Universidade do Minho

Mestrado em Engenharia Informática

Arquiteturas Aplicacionais

Ano lectivo 2014/2015

Trabalho Prático

Comparação de Frameworks

Autores:
José Morgado (pg27759)
Luís Miguel Pinto (pg27756)
Pedro Carneiro (pg25324)

Braga, 2 de Março de 2015

Conteúdo

1 Introdução

2 Camada de Dados

A camada de dados funciona como uma interface para persistir os dados/objetos manipulados pela camada de negócio. Esta interação entre as camadas pode ser baseada numa abordagem centrada em qualquer uma delas. Temos por isso de um lado as frameworks ORM, como o Hibernate, e por outro lado as frameworks que colocam a base de dados em primeiro lugar, como o jOOQ. Apresentaremos ainda a biblioteca DbUtils, uma alternativa que simplifica a utilização do JDBC.

2.1 Hibernate

O Hibernate é a framework ORM mais popular. O mapeamento entre objetos e os dados existentes nas tabelas da base de dados pode ser configurado através de um ficheiro XML ou Java Annotations.

Apresenta as seguintes vantagens:

- **Produtividade** Permite persistir o estado dos objetos de forma muito simples, não sendo sequer necessário conhecimentos de SQL.
- **Manutenibilidade** Permite reduzir o número de linhas de código, pelo que é mais fácil perceber o sistema e efetuar refactoring.
- **Portabilidade** Facilmente se muda de base de dados, bastando proceder a algumas mudanças no ficheiro de configuração.

A principal desvantagem tem que ver com o impacto em termos desempenho.

2.2 jOOQ

Esta ferramenta permite gerar código Java a partir da base de dados e escrever código Java com uma sintaxe mais próxima do SQL. Por exemplo:

- **SQL**

```
SELECT * FROM BOOK
WHERE BOOK.PUBLISHED_IN = 2011
ORDER BY BOOK.TITLE
```

- **Java**

```
create.selectFrom(BOOK)
      .where(BOOK.PUBLISHED_IN.eq(2011))
      .orderBy(BOOK.TITLE)
```

Surgiu como resposta para algumas das desvantagens inerentes ao uso de frameworks ORM. Alguns dos seus pontos fortes são:

- **Base de Dados em Primeiro Lugar** Permite utilizar todas as capacidades dos RDBMs e do SQL.
- **Typesafe SQL** A linguagem SQL é type safe, e essa característica é mantida no código Java.
- **Active Records** Usando este padrão, não é necessário escrever comandos SQL para lidar com as operações do tipo CRUD.
- **Standadização** Permite utilizar qualquer dialeto do SQL, pois as expressões são adaptadas automaticamente à base de dados utilizada. Assim, o mesmo código funcionará em qualquer base de dados.
- **Procedures** O gerador de código também gera métodos para cada procedimento.

2.3 DbUtils

Esta biblioteca foi concebida com o intuito de facilitar a utilização do JDBC, reduzindo o código necessário para efetuar queries e updates. Apresenta as seguintes vantagens:

- Redução da probabilidade de ocorrência de erros inerentes à gestão de recursos/ligações.
- Código mais limpo e legível.
- Geração automática de JavaBeans a partir dos ResultSets.

2.4 Selecção de uma alternativa

Todas as alternativas têm as suas vantagens e desvantagens, pelo que a escolha deve ser ponderada consoante o caso:

- **Hibernate** Se quisermos maior produtividade e abstrair ao máximo a camada de dados.
- **jOOQ** Se pretendermos tirar partido das vantagens do SQL, esta alternativa possibilita fazê-lo diretamente em Java de forma rápida e segura.
- **DbUtils** Se o projeto em causa for de dimensão reduzida e se apenas se pretender uma alternativa ao JDBC.

3 Camada Lógica

4 Camada de Apresentação

5 Conclusão

6 Anexos