Globalcode®
The Developers Company

Open-source Education

Hibernate 3

Iniciativa Globalcode

Palestrante

- RENATO BELLIA

  - Formado em Engenharia de Computadores pela FEI
  - Certificações:
    - Java Programmer, Business Component Developer e DBA certificado pela Oracle
  - Instrutor Globalcode;
    - Academia do Java
    - Academia do Web Developer

  - email: rbellia@globalcode.com.br

Agenda

- Entidades
- Persistência
- Mapeamento Objeto Relacional
- Porque utilizar Hibernate ?
- Como instalar ?
- Persistência com Hibernate
- Automatic Dirty Checking
- Hibernate Query Language

Agenda

- Entidades
- Persistência
- Mapeamento Objeto Relacional
- Porque utilizar Hibernate ?
- Como instalar ?
- Persistência com Hibernate
- Automatic Dirty Checking
- Hibernate Query Language

Entidades

São classes cujos objetos representam elementos existentes no mundo real.

- Cliente
- Cheque
- Registro de tarifação
- Passagem aérea
- Registros de vendas

Agenda

- Entidades
- Persistência
- Mapeamento Objeto Relacional
- Porque utilizar Hibernate ?
- Como instalar ?
- Persistência com Hibernate
- Automatic Dirty Checking
- Hibernate Query Language

Persistência

Os objetos de classes entidade precisam manter seu estado entre diversas sessões de utilização de software.

Ou seja, devem ser não-voláteis ou persistentes.

Persistência

Operações de persistência diante de um repositório de dados: CRUD

- Criação, ou inserção (Create)
- Leitura ou recuperação do estado (Read)
- Atualização do estado persistido (Update)
- Eliminação dos dados (Delete)

Persistência

Os bancos de dados relacionais surgiram na década de 70 e representam hoje a principal solução de armazenamento de dados em sistemas corporativos.

API JDBC:

Java suporta conectividade com bancos de dados relacionais desde a versão 1.1

Agenda

- Entidades
- Persistência
- Mapeamento Objeto Relacional
- Porque utilizar Hibernate ?
- Como instalar ?
- Persistência com Hibernate
- Automatic Dirty Checking
- Hibernate Query Language

Mapeamento Objeto Relacional (ORM)

Estrutura Orientada a Objetos	Estrutura Relacional
Classe (Entidade)	Tabela
Objeto	Linha
Atributo	Coluna
Método	- - -
Associação	Chave estrangeira

Agenda

- Entidades
- Persistência
- Mapeamento Objeto Relacional
- Porque utilizar Hibernate ?
- Como instalar ?
- Persistência com Hibernate
- Automatic Dirty Checking
- Hibernate Query Language

Porque utilizar o Hibernate?

- **custo:** é open-source LGPL
- **benefício:** é uma solução poderosa, madura e portável - compatível com diversos bancos de dados relacionais e servidores de aplicação JEE
- **curva de aprendizado:** é rápida comparada com as outras soluções
- **documentação:** livros publicados e diversos tutoriais e artigos disponíveis na internet
- **suporte:** pode ser contratado comercialmente ou pode se recorrer a uma comunidade extremamente ativa nos fóruns de discussão

Porque utilizar o Hibernate?

- Padrão “De Facto” : amplamente adotado pelo mercado superando as especificações EJB 2.x e JDO

- Os conceitos do projeto Hibernate foram adotados para os entity beans segundo a especificação EJB 3

Agenda

- Entidades
- Persistência
- Mapeamento Objeto Relacional
- Porque utilizar Hibernate ?
- Como instalar ?
- Persistência com Hibernate
- Automatic Dirty Checking
- Hibernate Query Language

Como Instalar?

- Download (.zip) em www.hibernate.org – versão 3.0.5 ou outra versão superior em estágio produção
- Configurar o CLASSPATH de seu projeto com os seguintes arquivos:

dom4j.jar *
cglib.jar *
asm.jar *
antlr.jar *
jta.jar *
eh-cache.jar *
commons-collections.jar *
commons-logging.jar *
log4j.jar *

hibernate.jar
[seu driver JDBC] .jar

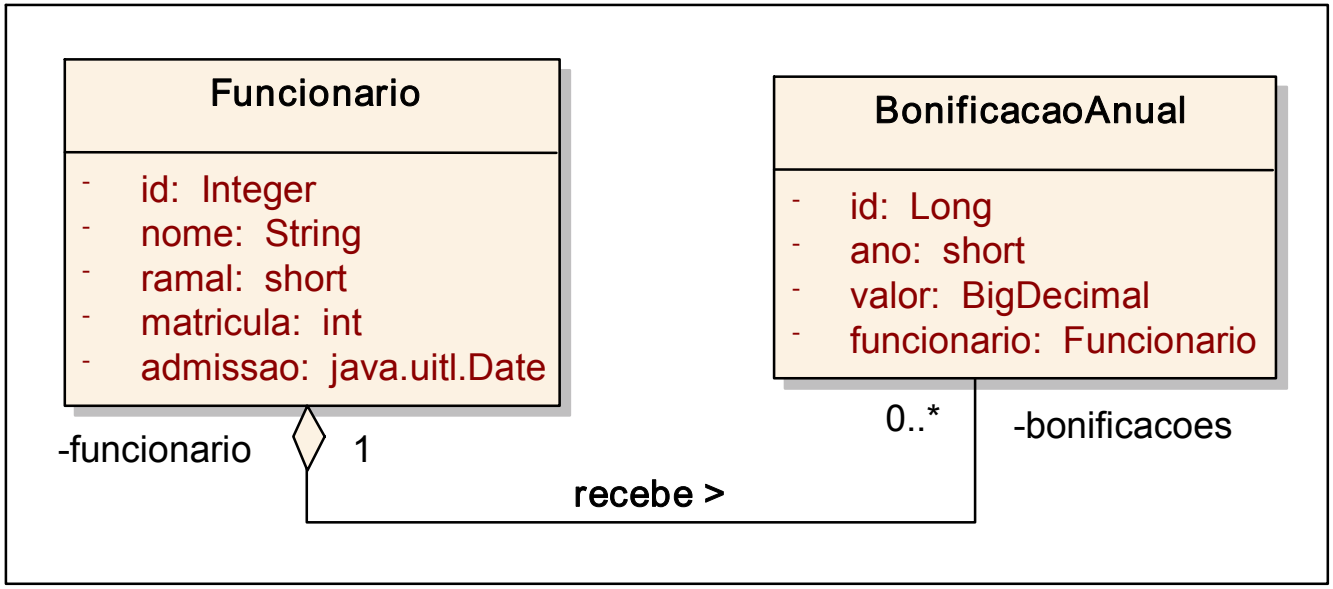
(*) presentes no diretório lib
da instalação do hibernate

Agenda

- Entidades
- Persistência
- Mapeamento Objeto Relacional
- Porque utilizar Hibernate ?
- Como instalar ?
- Persistência com Hibernate
- Automatic Dirty Checking
- Hibernate Query Language

Persistência com Hibernate

Funcionário e Bonificações



Persistência com Hibernate

```
public class Funcionario {

  private Integer id; // chave artificial
  private String nome;
  private short ramal;
  private int matricula; // chave natural
  private Date admissao;
  private Collection bonificacoes = new ArrayList();

  // getters & setters
  // equals & hashCode

}
```

```
public class Funcionario {
  ...
  private int matricula; // chave natural
  ...
  public boolean equals(Object o) {
    if (this == o) return true;
    if (o == null) return false;
    if (!(o instanceof Funcionario)) return false;
    Funcionario that = (Funcionario) o;
    return this.getMatricula() == that.getMatricula();
  }
  public int hashCode() {
    return this.getMatricula();
  }
}
```

Persistência com Hibernate

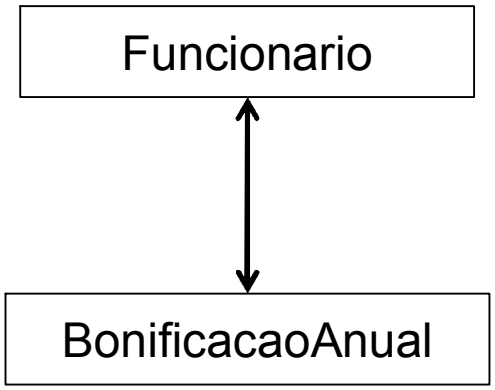
```
public class BonificacaoAnual {

  private Long id; // chave artificial
  private short ano; // chave natural
  private BigDecimal valor;
  private Funcionario funcionario; // chave natural

  // getters & setters
  // equals & hashCode

}
```

```
public class Funcionario {
   ...
  private Collection bonificacoes = new ArrayList();
   ...
   // sugestão para gerenciar associação bi-direcional
  public boolean addBonificacao(BonificacaoAnual ba) {
     if (getBonificacoes() == null)  return false;
     if (! this.equals(ba.getFuncionario())){
        if (ba.getFuncionario() != null) {
          ba.getFuncionario().getBonificacoes().remove(ba);
        }
        ba.setFuncionario(this);
     }
     return getBonificacoes().add(ba);
  }
}
```



hibernate.cfg.xml – arquivo de configuração

```
<hibernate-configuration>
  <session-factory>
    <property name="dialect">org.hibernate.dialect.HSQLDialect</property>
    <property name="connection.driver_class">
        org.hsqldb.jdbcDriver </property>
    <property name="connection.url">
        jdbc:hsqldb:file:db/testdb</property>
    <property name="connection.username"> usuario </property>
    <property name="connection.password"> senha </property>

    <mapping resource="my/model/Funcionario.hbm.xml" />
    <mapping resource="my/model/BonificacaoAnual.hbm.xml" />

  </session-factory>
</hibernate-configuration>
```

descrição da base de dados

arquivos de mapeamento objeto-relacional

Funcionario.hbm.xml – arquivo de mapeamento

```
<hibernate-mapping>
 <class table="tb_funcionario" name="my.model.Funcionario">
  <id column="id_funcionario" name="id">
    <generator class="identity"/>
  </id>
  <property name="admissao" not-null="true" type="date"
                                                column="dt_admissao"/>
  <property name="matricula" column="nr_matricula"/>
  <property name="nome" not-null="true" length="50" column="tx_nome"/>
  <property name="ramal" column="nr_ramal"/>
  <bag lazy="false" cascade="all" name="bonificacoes">
     <key column="id_funcionario"/>
     <one-to-many class="my.model.BonificacaoAnual"/>
  </bag>
 </class>
</hibernate-mapping>
```

primary key
chave artificial

coleção de bonificações

BonificacaoAnual.hbm.xml – arquivo de mapeamento

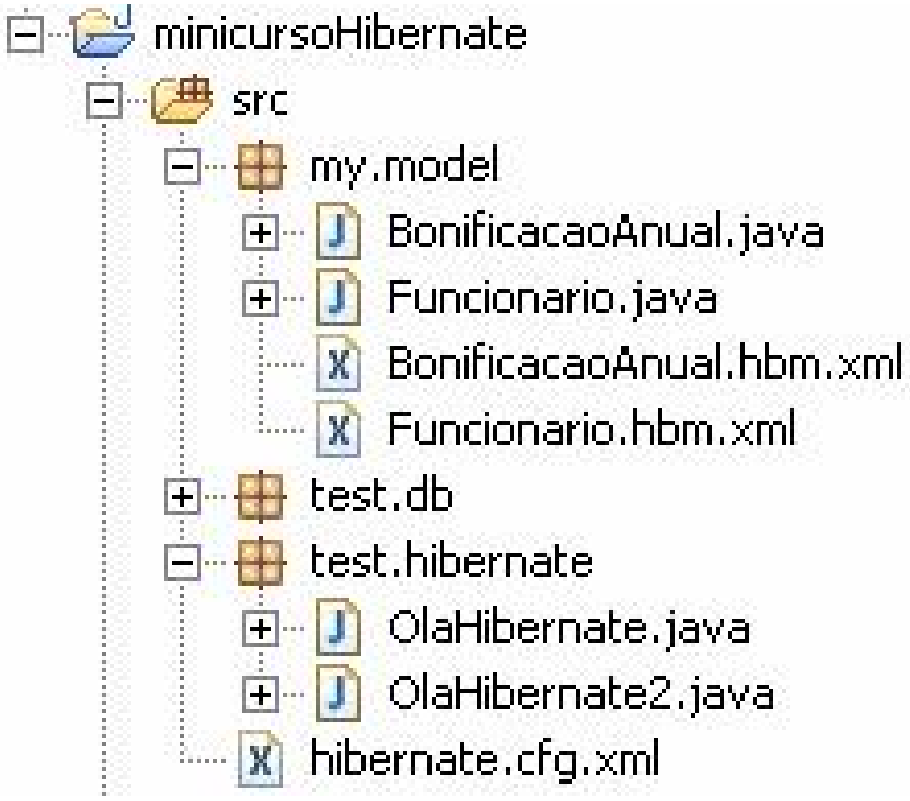
```
<hibernate-mapping>
  <class table="tb_bonus" name="my.model.BonificacaoAnual">
    <id column="id_bonus" name="id">
      <generator class="identity"/>
    </id>
    <property name="ano" not-null="true" column="nr_ano"/>
    <property name="valor" not-null="true" type="big_decimal"
                                                    column="vl_bonus"/>
    <many-to-one not-null="true" column="id_funcionario" name="funcionario"/>
  </class>
</hibernate-mapping>
```

primary key
chave artificial

referência para Funcionario

Persistência com Hibernate

Estrutura de diretórios do projeto



Persistência com Hibernate

- **Session:** representa uma sessão de interação com o banco de dados. Através de um objeto Session podemos realizar as operações de persistência (CRUD)
- **SessionFactory**: Fábrica de objetos **Session.** É configurada com os arquivos de mapeamento (*.hbm.xml) e com as informações sobre o banco de dados (hibernate.cfg.xml)

Persistência com Hibernate

```
import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.cfg.Configuration;

public static void main(String[ ] args) {

   Configuration cfg = new Configuration();
    cfg.configure(); // hibernate.cfg.xml na raiz da estrutura de pacotes
    SessionFactory sessionFactory = cfg.buildSessionFactory();
    Session session = sessionFactory.openSession();

    // operações de persistência

    session.close();
    sessionFactory.close();
}
```

Persistência com Hibernate

Demonstração

- OlaHibernate.java : save
- OlaHibernate2.java : get
- OlaHibernate3.java : get + modificação

Agenda

- Entidades
- Persistência
- Mapeamento Objeto Relacional
- Porque utilizar Hibernate ?
- Como instalar ?
- Persistência com Hibernate
- Automatic Dirty Checking
- Hibernate Query Language

Automatic dirty checking

- **Objeto Transiente :** objeto ignorado pela Session do Hibernate
- **Objeto Persistente :** objeto com estado persistente, gerenciado pela Session do Hibernate, sujeito ao mecanismo “automatic dirty checking”
- **Automatic dirty checking :** verificação automática de mudanças nos objetos persistentes. O estado do objeto em memória será refletido no banco de dados quando as operações ***session.flush()*** da ou ***transaction.commit()*** são executadas.

Agenda

- Entidades
- Persistência
- Mapeamento Objeto Relacional
- Porque utilizar Hibernate ?
- Como instalar ?
- Persistência com Hibernate
- Automatic Dirty Checking
- Hibernate Query Language

Hibernate Query Language (HQL)

Linguagem de consulta aos objetos persistidos, semelhante ao SQL.

Exemplos:

Para recuperar todas bonificações com valor superior a 1700:

```
from BonificacaoAnual b
where b.valor > 1700
```

Para recuperar todos os funcionarios que receberam bonificação no ano de 2005

```
from Funcionario f
where f.bonus.ano = 2005
```

Hibernate Query Language (HQL)

A interface Query é utilizada para executar uma consulta em HQL e retornar uma lista (java.util.List) de objetos recuperados do banco de dados.

```
import org.hibernate.Query
...
public void metodo(...) {
    ...
    String hql = "from BonificacaoAnual b where b.valor > 1700";
    Query query = session.createQuery(hql);
    List bonificacoes = query.list(); // lista de objetos BonificacaoAnual
    for (...) { ...}
}
```

Hibernate Query Language (HQL)

- Demonstração
- OlaHQL.java : consulta parametrizada

Hibernate Query Language (HQL)

- Demonstração
- OlaHQL.java : consulta parametrizada

Dúvidas ?