Trabajando con JPA

Más configuración

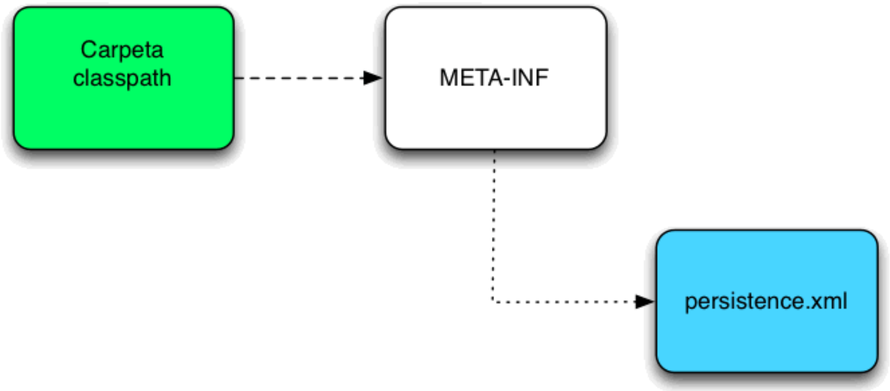
Índice del capítulo

Introducción

- ¿Existe alguna forma de configurar de una manera **más flexible** el fichero persistence.xml?
- Es una pregunta interesante ya que aunque muchas veces con la configuración de JPA por defecto nos es suficiente, **existen otras situaciones** en las que podemos necesitar una **mayor flexibilidad** a nivel de **persistence.xml**.
- Este fichero se suele ubicar en la **carpeta META-INF** de nuestra aplicación, es lo que recomienda la especificación.
- Ahora bien ¿puede estar el fichero persistence.xml en algún otro sitio?

El fichero persistence.xml y classpath

- La respuesta es **SÍ**. El fichero de persistence.xml se puede ubicar en cualquier carpeta que pertenezca al classpath siempre que **esté ubicado dentro de una carpeta META-INF**.



- De esa forma, podemos ubicar el fichero de persistencia **en otras rutas** que nos resulten más interesantes. Ahora bien, normalmente **esto no es suficiente**.

- Aparece un segundo fichero de configuración llamado orm.xml.

El fichero orm.xml

- Introducción.
- Características fundamentales.
- Mapeo de entidades.
- Configuración de consultas nombradas.
- Configuración de consultas nativas.
- Ajustes de configuración y propiedades.

Introducción

- El **fichero orm.xml** en JPA se utiliza para proporcionar información de **mapeo** y **configuración de entidades** cuando:
  - las **anotaciones** en las clases **no son suficientes** o
  - cuando se desea utilizar **configuración XML** en lugar de **anotaciones**.
- Este archivo es **opcional** y se puede utilizar como **alternativa a las anotaciones** para definir el mapeo objeto-relacional (ORM) en una aplicación JPA.

Características fundamentales

- **Mapeo de Entidades**:
  - Puedes utilizar el fichero orm.xml para especificar el **mapeo de entidades** y sus atributos con la base de datos. Esto incluye información sobre cómo se deben mapear los campos, las relaciones entre entidades, las claves primarias, etc.
- **Configuración de Consultas Nativas**:
  - Puedes definir **consultas SQL nativas** en el fichero orm.xml. Esto puede ser útil cuando necesitas ejecutar consultas personalizadas que no pueden expresarse fácilmente con JPQL (Java Persistence Query Language).
- **Configuración de Consultas Nombradas**:
  - Las **consultas nombradas** también pueden definirse en el fichero orm.xml. Esto es útil para centralizar y organizar las consultas en un solo lugar.

Características fundamentales

- Ajustes de **Configuración y Propiedades**:
  - Puedes incluir **configuraciones** y propiedades específicas de proveedores de persistencia en el fichero orm.xml. Esto podría incluir detalles sobre el proveedor de persistencia, ajustes de rendimiento y otras configuraciones específicas.
- Definición de **generadores de Identificadores**:
  - Puedes definir estrategias de generación de identificadores para las entidades en el fichero orm.xml. Esto incluye detalles sobre **cómo generar claves primarias**, como la generación automática, el uso de secuencias, etc.

Mapeo de entidades

```
<entity-mappings xmlns="http://xmlns.jcp.org/xml/ns/persistence/orm"
            xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/persistence/orm
                        http://xmlns.jcp.org/xml/ns/persistence/orm_2_1.xsd"
            version="2.1">
  <entity class="com.example.Employee">
    <table name="EMPLOYEE_TABLE"/>
    <attributes>
      <id name="id">
        <generated-value strategy="AUTO"/>
      </id>
      <basic name="firstName"/>
      <basic name="lastName"/>
    </attributes>
  </entity>
</entity-mappings>
```

Configuración de consultas nombradas

- Supongamos que tienes una entidad **Employee** y deseas definir una **consulta nombrada** para recuperar todos los empleados con un salario superior a un valor específico:

```
<entity-mappings xmlns="http://xmlns.jcp.org/xml/ns/persistence/orm"
            xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/persistence/orm
                        http://xmlns.jcp.org/xml/ns/persistence/orm_2_1.xsd"
            version="2.1">
  <named-query name="Employee.findBySalary">
    <query>
      <![CDATA[
        SELECT e FROM Employee e WHERE e.salary > :minSalary
      ]]>
    </query>
  </named-query>
</entity-mappings>
```

Configuración de consultas nombradas

- En este ejemplo:
  - **Employee.findBySalary** es el nombre de la consulta nombrada.
  - La consulta SQL JPQL está definida dentro de la etiqueta <**query**>.
  - Se utiliza el **parámetro :minSalary** para representar el **salario mínimo**.
- Después de definir esta consulta nombrada en el fichero orm.xml, podrías utilizarla en tu código Java de la siguiente manera:

```
TypedQuery<Employee> query = entityManager.createNamedQuery("Employee.findBySalary", Employee.class);
query.setParameter("minSalary", 50000);
List<Employee> employees = query.getResultList();
```

Configuración de consultas nativas

- Supongamos que tienes una entidad Employee y deseas definir una consulta nativa para recuperar todos los empleados cuyos salarios estén en un rango específico:

```
<entity-mappings xmlns="http://xmlns.jcp.org/xml/ns/persistence/orm"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/persistence/orm
                http://xmlns.jcp.org/xml/ns/persistence/orm_2_1.xsd"
        version="2.1">
  <named-native-query name="Employee.findBySalaryRange">
    <query>
      <![CDATA[
        SELECT * FROM Employee WHERE salary BETWEEN :minSalary AND :maxSalary
      ]]>
    </query>
    <result-class>com.example.Employee</result-class>
  </named-native-query>
</entity-mappings>
```

Configuración de consultas nativas

- En este ejemplo:
  - Employee.findBySalaryRange es el nombre de la consulta nativa.
  - La consulta SQL nativa está definida dentro de la etiqueta <query>.
  - Se utilizan parámetros :minSalary y :maxSalary para representar el rango de salarios.
- Después de definir esta consulta nativa en el fichero orm.xml, podrías utilizarla en tu código Java de la siguiente manera:

```
Query query = entityManager.createNamedQuery("Employee.findBySalaryRange");
query.setParameter("minSalary", 50000);
query.setParameter("maxSalary", 80000);
List<Employee> employees = query.getResultList();
```

Configuración de consultas nativas

- Este código utiliza la **consulta nativa Employee.findBySalaryRange** para recuperar todos los empleados cuyos salarios estén entre 50,000 y 80,000.
- Ten en cuenta que la **clase Employee** debe tener un constructor que coincida con las **columnas devueltas por la consulta nativa**.
- El **uso de consultas nativas** puede ser necesario en casos donde las **consultas JPQL** no son suficientes o cuando necesitas ejecutar sentencias SQL específicas de tu base de datos.
- Sin embargo, ten en cuenta que el uso de consultas nativas puede hacer que tu aplicación sea **menos portátil** entre **diferentes proveedores** de bases de datos.

Ajustes de configuración y propiedades

```
<entity-mappings xmlns="http://xmlns.jcp.org/xml/ns/persistence/orm"
            xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/persistence/orm
                        http://xmlns.jcp.org/xml/ns/persistence/orm_2_1.xsd"
            version="2.1">
  <persistence-unit-metadata>
    <persistence-unit-defaults>
      <!-- Configuración de Propiedades -->
      <schema>myschema</schema>
      <catalog>mycatalog</catalog>
      <delimited-identifiers/>

      <!-- Configuración de Ajustes -->
      <cascade-persist/>
      <validation-mode>NONE</validation-mode>
    </persistence-unit-defaults>
  </persistence-unit-metadata>
</entity-mappings>
```

Ajustes de configuración y propiedades

- En este ejemplo:
  - **<schema>myschema</schema>** y **<catalog>mycatalog</catalog>** establecen el esquema y el catálogo predeterminados para las entidades. Estos valores pueden variar según tu base de datos.
  - **<delimited-identifiers/>** indica que los identificadores deben ser tratados como identificadores delimitados. Esto puede ser útil si tienes identificadores que coinciden con palabras clave del SQL.
  - **<cascade-persist/>** especifica que la cascada de persistencia está habilitada por defecto. Esto significa que cuando persistes una entidad que contiene otras entidades, las entidades secundarias también se persistirán.
  - **<validation-mode>NONE</validation-mode>** establece el modo de validación en "NONE". Esto desactiva la validación de la entidad al realizar operaciones de persistencia.
- Estos son solo ejemplos y las propiedades y ajustes disponibles pueden variar según el proveedor de persistencia que estés utilizando (Hibernate, EclipseLink, etc.). Asegúrate de consultar la documentación específica de tu proveedor para obtener información detallada sobre las propiedades y ajustes disponibles.