

Java Enterprise Edition - JEE

15. Modelo de arquitetura JEE



Esp. Márcio Palheta
gtalk: marcio.palheta@gmail.com



Agenda

- Revisão geral
- Estudo de caso
- Implementação das camadas
- Exercícios



Aula de revisão geral

- Rever os conceitos JEE;
- Ambiente;
- Arquitetura;
 - MVC
 - Design Patterns;
- Análise e Projeto de Software;
- Estudo de caso – Cadastro de disciplinas

Estudo de caso – Cadastro de Disciplinas

- A escola MPTeach está desenvolvendo uma aplicação para administração acadêmica;
- Você foi contratado para implementar um dos módulos do projeto, o cadastro de Disciplinas;
- Por se tratar do módulo inicial, você além de implementar o requisito, deve projetar a arquitetura do sistema;



Passos necessários:

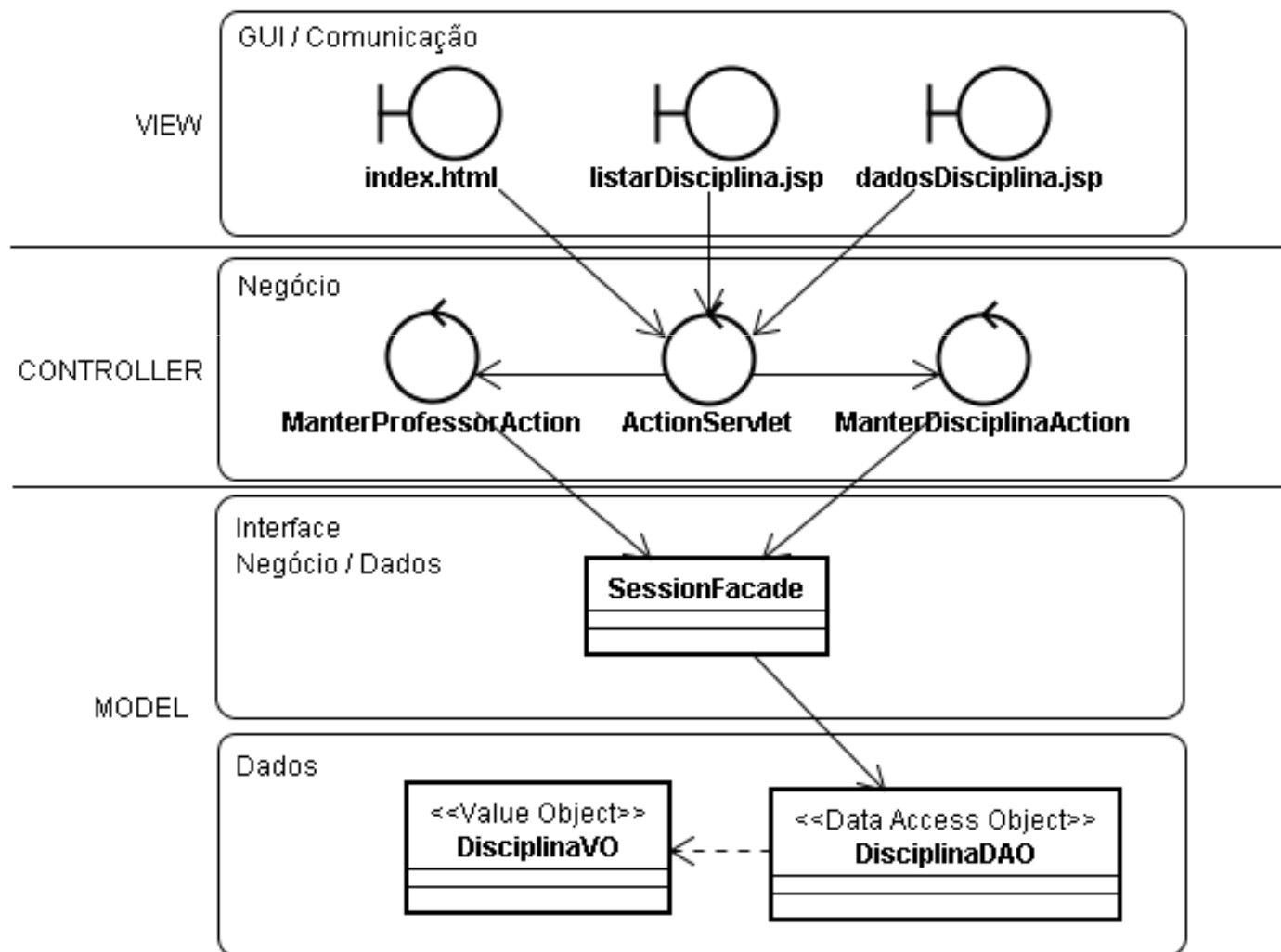
1. Definição da tecnologia;
2. Definição da arquitetura;
3. Definição do modelo de dados;
4. Criação da estrutura de armazenamento;
5. Criação do projeto WEB;
6. Implementação da camada MODEL;
7. Implementação da camada CONTROL;
8. Implementação da camada VIEW;



1. Definição da tecnologia

- Banco de dados: MySQL;
- Linguagem de programação: JAVA
- Padrão de Arquitetura: JEE com MVC;
- Model: Value Object, DAO, Hibernate 3
 - Annotations e Session Facade;
- View: JSP e HTML;
- Controller: Struts 1.1;

2. Definição da arquitetura – Padrão estudado

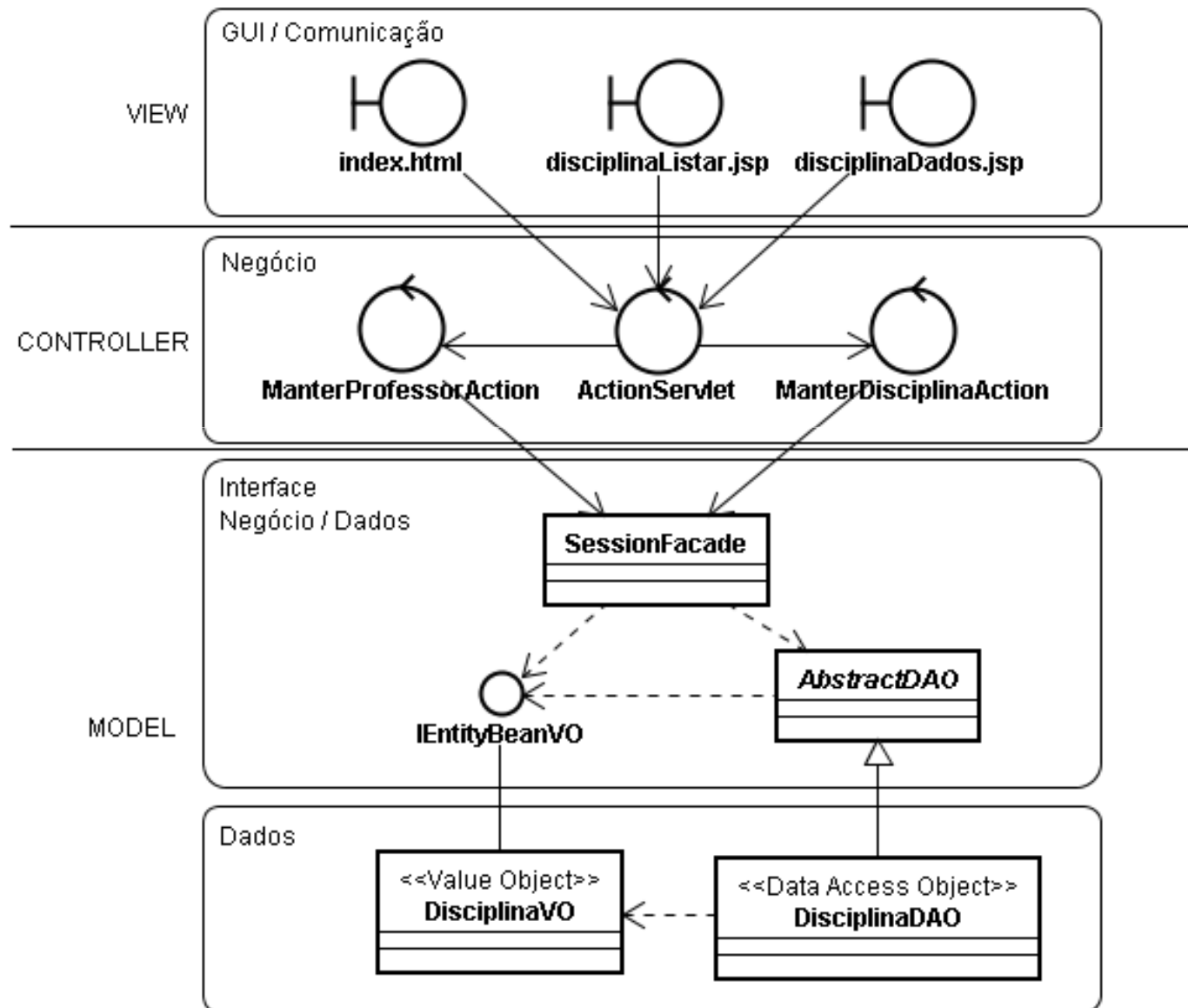
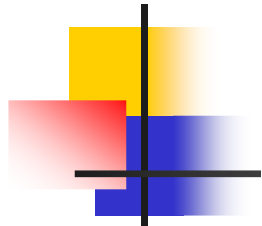




Itens a discutir em sala:

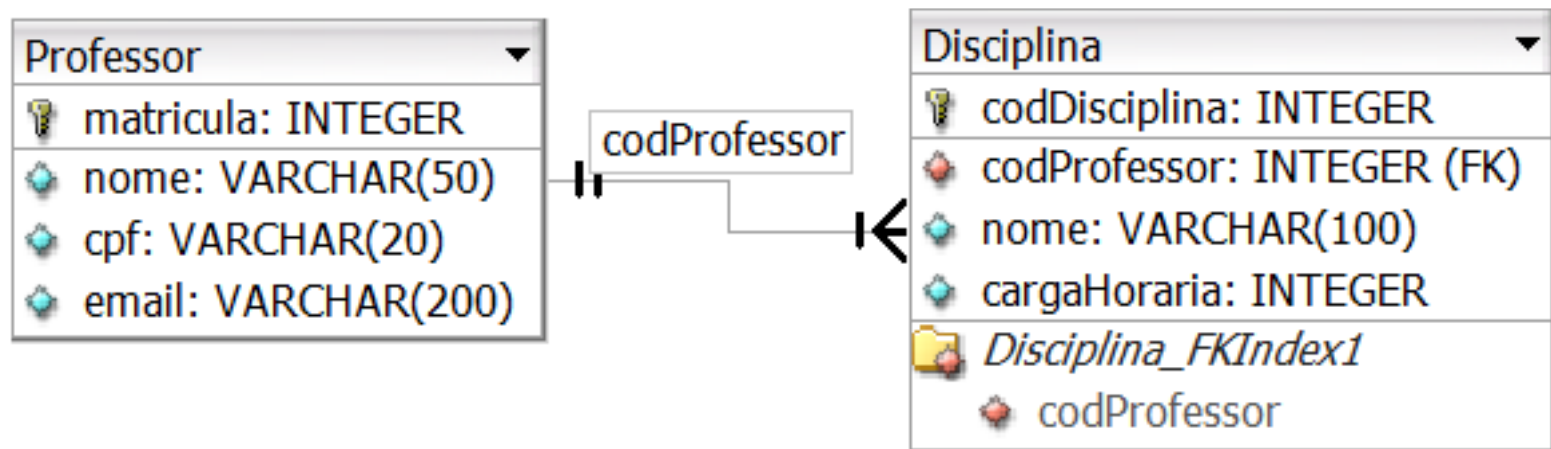
- O que acontecerá quando for implementado o módulo de professor?
- O que acontece na camada **model**?
- Como seria o código da classe ProfessorDAO?
- Podemos melhorar a arquitetura?
- Você lembra de: Interface, Classes Abstratas e Polimorfismo?

Arquitetura do projeto

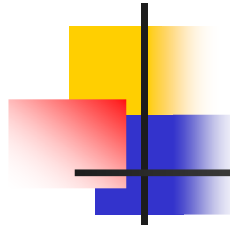


3. Modelo de dados

- Após o desenvolvimento da fase de Análise, foi identificado o seguinte MD:



4. Estrutura de armazenamento



```
Create database revisaojee;
use revisaojee;
CREATE TABLE Disciplina (
    codDisciplina INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
    codProfessor INTEGER UNSIGNED NOT NULL,
    nome VARCHAR(100) NULL,
    cargaHoraria INTEGER UNSIGNED NULL,
    PRIMARY KEY(codDisciplina),
    INDEX Disciplina_FKIndex1(codProfessor)
);
CREATE TABLE Professor (
    matricula INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
    nome VARCHAR(50) NULL,
    cpf VARCHAR(20) NULL,
    email VARCHAR(200) NULL,
    PRIMARY KEY(matricula)
);
```



Povoando a base de dados

Use revisaojee;

```
INSERT INTO `revisaojee`.`professor` (`matricula`, `nome`, `cpf`, `email`)
VALUES (NULL, 'Márcio Palheta', '33245512341',
'marcio.palheta@gmail.com');
```

```
INSERT INTO `revisaojee`.`professor` (`matricula`, `nome`, `cpf`, `email`)
VALUES (NULL, 'Maria Joaquina', '77788854312',
'maria.joaquina@qqcoisa.com.br');
```

```
INSERT INTO `revisaojee`.`disciplina` (`codDisciplina`, `codProfessor`,
`nome`, `cargaHoraria`) VALUES (NULL, '1', 'Linguagem de Programação
WEB', '30');
```

```
INSERT INTO `revisaojee`.`disciplina` (`codDisciplina`, `codProfessor`,
`nome`, `cargaHoraria`) VALUES (NULL, '1', 'Programação Orientada a
Objetos - Avançada', '30');
```

```
INSERT INTO `revisaojee`.`disciplina` (`codDisciplina`, `codProfessor`,
`nome`, `cargaHoraria`) VALUES (NULL, '2', 'Engenharia de software', '45');
```

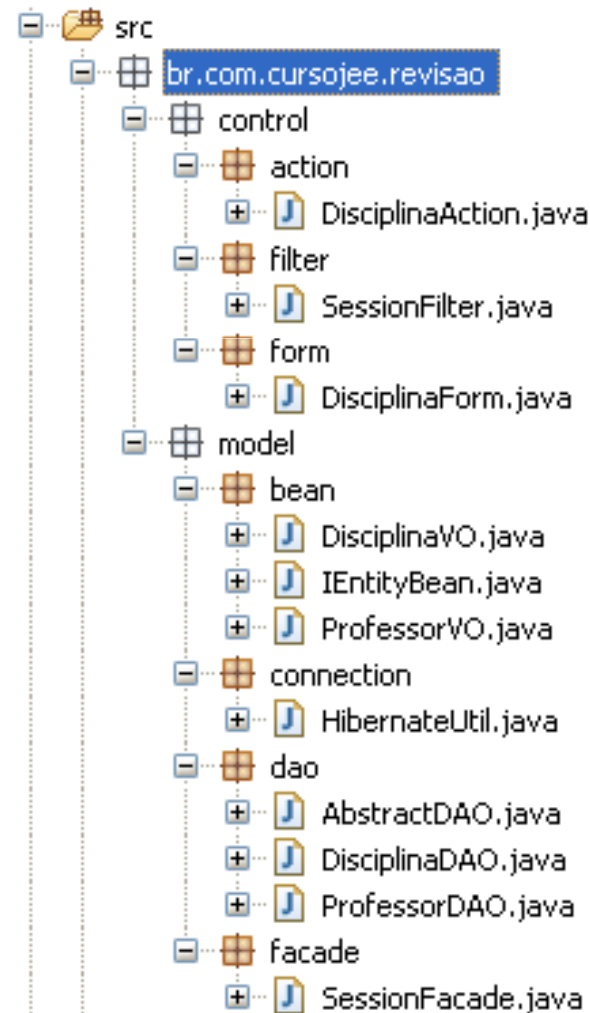
```
INSERT INTO `revisaojee`.`disciplina` (`codDisciplina`, `codProfessor`,
`nome`, `cargaHoraria`) VALUES (NULL, '1', 'Java Standard Edition - JSE',
'45');
```

5. Criação do projeto web no eclipse - revisaojee

- Após a criação, copie as libs necessárias



Visão da estrutura de pacotes do novo projeto **revisaojee**





6. Camada MODEL

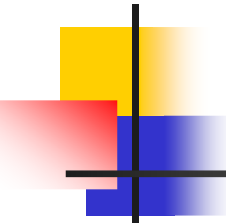
- IEntityBean
- ProfessorVO
- DisciplinaVO
- hibernate.properties
- HibernateUtil
- AbstractDAO
- DisciplinaDAO
- SessionFacade



IEntityBean

```
1 package br.com.cursojee.revisao.model.bean;
2
3 import java.io.Serializable;
4
5 //Interface padrão para classes VO
6 public interface IEntityBean
7     extends Serializable {
8 }
```

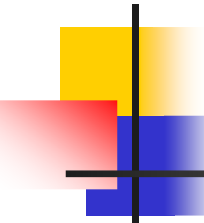

A Classe ProfessorVO



```
@Entity
@Table(name = "professor")
1
2
public class ProfessorVO implements IEntityBean {
3
    @Id
    @GeneratedValue
    private Long matricula;
4
    @Column(name="nome", nullable=false, length=50)
    private String nome;
    private String cpf;
    private String email;

    public Long getMatricula() {}
    public void setMatricula(Long matricula) {}
    public String getNome() {}
    public void setNome(String nome) {}
    public String getCpf() {}
    public void setCpf(String cpf) {}
    public String getEmail() {}
    public void setEmail(String email) {}
}
```

DisciplinaVO



```
1 package br.com.cursojee.revisao.model.bean;
2 import javax.persistence.Column;
9 @SuppressWarnings("serial")
10 @Entity
11 //Mapeamento da tabela
12 @Table(name="disciplina")
13 public class DisciplinaVO implements IEntityBean {
14     @Id
15     @GeneratedValue
16     private Long codDisciplina;
17     //Mapeamento da colua
18     @Column(name="nome", nullable=false, length=100)
19     private String nome;
20     private Integer cargaHoraria;
21     //Tipo de mapeamento
22     @ManyToOne
23     //Nome do campo na tabela Disciplina
24     @JoinColumn(name="codProfessor")
25     private ProfessorVO professor;
26     public Long getCodDisciplina() {}
29     public void setCodDisciplina(Long codDisciplina) {}
32     public String getNome() {}
35     public void setNome(String nome) {}
38     public Integer getCargaHoraria() {}
41     public void setCargaHoraria(Integer cargaHoraria) {}
44     public ProfessorVO getProfessor() {}
47     public void setProfessor(ProfessorVO professor) {}
50 }
```



src/hibernate.properties

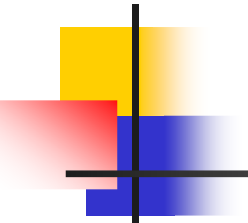
- #Configuracoes de acesso ao banco de dados
 - hibernate.dialect = org.hibernate.dialect.MySQLDialect
 - hibernate.connection.driver_class = com.mysql.jdbc.Driver
 - hibernate.connection.url = jdbc:mysql://localhost/revisaojee
 - hibernate.connection.username = root
 - hibernate.connection.password =
 - hibernate.show_sql = false
 - hibernate.format_sql = true
- #Controle de transacoes
 - hibernate.transaction.factory_class = org.hibernate.transaction.JDBCTransactionFactory
 - hibernate.current_session_context_class = thread
- #Configuração do pool de conexão
 - hibernate.c3p0.max_size=100
 - hibernate.c3p0.min_size=10
 - hibernate.c3p0.timeout=5000
 - hibernate.c3p0.max_statements=100
 - hibernate.c3p0.idle_test_period=300
 - hibernate.c3p0.acquire_increment=2



A classe HibernateUtil.java

```
public class HibernateUtil {  
    private static SessionFactory sessionFactory;  
    static{  
        //Criacao do objeto de configuracao  
        AnnotationConfiguration config =  
            new AnnotationConfiguration();  
        1 //Inclusao das classes mapeadas  
        config.addAnnotatedClass(ProfessorVO.class);  
        config.addAnnotatedClass(DisciplinaVO.class);  
        //Criacao do Sigleton  
        sessionFactory = config.buildSessionFactory();  
    }  
    public static Session getSessionFactory() {  
        2 //Retorna uma sessao do pool ou cria uma nova  
        return sessionFactory.getCurrentSession();  
    }  
}
```

Classe AbstractDAO



```
public abstract class AbstractDAO {  
    private Session session;  
    1 public AbstractDAO(Session session) {  
        this.session = session;  
    }  
    public void salvar(IEntityBean entityBean){  
        //Limpa o cache da sessão  
        session.clear();  
        2 //Cadastra ou Altera o entityBean  
        session.saveOrUpdate(entityBean);  
    }  
    public void excluir(IEntityBean entityBean){  
        session.delete(entityBean);  
    }  
    public IEntityBean consultar(Class classe, Long id){  
        IEntityBean entityBean = null;  
        //Realiza a consulta ao banco de dados  
        entityBean = (IEntityBean)session.load(classe, id);  
        return entityBean;  
    }  
    public List<IEntityBean> listar(Class classe){  
        List<IEntityBean> listaResultado = null;  
        Query query = null;  
        //Criação de um objeto de pesquisa - Query  
        query = session.createQuery("from "+classe.getName());  
        //Realiza a consulta ao banco de dados  
        listaResultado = query.list();  
        return listaResultado;  
    }  
}
```



Classe DisciplinaDAO

```
package br.com.cursojee.revisao.model.dao;

import java.util.Collection;

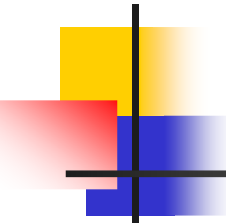
public class DisciplinaDAO extends AbstractDAO {

    public DisciplinaDAO(Session session) {
        //Chamada ao construtor da super classe;
        super(session);
    }

    public Collection<IEntityBean> listar(){
        //chamada ao metodo da super classe
        return super.listar(DisciplinaVO.class);
    }

    public DisciplinaVO consultar(Long id){
        //chamada ao metodo da super classe
        return (DisciplinaVO)
            super.consultar(DisciplinaVO.class, id);
    }
}
```

SessionFacade



```
public class SessionFacade {
    private DisciplinaDAO disciplinaDAO;
    private ProfessorDAO professorDAO;

    public void setSession(Session session) {
        disciplinaDAO = new DisciplinaDAO(session);
        professorDAO = new ProfessorDAO(session);
    }

    // Metodos de Disciplina *****
    public void salvarDisciplina(DisciplinaVO disciplina){
        disciplinaDAO.salvar(disciplina);
    }
    public void excluirDisciplina(DisciplinaVO disciplina){
        disciplinaDAO.excluir(disciplina);
    }
    public DisciplinaVO consultarDisciplina(Long id){
        return disciplinaDAO.consultar(id);
    }
    public List<IEntityBean> listarDisciplina(){
        return disciplinaDAO.listar();
    }

    // Metodos de Professor *****
    public List<IEntityBean> listarProfessor(){
        return professorDAO.listar();
    }
}
```



7. Camada de controle

- struts-config.xml
- DisciplinaForm.java
- DisciplinaAction.java

struts-config.xml



```
<struts-config>
  <form-beans>
    <form-bean name="DisciplinaForm"
      type="br.com.cursojee.revisao.control.form.DisciplinaForm"/>
  </form-beans>
  <action-mappings>
    <action input="disciplinaDados.jsp" name="DisciplinaForm"
      validate="true" path="/manterDisciplinas" parameter="method"
      type="br.com.cursojee.revisao.control.action.DisciplinaAction">
      <forward name="telaListagem" path="/disciplinaListar.jsp"/>
      <forward name="telaDados" path="/disciplinaDados.jsp"/>
      <forward name="telaSucesso" path="/sucesso.html"/>
    </action>

    <action path="/inicio" forward="/index.html"/>
  </action-mappings>

  <message-resources parameter="MessageResources" />

  <!-- ===== Validator plugin ===== -->
  <plug-in className="org.apache.struts.validator.ValidatorPlugIn">
    <set-property
      property="pathnames"
      value="/WEB-INF/validator-rules.xml,/WEB-INF/validation.xml"/>
  </plug-in>
</struts-config>
```



DisciplinaForm

```
1 package br.com.cursojee.revisao.control.form;
2
3+ import org.apache.struts.action.ActionForm;
4
5
6
7 @SuppressWarnings("serial")
8 public class DisciplinaForm extends ActionForm {
9     private DisciplinaVO disciplina = new DisciplinaVO();
10-    public DisciplinaVO getDisciplina() {
11        return disciplina;
12    }
13 }
```



DisciplinaAction

```
public class DisciplinaAction extends DispatchAction {

    private SessionFacade facade = new SessionFacade();

    public void startSession(HttpServletRequest request){
        Session session = (Session)request.getAttribute("session");
        facade.setSession(session);
    }

    public ActionForward listar(ActionMapping mapping,
        ActionForm form, HttpServletRequest request,
        HttpServletResponse response)
        throws Exception {
        startSession(request);
        List<IEntityBean> lista = facade.listarDisciplina();
        request.setAttribute("listaDisciplina", lista);
        return mapping.findForward("telaListagem");
    }

    //Outros metodos do caso de uso Manter Disciplina
}
```



8. Camada VIEW

- index.html
- disciplinaListar.jsp



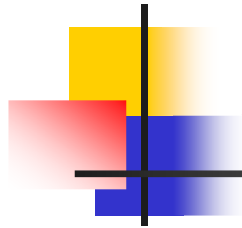
index.html

```
1 <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01
2     Transitional//EN"
3     "http://www.w3.org/TR/html4/loose.dtd">
4 <html>
5 <head>
6 <title>Página inicial</title>
7 </head>
8 <body>
9     <h1>Tela inicial</h1>
10    <a href="manterDisciplinas.do?method=listar">
11        Listar disciplinas</a>
12 </body>
13 </html>
```

disciplinaListar.jsp



```
1<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
2<%@ page language="java"
3    contentType="text/html; charset=ISO-8859-1"
4    pageEncoding="ISO-8859-1"%>
5<html>
6<head><title>Tela de listagem de disciplinas</title></head>
7<body>
8    <h1>Lista de disciplinas</h1>
9    <table border="1">
10        <tr>
11            <th>Código</th>
12            <th>Nome</th>
13            <th>CH</th>
14            <th>Professor</th>
15        </tr>
16        <c:forEach var="disciplina" items="${listaDisciplina}">
17            <tr>
18                <td>${disciplina.codDisciplina}</td>
19                <td>${disciplina.nome}</td>
20                <td>${disciplina.cargaHoraria}</td>
21                <td>${disciplina.professor.nome}</td>
22            </tr>
23        </c:forEach>
24    </table>
25    <br><a href="inicio.do">Página inicial</a>
26</body>
27</html>
```



Atividades em sala

- Criação e carga do banco de dados:
 - revisaojee
- Criação de um novo projeto web:
 - revisaojee_listagem
- Importação do projeto criado;
- Criação e alteração dos componentes necessários para **Listar Professores**;



Referências

- Hall, Marty, “Core Servlets and Java Server Pages”, Janeiro 2002, Sun Microsystems Press;
- <http://java.sun.com/>
- <http://java.sun.com/j2ee/1.6/docs/tutorial/doc/index.html>
- <http://java.sun.com/products/jndi/docs.html>
- <http://java.sun.com/blueprints/corej2eepatterns/Patterns/index.html>

Java Enterprise Edition - JEE

15. Modelo de arquitetura JEE



Esp. Márcio Palheta
gtalk: marcio.palheta@gmail.com