

LPIII - Filtragem e Agrupamento em Abas de Páginas Dinâmicas

1 - Filtragem por Prefixo de Nome

Nesta seção você aprenderá a filtrar os diretores cadastrados acrescentando, na tabela de visualização de diretores, uma coluna associada a um filtro, no qual você poderá informar o prefixo do nome do diretor para restringir a pesquisa dos diretores cadastrados no banco de dados.

Na classe DiretorBean

- incluir variável : `List<Diretor> diretoresFiltrados`
 - e seu par de métodos get e set

Na página cadastrarDiretores

- no componente `dataTable`
 - incluir o atributo `filteredValue="#{diretorBean.diretoresFiltrados}"`
 - para armazenar os objetos filtrados na variável `diretoresFiltrados`
 - no componente `column` referente ao nome do diretor
 - incluir os atributos: `filterBy="nome"` e `footerText="startsWith"`

Seguem as ilustrações da página cadastrarDiretores antes e depois de utilizar o filtro com o prefixo do nome do diretor (letras maiúsculas ou minúsculas):

Clube de Amigos do Cinema

Página Inicial

Amigos

Cadastrar Amigos

Filmes

Cadastrar Diretores

Cadastrar Atores

Cadastrar Filmes

Comentários

Cadastrar Comentários

Cadastrar Diretores

Diretores Cadastrados

Nome	Ganhador de Oscar	Ação
James Cameron	true	<button>Consultar</button>
Steven Spielberg	true	<button>Consultar</button>
Christopher Nolan	false	<button>Consultar</button>
startsWith		

Cadastrar

Clube de Amigos do Cinema

Página Inicial

Amigos

Cadastrar Amigos

Filmes

Cadastrar Diretores

Cadastrar Atores

Cadastrar Filmes

Comentários

Cadastrar Comentários

Cadastrar Diretores

Diretores Cadastrados

Nome	Ganhador de Oscar	Ação
S Steven Spielberg	true	<button>Consultar</button>
startsWith		

Cadastrar

Após concluir com a entidade Diretor, você poderá acrescentar a entidade Ator, no projeto:

- copie a classe `Diretor` e cole no pacote `entities`, refatorando para a classe `Ator`
 - substitua o atributos (e seus métodos de leitura e alteração) da entidade `Diretor` pelos atributos da entidade `Ator`
 - `String nome`
 - `int totalOscars`
- copie a classe `DiretorConverter`, do pacote `converters`, refatorando para a classe `AtorConverter`
 - realizando as adaptações necessárias em seu código
- copie a classe `DiretorService`, do pacote `services`, renomeando para a classe `AtorService`
 - realizando as adaptações necessárias em seu código
- copie a classe `DiretorBean`, do pacote `beans`, renomeando para a classe `AtorBean`
 - realizando as adaptações necessárias em seu código
- copie a página `cadastrarDiretores`, e refatore renomeando para a classe `cadastrarAtores`
 - realizando as adaptações necessárias em seu código
- na página `template`
 - no componente `menuitem` `value="Cadastrar Atores" url="/index.jsf"`
 - substitua a página `index` por `cadastrarAtores`

Seguem as ilustrações da página `cadastrarAtores` antes e depois de utilizar o filtro com o prefixo do nome do ator (letras maiúsculas ou minúsculas):

Clube de Amigos do Cinema

[Página Inicial](#)
[Amigos](#)
[Cadastrar Amigos](#)
[Filmes](#)
[Cadastrar Diretores](#)
[Cadastrar Atores](#)
[Cadastrar Filmes](#)
[Comentários](#)
[Cadastrar Comentários](#)

Cadastrar Atores

Atores Cadastrados

Nome	Ganhador de Oscar	Ação
Leonardo DiCaprio	1	Consultar
Kate Winslet	1	Consultar
Sam Worthington	0	Consultar
Zoe Saldanha	0	Consultar
Liam Neeson	0	Consultar
Ralph Fiennes	0	Consultar
startsWith		
Cadastrar		

Clube de Amigos do Cinema

[Página Inicial](#)
[Amigos](#)
[Cadastrar Amigos](#)
[Filmes](#)
[Cadastrar Diretores](#)
[Cadastrar Atores](#)
[Cadastrar Filmes](#)
[Comentários](#)
[Cadastrar Comentários](#)

Cadastrar Atores

Atores Cadastrados

Nome	Ganhador de Oscar	Ação
Leonardo DiCaprio	1	Consultar
Liam Neeson	0	Consultar
startsWith		
Cadastrar		

2 - Formulários Agrupados em Abas e Filtragem por Seleção de Valor

Nesta sessão, será criada a entidade Amigo, com suas classes do modelo objeto-relacional e página de cadastro. A definição dessa entidade será utilizada para ilustrar: (a) a criação um formulário de cadastro com abas; (b) a definição de um enumerado como uma classe interna de uma classe entidade; (c) a criação de filtros baseados em seleção de valor; e (d) anotações utilizadas para caracterizar variáveis associadas a enumerado, data de nascimento e expressões regulares (para checar o formato de entrada de uma informação).

No Clube de Amigos do Cinema, cada amigo é cadastrado com os dados de interesse para a troca de informações sobre filmes. Como dados pessoais são cadastrados: nome, apelido, sexo, estado civil, data de nascimento e cidade. Como dados de contato são cadastrados: email, celular utilizado no Whatsapp, identificador no Facebook, identificador do Instagram. A data de nascimento poderá ser utilizada para você parabenizar seu amigo. O prefixo do nome e do apelido são utilizados como filtros para localizar seu amigo. Adicionalmente, é possível filtrar os amigos por sexo, estado civil e cidade que reside. É interessante a possibilidade de verificar se o sexo, estado civil e características da cidade de residência (tais como: tamanho da cidade ou região do país a que pertence) influenciam, em média, no teor dos comentários dessas pessoas sobre filmes.

Da mesma forma, como foi feito com a entidade [Ator](#), copiar a entidade [Diretor](#) e colar reformatando seu nome para [Amigo](#). Seguem as importações que foram utilizadas para construir o código da classe [Amigo](#).

```
package entities;

import java.io.Serializable;
import java.util.Date;
import javax.persistence.Entity;
import javax.persistence.EnumType;
import javax.persistence.Enumerated;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.Temporal;
import javax.persistence.TemporalType;
import javax.validation.constraints.Pattern;

@Entity
public class Amigo implements Serializable, PersistentEntity {
```

Criar o enumerado `EstadoCivil` como uma classe interna da entidade `Amigo` caracterizada pela palavra chave `enum`:

- definindo os valores do enumerado e os strings que serão mostrados para cada um desses valores
- definindo uma variável `label` que será inicializada no construtor
- e associando à variável `label` um método de leitura

```
@Entity
public class Amigo implements Serializable, PersistentEntity {

    public enum EstadoCivil {
        solteiro("solteiro"), casado("casado"), divorciado("divorciado"), viúvo("viúvo");

        private final String label;

        private EstadoCivil(String label) { this.label = label; }
        public String getLabel() { return this.label; }
    };
};
```

Substitua os atributos da classe que você copiou pelos atributos da classe `Amigo`:

- denote o atributo `estadoCivil` como um enumerado associado a String
 - `@Enumerated(EnumType.STRING)`
- defina o atributo `dataNascimento` com a classe `java.util.Date`
 - e denote como data : `@Temporal(TemporalType.DATE)`
- restrinja o formato a ser informado para os atributos `email` e `whatsapp` (celular utilizado no Whatsapp) para combinar com expressões regulares
 - `@Pattern(regex = "expressão regular")`
- defina os métodos de leitura e escrita de cada um desses atributos

```
private String nome;
private String apelido;
private String sexo;
@Enumerated(EnumType.STRING)
private EstadoCivil estadoCivil;
@Temporal(TemporalType.DATE)
private Date dataNascimento;
private String cidade;

@Pattern(regexp = "\\w+([-+.'\\w+)*@\\w+([-.'\\w+)*\\.\\w+([-.'\\w+)*")
private String email;
@Pattern(regexp = "((\\d{2}\\d{2}) ?| (\\d{2}-))?(\\d{5}|\\d{4})-\\d{4}")
private String whatsapp;
private String facebook;
private String instagram;
```

Seguindo o mesmo procedimento adotado para a entidade `Ator`, criar as classes: `AmigoConverter`, `AmigoService`, `AmigoBean` e página `cadastrarAmigos`.

Na classe `AmigoBean`, criar o método que utiliza os valores do enumerado `EstadoCivil` para gerar um array de `SelectItem`, para ser utilizado na definição do filtro da tabela de visualização de amigos e para preenchimento do valor do atributo `estadoCivil` no formulário de cadastro. Segue o código do método `getEstadosCivis`:

- o parâmetro `filtrar`
 - `true` : indica que o array retornado será utilizado no filtro
 - no qual a primeira opção deve estar em branco
 - para sinalizar que o filtro não está sendo utilizado
 - `false` : indica que o array retornado será utilizado para preencher o formulário
 - na qual só devem constar os strings correspondentes aos valores do enumerado
 - porque o campo precisa ser preenchido

```
public SelectItem[] getEstadosCivis(boolean filtrar) {
    SelectItem[] items;
    int length = Amigo.EstadoCivil.values().length;
    int n = 0;
    if (filtrar) {
        items = new SelectItem[length + 1];
        items[0] = new SelectItem("", "");
        n++;
    } else items = new SelectItem[length];
    for (Amigo.EstadoCivil estado_civil : Amigo.EstadoCivil.values()) {
        items[n++] = new SelectItem(estado_civil, estado_civil.getLabel());
    }
    return items;
}
```

Adaptar o método acima para o método `getSexos`. Neste caso, os valores de preenchimento do `SelectItem` não se originam de um enumerado; são valores fixos: os pares `"feminino"` e `"Feminino"` e idem para `masculino`.

Na classe `AmigoBean`, criar o método `onFlowProcess`

- que será utilizado no formulário de cadastro com abas

```
public String onFlowProcess(FlowEvent event) { return event.getNewStep(); }
```

Na página `cadastrarAmigos`:

- adaptar as colunas da tabela para visualizar os atributos da entidade `Amigo`: `nome`, `apelido`, `sexo`, `estadoCivil`, `cidade`
- utilizar os filtros com prefixo de nome, conforme visto na seção 1 deste tutorial, para os seguintes atributos da entidade `Amigo`: `nome`, `apelido`, `cidade`
- utilizar os filtros com seleção de valores, conforme código abaixo (ilustrado para o atributo `estadoCivil`), para os seguintes atributos da entidade `Amigo`: `sexo`, `estadoCivil`

```
<p:column filterBy="estadoCivil"
           headerText="Estado Civil" footerText="exact"
           filterOptions="#{amigoBean.getEstadosCivis(true)}"
           filterMatchMode="exact">
    <h:outputText value="#{amigo.estadoCivil}" />
</p:column>
```

Adaptar o formulário da página [cadastrarAmigos](#):

- agrupado em abas utilizando o componente [wizard](#) com duas abas
 - [Dados Pessoais](#) e [Contatos](#)

```
<p:panel header="Dados do Amigo" id="dadosAmigo" widgetVar="panelAmigo"
    visible="false" closable="true" style="margin-top:10px;">
    <p:messages id="erroAmigo"/>
    <h:panelGrid id="displayAmigo" columns="1" styleClass="grid">
        <p:panel>
            <p:wizard
                flowListener="#{amigoBean.onFlowProcess}"
                backLabel="Anterior"
                nextLabel="Próximo">
                <p:tab id="dadosPessoais" title="1. Dados Pessoais">
                    <p:panel header="Dados Pessoais">
                        <h:panelGrid columns="3" columnClasses="label, value" styleClass="grid">
```

- utilizar [selectOneMenu](#) para o preenchimento do atributo [estadoCivil](#)

```
<p:selectOneMenu id="estadoCivilSelectOneMenu"
    value="#{amigoBean.value.estadoCivil}"
    required="false" label="Estado Civil" style="width:100px">
    <f:selectItems value="#{amigoBean.getEstadosCivis(false)}" />
</p:selectOneMenu>
```

- utilizar [selectOneRadio](#) para o preenchimento do atributo [sexo](#), conforme código acima, omitindo o parâmetro [style](#)
- utilizar o componente [inputMask](#) para o preenchimento do atributo [dataNascimento](#), conforme código abaixo

```
<p:inputMask id="dataNascimentoInputMask" required="false"
    label="Data de Nascimento" mask="99/99/9999"
    value="#{amigoBean.value.dataNascimento}">
    <f:convertDateTime pattern="DD/MM/YYYY" locale="pt_BR"/>
</p:inputMask>
```

Na página [template](#):

- no componente [menuitem](#) [value](#)="Cadastrar Amigos" [url](#)="/index.jsf"
 - substitua a página [index](#) por [cadastrarAmigos](#)

Linguagem de Programação III - Tutorial 3 : Filtragem e Agrupamento em Abas de Páginas Dinâmicas - 7/8

Segue a página cadastrarAmigos, deslocada na vertical para mostrar a primeira aba do formulário de cadastro:

The screenshot shows a web application interface. On the left is a sidebar with a menu containing: 'Página Inicial', 'Amigos' (highlighted), 'Cadastrar Amigos', 'Filmes', 'Cadastrar Diretores', 'Cadastrar Atores', 'Cadastrar Filmes', 'Comentários', and 'Cadastrar Comentários'. The main content area has a header with a table of tabs: 'startsWith', 'startsWith', 'exact', 'exact', 'startsWith'. Below this is a green 'Inserir' button. The main section is titled 'Dados do Amigo' and contains two tabs: '1. Dados Pessoais' (active) and '2. Contatos'. The 'Dados Pessoais' tab contains a form with the following fields: 'Nome Completo: *' (Adriana Andrade), 'Apelido: *' (Dri), 'Sexo: *' (radio buttons for 'Feminino' and 'Masculino', with 'Feminino' selected), 'Estado Civil: *' (dropdown menu showing 'solteiro'), 'Data de Nascimento: *' (14/10/2000), and 'Cidade: *' (Dourados). At the bottom of the form is a green button with a right arrow and the text 'Próximo'. Below the form are two buttons: a green checkmark icon followed by 'Alterar', and a trash can icon followed by 'Remover'.

Segue a página cadastrarAmigos, deslocada na vertical para mostrar a segunda aba do formulário de cadastro:

The screenshot shows the same web application interface, but the '2. Contatos' tab is now active. Above the form, a table displays two rows of friend data. The first row is for 'Adriana Andrade' with details: 'Dri', 'feminino', 'solteiro', 'Dourados', and a green 'Consultar' button. The second row is for 'Carlos de Campos Filho' with details: 'Caco', 'masculino', 'casado', 'Itaporã', and a green 'Consultar' button. Below this table is a green 'Inserir' button. The 'Dados do Amigo' section now shows the '2. Contatos' tab active, with a form containing: 'Email: *' (adrianaandrade@gmail.com), 'Whatsapp: *' (67-91111-1111), 'Facebook: *' (driface), and 'Instagram: *' (driinstag). At the bottom of the form is a green button with a left arrow and the text 'Anterior'.

Segue a página cadastrarAmigos, com dois amigos cadastrados:

Página Inicial

Amigos

Cadastrar Amigos

Filmes

Cadastrar Diretores

Cadastrar Atores

Cadastrar Filmes

Comentários

Cadastrar Comentários

Clube de Amigos do Cinema

Cadastrar Amigos

Amigos Cadastrados

Nome	Apelido	Sexo	Estado Civil	Cidade	Ação
<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	
Adriana Andrade	Dri	feminino	solteiro	Dourados	<input type="button" value="Consultar"/>
Carlos de Campos Filho	Caco	masculino	casado	Itaporã	<input type="button" value="Consultar"/>
startsWith	startsWith	exact	exact	startsWith	
<input type="button" value="Inserir"/>					

Segue a página cadastrarAmigos, ilustrando a utilização do filtro **Sexo**:

Página Inicial

Amigos

Cadastrar Amigos

Filmes

Cadastrar Diretores

Cadastrar Atores

Cadastrar Filmes

Comentários

Cadastrar Comentários

Clube de Amigos do Cinema

Cadastrar Amigos

Amigos Cadastrados

Nome	Apelido	Sexo	Estado Civil	Cidade	Ação
<input type="text"/>	<input type="text"/>	Feminino	<input type="text"/>	<input type="text"/>	
Adriana Andrade	Dri	feminino	solteiro	Dourados	<input type="button" value="Consultar"/>
startsWith	startsWith	exact	exact	startsWith	
<input type="button" value="Inserir"/>					

Segue a página cadastrarAmigos, ilustrando a utilização do filtro **Estado Civil**:

Página Inicial

Amigos

Cadastrar Amigos

Filmes

Cadastrar Diretores

Cadastrar Atores

Cadastrar Filmes

Comentários

Cadastrar Comentários

Clube de Amigos do Cinema

Cadastrar Amigos

Amigos Cadastrados

Nome	Apelido	Sexo	Estado Civil	Cidade	Ação
<input type="text"/>	<input type="text"/>	<input type="text"/>	casado	<input type="text"/>	
Carlos de Campos Filho	Caco	masculino	casado	Itaporã	<input type="button" value="Consultar"/>
startsWith	startsWith	exact	exact	startsWith	
<input type="button" value="Inserir"/>					