Introducción

CDI Context Dependency Injection

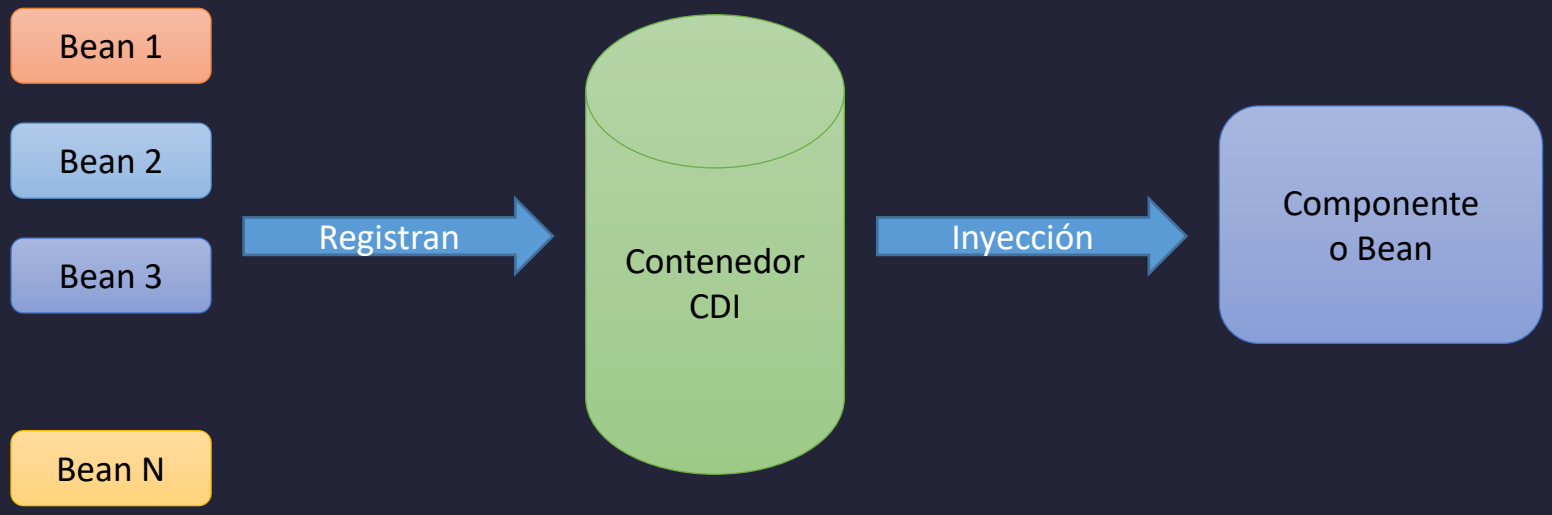
JAKARTA™ EE

Andrés Guzmán F

JAKARTA EE

CDI

Es una especificación estándar y framework para la inyección de dependencia y manejo de contextos incluido desde Java EE 6 en adelante.



JAKARTA EE

Inyección de dependencia

Es un patrón de diseño que forma parte de la programación orientada a objetos en la plataforma Java EE, es parte de la especificación JSR-330.

Especifica que se inyectará un componente o variable de contexto en un atributo de otro componente CDI, es decir para inyectar componentes de la aplicación en el componente actual

'Inyectar es justamente suministrar a un objeto una referencia de otros que necesite según la relación, tiene que plasmarse mediante la anotación @Inject'

JAKARTA EE

Características CDI

Promueve la composición y modularidad entre las partes que componen una aplicación

Mantienen su código limpio, simple y modular, bajo acoplamiento y alta cohesión

Las aplicaciones con Java EE con Weld son simples y requieren mucho menos código (Java y XML) para la misma funcionalidad

Se elimina código repetitivo y configuraciones XML y mayor uso de anotaciones

JAKARTA EE

Registrar e inyectar

Registrar o publicar un bean:

- Se crea de forma automática, no hay que hacer nada especial para publicar un bean en el contexto de CDI:

```
public interface Repositorio { }

public class RepositorioImpl implements Repositorio { }
```

Inyectar un bean existente en otro bean:

```
public class ServiceImpl implements Service {

   @Inject
   private Repositorio repositorio;
}
```

JAKARTA EE

Manejo de contextos

Podríamos no definer ningún contexto explícitamente y queda como @Dependent

```
public interface Repositorio { }

public class RepositorioImpl implements Repositorio { }
```

Pero también podríamos definir un contexto explícitamente

```
@ApplicationScoped
public class ServiceImpl implements Service {

   @Inject
   private Repositorio repositorio;
}
```

JAKARTA EE

Contextos CDI

- @Dependent
- @RequestScoped
- @SessionScoped
- @ConversationScoped
- @ApplicationScoped

JAKARTA EE

Anotación @Named

Hasta ahora, hemos visto cómo se registran los beans y a definir puntos de inyección con la anotación @Inject.
CDI también nos permite dar nombres a los beans y realizar la inyección mediante el nombre con la anotación @Named.

```
public interface Repositorio { }

@Named("jdbcRepositorio")
public class RepositorioImpl implements Repositorio { }
```

Luego en el Service lo inyectamos vía nombre del beans

```
public class ServiceImpl implements Service {

    @Inject
    @Named("jdbcRepositorio")
    private Repositorio repositorio;
}
```

JAKARTA EE

Anotación @Produces

Otra forma para registrar un bean mediante método, el objeto que devuelve este método (anotado con @Produces) queda registrado en el contenedor CDI.

```
@Produces
public Conexion produceConexion() {
   return new Conexion();
}
```

Opcionalmente también puede tener un nombre y contexto

```
@Produces
@RequestScoped
@Named("conn")
public Conexion produceConexion() {
   return new Conexion();
}
```

JAKARTA EE

Integración con EL (Lenguaje de Expresión)

También se integra con la librería EL de JSP donde nos permite acceder a métodos y atributos de los beans o componentes CDI mediante el nombre definido con la anotación @Named, es decir son asignaciones (o mapping) hacia estos objetos.

```
@SessionScoped
@Named
public class Carro implements Serializable {
    ...
}
```

Accedemos al carro en las vistas JSP mediante EL:

```
<c:forEach items="${carro.items}" var="item">
...
</c:forEach>

Total: ${carro.total}
```