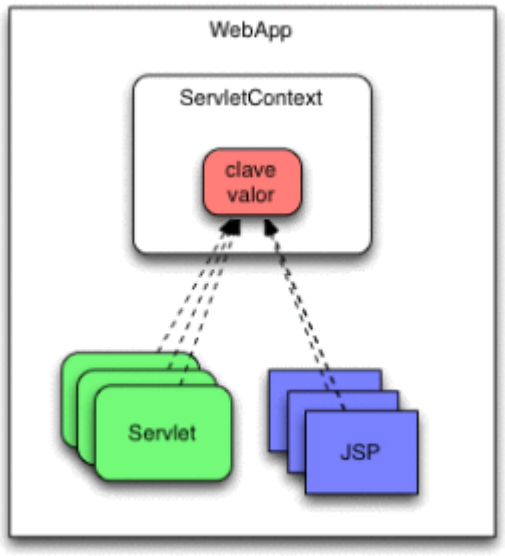
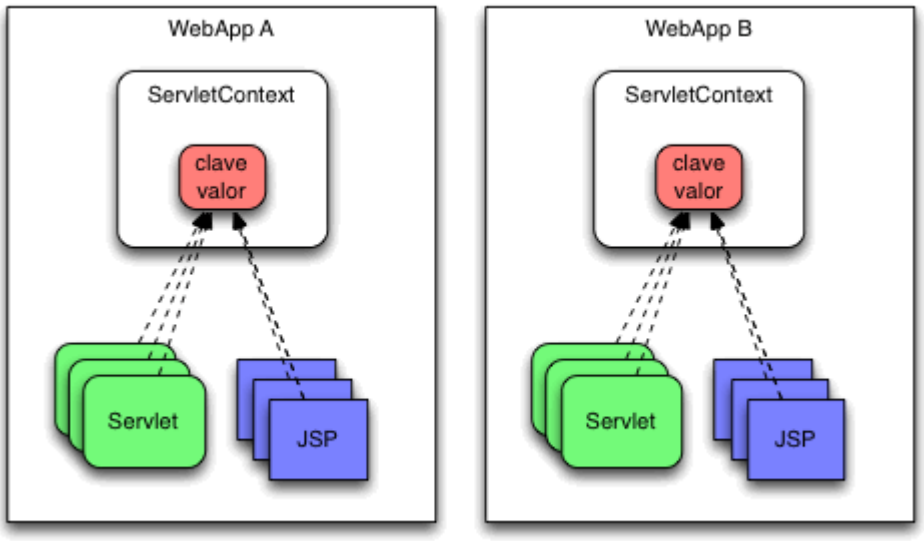
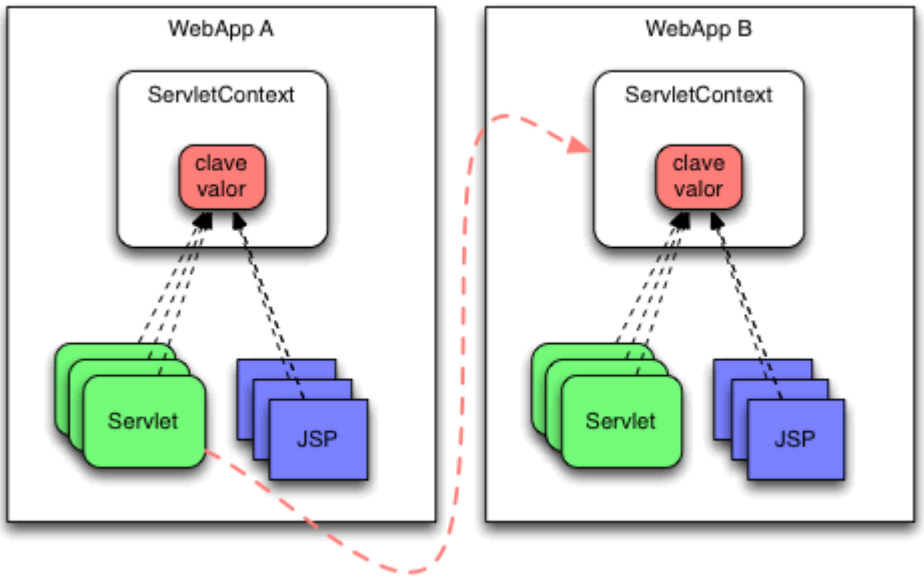
El ServletContext es uno de los objetos más utilizados de las aplicaciones web a la hora de compartir información entre los distintos componentes web como Servlets y JSP. Su funcionalidad esta orientada a almacenar claves/valores que sean comunes para toda la aplicación.



En principio cada aplicación web se encuentra aislada y es independiente de las otras.



Sin embargo muchas veces la gente me pregunta si hay alguna forma de comunicar ambas aplicaciones a nivel de API de Java.



Java ServletContext y comunicación

La respuesta es sí. La comunicación entre aplicaciones es posible a través de su ServletContext aunque no siempre es recomendable y depende siempre la configuración del Servidor de Aplicaciones. Vamos a ver un ejemplo partiendo de una aplicación web que contenga el siguiente fichero web.xml

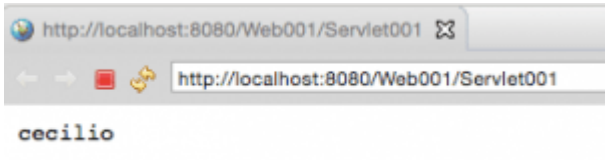
```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://java.sun.com/xml/ns/javaee"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd" version="3.0">
<display-name>Web001</display-name>

<context-param>
<param-name>nombre</param-name>
<param-value>cecilio</param-value>
</context-param>
</web-app>
```

Para acceder al nombre utilizaremos un Servlet que obtiene una referencia al ServletContext y nos los imprime por pantalla.

```
package com.arquitecturajava;

import java.io.IOException;
import java.io.PrintWriter;
```

```
import javax.servlet.ServletContext;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

@WebServlet("/Servlet001")
public class Servlet001 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();
pw.println(contexto.getInitParameter("nombre"));
pw.close();
}

 
```

El resultado es el siguiente:



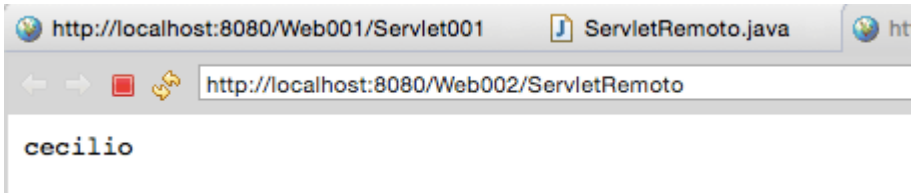
Como vemos estamos en la aplicación Web001 y accedemos al Servlet001. Ahora bien ¿cómo podemos hacer lo mismo desde otra aplicación?. El código a nivel de API es muy sencillo simplemente solicitaremos a la aplicación Web002 que acceda al contexto de

la Web001 vamos a verlo.

```
package com.arquitecturajava;

import java.io.IOException;
import java.io.PrintWriter;

import javax.servlet.ServletContext;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

/**
 * Servlet implementation class ServletRemoto
 */
@WebServlet("/ServletRemoto")
public class ServletRemoto 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();
 ServletContext otroContexto=contexto.getContext("/Web001");
 otroContexto.getInitParameter("nombre");
 pw.println(otroContexto.getInitParameter("nombre"));
```

```
 pw.close();
 }

}
```

El resultado será el siguiente que como vemos es otra aplicación:



Para que nos funcione correctamente deberemos configurar alguna cosa especifica de nuestro servidor que permita llamadas entre contextos ya que de entrada pueden estar aislados. En el caso del tomcat vale ir al server.xml y añadir que las aplicaciones soportan crossContext=”true”.

```
<Context crossContext="true" docBase="Web001" path="/Web001"
reloadable="true" source="org.eclipse.jst.jee.server:Web001"/>
 <Context crossContext="true" docBase="Web002" path="/Web002"
reloadable="true" source="org.eclipse.jst.jee.server:Web002"/>
```

Hay que tener claro que estas casuísticas deberían reducirse al mínimo para reducir el acomplamiento, pero a veces nos pueden ser útiles.

Otros artículos relacionados: ServletContext ,ServletContextListener , Servlet 3.0