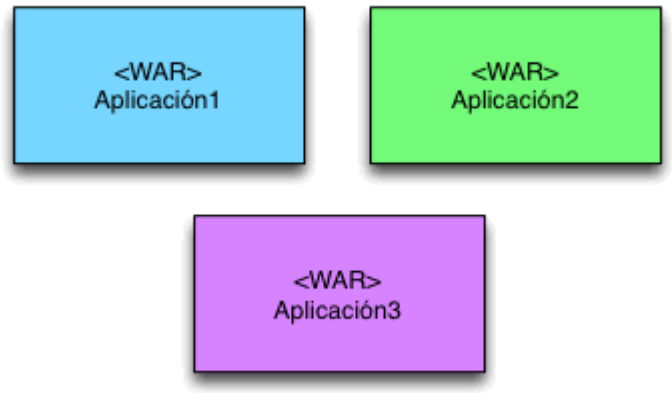
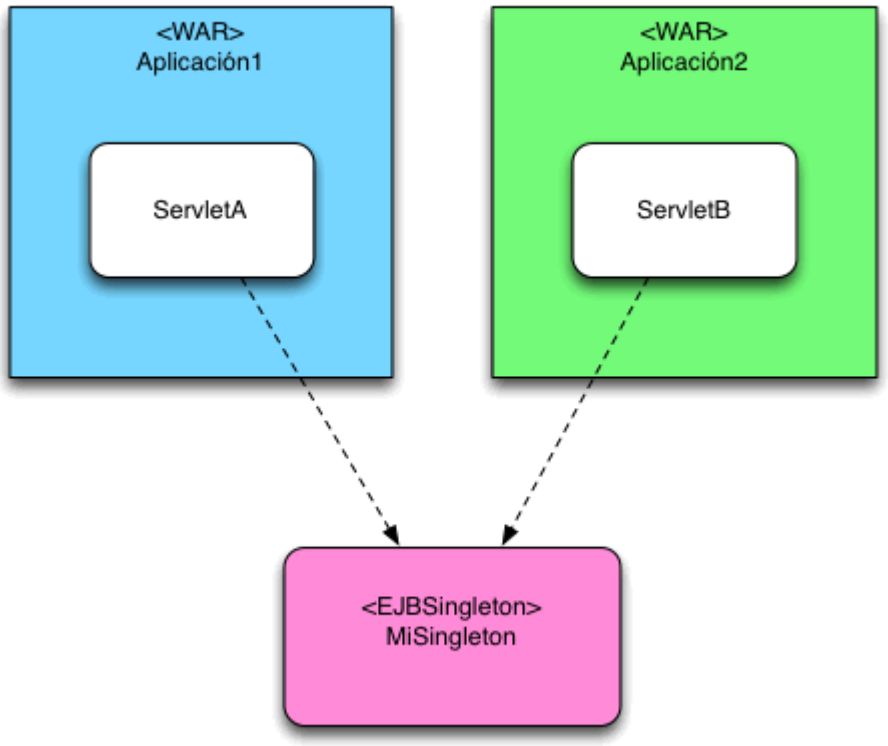
En muchas ocasiones desarrollamos aplicaciones que pueden empaquetarse en un WAR y es suficiente. En otras ocasiones tenemos que diseñar una aplicación que contiene varios de estos módulos. Es aquí donde nos encontramos con mayores problemáticas y donde los EJB Singleton nos pueden ser de ayuda.



EJB Singleton

¿Para que sirve un EJB Singleton? . Bueno se trata de un EJB disponible a partir de Java EE 6 que tiene la particularidad de que solo existe una instancia de él para toda nuestra aplicación Enterprise. Puede servir para acceder a datos que estén compartidos entre varias aplicaciones y se almacenen en el EJB como si se tratará de una pequeña cache. Por ejemplo podríamos disponer de un conjunto de rutas a carpetas o algo similar. En este artículo vamos a crear el EJB Singleton de “Hola Mundo” que es el típico contador que esta compartido entre varias aplicaciones. En este caso nos apoyaremos en dos Servlets cada uno almacenado en un fichero WAR que irán incrementando el contador.



Vamos a ver el código del EJB tanto su interface local como de su implementación:

```
@Local
public interface MiSingletonLocal {

public void incrementar();
public int getNumero();
}
```

```
package com.arquitecturajava;
```

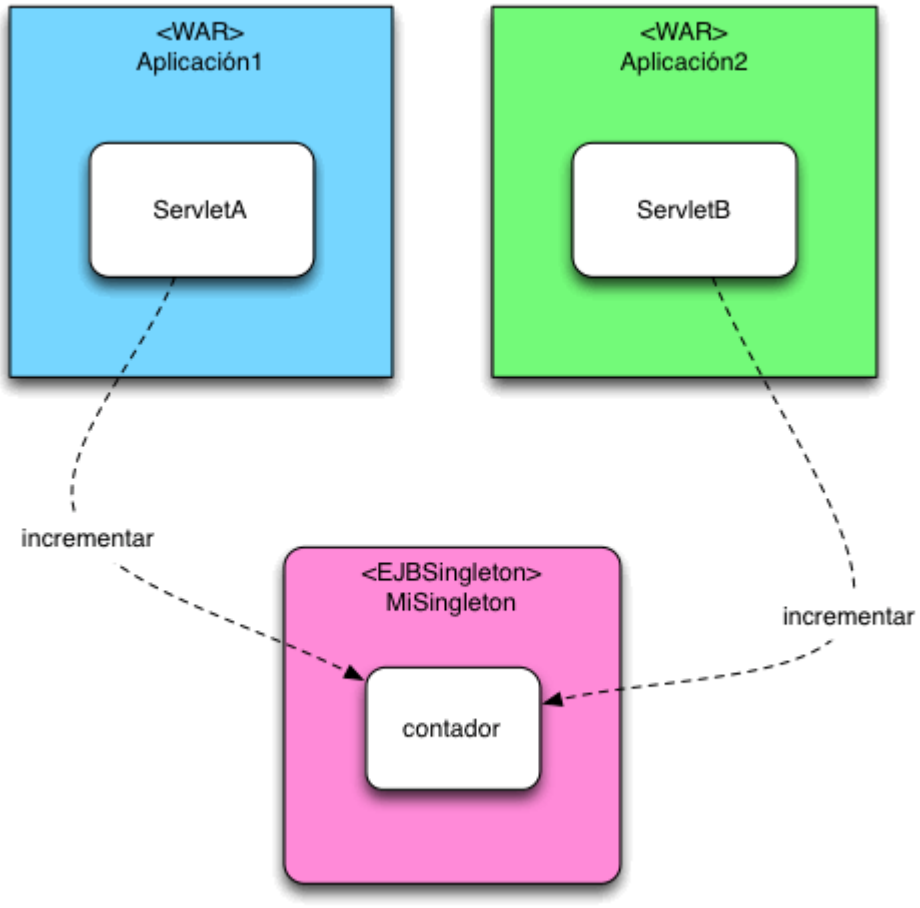
```
import javax.ejb.Singleton;

@Singleton
public class MiSingleton implements MiSingletonLocal {

private int numero;

public MiSingleton() {

}

@Override
public void incrementar() {
numero++;

}

@Override
public int getNumero() {
// TODO Auto-generated method stub
return numero;
}

}
```

Una vez construido el EJB acceder a él será tan sencillo como realizar una inyección de dependencia en alguno de los Servlets que se encuentren en cada uno de los WARs. Vamos a verlo:

```
package com.arquitecturajava.singleton;

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

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

import com.arquitecturajava.MiSingletonLocal;

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

@EJB
MiSingletonLocal misingleton;

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

misingleton.incrementar();
PrintWriter pw= response.getWriter();
pw.println(misingleton.getNumero());
}
protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
```

```
}

}
```

De esta forma cada uno de los Servlets podrá incrementar el valor del contador del EJB que compartirá estado entre las distintas aplicaciones Web.



Otros artículos relacionados :

Introducción a EJB

EJB Remotos

Cliente de EJB