01

Arquivo persistence.xml

Transcrição

Já temos o projeto criado e as dependências do Hibernate e do nosso banco de dados H2 e podemos continuar trabalhando com a JPA. Uma das coisas importantes que temos que fazer em uma aplicação que utiliza JPA é a parte de configuração. Na JPA, as configurações ficam no arquivo `.xml`. É possível configurar também via código Java, mas, geralmente, ficam no arquivo `.xml` chamado `persistence.xml`.

Nesse vídeo, entenderemos como é esse arquivo, quais são as configurações e como elas funcionam. Então, vamos criar esse arquivo no nosso projeto. Como se trata de um arquivo de configuração, colocaremos no "src/main/resources". Mas, ele precisa ficar dentro de uma pasta, então, primeiro criaremos a pasta.

Para isso, apertaremos com o botão direito no "src/main/resources", depois selecionaremos "New > Folder". Na próxima tela, preencheremos "Folder name" com o nome da pasta, que tem que ser "META-INF" (tudo em letra maiúscula). Esse é o nome do Diretório.

Criada a pasta, vamos até ela e selecionaremos "New > Other". Na próxima tela, selecionaremos "XML File" e depois chamaremos o arquivo, em "File name", de "persistence.xml". Agora basta apertar "Finish". Ele criará um arquivo `.xml` em branco, só com o cabeçalho do `.xml` que é: `<?xml version="1.0" encoding="UTF-8"?>`.

E esse é o arquivo `.xml`. Ele precisa ter o nome `persistence.xml` e tem que estar na pasta "META-INF", e, como se trata de uma aplicação Maven, fica no

"rsc/main/resources". Dentro do `.xml` vêm as tags de configuração da JPA. Vamos colar a tag raiz (principal) e o cabeçalho de configuração do `.xml` .

```
<?xml version="1.0" encoding="UTF-8"?>
<persistence version="2.2"
    xmlns="http://xmlns.jcp.org/xml/ns/persistence"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"

xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/persistence
http://xmlns.jcp.org/xml/ns/persistence/persistence_2_2.xsd">


</persistence>
```

COPIAR CÓDIGO

Pronto, aqui está a tag raiz `persistence` e as configurações de *namespace* do `.xml` . Não é necessário decorar, podemos pegar da internet ou de algum outro projeto. Dentro do arquivo, existe a tag raiz chamada `persistence` , e todas as configurações ficam dentro dela. Inclusive, ele já está reclamando, como se dissesse: é obrigatório ter uma tag `persistence-unit` e ela não está aqui.

Então, vamos adicionar a tag `persistence-unit` e dividí-la em dois, da seguinte maneira:

```
<persistence-unit name="">

</persistence-unit>
```

COPIAR CÓDIGO

A tag `persistence-unit` tem duas propriedades principais. Temos a `name` , e nela podemos colocar o nome que quisermos. No caso, escolheremos o nome `"loja"` , o mesmo do projeto. Há outra tag importante chamada `transaction`

`type="JTA"` com dois valores possíveis: JTA ou RESOURCE LOCAL. Nós utilizaremos a `"RESOURCE_LOCAL"` .

```
<persistence-unit name="loja" transaction-
type="RESOURCE_LOCAL">

</persistence-unit>
```

COPIAR CÓDIGO

A opção pela `JTA` seria mais adequada para quando estamos utilizando um servidor de aplicação, quando vai trabalhar com EJB, JMS ou outras tecnologias do Java EE, e o servidor se encarrega de cuidar da transação. Como, no nosso caso, se trata de uma aplicação *stand-alone*, sem servidor de aplicação, então será `"RESOURCE_LOCAL"` . Nós que gerenciaremos a transação. Depois veremos com calma essa parte de transação.

Então, a tag `persistence-unit` , tem que ficar dentro da tag raiz `persistence` , e dentro da tag `persistence-unit` que vão todas as configurações da nossa aplicação. Vamos pensar no `persistence-unit` como se ele fosse um banco de dados.

Se a nossa aplicação fosse trabalhar com dois, três bancos de dados, deveríamos ter, dentro da tag `persistence` , duas ou três tags `persistence-unit` . Ou seja, é um `persistence-unit` para cada banco de dados.

Dentro da tag `persistene-unit` nós adicionamos algumas propriedades para ensinar a JPA detalhes referentes ao nosso projeto (banco de dados e coisas do gênero). Então, existe uma tag chamada `properties` , e dentro dela, cada propriedade que configurarmos, fica em uma tag chamada `property` que tem um nome e um valor.

```
    <persistence-unit name="loja" transaction-
type="RESOURCE_LOCAL">
```

```
        <properties>
                <property name="" value=""/>
            </properties>
    </persistence-unit>
</persistence>
```

COPIAR CÓDIGO

Existem algumas propriedades que são obrigatórias, que precisamos informar para a JPA. Essas propriedades são específicas da JPA, então, elas têm um nome específico. Colocamos `"javax.persistence.` e a configuração queremos fazer. Se quisermos configurar, por exemplo, o driver do banco de dados, então, se estamos utilizando o H2, precisamos dizer para a JPA qual a classe do driver do H2.

Então, a propriedade se chama `javax.persistence.jdbc.driver` . Esse é o nome da propriedade. Não é necessário decorar, essas linhas de `.xml` ficarão disponíveis. É possível também pegar alguma como exemplo na internet ou de outro projeto. Nosso objetivo é apenas entender o que significa cada uma dessas propriedades.

No `value` , passaremos qual a classe do driver do JDBC. No caso do H2, é `"org.h2.Driver"` . Esse driver mudará de acordo com o banco de dados que quisermos utilizar. Se fosse o MySQL, seria `com.mysql.` . Se fosse PostgreSQL, `org.postgresql.` . Ou seja, varia de acordo com o banco de dados.

```
<property name="javax.persistence.jdbc.driver"
value="org.h2.Driver"/>
```

COPIAR CÓDIGO

Essa linha serve para dizer à JPA qual é a classe e onde está o driver do JDBC. Estamos usando a JPA, mas ela nada mais é do que uma camada de abstração em cima do JDBC. Por "baixo dos panos", de forma oculta, a JPA trabalha com o JDBC. Por isso utilizamos as propriedades do JDBC.

Além do driver, precisamos configurar a JPA indicando qual é a URL do banco de dados, isto é, onde está o endereço de conexão com o banco de dados. Esse endereço também varia de acordo com o banco de dados. No caso do H2, será `"h2:mem:loja"` . Isto é, queremos que o *database* no H2 se chame loja.

```
<property name="javax.persistence.jdbc.url"
value="h2:mem:loja"/>
```

COPIAR CÓDIGO

Todo banco de dados tem um usuário e uma senha. Portanto, teremos mais duas propriedades, a `jdbc.username"` , e a `jdbc.passwor"` . Então, essas são as duas propriedades para configurar o usuário e a senha. No caso do H2, será `"sa"` (geralmente usamos esse usuário no H2), e a senha ficará em branco `value""` .

```
<property name="javax.persistence.jdbc.username"
value="sa"/>
<property name="javax.persistence.jdbc.password" value=""/>
```

COPIAR CÓDIGO

Da JPA, são só essas quatro propriedades: qual é o driver do banco de dados; onde está a URL de conexão com o banco de dados; o usuário e a senha. Feito isso, a JPA consegue gerar as conexões para acessar o nosso banco de dados.

Existem propriedades específicas da implementação da JPA que estamos utilizando. Por exemplo, existem propriedades específicas do Hibernate, do TopLink, do OpenJPA, do EclipseLink, enfim, de cada uma das implementações da JPA. Como estamos utilizando o Hibernate, podemos colocar algumas propriedades específicas dele.

Então, vamos adicionar mais uma propriedade que terá o nome `"hibernate.dialect"` . O Hibernate precisa saber qual é a classe que tem o

dialeto do banco de dados. Ou seja, saber das particularidades do banco de dados. Por exemplo, no H2 não existe booleano (booleano é inteiro, 0 e 1), no MySQL existe.

Como cada banco de dados pode ter as suas particularidades, o dialeto é o que fará a comunicação correta com o banco de dados. O Hibernate precisa que essa propriedade será fornecida. O valor será a classe do Hibernate que representa o dialeto. No caso do H2, será `"org.hibernate.dialect.H2Dialect"` .

```
<property name="hibernate.dialect"
value="org.hibernate.dialect.H2Dialect"/>
```

COPIAR CÓDIGO

Essas são as principais propriedades de configuração do banco de dados (existe outra propriedade que é interessante configurar, mas adicionaremos posteriormente). O arquivo `persistence.xml` pode até parecer muito complicado, mas se trata, basicamente, de configuração, e precisamos configurar apenas uma vez.

A ideia é ensinar para a JPA detalhes do nosso banco de dados para que ela consiga se conectar e acessar o banco de dados corretamente. A JPA depende disso para se unir ao JDBC e acessar o banco de dados.

Apenas recapitulando, temos o `xml` com a tag raiz chamada `persistence` . Tudo estará dentro desta tag principal `persistence` . Depois temos os *namespaces* (padrão da JPA). Em seguida, vem a tag `pesustence-unit` , que representa uma unidade de persistência (podemos pensar nela como um banco de dados), por isso ela precisa ter um nome (que, no nosso caso, é `"loja"`).

Se tivéssemos vários bancos de dados, teríamos várias tags `persistence-unit` e conseguiríamos diferenciá-las pelo `name` , que precisa ser único para cada uma

delas. No caso, colocamos `"loja"`, que é o mesmo nome do projeto, mas não é obrigatório, poderia ser qualquer outro.

Adicionamos também o tipo de transação, `transaction-type`, que é `"RESOURCE_LOCAL"`, no caso de gerenciarmos a transação, ou JTA, se tivermos usando algum Java EE, e o servidor se encarregará de cuidar do controle transacional, que não é o nosso caso.

Dentro da tag `persistence-unit`, temos `properties` e as propriedades da JPA: driver do banco de dados; URL; usuário; e senha do banco de dados. Como implementação do Hibernate, temos o dialeto. Depois veremos outras propriedades.

O objetivo do vídeo era conhecer o arquivo de configuração da JPA, as tags, as propriedades e para que servem cada uma delas. Espero que vocês tenham aprendido um pouco. Na próxima aula continuamos trabalhando no projeto, vendo a questão das entidades e como fazer para acessar de fato o banco de dados. Vejo vocês lá!! Abraços!!