Consulta EJB

Mauricio David Molineros Molineros

Instituto Tecnológico Quito

Febrero 26 de 2021

Nota del autor

Mauricio David Molineros Landines, Análisis de Sistemas, Instituto Tecnólogico Quito

La correspondencia relacionada con esta investigación debe ser dirigida a nombre de

Mauricio David Molineros Landines, Instituto Tecnológico Quito, Av. 10 de Agosto Y Colon

Quito – Ecuador

Contacto: mauro8668@gmail.com

Contenido

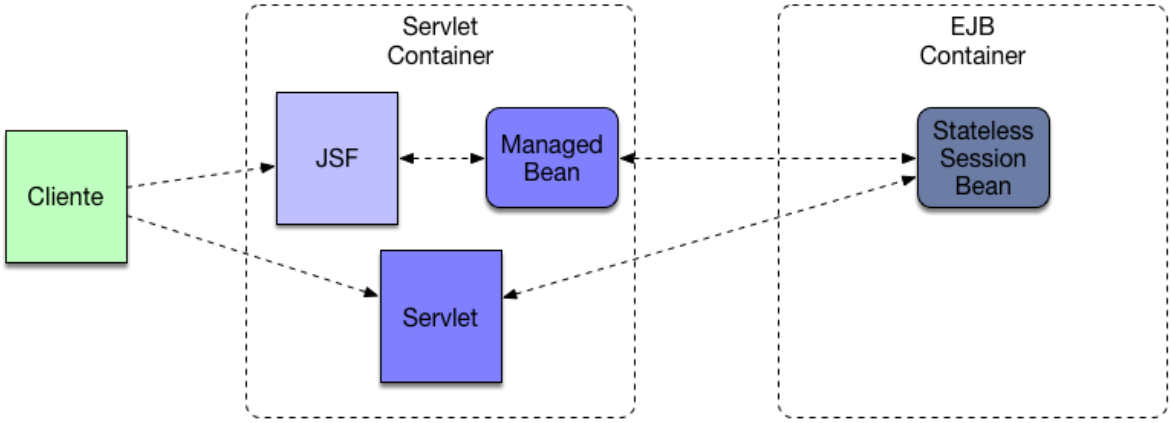
REPOSITORIO GIT

El presente trabajo se encuentra el siguiente repositorio GitHub.

Link: https://github.com/mmolineros/DEBRES.git

Que es EJB

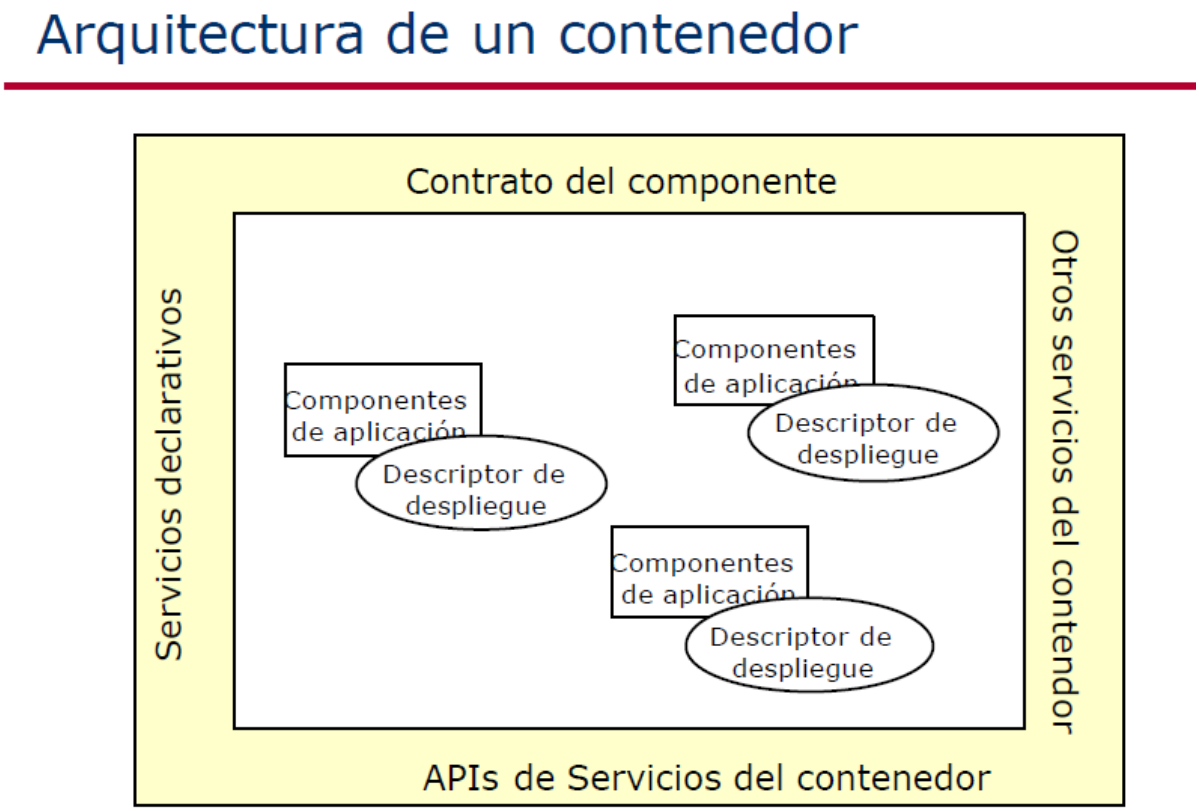
Los EJB son componentes fundamentales en el desarrollo de aplicaciones Java Enterprise Edition. Cabe recalcar, que muchas veces surgen dudas sobre cómo funciona un EJB a detalle.



El funcionamiento de los componentes EJB se basa fundamentalmente en el trabajo del contenedor EJB. El contenedor EJB es un programa Java que corre en el servidor y que contiene todas las clases y objetos necesarios para el correcto funcionamiento de los Enterprise Beans.

Que es un Conector de EJB

Contenedor que implementa el contrato del componente EJB de la arquitectura J2EE. Este contrato especifica un entorno de tiempo de ejecución de un Enterprise Bean que incluye servicios de seguridad, concurrencia, administración del ciclo de vida, transacciones, implementación, asignación de nombres y otros servicios.



Tipos de Beans, explique cada uno

Beans de Sesión: representa un componente que encapsula un conjunto de métodos o acciones de negocio que pueden ser llamados de forma síncrona. Ejemplos de EJBs de sesión podrían ser un sistema de peticiones de reserva de libros de una biblioteca o un carrito de la compra. En general, cualquier componente que ofrezca un conjunto de servicios (métodos) a los que se necesite acceder de forma distribuida, segura y transaccional.

Beans dirigidos por mensaje: Se diferencian de los anteriores en su carácter asíncrono. Los clientes de estos EJBs nunca los llaman directamente, sino que envían mensajes JMS para comunicarse con ellos. Un ejemplo podría ser un EJB ListenerNuevoCliente que se activara cada vez que se envía un mensaje comunicando que se ha dado de alta a un nuevo cliente.

Vemtajas de los EJB

Un EJB a través de un "EJB Container" ofrece varios servicios y funcionalidades no disponibles en un "Java Bean", algunas son las siguientes: Servicios ("Middleware")

Esta posiblemente sea la mayor ventaja de un EJB. Cuando se diseña un componente de Software se deben definir varios servicios para su funcionamiento, algunos pueden ser:

- Si ocurre un error que procedimiento debe ejecutarse.
- Si la base de datos especificada se encuentra desactivada.
- No fue posible cumplir exitosamente "x" procedimiento, se deben retractar sus acciones parciales o reinvocar la transacción.

Estos Servicios (Middleware) por lo general son requeridos además de la lógica contenida en los componentes principales, obviamente estos servicios aún deben ser diseñados, sin embargo, mediante un "EJB Container" se ofrecen estos servicios y es a través de un "Enterprise Java Bean" que es posible desarrollar los componentes principales.

División de Trabajo

La posibilidad de dividir servicios EJB Container de Componentes Principales EJB permite una clara división de trabajo, esto es, un diseñador de componentes EJB que puede concentrar sus esfuerzos en la lógica de proceso sin preocuparse del diseño de servicios. Y de la misma manera un diseñador de servicios concentrarse en su área.

Diversos Vendedores

El uso de especificaciones para EJB permite que existan diversos vendedores tanto de EJB Containers los cuales son incluidos en un java application server , así como Enterprise Java Bean, los cuales resuelven algún tipo de lógica.

Lo anterior permite ejecutar cualquier EJB en cualquier EJB Container, esto es, que puede adquirir un conjunto de EJB producidos por Inprise o inclusive desarrollarlos dentro de su empresa y estos podrán ser ejecutados en un EJB Container de IBM,Inprise o JBoss.

Procedimientos Remotos (RMI)

Debido a la solución que intentan ofrecer EJB su diseño gira alrededor de procedimientos remotos.

Diversos Clientes

Un EJB puede interactuar con una gran gamma de clientes desde: JSP o Servlets , bases de datos , Applets , sistemas ERP (SAP,JDEdward's).

Que es el JPQL

El Java Persistence Query Language (JPQL) es el lenguaje estándar de consultas de JPA. Es un lenguaje diseñado para combinar la simplicidad de la semántica y sintaxis del lenguaje SQL con la expresividad de un lenguaje orientado a objetos.

Tipos de Sentencias

- SENTENCIAS SELECT

La cláusula SELECT no enumera los campos o utiliza un "*" para seleccionar todos los campos. En su lugar, utiliza la variable "e" que indica que el tipo de resultado es una entidad Employe e

La cláusula SELECT utiliza una expresión de ruta para entidad Deparment

```
SELECT e.name, e.salary
FROM Employee e
```

```
SELECT e.department
FROM Employee e
```

- SENTENCIAS UPDATE

Para modificar las instancias de objetos que se encuentren dentro de la tabla alumnos se usa la cláusula UPDATE. También es muy similar a la que se utiliza en el lenguaje SQL.

```
UPDATE Alumnos  a
SET a.nota_matematicas = 5
WHERE a.cod_alumno = 1001
```

En este caso modificamos la nota de matemáticas del alumno cuyo código es el 1001. Vemos que se puede utilizar la cláusula WHERE para acotar las filas modificadas, de lo contrario se cambiarían todos los objetos de la tabla.

- SENTENCIAS DELETE

EL comando para borrar filas es DELETE. También se utiliza con la cláusula WHERE para acotar las filas que se deben borrar.

```
DELETE FROM Alumnos a
WHERE a.alumno = 1001
```

Unión de Objetos con JPQL

JOIN

Un Join es una consulta que combina resultados de distintas entidades. Cuando la consulta es traducida a SQL es común que las join entre entidades produzcan join similares entre las tablas

Inner Join

Un inner join devuelve todas las entidades del lado izquierdo que tienen una relación con la entidad del lado derecho de la relación

Se pueden definir de forma explícita utilizando el operador JOIN en la cláusula FROM

```
SELECT p
FROM Employee e JOIN e.phones p
```

Su traducción a SQL será:

```
SELECT p.id, p.phone_num, p.type, p.emp_id
FROM emp e, phone p
WHERE e.id = p.emp_id
```

Otra manera es mediante una variable de identificación en la cláusula FROM y condiciones de unión sobre ella en la cláusula WHERE. Este tipo de consulta se utiliza cuando no hay una relación explícita entre dos entidades en el modelo de dominio

```
SELECT DISTINCT d
FROM Department d, Employee e
WHERE d = e.department
```

Outer Join

Un outer join devuelve todas las entidades del lado izquierdo de la relación y la entidad de lado derecho con la que se relaciona

```
SELECT e, d
FROM Employee e LEFT JOIN e.department d
```

Su traducción a SQL será:

```
SELECT e.id, e.name, e.salary, e.manager_id, e.dept_id, e.address_id,
d.id, d.name
FROM employee e LEFT OUTER JOIN department d
ON (d.id = e.department_id)
```

En el ejemplo anterior sino existe departamento el segundo objeto del Objeto devuelto será null

Fetch Join

Permite seleccionar una o varias relaciones que deben ser traídas de forma “ansiosa”, es decir, modificar el comportamiento por defecto que lo que hace es cargar las entidades relacionadas de forma “perezosa”

```
SELECT e

FROM Employee e JOIN FETCH e.address
```

Notas al pie

Bibliografía

https://www.tutorialspoint.com/es/jpa/jpa_jpql.htm

http://javeritos.blogspot.com/2013/04/ventajas-y-desventajas-de-enterprice.html

https://users.dcc.uchile.cl/~jbarrios/J2EE/node46.html

https://users.dcc.uchile.cl/~jbarrios/J2EE/node47.html

https://www.arquitecturajava.com/introduccion-a-ejb-3-1-i/