



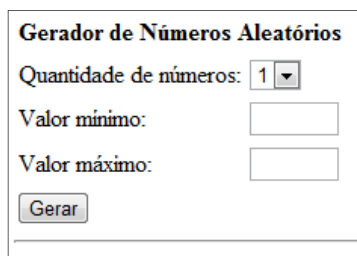
Java EE Web Total

Exercícios Propostos

Tags e Componentes do JSF

1 Exercício

Crie um sistema de geração de números aleatórios. A tela para entrada dos dados deve ser semelhante à tela abaixo:



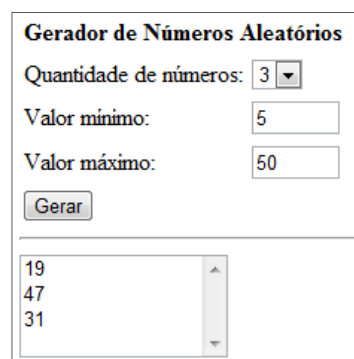
Gerador de Números Aleatórios

Quantidade de números: 1 ▼

Valor mínimo:

Valor máximo:

O usuário deve escolher uma quantidade de números a serem gerados (pode variar de 1 a 5) e também os valores mínimo e máximo (devem ser números inteiros). Ao clicar no botão *Gerar*, o sistema gera os números de acordo com os critérios e apresenta a lista de números dentro de um componente `h:selectManyListbox`, conforme a figura abaixo:



Gerador de Números Aleatórios

Quantidade de números: 3 ▼

Valor mínimo: 5

Valor máximo: 50

19
47
31

2 Exercício

Crie um sistema de cadastro de dados de um usuário. O sistema deve ter três telas de preenchimento de dados, onde o usuário deve, sequencialmente, fornecer seus dados pessoais, dados comerciais e indicar suas áreas de interesse pessoal. Após finalizar o preenchimento de cada tela, ele deve clicar nos botões *Próximo* ou *Anterior* para navegar entre as telas. Os dados preenchidos em uma tela anterior devem ser reexibidos caso a tela seja aberta novamente.

Ao final da terceira tela, existirá um botão *Finalizar*. Ao clicar neste botão, o usuário deverá ser levado à outra tela que exibe todas as informações preenchidas nas telas anteriores.

Veja abaixo exemplos de como as telas do sistema podem ser criadas:

Dados Pessoais	
Nome:	<input type="text" value="José Silva"/>
E-mail:	<input type="text" value="jose@email.com"/>
Data de nascimento:	<input type="text" value="10/12/1977"/>
Rua:	<input type="text" value="R. das Jabuticabas"/>
Número:	<input type="text" value="302"/>
Complemento:	<input type="text" value="Ap. 300"/>
Bairro:	<input type="text" value="Indaial"/>
CEP:	<input type="text" value="82091-000"/>
Cidade:	<input type="text" value="Rio de Janeiro"/>
Estado:	<input type="text" value="RJ - Rio de Janeiro"/>
Telefone residencial:	<input type="text" value="21"/> <input type="text" value="5555 0398"/>
Telefone celular:	<input type="text" value="21"/> <input type="text" value="5555 9090"/>
<input type="button" value="Próximo >>"/>	

Dados Comerciais	
Nome da Empresa:	<input type="text" value="Choque Materiais Elétrico"/>
Cargo:	<input type="text" value="Gerente"/>
Salário:	<input type="text" value="3500"/>
Rua:	<input type="text" value="Rua das Amoras"/>
Número:	<input type="text" value="958"/>
Complemento:	<input type="text" value="Cj. 100"/>
Bairro:	<input type="text" value="Santa Bernadete"/>
CEP:	<input type="text" value="48592-990"/>
Cidade:	<input type="text" value="Curitiba"/>
Estado:	<input type="text" value="PR - Paraná"/>
Telefone comercial:	<input type="text" value="41"/> <input type="text" value="5555 4890"/>
<input type="button" value="Anterior <<"/> <input type="button" value="Próximo >>"/>	

Áreas de Interesse	
<input type="checkbox"/>	Esportes
<input checked="" type="checkbox"/>	Música
<input type="checkbox"/>	Artes Marciais
<input checked="" type="checkbox"/>	Viagens
<input checked="" type="checkbox"/>	Cinema
<input type="checkbox"/>	Dança
<input type="button" value="Anterior <<"/> <input type="button" value="Finalizar"/>	

Dados Pessoais		Dados Comerciais		Áreas de Interesse
Nome:	José Silva	Nome da Empresa:	Choque Materiais Elétricos	
E-mail:	jose@email.com	Cargo:	Gerente	
Data de nascimento:	10/12/1977	Salário:	R\$ 3.500,00	
Rua:	R. das Jabuticabas	Rua:	Rua das Amoras	
Número:	302	Número:	958	
Complemento:	Ap. 300	Complemento:	Cj. 100	
Bairro:	Indaial	Bairro:	Santa Bernadete	
CEP:	82091-000	CEP:	48592-990	
Cidade:	Rio de Janeiro	Cidade:	Curitiba	
Estado:	RJ	Estado:	PR	
Telefone residencial:	(21) 5555 0398	Telefone comercial:	(41) 5555 4890	
Telefone celular:	(21) 5555 9090			

Áreas de Interesse

☐ Esportes

☒ Música

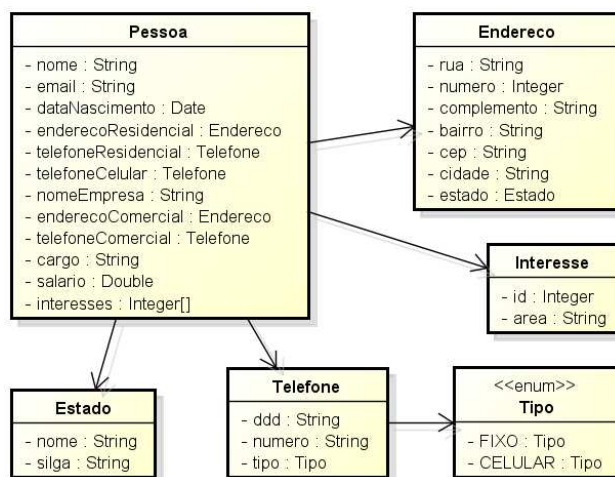
☐ Artes Marciais

☒ Viagens

☒ Cinema

☐ Dança

Segue abaixo uma sugestão de classes para armazenar as informações do cadastro (os métodos getters e setters foram omitidos para simplificar o diagrama).



Dica: Para o bean que será usado em conjunto com as telas, utilize o escopo de sessão (@SessionScoped). Esta annotation garante que o bean vai existir enquanto a janela do navegador estiver ativa, permitindo que você armazene os dados de todas as telas do fluxo de cadastro sem perder as informações.

3 Exercício

Crie um sistema simples de cadastro de livros, que permita a inserção, alteração e exclusão de registros. Os livros devem ser listados em um componente `h:dataTable`, e as três operações citadas anteriormente devem ser efetuadas diretamente na tabela. Veja um exemplo na figura abaixo:

Lista de Livros					
#	Título	Autor	Editora	Núm. Páginas	
1	Java para Web	José Silva	Livro & Cia	256	Gravar
2	Java Server Faces	Maria Oliveira	Novo Livro	344	Alterar Excluir
3					Gravar

Inserir

A linha 1 corresponde a um registro em modo de edição; a linha 2 é um registro existente, que pode ser alterado ou excluído; e a linha 3 corresponde a um novo registro sendo inserido.

Os dados devem ficar armazenados dentro do bean (não é necessário gravá-los em arquivos ou em um banco de dados).

Dica: Para o bean que será usado em conjunto com a tela, utilize o escopo de sessão (@SessionScoped). Esta annotation garante que o bean vai existir enquanto a janela do navegador estiver ativa e vai manter dentro do bean os livros que você já cadastrou.

4 Exercício

Crie um sistema que simula pagamentos via cartão de crédito. A tela de entrada de dados deve ser semelhante à figura abaixo:

Bandeira:	<input type="text" value="Selecione"/>
Número do cartão:	<input type="text"/>
Nome (como consta no cartão de crédito):	<input type="text"/>
Data de validade (mm/aaaa):	<input type="text"/>
Valor da compra (R\$):	<input type="text"/>
<input type="button" value="Processar"/>	

Para trabalhar com os conversores do JSF, considere o seguinte:

- A bandeira é um enum com as opções VISA, MASTERCARD, AMEX e DINERS. O valor selecionado deve ser mapeado para um atributo no bean cujo tipo é do enum criado.
- A data de validade deve ser mapeada para um objeto do tipo Date no bean.
- O valor da compra deve ser mapeado para um objeto do tipo Double no bean.
- O número de cartão e o nome são mapeados para o tipo String.

Caso ocorram erros de conversão, as mensagens devem aparecer ao lado do componente onde o erro foi originado. As mensagens exibidas devem ser lidas de um arquivo de recursos próprio da aplicação, que sobrescreve as mensagens relevantes do arquivo de recursos Messages_xx_XX.properties do JSF.

5 Exercício

Utilize o mesmo sistema criado no *Exercício 4* e adicione algumas regras de validação ao formulário de pagamento. As regras são as seguintes:

- Todos os campos são de preenchimento obrigatório.
- O número do cartão deve ser composto por 16 dígitos (**Dica:** esta validação pode ser facilmente realizada por uma expressão regular).
- O nome pode ter, no máximo, 50 caracteres.
- A data de validade do cartão não pode ser anterior à data atual (*mm/aaaa*). **Dica:** um validador customizado pode ser utilizado para fazer esta validação
- A valor da compra não pode ser um número negativo.

Assim como no *Exercício 4*, caso ocorram erros de validação, as mensagens devem aparecer ao lado do componente onde o erro foi originado. As mensagens exibidas devem ser lidas de um arquivo de recursos próprio da aplicação, que sobrescreve as mensagens relevantes do arquivo de recursos `Messages_xx_XX.properties` do JSF.

Desafio: Tente fazer com que o validador customizado também leia a mensagem a partir do arquivo de recursos que você criou para a aplicação. Para isso, pesquise sobre os métodos `getApplication()` da classe `FacesContext` e também `getMessageBundle()` da classe `Application`.