Hibernate

Marcelo Werneck

Hibernate

Define classes persistentes que são mapeadas a tabelas de um banco de dados relacional.

Hibernate

Recursos iniciais:

- Como é uma classe persistente simples
- Como especificar o mapeamento
- Recursos das instâncias das classes persistentes.

Hibernate

Identificar entidades de negócio em um domínio de problema.

Modelo de domínio: modelo conceitual destas entidades

Modelo criado em Java criando uma classe persistente para cada entidade.

Hibernate

Metadados de mapeamento são definidos.

Como estas classes e atributos se relacionam com tabelas e colunas.

Envolve escrever ou gerar documentos XML.

Podem ser distribuídos junto com a aplicação e usados em tempo de execução.

Modelo de Domínio

Deve realizar somente a modelagem do domínio do negócio (separation of concerns)

Modelo pode ser reusado em qualquer aplicação.

Pode ser testado automaticamente (testes de unidade – junit)

Persistência transparente

Separação completa entre as classes persistentes do modelo de domínio e a lógica de persistência.

Classes persistentes não conhecem e não têm dependência do mecanismo de persistência.

Hibernate permite persistência transparente.

POJO

POJO – Plain Old Java Objects

Abordagem básica.

Modelo de componente para desenvolvimento

Alguns consideram sinônimos de Java Beans.

Hibernate e POJO

Hibernate funciona melhor com um modelo de domínio implementado como POJOs.

Requisitos impostos por Hibernate são basicamente os de boa prática de programação

Exemplo de POJO

Exemplo de Mapeamento

Mapeamento

Um mapeamento hibernate típico define um nome de propriedade JavaBeans, um nome de coluna em um banco de dados e um nome de um tipo Hibernate.

Mapeamento

Mapeia um estilo JavaBean para uma coluna de tabela.

```
<property name="description"
column="DESCRIPTION" type="string"/>
```

```
<property name="description"
column="DESCRIPTION"/>
```

Mapeamento

Elemento <column> pode ser usado.

```
<property name="description"
column="DESCRIPTION" type="string"/>

<property name="description" type="string">
        <column name="DESCRIPTION"/>
    </property>
```

Propriedades derivadas

```
<property name="totalIncludingTax"
      formula="TOTAL + TAX_RATE * TOTAL"
      type="big_decimal"/>
```

Propriedades

É possível controlar se uma propriedade pode aparecer em uma declaração insert ou update

```
<property name="name"
        column="NAME"
        type="string"
        insert="false"
        update="false"/>
```

Declaração de Classes

```
<hibernate-mapping>
    <class
    name="org.hibernate.auction.model.Category"
    table="CATEGORY">
        ...
    </class>
</hibernate-mapping>
```

XDoclet

Xdoclet é implementado como uma tarefa Ant que gera código ou metadados XML como parte do processo de build.

Criar um documento de mapeamento com XDoclet é direto.

Em vez de se escrevê-lo a mão, código fonte é marcado com tags javadoc.

Exemplo Xdoclet

```
/**
* @hibernate.class
* table="CATEGORY"
*/
public class Category {
...
}
```

Exemplo Xdoclet

```
/**
* @hibernate.id
* generator-class="native"
* column="CATEGORY_ID"
*/
public Long getId() {
return id;
}
```

Exemplo XDoclet

```
...
/**
* @hibernate.property
*/
public String getName() {
return name;
}
```