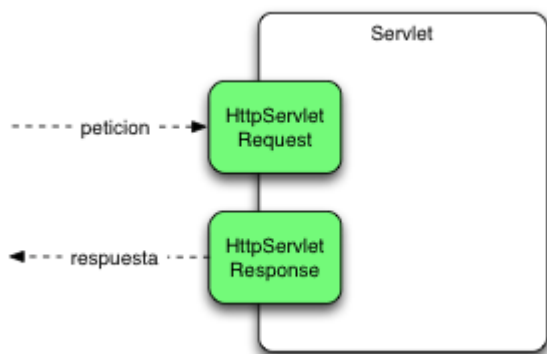


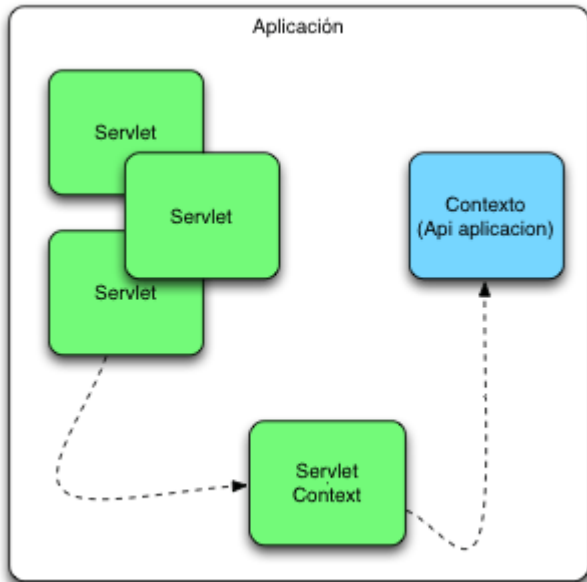
Muchas veces cuando uno comienza a programar en el entorno web de Java suelen quedarle muy claros los conceptos de `HttpServletRequest` (petición) y `HttpServletResponse` (respuesta) junto con el propio concepto de `Servlet`.



Sin embargo suelen aparecer muchas más dudas a la hora de gestionar otro de los conceptos fundamentales, el concepto de `ServletContext`. ¿Para que sirve este objeto? .

ServletContext

El `ServletContext` como su nombre indica permite acceder a un `Servlet` a la información de su Contexto o dicho de otra manera a la información asociada con la propia Aplicación y que es común a todos los `Servlets` que desplaguemos dentro de esa aplicación.



Así pues el fabricante del contenedor de Servlet (Tomcat por Ejemplo) suele generar algún objeto que aporta información a nivel del API sobre la propia aplicación web .Este objeto puede ser perfectamente una implementación propietaria del servidor . El objeto ServletContext que pertenece al Standard de JEE y por lo tanto todos los servidores lo incorporan accede a parte de la información de este otro objeto. Información que en muchos casos es critica para los desarrolladores. Vamos a ver un ejemplo en código en el cual usamos el objeto ServletContext para leer una variable global de la aplicación web y que esta declarada en el fichero web.xml

```
</pre>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://java.sun.com/xml/ns/javaee"
xmlns:web="http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd" version="3.0">
  <display-name>ServletContext</display-name>
```

```
<context-param>
  <param-name>directorioPDF</param-name>
  <param-value>pdfs</param-value>
</context-param>
</web-app>
<pre>
```

Como podemos ver hemos declarado una variable denominada “directorioPDF” que es común para toda la aplicación y queremos compartirla entre los distintos Servlets para ellos nos apoyaremos en el objeto ServletContext y su método getInitParameter().

```
@WebServlet("/ServletAplicacion")
public class ServletAplicacion extends HttpServlet {
    private static final long serialVersionUID = 1L;

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

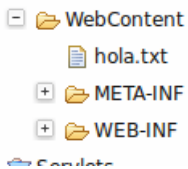
        PrintWriter pw= response.getWriter();
        ServletContext contexto=request.getServletContext();

        String
        directorio=contexto.getInitParameter("&quot;directorioPDF&quot;");
        pw.println(directorio);

    }
}
```

Igual que podemos acceder a una variable de contexto declarada a nivel de web.xml podemos por ejemplo acceder a la URL absoluta de la aplicación y leer un fichero que se

encuentre dentro de ella como el siguiente "hola.txt"



```
@WebServlet("/ServletAplicacion")
public class ServletAplicacion extends HttpServlet {

    private static final long serialVersionUID = 1L;
    protected void doGet(HttpServletRequest request, HttpServletResponse
    response) throws ServletException, IOException {

        PrintWriter pw= response.getWriter();
        File f= new File(contexto.getRealPath(""hola.txt""));

        BufferedReader lectorCadenas= new BufferedReader(new FileReader(f));
        String mensaje=lectorCadenas.readLine();
        lectorCadenas.close();
        pw.print(mensaje);
        pw.close();
    }

}
```

En este caso hemos utilizado el método del ServletContext que se denomina `getRealPath()` y nos devuelve la URL absoluta para nuestra URL relativa. De esta manera leeremos el fichero sin problemas.