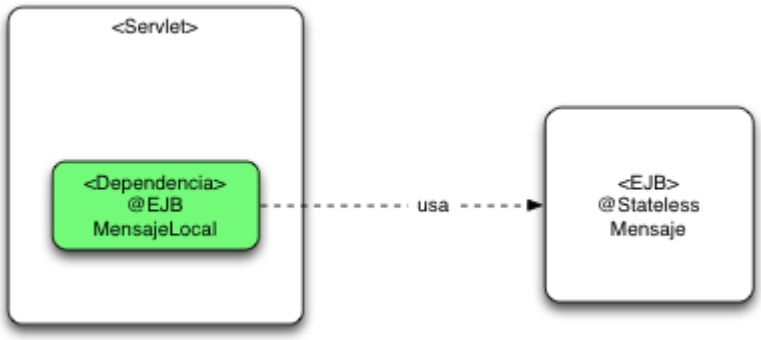
La inyección de dependencia es uno de los conceptos que ya hemos abordado a nivel de EJBs en donde habitualmente un Servlet accede a un EJB vía inyección de dependencia usando la anotación @EJB



Vamos a ver el código del EJB Interface MensajeLocal

```
package com.arquitecturajava;

import javax.ejb.Local;

@Local
public interface MensajeLocal {
 public String hola();
}
```

EJB Implementación

```
package com.arquitecturajava;
import javax.ejb.Stateless;

@Stateless
```

```
public class Mensaje implements MensajeLocal {

@Override
 public String hola() {
 return "accediendo al EJB de hola";
 }

}
```

Servlet

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

import javax.ejb.EJB;
import javax.inject.Inject;
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("/ServletCDI")
public class ServletCDI extends HttpServlet {
 private static final long serialVersionUID = 1L;

 @EJB
 MensajeLocal mensaje;

 protected void doGet(HttpServletRequest request, HttpServletResponse
```

```
response) throws ServletException, IOException {

 PrintWriter pw= response.getWriter();
 pw.println(mensaje.hola());

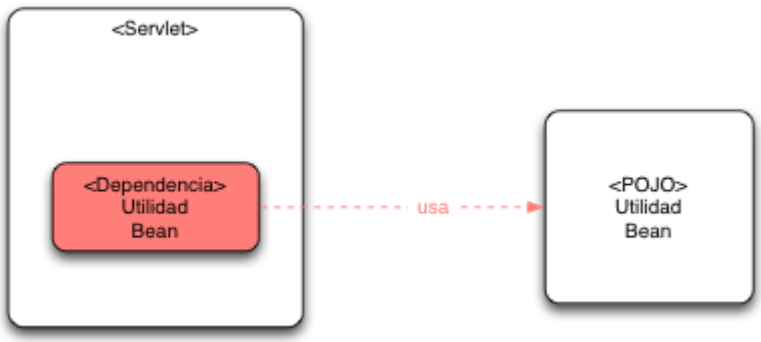
 pw.close();
 }
}
```

Java EE 5 Limitaciones

Este código funciona perfectamente a nivel de Java EE 5 .Sin embargo tiene una serie de limitaciones entre ellas que no podemos inyectar una clase como la siguiente .

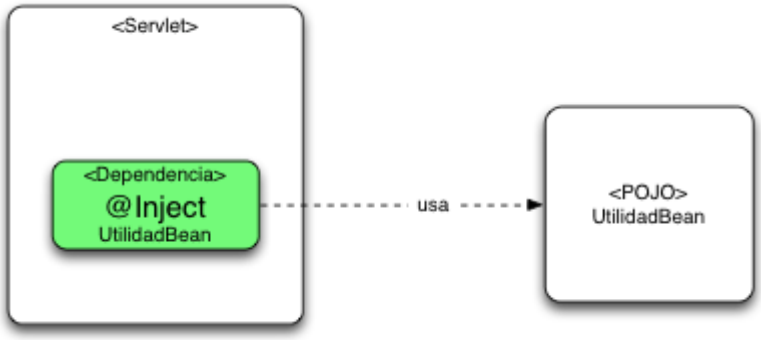
```
package com.arquitecturajava;
public class UtilidadBean {
 public String hola() {
 return "Hola desde la utilidad";
 }
}
```

Java EE 5 no permite la inyección de dependencias a nivel de POJOs, solo a nivel de EJBs.



Java EE 6 (Java y CDI)

A partir de la versión 6 de Java EE podemos hacer uso de CDI (Context Dependency Inyection) que es la JSR que nos permite definir inyección de dependencias tanto para EJBs como para POJOs .De esta forma podremos construir el Servlet que utilizando la anotacion @Inject sea capaz de inyectar una clase de tipo POJO.




```
package com.arquitecturajava;

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

import javax.ejb.EJB;
import javax.inject.Inject;
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("/ServletCDI")
public class ServletCDI extends HttpServlet {
```

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

 @EJB
 MensajeLocal mensaje;
 @Inject
 UtilidadBean mensajeUtilidad;

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

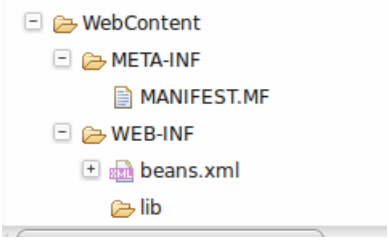
 PrintWriter pw= response.getWriter();
 pw.println(mensaje.hola());
 pw.println(mensajeUtilidad.hola());
 pw.close();
 }
}
```

Una vez inyectadas las dependencias si invocamos la URL del servlet el resultado será el siguiente.



Beans.xml

Para que todo esto funcione correctamente deberemos añadir el fichero de beans.xml (JSR 330) a la carpeta WEB-INF. Sino la aplicación no será capaz de inyectar correctamente las



dependencias.

El contenido de este fichero esta vacío para un ejemplo tan sencillo pero el fichero es obligatorio .

```
<?xml version="1.0"?>
<beans xmlns="http://java.sun.com/xml/ns/javaee"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://jboss.org/schema/cdi/beans_1_0.xsd"/>
```

EJBs y @Inject

Hemos usado @Inject para inyectar POJOS pero lo podríamos utilizar de igual manera para inyectar EJBs

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

import javax.inject.Inject;
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("/ServletCDI")
```

```
public class ServletCDI extends HttpServlet {
 private static final long serialVersionUID = 1L;

 @Inject
 MensajeLocal mensaje;
 @Inject
 UtilidadBean mensajeUtilidad;

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

 PrintWriter pw= response.getWriter();
 pw.println(mensaje.hola());
 pw.println(mensajeUtilidad.hola());
 pw.close();
 }
}
```

De esta forma JEE 6 nos simplifica el uso de anotaciones