POLO SETOR O – CEILÂNDIA – DF

DESENVOLVIMENTO FULL STACK
VAMOS INTEGRAR SISTEMAS
2024.2
MAYCON MOURA

Título da Prática:

Explorando o Padrão Front Controller e a Arquitetura MVC em Aplicações Web Java

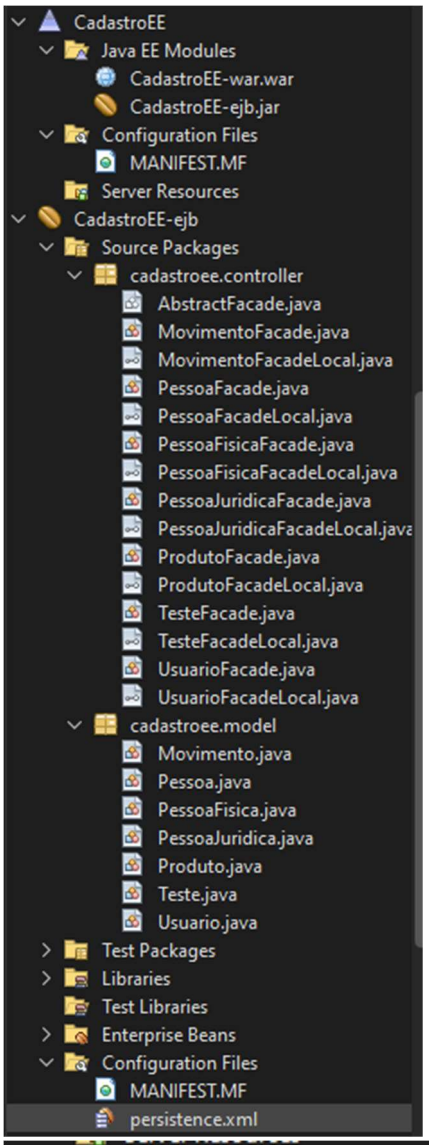
Camadas de Persistência e Controle

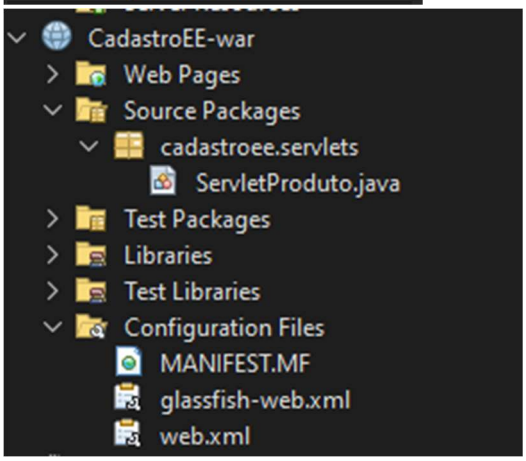
Objetivo da Prática:

O objetivo desta prática é compreender o funcionamento do padrão de design Front Controller e sua implementação em aplicativos Web Java dentro da arquitetura MVC (Model-View-Controller). Além disso, a prática busca explorar as diferenças e semelhanças entre Servlets e JSPs, assim como distinguir entre um redirecionamento simples e o uso do método forward com RequestDispatcher, e entender a função de parâmetros e atributos nos objetos HttpRequest.

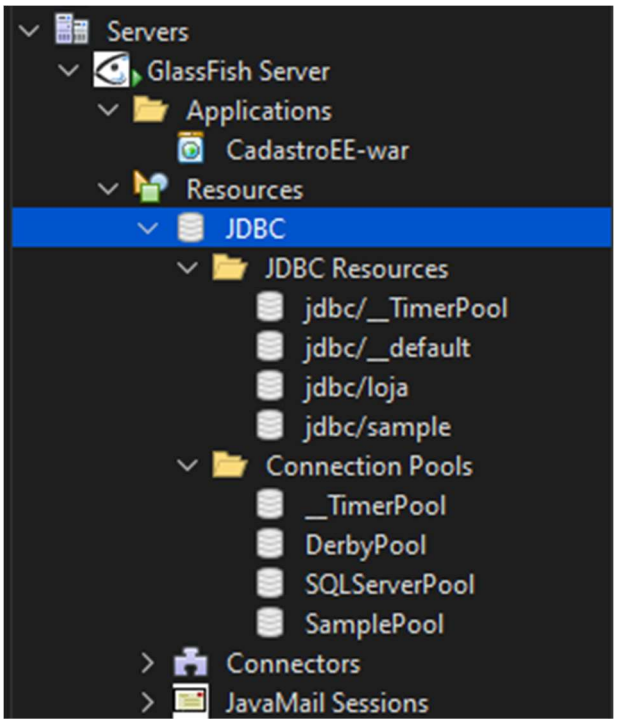
Códigos:

Estrutura:


CadastroEE
Java EE Modules
CadastroEE-war.war
CadastroEE-ejb.jar
Configuration Files
MANIFEST.MF
Server Resources
CadastroEE-ejb
Source Packages
cadastroee.controller
AbstractFacade.java
MovimentoFacade.java
MovimentoFacadeLocal.java
PessoaFacade.java
PessoaFacadeLocal.java
PessoaFisicaFacade.java
PessoaFisicaFacadeLocal.java
PessoaJuridicaFacade.java
PessoaJuridicaFacadeLocal.java
ProdutoFacade.java
ProdutoFacadeLocal.java
TesteFacade.java
TesteFacadeLocal.java
UsuarioFacade.java
UsuarioFacadeLocal.java
cadastroee.model
Movimento.java
Pessoa.java
PessoaFisica.java
PessoaJuridica.java
Produto.java
Teste.java
Usuario.java
Test Packages
Libraries
Test Libraries
Enterprise Beans
Configuration Files
MANIFEST.MF
persistence.xml


CadastroEE-war
Web Pages
Source Packages
cadastroee.servlets
ServletProduto.java
Test Packages
Libraries
Test Libraries
Configuration Files
MANIFEST.MF
glassfish-web.xml
web.xml

Serviço e DB:



AbstractFacade.java

```
package cadastroee.controller;

import jakarta.persistence.EntityManager;
import java.util.List;

/**
 *
 * @author maycon
 */
public abstract class AbstractFacade<T> {

    private Class<T> entityClass;

    public AbstractFacade(Class<T> entityClass) {
        this.entityClass = entityClass;
    }

    protected abstract EntityManager getEntityManager();

    public void create(T entity) {
        getEntityManager().persist(entity);
    }

    public void edit(T entity) {
        getEntityManager().merge(entity);
    }
```

```
    public void remove(T entity) {
        getEntityManager().remove(getEntityManager().merge(entity));
    }

    public T find(Object id) {
        return getEntityManager().find(entityClass, id);
    }

    public List<T> findAll() {
        jakarta.persistence.criteria.CriteriaQuery cq =
getEntityManager().getCriteriaBuilder().createQuery();
        cq.select(cq.from(entityClass));
        return getEntityManager().createQuery(cq).getResultList();
    }

    public List<T> findRange(int[] range) {
        jakarta.persistence.criteria.CriteriaQuery cq =
getEntityManager().getCriteriaBuilder().createQuery();
        cq.select(cq.from(entityClass));
        jakarta.persistence.Query q = getEntityManager().createQuery(cq);
        q.setMaxResults(range[1] - range[0] + 1);
        q.setFirstResult(range[0]);
        return q.getResultList();
    }

    public int count() {
        jakarta.persistence.criteria.CriteriaQuery cq =
getEntityManager().getCriteriaBuilder().createQuery();
        jakarta.persistence.criteria.Root<T> rt = cq.from(entityClass);
        cq.select(getEntityManager().getCriteriaBuilder().count(rt));
        jakarta.persistence.Query q = getEntityManager().createQuery(cq);
        return ((Long) q.getSingleResult()).intValue();
    }}
```

MovimentoFacade.java

```
package cadastroee.controller;

import cadastroee.model.Movimento;
import jakarta.ejb.Stateless;
import jakarta.persistence.EntityManager;
import jakarta.persistence.PersistenceContext;

/**
 *
 * @author maycon
 */
```

```
@Stateless
public class MovimentoFacade extends AbstractFacade<Movimento> implements
MovimentoFacadeLocal {

    @PersistenceContext(unitName = "CadastroEE-ejbPU")
    private EntityManager em;

    @Override
    protected EntityManager getEntityManager() {
        return em;
    }

    public MovimentoFacade() {
        super(Movimento.class);
    }

}
```

MovimentoFacadeLocal.java

```
package cadastroee.controller;

import cadastroee.model.Movimento;
import jakarta.ejb.Local;
import java.util.List;

/**
 *
 * @author maycon
 */
@Local
public interface MovimentoFacadeLocal {

    void create(Movimento movimento);

    void edit(Movimento movimento);

    void remove(Movimento movimento);

    Movimento find(Object id);

    List<Movimento> findAll();

    List<Movimento> findRange(int[] range);

    int count();

}
```

PessoaFacade.java

```
package cadastroee.controller;

import cadastroee.model.Pessoa;
import jakarta.ejb.Stateless;
import jakarta.persistence.EntityManager;
import jakarta.persistence.PersistenceContext;

/**
 *
 * @author maycon
 */
@Stateless
public class PessoaFacade extends AbstractFacade<Pessoa> implements
PessoaFacadeLocal {

    @PersistenceContext(unitName = "CadastroEE-ejbPU")
    private EntityManager em;

    @Override
    protected EntityManager getEntityManager() {
        return em;
    }

    public PessoaFacade() {
        super(Pessoa.class);
    }

}
```

PessoaFacadeLocal.java

```
package cadastroee.controller;

import cadastroee.model.Pessoa;
import jakarta.ejb.Local;
import java.util.List;

/**
 *
 * @author maycon
 */
@Local
public interface PessoaFacadeLocal {

    void create(Pessoa pessoa);
```

```
    void edit(Pessoa pessoa);

    void remove(Pessoa pessoa);

    Pessoa find(Object id);

    List<Pessoa> findAll();

    List<Pessoa> findRange(int[] range);

    int count();

}
```

PessoaFisicaFacade.java

```
package cadastroee.controller;

import cadastroee.model.PessoaFisica;
import jakarta.ejb.Stateless;
import jakarta.persistence.EntityManager;
import jakarta.persistence.PersistenceContext;

/**
 *
 * @author maycon
 */
@Stateless
public class PessoaFisicaFacade extends AbstractFacade<PessoaFisica>
implements PessoaFisicaFacadeLocal {

    @PersistenceContext(unitName = "CadastroEE-ejbPU")
    private EntityManager em;

    @Override
    protected EntityManager getEntityManager() {
        return em;
    }

    public PessoaFisicaFacade() {
        super(PessoaFisica.class);
    }

}
```

PessoaFisicaFacadeLocal.java

```
package cadastroee.controller;

import cadastroee.model.PessoaFisica;
import jakarta.ejb.Local;
import java.util.List;

/**
 *
 * @author maycon
 */
@Local
public interface PessoaFisicaFacadeLocal {

    void create(PessoaFisica pessoaFisica);

    void edit(PessoaFisica pessoaFisica);

    void remove(PessoaFisica pessoaFisica);

    PessoaFisica find(Object id);

    List<PessoaFisica> findAll();

    List<PessoaFisica> findRange(int[] range);

    int count();

}
```

PessoaJuridicaFacade.java

```
package cadastroee.controller;

import cadastroee.model.PessoaJuridica;
import jakarta.ejb.Stateless;
import jakarta.persistence.EntityManager;
import jakarta.persistence.PersistenceContext;

/**
 *
 * @author maycon
 */
@Stateless
public class PessoaJuridicaFacade extends AbstractFacade<PessoaJuridica>
implements PessoaJuridicaFacadeLocal {

    @PersistenceContext(unitName = "CadastroEE-ejbPU")
    private EntityManager em;
```

```
    @Override
    protected EntityManager getEntityManager() {
        return em;
    }

    public PessoaJuridicaFacade() {
        super(PessoaJuridica.class);
    }

}
```

PessoaJuridicaFacadeLocal.java

```
package cadastroee.controller;

import cadastroee.model.PessoaJuridica;
import jakarta.ejb.Local;
import java.util.List;

/**
 *
 * @author maycon
 */
@Local
public interface PessoaJuridicaFacadeLocal {

    void create(PessoaJuridica pessoaJuridica);

    void edit(PessoaJuridica pessoaJuridica);

    void remove(PessoaJuridica pessoaJuridica);

    PessoaJuridica find(Object id);

    List<PessoaJuridica> findAll();

    List<PessoaJuridica> findRange(int[] range);

    int count();

}
```

ProdutoFacade.java

```
package cadastroee.controller;
```

```
import cadastroee.model.Produto;
import jakarta.ejb.Local;
import jakarta.ejb.Stateless;
import jakarta.persistence.EntityManager;
import jakarta.persistence.PersistenceContext;
import java.util.List;

/**
 *
 * @author maycon
 */
@Stateless(name = "ProdutoFacade")
@Local(ProdutoFacadeLocal.class)
public class ProdutoFacade extends AbstractFacade<Produto> implements
ProdutoFacadeLocal {

    @PersistenceContext(unitName = "CadastroEE-ejbPU")
    private EntityManager em;

    @Override
    protected EntityManager getEntityManager() {
        return em;
    }

    @Override
    public List<Produto> findAll() {
        return em.createNamedQuery("Produto.findAll", Produto.class).getResultList();
    }

    public ProdutoFacade() {
        super(Produto.class);
    }

}
```

ProdutoFacadeLocal.java

```
package cadastroee.controller;

import cadastroee.model.Produto;
import jakarta.ejb.Local;
import java.util.List;

/**
 *
 * @author maycon
 */
@Local
public interface ProdutoFacadeLocal {
```

```
    void create(Produto produto);

    void edit(Produto produto);

    void remove(Produto produto);

    Produto find(Object id);

    List<Produto> findAll();

    List<Produto> findRange(int[] range);

    int count();

}
```

UsuarioFacade.java

```
package cadastroee.controller;

import cadastroee.model.Usuario;
import jakarta.ejb.Stateless;
import jakarta.persistence.EntityManager;
import jakarta.persistence.PersistenceContext;

/**
 *
 * @author maycon
 */
@Stateless
public class UsuarioFacade extends AbstractFacade<Usuario> implements
UsuarioFacadeLocal {

    @PersistenceContext(unitName = "CadastroEE-ejbPU")
    private EntityManager em;

    @Override
    protected EntityManager getEntityManager() {
        return em;
    }
    public UsuarioFacade() {
        super(Usuario.class);
    }
}
```

UsuarioFacadeLocal.java

```
package cadastroee.controller;

import cadastroee.model.Usuario;
import jakarta.ejb.Local;
import java.util.List;

/**
 *
 * @author maycon
 */
@Local
public interface UsuarioFacadeLocal {
    void create(Usuario usuario);
    void edit(Usuario usuario);
    void remove(Usuario usuario);
    Usuario find(Object id);
    List<Usuario> findAll();
    List<Usuario> findRange(int[] range);
    int count();
}
```

Movimento.java

```
package cadastroee.model;

import jakarta.persistence.Basic;
import jakarta.persistence.Column;
import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;
import jakarta.persistence.JoinColumn;
import jakarta.persistence.ManyToOne;
import jakarta.persistence.NamedQueries;
import jakarta.persistence.NamedQuery;
import jakarta.persistence.Table;
import jakarta.persistence.Temporal;
import jakarta.persistence.TemporalType;
import jakarta.validation.constraints.NotNull;
import jakarta.xml.bind.annotation.XmlRootElement;
import java.io.Serializable;
import java.math.BigDecimal;
import java.util.Date;

/**
 *
 * @author maycon
```

```
 */
@Entity
@Table(name = "Movimento")
@XmlRootElement
@NamedQueries({
    @NamedQuery(name = "Movimento.findAll", query = "SELECT m FROM
Movimento m"),
    @NamedQuery(name = "Movimento.findByIdCompra", query = "SELECT m FROM
Movimento m WHERE m.idCompra = :idCompra"),
    @NamedQuery(name = "Movimento.findByQuantidade", query = "SELECT m
FROM Movimento m WHERE m.quantidade = :quantidade"),
    @NamedQuery(name = "Movimento.findByPrecoUnitario", query = "SELECT m
FROM Movimento m WHERE m.precoUnitario = :precoUnitario"),
    @NamedQuery(name = "Movimento.findByDataMovimento", query = "SELECT m
FROM Movimento m WHERE m.dataMovimento = :dataMovimento"),
    @NamedQuery(name = "Movimento.findByTipo", query = "SELECT m FROM
Movimento m WHERE m.tipo = :tipo")})
public class Movimento implements Serializable {

    private static final long serialVersionUID = 1L;
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Basic(optional = false)
    @Column(name = "id_compra")
    private Integer idCompra;
    @Basic(optional = false)
    @NotNull
    @Column(name = "quantidade")
    private int quantidade;
    // @Max(value=?)  @Min(value=?)//if you know range of your decimal fields
consider using these annotations to enforce field validation
    @Basic(optional = false)
    @NotNull
    @Column(name = "preco_unitario")
    private BigDecimal precoUnitario;
    @Basic(optional = false)
    @NotNull
    @Column(name = "data_movimento")
    @Temporal(TemporalType.TIMESTAMP)
    private Date dataMovimento;
    @Basic(optional = false)
    @NotNull
    @Column(name = "tipo")
    private Character tipo;
    @JoinColumn(name = "id_pessoa", referencedColumnName = "id_pessoa")
    @ManyToOne(optional = false)
    private Pessoa idPessoa;
    @JoinColumn(name = "id_produto", referencedColumnName = "id_produto")
```

```
    @ManyToOne(optional = false)
    private Produto idProduto;
    @JoinColumn(name = "id_usuario", referencedColumnName = "id_usuario")
    @ManyToOne(optional = false)
    private Usuario idUsuario;

    public Movimento() {
    }

    public Movimento(Integer idCompra) {
        this.idCompra = idCompra;
    }

    public Movimento(Integer idCompra, int quantidade, BigDecimal precoUnitario,
Date dataMovimento, Character tipo) {
        this.idCompra = idCompra;
        this.quantidade = quantidade;
        this.precoUnitario = precoUnitario;
        this.dataMovimento = dataMovimento;
        this.tipo = tipo;
    }

    public Integer getIdCompra() {
        return idCompra;
    }

    public void setIdCompra(Integer idCompra) {
        this.idCompra = idCompra;
    }

    public int getQuantidade() {
        return quantidade;
    }

    public void setQuantidade(int quantidade) {
        this.quantidade = quantidade;
    }

    public BigDecimal getPrecoUnitario() {
        return precoUnitario;
    }

    public void setPrecoUnitario(BigDecimal precoUnitario) {
        this.precoUnitario = precoUnitario;
    }

    public Date getDataMovimento() {
        return dataMovimento;
```

```
}

public void setDataMovimento(Date dataMovimento) {
    this.dataMovimento = dataMovimento;
}

public Character getTipo() {
    return tipo;
}

public void setTipo(Character tipo) {
    this.tipo = tipo;
}

public Pessoa getIdPessoa() {
    return idPessoa;
}

public void setIdPessoa(Pessoa idPessoa) {
    this.idPessoa = idPessoa;
}

public Produto getIdProduto() {
    return idProduto;
}

public void setIdProduto(Produto idProduto) {
    this.idProduto = idProduto;
}

public Usuario getIdUsuario() {
    return idUsuario;
}

public void setIdUsuario(Usuario idUsuario) {
    this.idUsuario = idUsuario;
}

@Override
public int hashCode() {
    int hash = 0;
    hash += (idCompra != null ? idCompra.hashCode() : 0);
    return hash;
}

@Override
public boolean equals(Object object) {
    // TODO: Warning - this method won't work in the case the id fields are not set
```

```
        if (!(object instanceof Movimento)) {
            return false;
        }
        Movimento other = (Movimento) object;
        if ((this.idCompra == null && other.idCompra != null) || (this.idCompra != null &&
!this.idCompra.equals(other.idCompra))) {
            return false;
        }
        return true;
    }

    @Override
    public String toString() {
        return "cadastroee.model.Movimento[ idCompra=" + idCompra + " ]";
    }

}
```

Pessoa.java

```
package cadastroee.model;

import jakarta.persistence.Basic;
import jakarta.persistence.CascadeType;
import jakarta.persistence.Column;
import jakarta.persistence.Entity;
import jakarta.persistence.Id;
import jakarta.persistence.NamedQueries;
import jakarta.persistence.NamedQuery;
import jakarta.persistence.OneToMany;
import jakarta.persistence.OneToOne;
import jakarta.persistence.Table;
import jakarta.validation.constraints.NotNull;
import jakarta.validation.constraints.Size;
import jakarta.xml.bind.annotation.XmlRootElement;
import jakarta.xml.bind.annotation.XmlTransient;
import java.io.Serializable;
import java.util.Collection;

/**
 *
 * @author maycon
 */
@Entity
@Table(name = "Pessoa")
@XmlRootElement
@NamedQueries({
    @NamedQuery(name = "Pessoa.findAll", query = "SELECT p FROM Pessoa p"),
```

```
    @NamedQuery(name = "Pessoa.findByIdPessoa", query = "SELECT p FROM
Pessoa p WHERE p.idPessoa = :idPessoa"),
    @NamedQuery(name = "Pessoa.findByNome", query = "SELECT p FROM Pessoa
p WHERE p.nome = :nome"),
    @NamedQuery(name = "Pessoa.findByTipo", query = "SELECT p FROM Pessoa p
WHERE p.tipo = :tipo"),
    @NamedQuery(name = "Pessoa.findByEndereco", query = "SELECT p FROM
Pessoa p WHERE p.endereco = :endereco"),
    @NamedQuery(name = "Pessoa.findByTelefone", query = "SELECT p FROM
Pessoa p WHERE p.telefone = :telefone"),
    @NamedQuery(name = "Pessoa.findByEmail", query = "SELECT p FROM Pessoa
p WHERE p.email = :email")})
public class Pessoa implements Serializable {

    private static final long serialVersionUID = 1L;
    @Id
    @Basic(optional = false)
    @NotNull
    @Column(name = "id_pessoa")
    private Integer idPessoa;
    @Basic(optional = false)
    @NotNull
    @Size(min = 1, max = 100)
    @Column(name = "nome")
    private String nome;
    @Basic(optional = false)
    @NotNull
    @Column(name = "tipo")
    private Character tipo;
    @Basic(optional = false)
    @NotNull
    @Size(min = 1, max = 255)
    @Column(name = "endereco")
    private String endereco;
    @Basic(optional = false)
    @NotNull
    @Size(min = 1, max = 20)
    @Column(name = "telefone")
    private String telefone;
    //    @Pattern(regexp="[a-z0-9!#$%&'*+/=?^_`{|}~-]+(?:\\.[a-z0-9!#$%&'*+/=?^_`{|}~-
]+)*@(?:[a-z0-9](?:[a-z0-9-]*[a-z0-9])?\\.)+[a-z0-9](?:[a-z0-9-]*[a-z0-9])?",
message="Invalid email")//if the field contains email address consider using this
annotation to enforce field validation
    @Basic(optional = false)
    @NotNull
    @Size(min = 1, max = 100)
    @Column(name = "email")
    private String email;
```

```
    @OneToOne(cascade = CascadeType.ALL, mappedBy = "pessoa")
    private PessoaJuridica pessoaJuridica;
    @OneToOne(cascade = CascadeType.ALL, mappedBy = "pessoa")
    private PessoaFisica pessoaFisica;
    @OneToMany(cascade = CascadeType.ALL, mappedBy = "idPessoa")
    private Collection<Movimento> movimentoCollection;

    public Pessoa() {
    }

    public Pessoa(Integer idPessoa) {
        this.idPessoa = idPessoa;
    }

    public Pessoa(Integer idPessoa, String nome, Character tipo, String endereco,
String telefone, String email) {
        this.idPessoa = idPessoa;
        this.nome = nome;
        this.tipo = tipo;
        this.endereco = endereco;
        this.telefone = telefone;
        this.email = email;
    }

    public Integer getIdPessoa() {
        return idPessoa;
    }

    public void setIdPessoa(Integer idPessoa) {
        this.idPessoa = idPessoa;
    }

    public String getNome() {
        return nome;
    }

    public void setNome(String nome) {
        this.nome = nome;
    }

    public Character getTipo() {
        return tipo;
    }

    public void setTipo(Character tipo) {
        this.tipo = tipo;
    }
```

```
    public String getEndereco() {
        return endereco;
    }

    public void setEndereco(String endereco) {
        this.endereco = endereco;
    }

    public String getTelefone() {
        return telefone;
    }

    public void setTelefone(String telefone) {
        this.telefone = telefone;
    }

    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }

    public PessoaJuridica getPessoaJuridica() {
        return pessoaJuridica;
    }

    public void setPessoaJuridica(PessoaJuridica pessoaJuridica) {
        this.pessoaJuridica = pessoaJuridica;
    }

    public PessoaFisica getPessoaFisica() {
        return pessoaFisica;
    }

    public void setPessoaFisica(PessoaFisica pessoaFisica) {
        this.pessoaFisica = pessoaFisica;
    }

    @XmlTransient
    public Collection<Movimento> getMovimentoCollection() {
        return movimentoCollection;
    }

    public void setMovimentoCollection(Collection<Movimento> movimentoCollection)
{
        this.movimentoCollection = movimentoCollection;
```

```
    }

    @Override
    public int hashCode() {
        int hash = 0;
        hash += (idPessoa != null ? idPessoa.hashCode() : 0);
        return hash;
    }

    @Override
    public boolean equals(Object object) {
        // TODO: Warning - this method won't work in the case the id fields are not set
        if (!(object instanceof Pessoa)) {
            return false;
        }
        Pessoa other = (Pessoa) object;
        if ((this.idPessoa == null && other.idPessoa != null) || (this.idPessoa != null &&
!this.idPessoa.equals(other.idPessoa))) {
            return false;
        }
        return true;
    }

    @Override
    public String toString() {
        return "cadastroee.model.Pessoa[ idPessoa=" + idPessoa + " ]";
    }

}
```

PessoaFisica.java

```
package cadastroee.model;

import jakarta.persistence.Basic;
import jakarta.persistence.Column;
import jakarta.persistence.Entity;
import jakarta.persistence.Id;
import jakarta.persistence.JoinColumn;
import jakarta.persistence.NamedQueries;
import jakarta.persistence.NamedQuery;
import jakarta.persistence.OneToOne;
import jakarta.persistence.Table;
import jakarta.validation.constraints.NotNull;
import jakarta.validation.constraints.Size;
import jakarta.xml.bind.annotation.XmlRootElement;
import java.io.Serializable;
```

```
/**
 *
 * @author maycon
 */
@Entity
@Table(name = "PessoaFisica")
@XmlRootElement
@NamedQueries({
    @NamedQuery(name = "PessoaFisica.findAll", query = "SELECT p FROM
PessoaFisica p"),
    @NamedQuery(name = "PessoaFisica.findByIdPessoa", query = "SELECT p
FROM PessoaFisica p WHERE p.idPessoa = :idPessoa"),
    @NamedQuery(name = "PessoaFisica.findByCpf", query = "SELECT p FROM
PessoaFisica p WHERE p.cpf = :cpf")})
public class PessoaFisica implements Serializable {

    private static final long serialVersionUID = 1L;
    @Id
    @Basic(optional = false)
    @NotNull
    @Column(name = "id_pessoa")
    private Integer idPessoa;
    @Basic(optional = false)
    @NotNull
    @Size(min = 1, max = 11)
    @Column(name = "cpf")
    private String cpf;
    @JoinColumn(name = "id_pessoa", referencedColumnName = "id_pessoa",
insertable = false, updatable = false)
    @OneToOne(optional = false)
    private Pessoa pessoa;

    public PessoaFisica() {
    }

    public PessoaFisica(Integer idPessoa) {
        this.idPessoa = idPessoa;
    }

    public PessoaFisica(Integer idPessoa, String cpf) {
        this.idPessoa = idPessoa;
        this.cpf = cpf;
    }

    public Integer getIdPessoa() {
        return idPessoa;
    }
```

```
    public void setIdPessoa(Integer idPessoa) {
        this.idPessoa = idPessoa;
    }

    public String getCpf() {
        return cpf;
    }

    public void setCpf(String cpf) {
        this.cpf = cpf;
    }

    public Pessoa getPessoa() {
        return pessoa;
    }

    public void setPessoa(Pessoa pessoa) {
        this.pessoa = pessoa;
    }

    @Override
    public int hashCode() {
        int hash = 0;
        hash += (idPessoa != null ? idPessoa.hashCode() : 0);
        return hash;
    }

    @Override
    public boolean equals(Object object) {
        // TODO: Warning - this method won't work in the case the id fields are not set
        if (!(object instanceof PessoaFisica)) {
            return false;
        }
        PessoaFisica other = (PessoaFisica) object;
        if ((this.idPessoa == null && other.idPessoa != null) || (this.idPessoa != null &&
!this.idPessoa.equals(other.idPessoa))) {
            return false;
        }
        return true;
    }

    @Override
    public String toString() {
        return "cadastroee.model.PessoaFisica[ idPessoa=" + idPessoa + " ]";
    }

}
```

PessoaJuridica.java

```
package cadastroee.model;

import jakarta.persistence.Basic;
import jakarta.persistence.Column;
import jakarta.persistence.Entity;
import jakarta.persistence.Id;
import jakarta.persistence.JoinColumn;
import jakarta.persistence.NamedQueries;
import jakarta.persistence.NamedQuery;
import jakarta.persistence.OneToOne;
import jakarta.persistence.Table;
import jakarta.validation.constraints.NotNull;
import jakarta.validation.constraints.Size;
import jakarta.xml.bind.annotation.XmlRootElement;
import java.io.Serializable;

/**
 *
 * @author maycon
 */
@Entity
@Table(name = "PessoaJuridica")
@XmlRootElement
@NamedQueries({
    @NamedQuery(name = "PessoaJuridica.findAll", query = "SELECT p FROM
PessoaJuridica p"),
    @NamedQuery(name = "PessoaJuridica.findByIdPessoa", query = "SELECT p
FROM PessoaJuridica p WHERE p.idPessoa = :idPessoa"),
    @NamedQuery(name = "PessoaJuridica.findByCnpj", query = "SELECT p FROM
PessoaJuridica p WHERE p.cnpj = :cnpj")})
public class PessoaJuridica implements Serializable {

    private static final long serialVersionUID = 1L;
    @Id
    @Basic(optional = false)
    @NotNull
    @Column(name = "id_pessoa")
    private Integer idPessoa;
    @Basic(optional = false)
    @NotNull
    @Size(min = 1, max = 14)
    @Column(name = "cnpj")
    private String cnpj;
    @JoinColumn(name = "id_pessoa", referencedColumnName = "id_pessoa",
insertable = false, updatable = false)
    @OneToOne(optional = false)
```

```
private Pessoa pessoa;

public PessoaJuridica() {
}

public PessoaJuridica(Integer idPessoa) {
    this.idPessoa = idPessoa;
}

public PessoaJuridica(Integer idPessoa, String cnpj) {
    this.idPessoa = idPessoa;
    this.cnpj = cnpj;
}

public Integer getIdPessoa() {
    return idPessoa;
}

public void setIdPessoa(Integer idPessoa) {
    this.idPessoa = idPessoa;
}

public String getCnpj() {
    return cnpj;
}

public void setCnpj(String cnpj) {
    this.cnpj = cnpj;
}

public Pessoa getPessoa() {
    return pessoa;
}

public void setPessoa(Pessoa pessoa) {
    this.pessoa = pessoa;
}

@Override
public int hashCode() {
    int hash = 0;
    hash += (idPessoa != null ? idPessoa.hashCode() : 0);
    return hash;
}

@Override
public boolean equals(Object object) {
    // TODO: Warning - this method won't work in the case the id fields are not set
```

```
        if (!(object instanceof PessoaJuridica)) {
            return false;
        }
        PessoaJuridica other = (PessoaJuridica) object;
        if ((this.idPessoa == null && other.idPessoa != null) || (this.idPessoa != null &&
!this.idPessoa.equals(other.idPessoa))) {
            return false;
        }
        return true;
    }

    @Override
    public String toString() {
        return "cadastroee.model.PessoaJuridica[ idPessoa=" + idPessoa + " ]";
    }
}
```

Produto.java

```
package cadastroee.model;

import jakarta.persistence.Basic;
import jakarta.persistence.CascadeType;
import jakarta.persistence.Column;
import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;
import jakarta.persistence.NamedQueries;
import jakarta.persistence.NamedQuery;
import jakarta.persistence.OneToMany;
import jakarta.persistence.Table;
import jakarta.validation.constraints.NotNull;
import jakarta.validation.constraints.Size;
import jakarta.xml.bind.annotation.XmlRootElement;
import jakarta.xml.bind.annotation.XmlTransient;
import java.io.Serializable;
import java.math.BigDecimal;
import java.util.Collection;

/**
 *
 * @author maycon
 */
@Entity
@Table(name = "Produto")
@XmlRootElement
@NamedQueries({
```

```
    @NamedQuery(name = "Produto.findAll", query = "SELECT p FROM Produto p"),
    @NamedQuery(name = "Produto.findByIdProduto", query = "SELECT p FROM
Produto p WHERE p.idProduto = :idProduto"),
    @NamedQuery(name = "Produto.findByNome", query = "SELECT p FROM Produto
p WHERE p.nome = :nome"),
    @NamedQuery(name = "Produto.findByQuantidade", query = "SELECT p FROM
Produto p WHERE p.quantidade = :quantidade"),
    @NamedQuery(name = "Produto.findByPrecoVenda", query = "SELECT p FROM
Produto p WHERE p.precoVenda = :precoVenda")})
public class Produto implements Serializable {

    private static final long serialVersionUID = 1L;
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Basic(optional = false)
    @Column(name = "id_produto")
    private Integer idProduto;
    @Basic(optional = false)
    @NotNull
    @Size(min = 1, max = 100)
    @Column(name = "nome")
    private String nome;
    @Basic(optional = false)
    @NotNull
    @Column(name = "quantidade")
    private int quantidade;
    // @Max(value=?)  @Min(value=?)//if you know range of your decimal fields
consider using these annotations to enforce field validation
    @Basic(optional = false)
    @NotNull
    @Column(name = "preco_venda")
    private float precoVenda;
    @OneToMany(cascade = CascadeType.ALL, mappedBy = "idProduto")
    private Collection<Movimento> movimentoCollection;

    public Produto() {
    }

    public Produto(Integer idProduto) {
        this.idProduto = idProduto;
    }

    public Produto(Integer idProduto, String nome, int quantidade, float precoVenda) {
        this.idProduto = idProduto;
        this.nome = nome;
        this.quantidade = quantidade;
        this.precoVenda = precoVenda;
    }
```

```
    public Integer getIdProduto() {
        return idProduto;
    }

    public void setIdProduto(Integer idProduto) {
        this.idProduto = idProduto;
    }

    public String getNome() {
        return nome;
    }

    public void setNome(String nome) {
        this.nome = nome;
    }

    public int getQuantidade() {
        return quantidade;
    }

    public void setQuantidade(int quantidade) {
        this.quantidade = quantidade;
    }

    public float getPrecoVenda() {
        return precoVenda;
    }

    public void setPrecoVenda(float precoVenda) {
        this.precoVenda = precoVenda;
    }

    @XmlTransient
    public Collection<Movimento> getMovimentoCollection() {
        return movimentoCollection;
    }

    public void setMovimentoCollection(Collection<Movimento> movimentoCollection)
{
        this.movimentoCollection = movimentoCollection;
    }

    @Override
    public int hashCode() {
        int hash = 0;
        hash += (idProduto != null ? idProduto.hashCode() : 0);
        return hash;
```

```
    }

    @Override
    public boolean equals(Object object) {
        // TODO: Warning - this method won't work in the case the id fields are not set
        if (!(object instanceof Produto)) {
            return false;
        }
        Produto other = (Produto) object;
        if ((this.idProduto == null && other.idProduto != null) || (this.idProduto != null &&
!this.idProduto.equals(other.idProduto))) {
            return false;
        }
        return true;
    }

    @Override
    public String toString() {
        return "cadastroee.model.Produto[ idProduto=" + idProduto + " ]";
    }

}
```

Usuario.java

```
package cadastroee.model;

import jakarta.persistence.Basic;
import jakarta.persistence.CascadeType;
import jakarta.persistence.Column;
import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;
import jakarta.persistence.NamedQueries;
import jakarta.persistence.NamedQuery;
import jakarta.persistence.OneToMany;
import jakarta.persistence.Table;
import jakarta.validation.constraints.NotNull;
import jakarta.validation.constraints.Size;
import jakarta.xml.bind.annotation.XmlRootElement;
import jakarta.xml.bind.annotation.XmlTransient;
import java.io.Serializable;
import java.util.Collection;

/**
 *
 * @author maycon
```

```
 */
@Entity
@Table(name = "Usuario")
@XmlRootElement
@NamedQueries({
    @NamedQuery(name = "Usuario.findAll", query = "SELECT u FROM Usuario u"),
    @NamedQuery(name = "Usuario.findByIdUsuario", query = "SELECT u FROM
Usuario u WHERE u.idUsuario = :idUsuario"),
    @NamedQuery(name = "Usuario.findByNome", query = "SELECT u FROM Usuario
u WHERE u.nome = :nome"),
    @NamedQuery(name = "Usuario.findByEmail", query = "SELECT u FROM Usuario
u WHERE u.email = :email"),
    @NamedQuery(name = "Usuario.findBySenha", query = "SELECT u FROM
Usuario u WHERE u.senha = :senha")})
public class Usuario implements Serializable {

    private static final long serialVersionUID = 1L;
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Basic(optional = false)
    @Column(name = "id_usuario")
    private Integer idUsuario;
    @Basic(optional = false)
    @NotNull
    @Size(min = 1, max = 100)
    @Column(name = "nome")
    private String nome;
    //    @Pattern(regexp="[a-z0-9!#$%&'*+/=?^_`{|}~-]+(?:\\.[a-z0-9!#$%&'*+/=?^_`{|}~-
]+)*@(?:[a-z0-9](?:[a-z0-9-]*[a-z0-9])?\\.)+[a-z0-9](?:[a-z0-9-]*[a-z0-9])?",
message="Invalid email")//if the field contains email address consider using this
annotation to enforce field validation
    @Basic(optional = false)
    @NotNull
    @Size(min = 1, max = 100)
    @Column(name = "email")
    private String email;
    @Basic(optional = false)
    @NotNull
    @Size(min = 1, max = 255)
    @Column(name = "senha")
    private String senha;
    @OneToMany(cascade = CascadeType.ALL, mappedBy = "idUsuario")
    private Collection<Movimento> movimentoCollection;

    public Usuario() {
    }

    public Usuario(Integer idUsuario) {
```

```
    this.idUsuario = idUsuario;
}

public Usuario(Integer idUsuario, String nome, String email, String senha) {
    this.idUsuario = idUsuario;
    this.nome = nome;
    this.email = email;
    this.senha = senha;
}

public Integer getIdUsuario() {
    return idUsuario;
}

public void setIdUsuario(Integer idUsuario) {
    this.idUsuario = idUsuario;
}

public String getNome() {
    return nome;
}

public void setNome(String nome) {
    this.nome = nome;
}

public String getEmail() {
    return email;
}

public void setEmail(String email) {
    this.email = email;
}

public String getSenha() {
    return senha;
}

public void setSenha(String senha) {
    this.senha = senha;
}

@XmlTransient
public Collection<Movimento> getMovimentoCollection() {
    return movimentoCollection;
}
```

```
    public void setMovimentoCollection(Collection<Movimento> movimentoCollection)
{
        this.movimentoCollection = movimentoCollection;
    }

    @Override
    public int hashCode() {
        int hash = 0;
        hash += (idUsuario != null ? idUsuario.hashCode() : 0);
        return hash;
    }

    @Override
    public boolean equals(Object object) {
        // TODO: Warning - this method won't work in the case the id fields are not set
        if (!(object instanceof Usuario)) {
            return false;
        }
        Usuario other = (Usuario) object;
        if ((this.idUsuario == null && other.idUsuario != null) || (this.idUsuario != null &&
!this.idUsuario.equals(other.idUsuario))) {
            return false;
        }
        return true;
    }

    @Override
    public String toString() {
        return "cadastroee.model.Usuario[ idUsuario=" + idUsuario + " ]";
    }

}
```

ServletProduto.java

```
package cadastroee.servlets;

import cadastroee.controller.ProdutoFacadeLocal;
import java.io.IOException;
import java.io.PrintWriter;
import jakarta.servlet.ServletException;
import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;
import cadastroee.model.Produto;
import jakarta.ejb.EJB;
```

```
import java.util.List;
import javax.naming.InitialContext;
import javax.naming.NamingException;

/**
 *
 * @author maycon
 */
public class ServletProduto extends HttpServlet {

    @EJB
    private ProdutoFacadeLocal facade;

    /**
     * Processes requests for both HTTP <code>GET</code> and
<code>POST</code>
     * methods.
     *
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    protected void processRequest(HttpServletRequest request, HttpServletResponse
response)
            throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        try (PrintWriter out = response.getWriter()) {
            /* TODO output your page here. You may use following sample code. */
            out.println("<!DOCTYPE html>");
            out.println("<html>");
            out.println("<head>");
            out.println("<title>Servlet ServletProduto</title>");
            out.println("</head>");
            out.println("<body>");
            out.println("<h1>Servlet ServletProduto at " + request.getContextPath() +
"</h1>");
            out.println("</body>");
            out.println("</html>");
        }
    }

    // <editor-fold defaultstate="collapsed" desc="HttpServlet methods. Click on the +
sign on the left to edit the code.">
    /**
     * Handles the HTTP <code>GET</code> method.
     *
     * @param request servlet request
```

```
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
//        try {
//            InitialContext ctx = new InitialContext();
//                    facade = (ProdutoFacadeLocal) ctx.lookup("java:global/CadastroEE-
war/ProdutoFacade!cadastroee.controller.ProdutoFacadeLocal");
//            // Use o EJB conforme necessário
//        } catch (NamingException e) {
//            e.printStackTrace();
//            // Trate a exceção
//        }

        response.setContentType("text/html;charset=UTF-8");
        try (PrintWriter out = response.getWriter()) {
            out.println("<html>");
            out.println("<head>");
            out.println("<title>Produtos</title>");
            out.println("</head>");
            out.println("<body>");
            out.println("<h1>Lista de Produtos</h1>");

            if (facade != null) {
                List<Produto> produtos = facade.findAll();
                for (Produto produto : produtos) {
                    out.println("<p>" + produto.getNome() + "</p>");
                }
            } else {
                out.println("<p>Erro: EJB não injetado corretamente!</p>");
            }

            out.println("</body>");
            out.println("</html>");
        }
    }

    /**
     * Handles the HTTP <code>POST</code> method.
     *
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
```

```
    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse
response)
            throws ServletException, IOException {
        processRequest(request, response);
    }

    /**
     * Returns a short description of the servlet.
     *
     * @return a String containing servlet description
     */
    @Override
    public String getServletInfo() {
        return "Short description";
    }// </editor-fold>

}
```

Resultados da Execução:

Análise e Conclusão:

1. Como funciona o padrão Front Controller e como ele é implementado em um aplicativo Web Java, na arquitetura MVC?

O padrão Front Controller é um dos padrões de design usados para centralizar a manipulação de requisições em uma aplicação web. Ele atua como um ponto único de entrada para todas as requisições do cliente. Isso significa que, independentemente do tipo de solicitação ou da

ação solicitada, todas as requisições passam primeiro pelo controlador frontal antes de serem encaminhadas para os componentes apropriados (como outros controllers ou views).

Implementação na Arquitetura MVC:

Model: Representa os dados da aplicação e a lógica de negócio.

View: Responsável por renderizar a interface de usuário com base nos dados fornecidos pelo Model.

Controller: Atua como intermediário entre a View e o Model, processando as requisições do usuário, invocando os serviços do Model e selecionando a View apropriada para retornar ao usuário.

Na arquitetura MVC em Java, o Servlet geralmente atua como o Front Controller, processando todas as requisições, decidindo qual ação tomar e encaminhando a requisição para a View apropriada (JSP ou outro Servlet). Isso garante uma separação clara entre a lógica de negócio e a apresentação, facilitando a manutenção e escalabilidade da aplicação.

2. Quais as diferenças e semelhanças entre Servlets e JSPs?

Semelhanças:

- Ambos são componentes do lado do servidor que geram conteúdo dinâmico em resposta às requisições HTTP.
- Tanto Servlets quanto JSPs podem acessar os mesmos objetos de requisição e resposta (HttpServletRequest e HttpServletResponse) e compartilhar dados por meio de escopos (request, session, application).

Diferenças:

- **Servlets**: São classes Java puras que respondem às requisições HTTP. Eles são mais orientados à lógica de controle e processamento de dados. Como são escritos inteiramente em Java, a criação de conteúdo HTML diretamente em Servlets pode ser complicada e menos intuitiva.
- **JSPs (JavaServer Pages)**: São essencialmente Servlets com uma sintaxe mais amigável para a criação de conteúdo HTML. O JSP permite misturar código Java com marcação HTML, facilitando o desenvolvimento de interfaces de usuário. Ao serem compilados, os JSPs são convertidos em Servlets.

3. Qual a diferença entre um redirecionamento simples e o uso do método forward, a partir do RequestDispatcher?

Redirecionamento Simples:

- O redirecionamento é feito utilizando o método sendRedirect do objeto HttpServletResponse. Quando utilizado, ele envia uma resposta ao cliente, instruindo-o a fazer uma nova requisição para um novo URL. Isso resulta em uma mudança visível na URL do navegador, e uma nova solicitação HTTP é criada.
- **Uso**: Ideal quando se deseja redirecionar o cliente para uma página completamente diferente ou até mesmo um domínio externo.

Forward (via RequestDispatcher):

- O método forward do RequestDispatcher permite que um Servlet ou JSP encaminhe a requisição para outro recurso dentro do mesmo servidor sem que o cliente seja informado. A URL no navegador permanece a mesma, e nenhuma nova solicitação HTTP é criada.
- **Uso**: Comumente usado para encaminhar a requisição de um Servlet para um JSP que renderiza a resposta final, mantendo o controle dentro do servidor.

4. Para que servem parâmetros e atributos nos objetos HttpRequest?

Parâmetros (getParameter):

- São usados para obter dados enviados pelo cliente na requisição HTTP, como em formulários submetidos via GET ou POST. Estes são dados fixos que são transmitidos do cliente para o servidor.

Atributos (setAttribute e getAttribute):

- Servem para compartilhar dados entre componentes do servidor durante o processamento de uma requisição. Diferente dos parâmetros, os atributos não são enviados pelo cliente, mas são definidos e manipulados pelo servidor, podendo ser usados para passar informações de um Servlet para um JSP ou outro Servlet durante o ciclo de requisição/resposta.