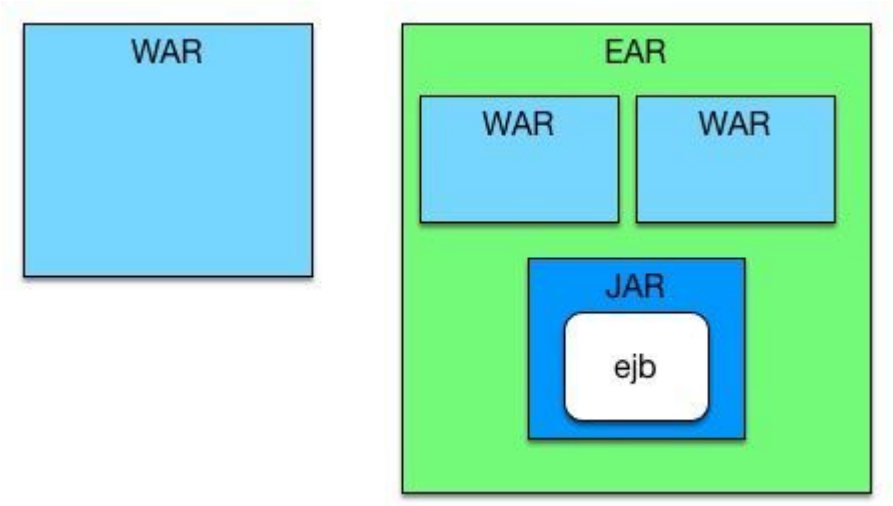
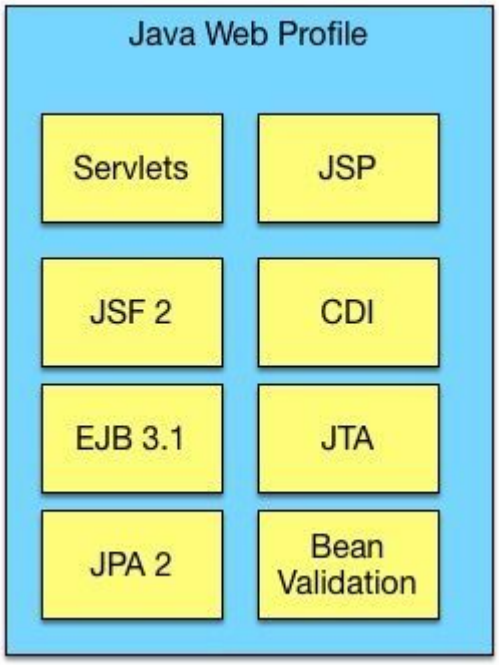
La de idea de EJB in WAR suena un poco rara al principio . Todos nosotros estamos acostumbrados a trabajar con aplicaciones Java web o aplicaciones Java Enterprise. Son en estas últimas en donde se ubican los EJB o Enterprise Java Beans. Así pues si tuvieramos que echar un vistazo a la estructura de un WAR y un EAR veríamos de forma simplificada algo así:

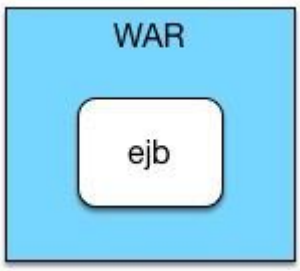


Java EE 6 EJB in WAR

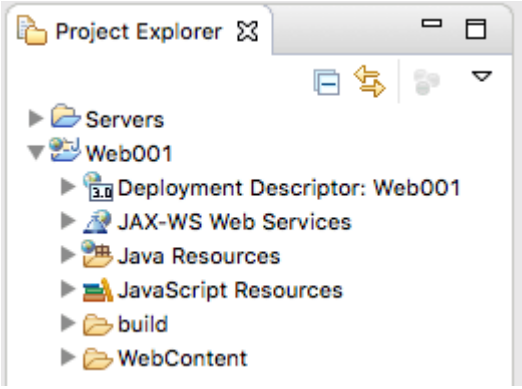
Lo que mucha gente no sabe es que a partir de Java EE 6 , el web profile es muy completo :



Soportando el despliegue de EJBs dentro de WARs .



Esto nos facilitará el futuro camino hacia los microservicios ya que el concepto de EAR que contiene varios WARs y JARs es bastante lejano a dicha filosofía y a la modularidad. Vamos a construir un ejemplo de esta casuística . Para ello necesitamos construir un Dynamic Web Project en Eclipse:



El siguiente paso es construir un EJB dentro del proyecto. En un principio nos parecerá algo extraño pero el entorno lo soporta sin problema:

```
package com.arquitecturajava.ejb;

import javax.ejb.Local;

@Local
public interface MensajeEJBLocal {

public String mensaje() ;
}
```

```
package com.arquitecturajava.ejb;

import javax.ejb.LocalBean;
import javax.ejb.Stateless;
@Stateless
@LocalBean
```

```
public class MensajeEJB implements MensajeEJBLocal {

@Override
public String mensaje() {

return "hola desde un EJB";
}

}
```

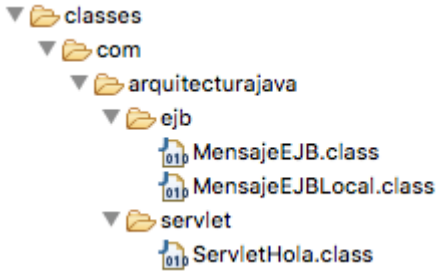
El siguiente paso es construir un Servlet que delegue en el EJB que acabamos de construir y nos imprima su mensaje.

```
package com.arquitecturajava.servlet;

import java.io.IOException;

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.ejb.MensajeEJBLocal;

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

```
private static final long serialVersionUID = 1L;

@EJB
MensajeEJBLocal mimensaje;

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

response.getWriter().append(mimensaje.mensaje());
}

}
```

Invocamos el Servlet y nos mostrará el mensaje del EJB por pantalla.

Si revisamos el código nos daremos cuenta que el EJB se ha ubicado como una clase normal de Java en la carpeta classes.



Acabamos de desplegar nuestro primer EJB dentro de un WAR .En muchos casos usaremos la filosofía de EJB in WAR para desplegar servicios REST y enfocar hacia los microservicios.

Otros artículos relacionados:

¿ Que es REST ?

Introducción a EJB 3.1 (I)

El porqué de los MicroServicios