Aula 10 – Exemplo de aplicação MVC web com JSF

Objetivo da aula

Compreender o funcionamento de uma aplicação web em Java que utiliza JSF, por meio de um exemplo básico. O exemplo contempla o padrão MVC e o uso de páginas JSF para realizar o cadastro de dados de objetos do tipo Animal e posteriormente exibe as informações em outra página.

Acesse:

Como criar um projeto web no Netbeans:

- https://netbeans.org/kb/docs/web/jsf20-intro_pt_BR.html

Tutorial sobre JSF 2.0

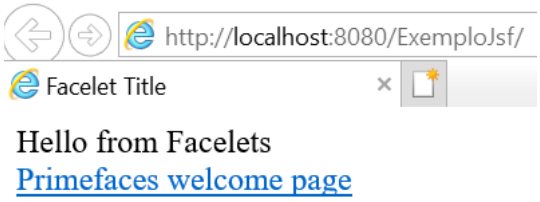
- http://www.mkyong.com/tutorials/jsf-2-0-tutorials/

Exibindo dados de objetos obtidos de determinada fonte organizados em uma tabela:

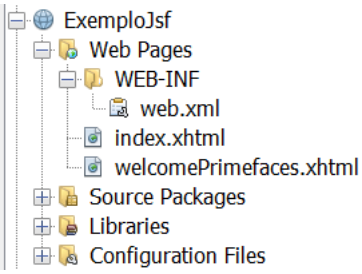
- http://www.mkyong.com/jsf2/jsf-2-0-jdbc-integration-example/

Instruções

1. Abra o Netbeans e crie um novo projeto web.
   - Arquivo (*File*)→ Novo Projeto (*New Project*)
     - Java Web → Web Application → Próximo (*Next*)
     - Nomeie o projeto como "ExemploJsf". Se desejar, mude o local onde o projeto será salvo → Próximo (*Next*)
     - Na janela sobre Servidores e Configurações (*Server and Settings*) deixe os parâmetros do jeito que aparecem → Próximo (*Next*)
     - Na janela Frameworks selecione o item **JavaServer Faces**
       - Na aba Componentes selecione **PrimeFaces** → Finalizar (*Finish*)
2. Ao executar o projeto, deverá ser carregado no navegador um resultado semelhante a figura a seguir.



3. Observe que o projeto web já possui uma estrutura contendo um arquivo web.xml com o mapeamento para o Faces Servlet, como mostra a figura seguinte.



4. Clique com o botão direito sobre "Pacotes de Códigos-fonte" ou "Source package" e crie um novo pacote (Novo → Pacote Java) com o seguinte padrão de nomenclatura: **`br.uem.din.petshop.model`**.
5. Dentro desse pacote, crie uma nova classe Java e nomeie-o como Animal.
6. Copie e cole o trecho de código em destaque:

```
package br.uem.din.petshop.model;

public class Animal {

    private String nome;
    private int idade;

    public Animal(String nome, int idade) {
        this.nome = nome;
        this.idade = idade;
    }

    public String getNome() {
        return nome;
    }

    public int getIdade() {
        return idade;
    }
}
```

7. Clique com o botão direito sobre "Pacotes de Códigos-fonte" ou "Source package" e crie um novo pacote (Novo → Pacote Java) com o seguinte padrão de nomenclatura: **`br.uem.din.petshop.controller`**.
8. Dentro desse pacote, crie uma nova classe Java com o nome de `CadastroController`.
   - A classe `CadastroController` será responsável por armazenar os dados de vários objetos do tipo animal em uma lista na memória.
     - Uma conexão com o banco de dados poderia ser criado nessa classe, com a chamada de métodos da classe responsável pela persistência de dados.
   - A classe `CadastroController` é um **Singleton**, que é um padrão de projeto no qual existe uma única instância de uma classe para toda a aplicação.
     - **Responda: Para que serve o padrão Singleton?**
   - Copie e cole a classe `CadastroController`.

```
package br.uem.din.petshop.controller;
```

```
import br.uem.din.petshop.model.Animal;
import java.util.ArrayList;
import java.util.List;

public class CadastroController {

    private List<Animal> animais;

    private static CadastroController instance;

    private CadastroController() {
        this.animais = new ArrayList<>();
    }

    public static CadastroController getInstance() {
        if (instance == null) {
            instance = new CadastroController();
        }
        return instance;
    }

    public void salvarAnimal(Animal animal) {
        this.animais.add(animal);
    }

    public List<Animal> listarAnimais() {
        return animais;
    }
}
```

9. Clique com o botão direito sobre "Pacotes de Códigos-fonte" ou "Source package" e crie um novo pacote (Novo → Pacote Java) com o seguinte padrão de nomenclatura: **`br.uem.din.petshop.bean`**.
10. Dentro desse pacote, crie uma nova classe JSF Managed Bean.
    - Clique com o botão direito sobre o pacote e acesse:
      - Outro (*Other*) → JavaServer Faces → JSF Managed Bean
      - **OU** Novo (*New*) → JSF Managed Bean
    - Nomeie a classe como **`AnimalCadastroBean`**.
    - Modifique o escopo da classe para session, na parte inferior da janela.
    - Finalizar (Finish).
11. Aparecerá uma classe bean com uma estrutura semelhante a seguinte:

```
package br.uem.din.petshop.bean;

import javax.inject.Named;
import javax.enterprise.context.SessionScoped;
import java.io.Serializable;

@Named(value = "animalCadastroBean")
@SessionScoped
```

```
public class AnimalCadastroBean implements Serializable {

      /**
       * Creates a new instance of AnimalCadastroBean
       */
      public AnimalCadastroBean() {
      }
}
```

12. A classe `AnimalCadastroBean` será responsável pelos tratamentos das requisições. Dessa forma, o próximo passo é colocar os atributos (propriedades) do bean, correspondente aos campos que aparecerão na tela e gerar os respectivos get /set. Essa classe também possuirá um método que cria um objeto a partir dos parâmetros do bean, chama o método do controlador para salvar esse objeto e retorna uma string que redireciona para uma página JSF para mostrar o resultado.

```
package br.uem.din.petshop.bean;

import br.uem.din.petshop.controller.CadastroController;
import br.uem.din.petshop.model.Animal;
import javax.inject.Named;
import javax.enterprise.context.SessionScoped;
import java.io.Serializable;

@Named(value = "animalCadastroBean")
@SessionScoped
public class AnimalCadastroBean implements Serializable {
    //referenciam os campos da tela
    //nao tem o objeto Animal para não causar a repeticao de referencia na lista
    private String nome;
    private int idade;

    /**
     * Creates a new instance of AnimalCadastroBean
     */
    public AnimalCadastroBean() {
    }

    public String getNome() {
        return nome;
    }

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

    public int getIdade() {
        return idade;
    }

    public void setIdade(int idade) {
        this.idade = idade;
    }

    public String cadastrar() {
        CadastroController.getInstance().salvarAnimal(new Animal(nome, idade));
        return "response";
    }

}
```

13. Dentro do pacote **`br.uem.din.petshop.bean`**, crie uma nova classe JSF Managed Bean e nomeie a classe como **`AnimalListagemBean`**.
    - Modifique o escopo da classe para session, na parte inferior da janela.
14. A classe `AnimalListagemBean` será responsável por mostrar todos os animais cadastrados na lista mantida pelo `CadastroController`.
15. Copie e cole o código da classe `AnimalListagemBean`:

```
package br.uem.din.petshop.bean;

import br.uem.din.petshop.controller.CadastroController;
import br.uem.din.petshop.model.Animal;
import javax.inject.Named;
import javax.enterprise.context.SessionScoped;
import java.io.Serializable;
import java.util.List;

@Named(value = "animalListagemBean")
@SessionScoped
public class AnimalListagemBean implements Serializable {

    /**
     * Creates a new instance of AnimalListagemBean
     */
    public AnimalListagemBean() {
    }

    public List<Animal> getAnimais() {
        return CadastroController.getInstance().listarAnimais();
    }

}
```

Na pasta WebPages:

16. Abra o arquivo index.xhtml e substitua pelo código a seguir.

```
<?xml version='1.0' encoding='UTF-8' ?>
<!DOCTYPE    html    PUBLIC    "-//W3C//DTD    XHTML    1.0    Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://xmlns.jcp.org/jsf/html">
    <h:head>
        <title>Exemplo de JSF</title>
    </h:head>
    <h:body>
        Cadastro de Animais
        <br />
        <br />
        <h:form>
            <h:outputText value="Nome" />
            <br />
                                        <h:inputText    id="nome"    title="Nome"
value="#{animalCadastroBean.nome}" />
            <br />
            <br />
            <h:outputText value="Idade" />
            <br />
                                      <h:inputText    id="idade"    title="Idade"
```

```
value="#{animalCadastroBean.idade}" />
            <br />
            <br />
                                <h:commandButton   id="submit"   value="Cadastrar"
action="#{animalCadastroBean.cadastrar}" />
        </h:form>
    </h:body>
</html>
```

17. Abra o arquivo welcomePrimefaces.xhtml e copie e cole o código a seguir.

```
<!DOCTYPE    HTML    PUBLIC    "-//W3C//DTD    HTML    4.01    Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://java.sun.com/jsf/html"
      xmlns:f="http://java.sun.com/jsf/core"
      xmlns:ui="http://java.sun.com/jsf/facelets">

    <f:view contentType="text/html">
        <h:head>
            <title>Cadastro de Animais</title>
        </h:head>
        <h:body>
            Animais Cadastrados
            <br />
            <br />
            <h:dataTable value="#{animalListagemBean.animais}" var="animal"
                         styleClass="order-table"
                         headerClass="order-table-header"
                         rowClasses="order-table-odd-row,order-table-even-row"
                         >
                <h:column>
                    <f:facet name="header">
                        Nome
                    </f:facet>
                    #{animal.nome}
                </h:column>

                <h:column>
                    <f:facet name="header">
                        Idade
                    </f:facet>
                    #{animal.idade}
                </h:column>

            </h:dataTable>
        </h:body>
    </f:view>
</html>
```

18. Renomeie o arquivo `welcomePrimefaces` para `response`.

19. Execute o projeto e observe o resultado.

Responda às seguintes questões:

1. O que é o padrão de projeto Singleton? Explique como ele está sendo aplicado no projeto dado como exemplo.
2. Explique como as requisições são tratadas, citando todas as classes do projeto.