4. Spring Data

A missão da Spring Data é fornecer um modelo de programação familiar e consistente baseado em Spring para acesso a dados, mantendo as características especiais do armazenamento de dados subjacente.

Facilita o uso de tecnologias de acesso a dados, bancos de dados relacionais e não relacionais, estruturas de redução de mapa e serviços de dados baseados em nuvem. Este é um projeto guarda-chuva que contém muitos subprojetos específicos para um determinado banco de dados. Os projetos são desenvolvidos trabalhando em conjunto com muitas das empresas e desenvolvedores que estão por trás dessas tecnologias interessantes.

Recursos:

- Repositório poderoso e abstrações de mapeamento de objeto personalizado
- Derivação de consulta dinâmica de nomes de métodos de repositório
- Classes base de domínio de implementação que fornecem propriedades básicas
- Suporte para auditoria transparente (criado, alterado pela última vez)
- Possibilidade de integrar código de repositório personalizado
- Integração fácil do Spring via JavaConfig e namespaces XML personalizados
- Integração avançada com controladores Spring MVC
- Suporte experimental para persistência entre lojas

4.1 Spring Data JPA

O Spring Data JPA, parte da família Spring Data maior, facilita a implementação de repositórios baseados em JPA. Este módulo lida com suporte aprimorado para camadas de acesso a dados baseadas em JPA. Facilita a criação de aplicativos baseados em Spring que usam tecnologias de acesso a dados.

Implementar uma camada de acesso a dados de um aplicativo tem sido complicado por um bom tempo. Muito código clichê precisa ser escrito para executar consultas simples, além de realizar paginação e auditoria. O Spring Data JPA visa melhorar significativamente a implementação das camadas de acesso a dados, reduzindo o esforço para a quantidade realmente necessária. Como desenvolvedor, você escreve suas interfaces de repositório, incluindo métodos localizadores personalizados, e o Spring fornecerá a implementação automaticamente.

4.2 JPA ou Spring Data JPA?

Durante muito tempo, os desenvolvedores tiveram que lidar com as dificuldades de mapeamento e manipulação de objetos em Java para as suas respectivas representações em um banco de dados, registros em uma tabela. O mundo relacional do banco de dados é incompatível com o mundo orientado a objetos porque eles lidam com os dados de maneiras distintas. Enquanto em Java os registros são representados por objetos que possuem comportamento, no banco de dados os registros são representados de forma textual e sem comportamento.

Para diminuir a incompatibilidade entre eles e permitir que objetos sejam representados no banco de dados, foi criada a especificação *Java Database Connectivity* API, o famoso JDBC. Apesar dos benefícios conquistados, também vieram muitas dificuldades, como problemas de conexão com bancos de dados diversos, casting excessivo de objetos, código complexo para ações simples, ausência de cache de objetos, etc.

Para resolver esse problema, a Sun lançou, em 2006, o primeiro release de sua nova especificação, a JPA, ou Java Persistence API, para a padronização do mapeamento entre o mundo orientado a objetos e o mundo relacional. A partir desse momento, os frameworks mais utilizados, como o Hibernate, o EclipseLink e o TopLink, passaram a seguir as regras dessa especificação, o que possibilitou a migração de uma solução para a outra de forma fácil, além de uma API simples e padronizada, permitindo o reaproveitamento do conhecimento de um framework em relação a outro.

Apesar das incríveis funcionalidades existentes na especificação JPA, muitas delas são repetitivas, fazendo com que o mesmo código seja reescrito diversas vezes, ou exigem muito código para operações simples. Tendo em vista esse cenário, o Spring criou o framework Spring Data JPA, cujo principal objetivo é permitir que o desenvolvimento com JPA se torne mais fácil, prático e menos repetitivo.

É importante saber que o Spring Data JPA utiliza as próprias funcionalidades da especificação JPA, mas encapsula os seus recursos.

4.3 Persistência de dados: Spring Data JPA

A seguir listamos algumas das principais funcionalidades do Spring Data JPA.

- Operações de criação, remoção, modificação e seleção, conhecidas como CRUD;
- Query Methods;
- Method Names;
- Query Annotation;
- Named Queries;
- Ordenação.

4.4 Configurando o Spring Data JPA

Para configurar o Spring Data JPA, utilizaremos o gerenciador de dependências Maven. Para que o Spring Data JPA seja configurado no Maven, basta adicionar no arquivo *pom.xml*:

```
<dependencies>
    <dependency>
        <groupId>org.springframework.data</groupId>
        <artifactId>spring-data-jpa</artifactId>
        <version>1.9.4.RELEASE</version>
    </dependency>
</dependencies>
```