SERVLETS E JSP (JEE) - JSP - Banco de Dados (Hibernate)

Diogo Cezar Teixeira Batista

diogo@diogocezar.com.br

http://www.diogocezar.com

Universidade Tecnológica Federal do Paraná - UTFPR

Cornélio Procópio - 2012

JSP Banco de Dados (Hibernate) I

- o *Hibernate* é um *framework* para o mapeamento objeto-relacional escrito na linguagem Java;
- este *framework* facilita o mapeamento dos atributos entre uma base tradicional de dados relacionais e o modelo objeto de uma aplicação;
- uso de arquivos (XML) ou anotações Java;

JSP Banco de Dados (Hibernate) II

- Arquivo de configuração: *hibernate.cfg.xml*

```
<hibernate-configuration>
<session-factory>
<property name="hibernate.dialect">org.hibernate.dialect.↩
    PostgreSQLDialect</property>
<propertyname="hibernate.connection.driver_class">org.postgresql.↩
    Driver</property>
<propertyname="hibernate.connection.url">jdbc:postgresql://↩
    localhost:5432/alunos</property>
<property name="hibernate.connection.username">postgres</property>
<property name="hibernate.connection.password">*****</property>
<property name="show_sql">true</property>
<property name="hibernate.hbm2ddl.auto">update</property>
<mapping class="model.Aluno" />
<mapping class="model.Endereco" />
</session-factory>
</hibernate-configuration>
```

JSP Banco de Dados (Hibernate) III

- MVC: é um modelo de desenvolvimento de *Software*;
- é um padrão de arquitetura de aplicações que visa separar a lógica da aplicação (*Model*), da interface do usuário (*View*), e do fluxo da aplicação (*Controller*);
- *Model* (Modelo): representa os dados;
- *View* (Visão): representa a interface de interação com o usuário;
- *Controller* (Controle): representa o elo de ligação entre visão e modelo, é reponsável também por tratar a regra de negócios;

JSP Banco de Dados (Hibernate) IV

- Organização dos arquivos:
  - ClasseField e ClasseList: Responsável por tratar a iteração com o usuário;
  - ClasseDAO: Responsável por invocar as interações com os dados de um modelo;
  - Classe: Modelo que representa os campos da tabela;

JSP Banco de Dados (Hibernate) V

- Exemplo Aluno - AlunoDAO:

```
public class AlunoDAO {
  private SessionFactory factory;
  public AlunoDAO(){
    ...
  }
  public List<Aluno> getAllList(){
    ...
  }
  public void insert(Aluno aluno){
    ...
  }
}
```

JSP Banco de Dados (Hibernate) VI

- Exemplo aluno - Aluno:

```
@Table(name="aluno")
@Entity
public class Aluno {

@Id
@SequenceGenerator(name = "idSeqAluno", sequenceName="aluno_id_seq")
@GeneratedValue(generator = "idSeqAluno")
private Integer id;

@Column(name = "nome")
private String nome;

@Column(name = "idade")
private Integer idade;

@Column(name = "fone")
private String fone;

@Column(name = "email")
private String email;

@OneToMany(mappedBy="aluno", fetch=FetchType.EAGER, cascade = ↩
     CascadeType.ALL, targetEntity = Endereco.class)
private List<Endereco> endereco = new ArrayList<Endereco>();
```

JSP Banco de Dados (Hibernate) VII

- Exemplo aluno - Endereco:

```
@Table(name="endereco")
@Entity
public class Endereco {
@Id
@SequenceGenerator(name = "idSeqEndereco", sequenceName="↩
     endereco_id_seq")
@GeneratedValue(generator = "idSeqEndereco")
private Integer id;

@Column(name = "descricao")
private String descricao;

@JoinColumn(name = "_aluno", referencedColumnName="id")
@ManyToOne
private Aluno aluno;
```

JSP Banco de Dados (Hibernate) VIII

- pode-se tratar os dados como se fossem listas de objetos;
- ao utilizar JSTL alterar o retorno das listas para *Collection*;
- existem métodos de criação automática destas classes (Netbeans);
- A organização do modelo MVC pode se alterar de acordo com as tecnologias utilizadas;
  - JSF + JPA + Facelets;