Curso de Especialização em Desenvolvimento de Aplicações para Dispositivos Móveis e Cloud Computing

DM110

Desenvolvimento Java EE

Prof. Roberto Ribeiro Rocha
rrocha.roberto@gmail.com

Na aula de hoje

- Enterprise Java Beans
  - Session Beans
  - Java Persistence API (Entity Beans)

Enterprise JavaBeans

- Componentes que encapsulam a lógica de negócios de uma aplicação
- Gerenciados pelo EJB container
- Existem dois tipos de EJB:
  - Session Beans
  - Message-Driven Beans
- Antigamente existiam os Entity Beans, mas foram substituídos pela especificação JPA

Enterprise JavaBeans

- Session Beans:
  - Componentes que encapsulam a lógica de negócios da aplicação e podem ser invocados *programaticamente* através de suas interfaces
- Message-Driven Beans (próxima aula):
  - Componentes capazes de processar mensagens de forma assíncrona

Session Beans

- Stateful session beans
  - Mantém estado entre as chamadas
  - Permanecem associados exclusivamente a um ciente
- Stateless session beans
  - Não mantém estado entre as chamadas
  - Podem ser compartilhados por mais de um cliente, sendo liberados após a invocação
- Singleton session beans
  - Possui apenas uma instância em todo o ciclo de vida da aplicação

Interfaces de um Session Bean

- Session Beans podem possuir interfaces locais ou remotas:
  - Local: para acessos locais, em uma mesma JVM
  - Remote: para acessos remotos, entre diferentes instâncias de JVMs
- Essa separação permite uma melhor estruturação dos componentes.

Definindo as Interfaces do Session Bean

```
public interface Hello {
    public String sayHello(String name);
    }
}

public interface HelloLocal extends Hello {}

public interface HelloRemote extends Hello { }
```

Criando um Session Bean

```
@Stateless
@Remote(HelloRemote.class)
@Local(HelloLocal.class)
public class HelloBean implements HelloLocal,HelloRemote{
    @Override
    public String sayHello(String name) {
        return "HelloBean saying hello to " + name;
    }
}
```

Usando o Session Bean dentro de um Servlet

```
@WebServlet("/helloServlet")
public class HelloServlet extends HttpServlet {
  @EJB(lookup = "ejb:dm110-ear-1.0/dm110-ejb-1.0/HelloBean!
br.inatel.dm110.hello.interfaces.HelloRemote")
private HelloRemote helloBean;
  @Override
  protected void doGet(HttpServletRequest req, HttpServletResponse resp)
            throws ServletException, IOException {
    String name = req.getParameter("name");
    resp.setContentType("text/html");
    resp.getWriter().print("<h1>" + helloBean.sayHello(name) + "</h1>");
  }
}
```

Prática com Session Beans

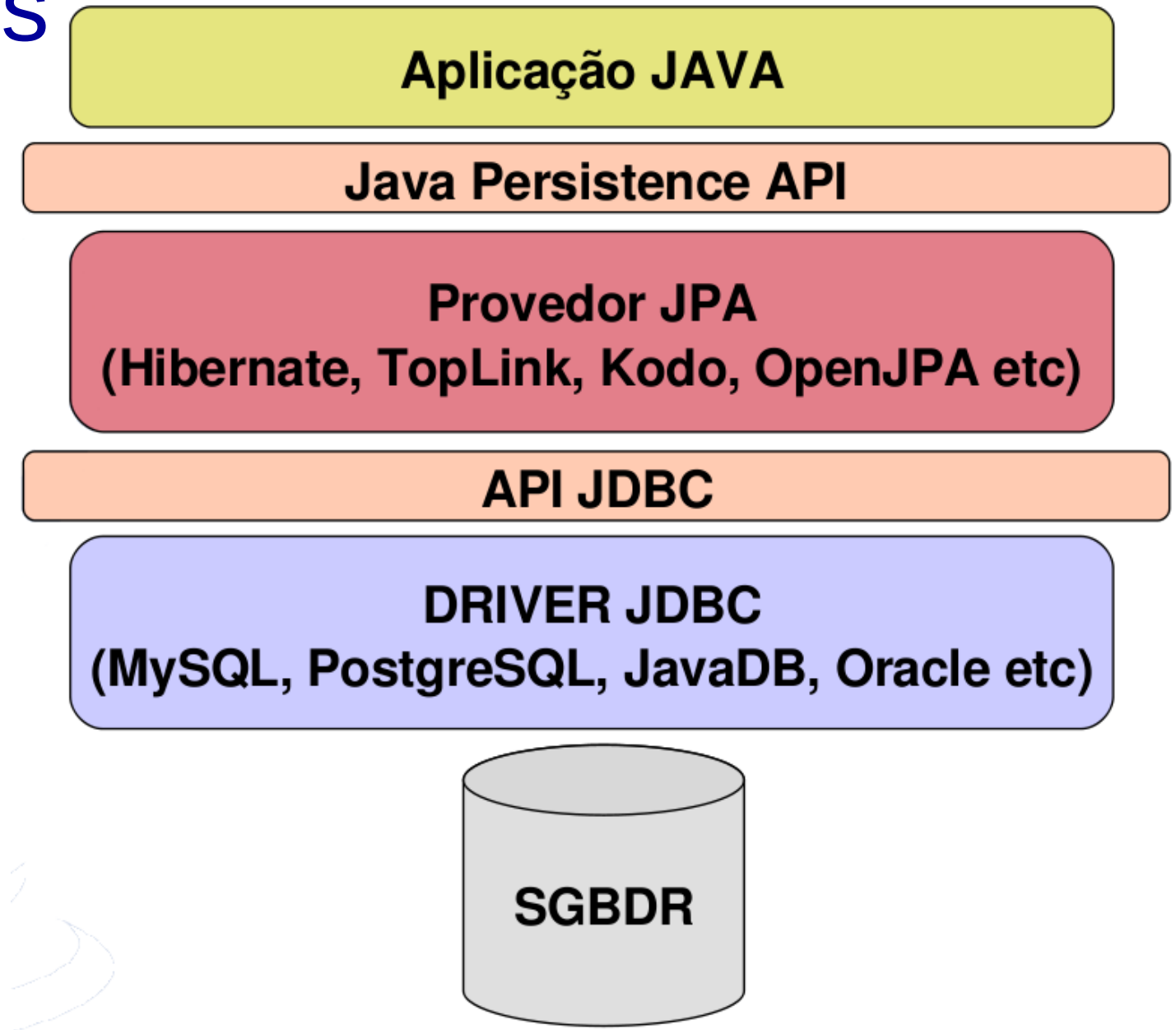
- Criar as interfaces: Hello, HelloLocal e HelloRemote
- Implementar a classe Session Bean (HelloBean)
  - Atenção ao JNDI name
- Chamar o HelloBean dentro de HelloServlet
- Chamar o HelloBean dentro de HelloServiceImpl

Java Persistence API

Java Persistence API

- Provê o mapeamento objeto-relacional
  - Fornece ferramentas para conversão dos dados dos objetos para tabelas do banco de dados
  - E vice-versa
- Usa anotações nas classes
- Possui uma linguagem de consulta e
- Ferramentas para manipular entidades
- Usa configuração por exceção.

JPA Layers


Aplicação JAVA
Java Persistence API
Provedor JPA
(Hibernate, TopLink, Kodo, OpenJPA etc)
API JDBC
DRIVER JDBC
(MySQL, PostgreSQL, JavaDB, Oracle etc)
SGBDR

Características de uma Entidade

- Deve ser anotada com @Entity
- Deve ter um construtor sem argumentos, público ou protegido
  - mas pode ter outros construtores;
- Não deve ser final;
- Deve implementar a interface Serializable.

ORM – Mapeamento Objeto Relacional

- Classe → Tabela
- Objeto → Linha da tabela
- Atributo → Coluna da tabela
- Associação → Chave estrangeira

Principais Annotations JPA

- **@Entity:** declara uma classe como entidade → tabela
- **@Table:** define a tabela de uma entidade
- **@Column:** mapeia o atributo para uma coluna
- **@Id:** especifica a chave primária de uma tabela
- **@GeneratedValue:** especifica valores gerados automaticamente (auto-incremento ou sequence)
  - **@SequenceGenerator:** especifica uma sequência
- **@Temporal:** Mapeia atributos de data/hora
- **@Transient:** Especifica um atributo não persistente

Principais Annotations JPA

- **@OneToOne:** mapeia um atributo com uma classe que possuem cardinalidade 1:1 entre si.
- **@OneToMany:** mapeia um atributo do lado N com uma classe do lado 1
- **@ManyToOne:** mapeia um atributo do lado 1 com uma classe do lado N
- **@JoinColumn:** usada em conjunto com @ManyToOne para indicar qual é o campo da chave estrangeira que está associado com a chave primária da classe atual

Principais métodos do EntityManager

void **persist**(Object entity) // Persiste uma instância

<T> **merge**(T entity) // Faz update da entidade

void **remove**(Object entity) // Remove a instância da entidade

<T> **find**(Class<T> entityClass, Object primaryKey) // Busca pela chave primaria.

Query **createNativeQuery**(String sql) // Cria uma de Query para uma SQL nativa.

TypedQuery<T> **createQuery**(String qlStr, Class<T> resultClass) //Cria Query “tipada”

EntityTransaction **getTransaction**() // Obtem o objeto que controla a transacao.

Exemplo de Mapeamento Objeto/Relacional (ORM) – tabela

```
CREATE TABLE ESTADO (
   IBGE INTEGER NOT NULL,
   SIGLA VARCHAR(2) NOT NULL,
   NOME VARCHAR(30) NOT NULL,
   AREA FLOAT NOT NULL,
   PRIMARY KEY IBGE
);
```

Exemplo de ORM – Entidade

```
@Entity
@Table(name = "ESTADO")
public class State implements Serializable {
    public State() {} //construtor default

    @Id
    private int ibge;
    private String sigla;
    private String nome;
    private float area;
    // Getters e setters...
```

Exemplo de DAO

```
public class IBGEDAO {

    @PersistenceContext(unitName = "IBGE_pu")
    private EntityManager em;

    public List<State> listAll() {
        Query query = em.createQuery("from State s", State.class)
        return query.getResultList();
    }

    public void insert(State state) {
        em.persist(state);
    }
}
```

Exemplo de Persistence Unit

```
<?xml version="1.0" encoding="UTF-8"?>
<persistence version="1.0" -- omição de várias propriedades>
<persistence-unit name="ibge_pu">

<provider>org.hibernate.jpa.HibernatePersistenceProvider</provider>
    <non-jta-data-source>java:/IBGE_DS</non-jta-data-source>
       <properties>
          <property name="hibernate.dialect"
                    value="org.hibernate.dialect.PostgreSQLDialect"/>
          <property name="hibernate.show_sql"
                    value="true" />
      </properties>
   </persistence-unit>
</persistence>
```

Prática com ORM/acesso ao banco

- Criar a tabela ESTADO e classe State
- Confirugar o Persistence Unit
- Criar o DAO
- Criar o Stateless Session Bean
  - IBGE, IBGELocal, IBGERemote e IBGEBean
- Criar o serviço Rest: IBGEService
  - Interface, classe e adicionar no RestApplication
- Configurar do Datasource
- Fazer o deploy e teste da aplicação

Exercício

- Fazer um serviço REST para efetuar as seguintes operações para objetos “cidades”.
  - inclusão de um registro
  - listagem de registros
  - busca de um registro através de seu identificador
  - atualização de um registro

Obrigado :)