05

Configurando o TransactionManager

Transcrição

Nosso próximo passo é disponibilizar essa configuração para o **Spring**. Faremos isso na nossa class: `ServletSpringMVC`. No Método: `getServletConfigClasses`.

```
@Override
protected Class<?>[] getServletConfigClasses() {
    return new Class[] {AppWebConfiguration.class, JPAConfiguration.class};
}
```

Mais um passo é necessário para podermos finalizar esta etapa de configuração. Como as nossas entidades serão gerenciadas pelo **Framework**, precisaremos setar mais um atributo, que essencialmente é utilizado sempre que usamos o banco de dados. O **id**. Então em nossa classe `Produto`, definiremos o `id`.

```
@Entity
public class Produto {

    @Id @GeneratedValue(strategy=GenerationType.IDENTITY)
    private int id;

    [...]
}
```

Basicamente, só precisaríamos do `@Id`. Mas para que não precisemos gerenciá-lo manualmente, usamos a segunda anotação (`@GeneratedValue(strategy=GenerationType.IDENTITY)`) para que o próprio *framework* o gere e o gerencie. Feito isso podemos acessar a página de cadastro de produtos e tentarmos cadastrar um produto. A resposta que temos logo após tentar cadastrar um produto será essa:

```
HTTP Status 500 - Request processing failed; nested exception is javax.persistence.TransactionRe
```

Como assim? Aparentemente estava tudo funcionando. Note que o erro é bem claro: `No transactional EntityManager available` do tipo `TransactionRequiredException`. Ou seja, nossa operação com o banco de dados deve ser gerenciada com uma transação.

Façamos então essas últimas configurações para conseguirmos cadastrar nossos produtos no banco de dados. Primeiro precisaremos de um `TransactionManager` que conheça nosso `EntityManager` para que assim ele possa gerenciar as transações de nossas entidades.

Na classe `JPAConfiguration` adicionaremos mais um método que criará o `TransactionManager`.

```
@EnableTransactionManagement
public class JPAConfiguration {

    [...]
```

```
    @Bean
    public JpaTransactionManager transactionManager(EntityManagerFactory emf){
        return new JpaTransactionManager(emf);
    }
}
```

Note que adicionamos a anotação `@EnableTransactionManagement`. Assim o **Spring** ativa o gerenciamento de transações e já reconhece o `TransactionManager`. Agora precisamos definir que o nosso `ProdutoDAO` é uma classe *Transancional* e fazemos isso através da anotação `@Transactional` do pacote org.springframework.transaction.annotation.Transactional.

```
@Repository
@Transactional
public class ProdutoDAO {
    [...]
}
```

É isso! Não precisamos configurar mais nada por hora. Desta forma já conseguiremos cadastrar nossos produtos sem nenhum problema. Não se esqueça de verificar as configurações do seu banco de dados e de criar o banco `casadocodigo` pois o **Hibernate** não cria o banco, mas as tabelas e campos sim.

Experimente cadastrar alguns livros da Casa do Código e verificar se estão realmente no banco de dados. Caso tenha problemas ou dúvidas, publique no fórum, você sempre encontrará alguém para ajudar!