServlet {

    protected void doPost(HttpServletRequest request, HttpServletResponse response
    )
        throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        PrintWriter out = response.getWriter();
        try {
            String name = request.getParameter("user");
            String password = request.getParameter("pass");
```

```

        if(password.equals("studytonight"))
        {
            RequestDispatcher rd = request.getRequestDispatcher("Welcome");
            rd.forward(request, response);
        }
        else
        {
            out.println("<font color='red'><b>You have entered incorrect password</b></font>");
            RequestDispatcher rd = request.getRequestDispatcher("index.html");
;
            rd.include(request, response);
        }
    }finally {
        out.close();
    }

}
}

```

Welcome.java

```

import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class Welcome extends HttpServlet {

    protected void doPost(HttpServletRequest request, HttpServletResponse response
    )

        throws ServletException, IOException {
        response.setContentType("text/html; charset=UTF-8");
    }
}

```

```
PrintWriter out = response.getWriter();
try {

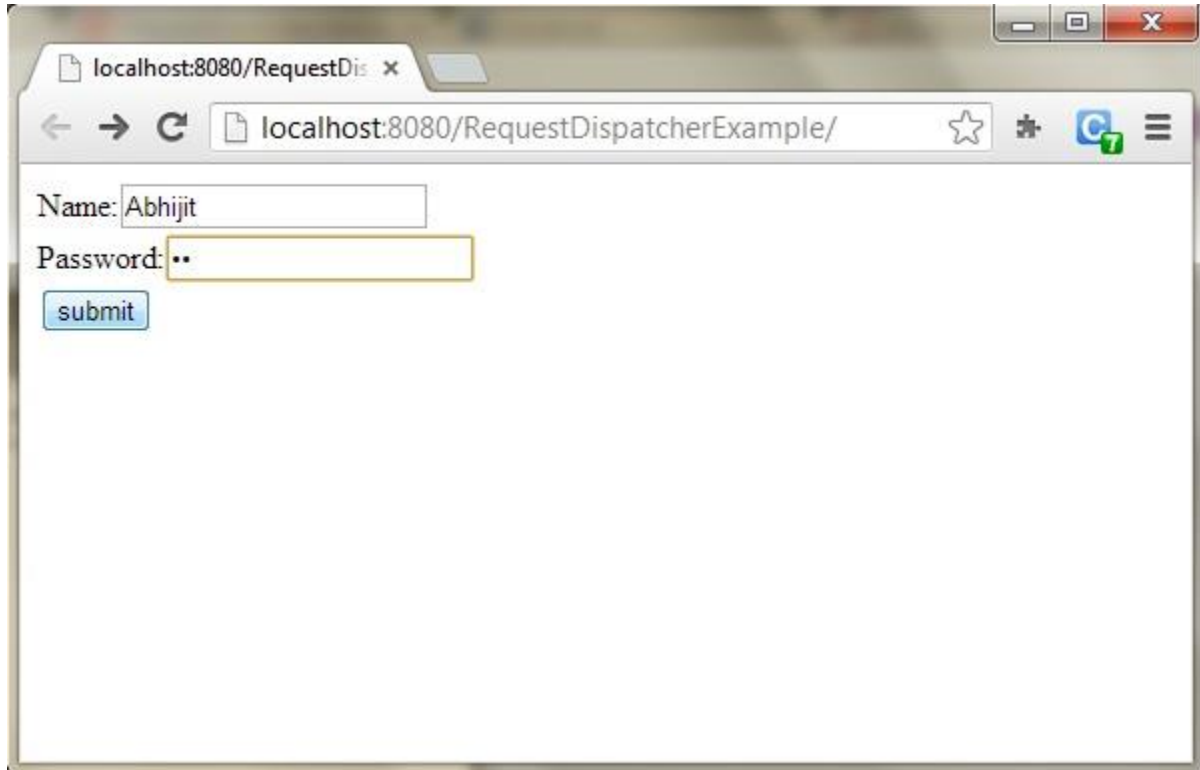
    out.println("<h2>Welcome user</h2>");
} finally {
    out.close();
}
}
```

web.xml

```
<web-app>
    <servlet>
        <servlet-name>Validate</servlet-name>
        <servlet-class>Validate</servlet-class>
    </servlet>
    <servlet>
        <servlet-name>Welcome</servlet-name>
        <servlet-class>Welcome</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>Validate</servlet-name>
        <url-pattern>/Validate</url-pattern>
    </servlet-mapping>
    <servlet-mapping>
        <servlet-name>Welcome</servlet-name>
        <url-pattern>/Welcome</url-pattern>
    </servlet-mapping>
    <welcome-file-list>
        <welcome-file>index.html</welcome-file>
    </welcome-file-list>
</web-app>
```

```
</welcome-file-list>  
</web-app>
```

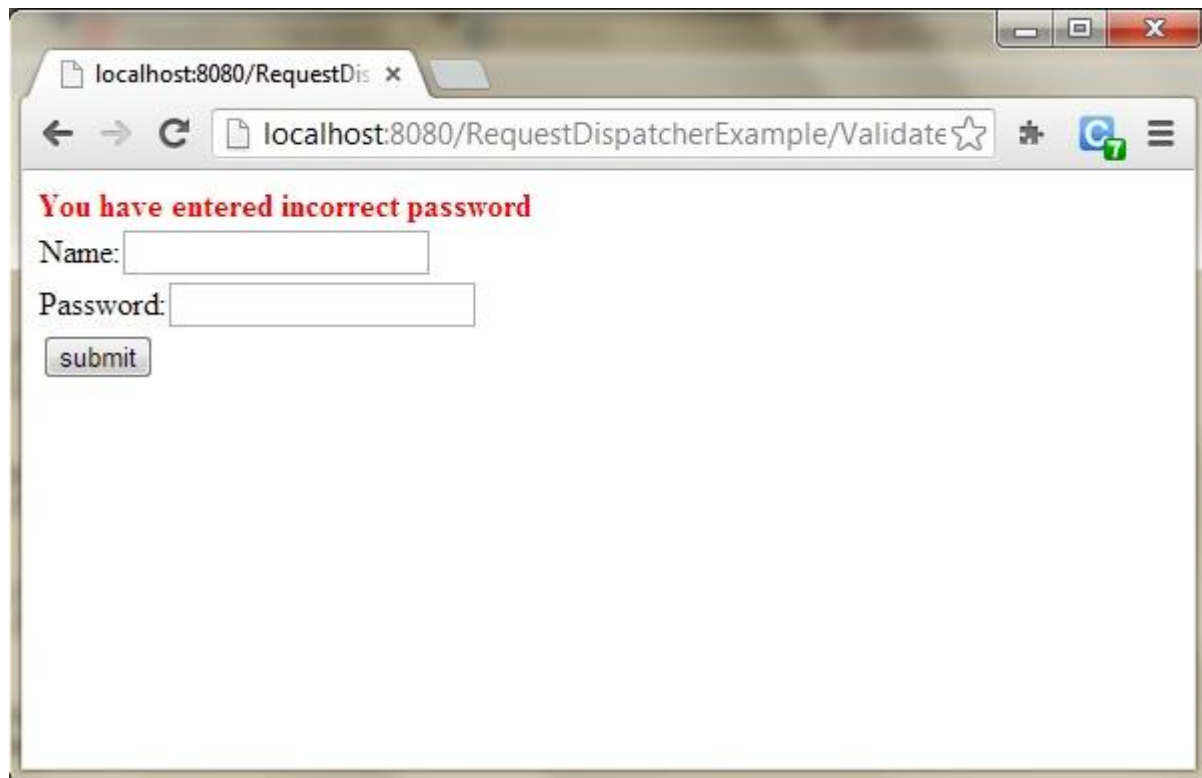
This will be the first screen. You can enter your Username and Password here.



The screenshot shows a web browser window with the address bar displaying 'localhost:8080/RequestDispatcherExample/'. The page contains a login form with the following elements:

- A text input field labeled 'Name:' containing the text 'Abhijit'.
- A text input field labeled 'Password:' containing two dots '..', indicating a password field.
- A blue button labeled 'submit'.

When you click on Submit, Password will be validated, if it is not 'studytonight' , error message will be displayed.



Enter any Username, but enter 'studytonight' as password.



Password will be successfully validated and you will be directed to the Welcome Servlet.



Introduction to sendRedirect() Method

`sendRedirect()` method redirects the response to another resource. This method actually makes the client(browser) to create a new request to get to the resource. The client can see the new url in the browser.

`sendRedirect()` accepts relative **URL**, so it can go for resources inside or outside the server.

sendRedirect() and Request Dispatcher

The main difference between a **redirection** and a **request dispatching** is that, redirection makes the client(browser) create a new request to get to the resource, the user can see the new URL while request dispatch get the resource in same request and URL does not changes.

Also, another very important difference is that, `sendRedirect()` works on **response** object while request dispatch work on **request** object.

Example demonstrating usage of sendRedirect()

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class MyServlet extends HttpServlet {

    protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        PrintWriter out = response.getWriter();
        try {
            response.sendRedirect("http://www.studytonight.com");
        }finally {
            out.close();
        }
    }
}
```

```
}  
}
```

Introduction to ServletConfig interface

When the **Web Container** initializes a servlet, it creates a **ServletConfig** object for the servlet. ServletConfig object is used to pass information to a servlet during initialization by getting configuration information from **web.xml** (Deployment Descriptor).

Methods of ServletConfig

- String `getInitParameter(String name):` returns a String value initialized parameter, or NULL if the parameter does not exist.
 - Enumeration `getInitParameterNames():` returns the names of the servlet's initialization parameters as an Enumeration of String objects, or an empty Enumeration if the servlet has no initialization parameters.
 - ServletContext `getServletContext():` returns a reference to the ServletContext
 - String `getServletName():` returns the name of the servlet instance
-

How to Initialize a Servlet inside web.xml

In the Deployment Descriptor(web.xml) file,

```
<servlet>
  <servlet-name>check</servlet-name>
  <servlet-class>MyServlet</servlet-class>
  <init-param>
    <param-name>email</param-name>
    <param-value>we@studytonight.com</param-value>
  </init-param>
</servlet>
```

Or, Inside the Servlet class, using following code,

```
ServletConfig sc = getServletConfig();
out.println(sc.getInitParameter("email"));
```

Example demonstrating usage of ServletConfig

web.xml

```
<web-app...>
  <servlet>
    <servlet-name>check</servlet-name>
    <servlet-class>MyServlet</servlet-class>
    <init-param>
      <param-name>email</param-name>
      <param-value>we@studytonight.com</param-value>
    </init-param>
  </servlet>
```

```
<servlet-mapping>
    <servlet-name>check</servlet-name>
    <url-pattern>/check</url-pattern>
</servlet-mapping>
<welcome-file-list>
    <welcome-file>index.html</welcome-file>
</welcome-file-list>
</web-app>
```

MyServlet class :

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class MyServlet extends HttpServlet {

    protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {

        response.setContentType("text/html;charset=UTF-8");
        PrintWriter out = response.getWriter();
        ServletConfig sc=getServletConfig();
        out.println(sc.getInitParameter("email"));
    }
}
```

Introduction to ServletContext Interface

For every **Web application** a **ServletContext** object is created by the web container. ServletContext object is used to get configuration information from **Deployment Descriptor**(web.xml) which will be available to any servlet or JSPs that are part of the web app.

Some Important method of ServletContext

Methods	Description
Object <code>getAttribute(String name)</code>	returns the container attribute with the given name, or NULL if there is no attribute by that name.
String <code>getInitParameter(String name)</code>	returns parameter value for the specified parameter name, or NULL if the parameter does not exist
Enumeration <code>getInitParameterNames()</code>	returns the names of the context's initialization parameters as an Enumeration of String objects
void <code>setAttribute(String name, Object obj)</code>	set an object with the given attribute name in the application scope
void <code>removeAttribute(String name)</code>	removes the attribute with the specified name from the application context

How Context Parameter is Initialized inside web.xml

```

<web-app ...>
<context-param>
    <param-name>driverName</param-name>
    <param-value>sun.jdbc.JdbcOdbcDriver</param-value>
</context-param>
<servlet>
    ....
</servlet>
</web-app>

```

The <context-param> is for whole application, so it is put inside the <web-app> tag but outside any <servlet> declaration

Parameter name

Parameter value

How to get the Object of ServletContext

```
ServletContext app = getServletContext();
```

OR

```
ServletContext app = getServletConfig().getServletContext();
```

Advantages of ServletContext

- Provides communication between servlets
- Available to all servlets and JSPs that are part of the web app
- Used to get configuration information from web.xml

Difference between Context Init Parameters and Servlet Init Parameter

Context Init parameters	Servlet Init parameter
-------------------------	------------------------

Available to all servlets and JSPs that are part of web	Available to only servlet for which the <code><init-param></code> was configured
Context Init parameters are initialized within the <code><web-app></code> not within a specific <code><servlet></code> elements	Initialized within the <code><servlet></code> for each specific servlet.
ServletContext object is used to get Context Init parameters	ServletConfig object is used to get Servlet Init parameters
Only one ServletContext object for entire web app	Each servlet has its own ServletConfig object

Example demonstrating usage of ServletContext

web.xml

```
<web-app ...>
```

```
    <context-param>
```

```
        <param-name>driverName</param-name>
```

```
        <param-value>sun.jdbc.JdbcOdbcDriver</param-value>
```

```
    </context-param>
```

```
    <servlet>
```

```
        <servlet-name>hello</servlet-name>
```

```
        <servlet-class>MyServlet</servlet-class>
```

```
    </servlet>
```

```
    <servlet-mapping>
```

```
        <servlet-name>hello</servlet-name>
```

```
        <url-pattern>/hello</url-pattern>
    </servlet-mapping>

</web-app>
```

MyServlet class :

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class MyServlet extends HttpServlet {

    protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        PrintWriter out = response.getWriter();
        ServletContext sc = getServletContext();
        out.println(sc.getInitParameter("driverName"));
    }
}
```

Introduction to Attribute

An **attribute** is an object that is used to share information in a web app. Attribute allows Servlets to share information among themselves. Attributes can be SET and GET from one of the following scopes :

1. request
2. session
3. application

Application Scope:

```
ServletContext sc=getServletContext();  
sc.setAttribute("user", "Abhijit");  
sc.getAttribute("user");  
sc.removeAttribute("user");
```

Diagram annotations for Application Scope:

- context object**: points to `ServletContext`
- attribute name**: points to `"user"` in `setAttribute`
- attribute value**: points to `"Abhijit"`
- getting an attribute**: points to `getAttribute`
- removing attribute**: points to `removeAttribute`

request Scope:

```
request.setAttribute("user", "Abhijit");  
request.getAttribute("user");  
request.removeAttribute("user");
```

Diagram annotations for request Scope:

- setting an attribute on request scope**: points to `setAttribute`
- getting an attribute**: points to `getAttribute`
- removing an attribute**: points to `removeAttribute`
- ServletRequest object**: points to the `request` object in all three methods

How to SET an Attribute

`public void setAttribute(String name, Object obj)` method is used to SET an Attribute.

Example demonstrating Setting Attribute

```
import java.io.*;  
import javax.servlet.*;  
import javax.servlet.http.*;  
  
public class First extends HttpServlet {
```

```

protected void doPost(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    response.setContentType("text/html;charset=UTF-8");
    PrintWriter out = response.getWriter();
    ServletContext sc = getServletContext();
    sc.setAttribute("user", "Abhijit");    //setting attribute on context sc
ope
    }
}

```

How to GET an Attribute

Object `getAttribute(String name)` method is used to GET an attribute.

Example demonstrating getting a value of set Attribute

```

import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class Second extends HttpServlet {

    protected void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        PrintWriter out = response.getWriter();
        ServletContext sc = getServletContext();

        //getting attribute from context scope
        String str = sc.getAttribute("user");
    }
}

```

```
        out.println("Welcome"+str); // Prints : Welcome Abhijit
    }
}
```