Módulo 8

No módulo anterior discutimos sobre o padrão MVC e um pouco mais sobre arquitetura em camadas. Refatoramos nosso projeto, agora ele é *web*! Construímos elementos da camada de apresentação e conectamos a tela à lógica da aplicação. Listamos e matriculamos alunos.

Neste módulo – o último – exploraremos a manipulação das mensagens, algumas funcionalidades da extensão JSF e facilidades para criar aplicações *web* de forma rápida. Com isto finalizaremos o curso básico do Demoiselle.

FacesMessage

Muitas vezes precisamos enviar mensagens para o usuário. No caso de aplicações JSF, esta funcionalidade está disponível na classe *javax.faces.context.FacesContext*.

1. Abra a classe ***TurmaMB***.
2. Acrescente o atributo *@Inject private FacesContext facesContext*.
3. No método ***matricular*** lance a mensagem “Cadastro realizado com sucesso”.
4. Abra o arquivo ***turma.xhtml*** e inclua a tag *<h:messages />* antes do formulário.

A classe ***TurmaMB*** vai ficar assim:

```
...

@Inject
private FacesContext facesContext;

...

public void matricular() {
	bc.matricular(new Aluno(nomeAluno));
	facesContext.addMessage("sucesso", new FacesMessage("Cadastro realizado com sucesso"));
}
...
```

E ***turma.xhtml*** assim:

```
<html xmlns:f="http://java.sun.com/jsf/core" xmlns:h="http://java.sun.com/jsf/html">
	<h:messages />

	<h:form>
		Nome do aluno
		<h:inputText id="nomeAluno" value="#{turmaMB.nomeAluno}"/>
		<h:commandButton value="Matricular" action="#{turmaMB.matricular}" />
	</h:form>
	<h:body>
		<h:dataTable id="lista" var="aluno" value="#{turmaMB.alunosMatriculados}">
			<h:column>
				<h:outputText value="#{aluno.matricula}" />
			</h:column>
			<h:column>
				<h:outputText value="#{aluno.nome}" />
			</h:column>
		</h:dataTable>
	</h:body>
</html>
```

Como a injeção do *FacesContext* não é suportada nativamente pelo JSF, o Demoiselle disponibiliza este recurso na extensão *demoiselle-jsf*. Dê uma olhada no *pom.xml*, você não encontrará nenhuma referência à extensão. Tudo foi definido no *demoiselle-jsf-parent* e herdado pelo POM da nossa aplicação.

Acesse http://localhost:8080/inscricao/turma.jsf, matricule um aluno e veja a mensagem na tela.

Confira o vídeo de demonstração clicando no link abaixo:

http://www.frameworkdemoiselle.gov.br/documentacaodoprojeto/manuais-e-tutoriais/tutorial-da-versao-2-2-3-0/videos/modulo-8-video-1

Mensagem

Supondo que precisássemos disparar mensagens da camada de negócio, e não mais da apresentação, seria considerado um desvio grave de arquitetura se referenciássemos o *FacesContext* – um elemento da apresentação – na classe *TurmaBC*. Para suprir esta necessidade, o Demoiselle disponibilizou a interface *MessageContext*.

1. Abra a classe ***TurmaBC***.
2. Acrescente o atributo *@Inject private MessageContext messageContext*.
3. No método ***matricular***, lance para o usuário a mesma mensagem gerada no *log*.
4. Remova o lançamento da mensagem na classe ***TurmaMB***.

A classe ***TurmaBC*** vai ficar assim:

```
…

@Inject
private MessageContext messageContext;

@Transactional
@RequiredPermission(resource = "turma", operation = "matricular")
public void matricular(Aluno aluno) {
    …

    String mensagem = bundle.getString("matricula.sucesso", aluno.getNome());
    logger.info(mensagem);
    messageContext.add(mensagem);
}

...
```

Como *br.gov.frameworkdemoiselle.message.MessageContext* independe de elementos das camadas, a arquitetura da aplicação será preservada. As mensagens enfileiradas no *MessageContext* serão convertidas automaticamente em *FacesMessage* pela extensão JSF.

Acesse http://localhost:8080/inscricao/turma.jsf, matricule um aluno e veja a mensagem na tela.

Confira o vídeo de demonstração clicando no link abaixo:

http://www.frameworkdemoiselle.gov.br/documentacaodoprojeto/manuais-e-tutoriais/tutorial-da-versao-2-2-3-0/videos/modulo-8-video-2

Exceções

Outra responsabilidade da extensão JSF é tratar as exceções da aplicação e convertê-las em mensagens.

1. Abra a classe ***TurmaException***.
2. Sobrescreva o construtor padrão com *super("Erro na matrícula!")*.

A classe ***TurmaException*** ficará assim:

```
@ApplicationException(rollback = true)
public class TurmaException extends RuntimeException {

	public TurmaException() {
		super("Erro na matrícula!");
	}
}
```

Como *TurmaException* é uma exceção de aplicação, pois está anotada com *@ApplicationException*, a extensão JSF fará um tratamento especial. Acesse http://localhost:8080/inscricao/turma.jsf, exceda o limite da turma e veja o que ocorre.

Terminamos por aqui os experimentos com nossa aplicação. Sinta-se à vontade para refatorá-la, discriminar as falhas (duplicação ou de limite excedido), exibir mensagens mais intuitivas para o usuário, formatar melhor as telas, etc.

Confira o vídeo de demonstração clicando no link abaixo:

http://www.frameworkdemoiselle.gov.br/documentacaodoprojeto/manuais-e-tutoriais/tutorial-da-versao-2-2-3-0/videos/modulo-8-video-3

Arquétipo

No decorrer dos módulos criamos incrementalmente uma aplicação onde experimentamos diversas funcionalidades do Framework. No módulo anterior convertemos nosso projeto para a plataforma *web* aderente às boas práticas de arquitetura em camada e ao padrão MVC. Não foi um caminho curto, é verdade. Por isto, vamos pegar um atalho com o arquétipo *demoiselle-jsf-jpa*:

1. No Eclipse, acesse *File / New / Project*.
2. Agora vá em *Maven / Maven project* e clique em *Next* duas vezes.
3. Escolha o Catalog *Demoiselle*.
4. Na listagem, escolha o *demoiselle-jsf-jpa* e clique em *Next*.
5. Defina o Group Id *br.gov.serpro*, o Artifact Id *bookmark* e clique em *Finish*.
6. Aguarde o Maven criar o seu novo projeto.

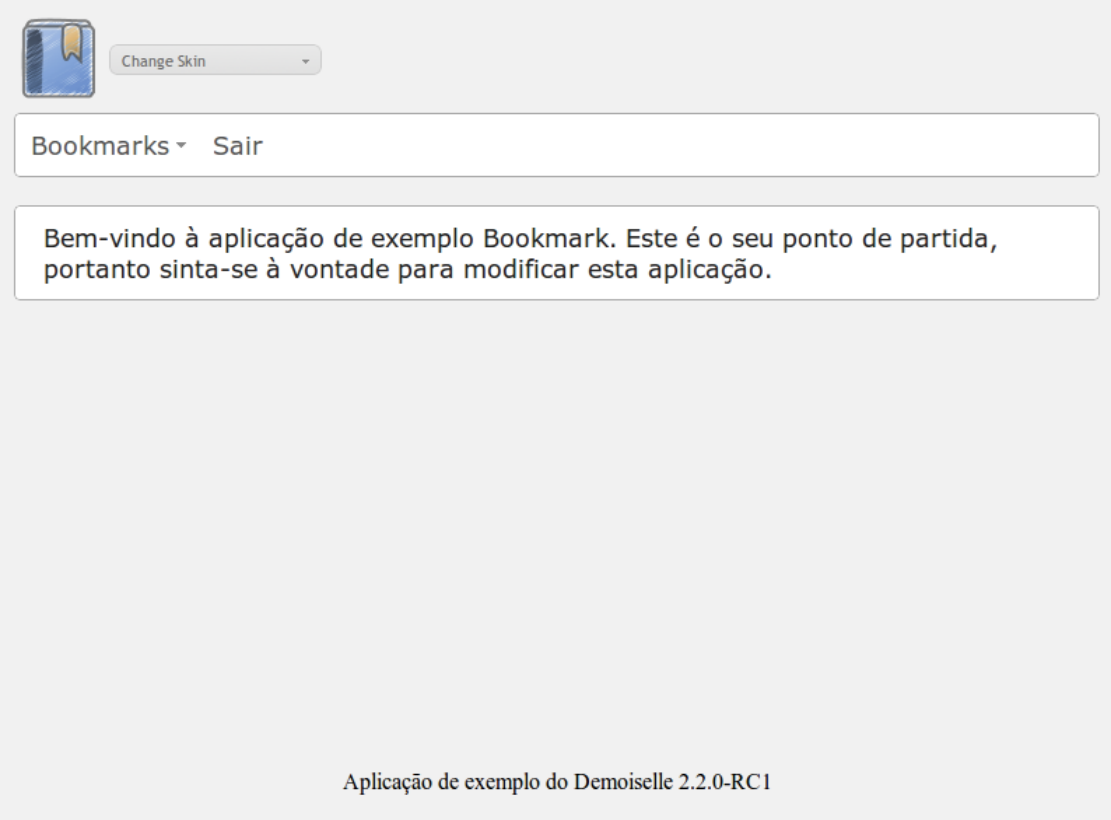
Praticamente tudo o que aprendemos ao longo do curso está consolidado neste novo projeto. Ao criar projetos com o arquétipo *demoiselle-jsf-jpa*, você leva de brinde um exemplo de CRUD já implementado. Trata-se do cadastro básico de *links* favoritos, similar ao *bookmark* dos navegadores *web*. Os artefatos gerados servirão como guia para a criação de novas telas, classes e configurações.

Para executar a aplicação gerada pelo arquétipo, faça isso:

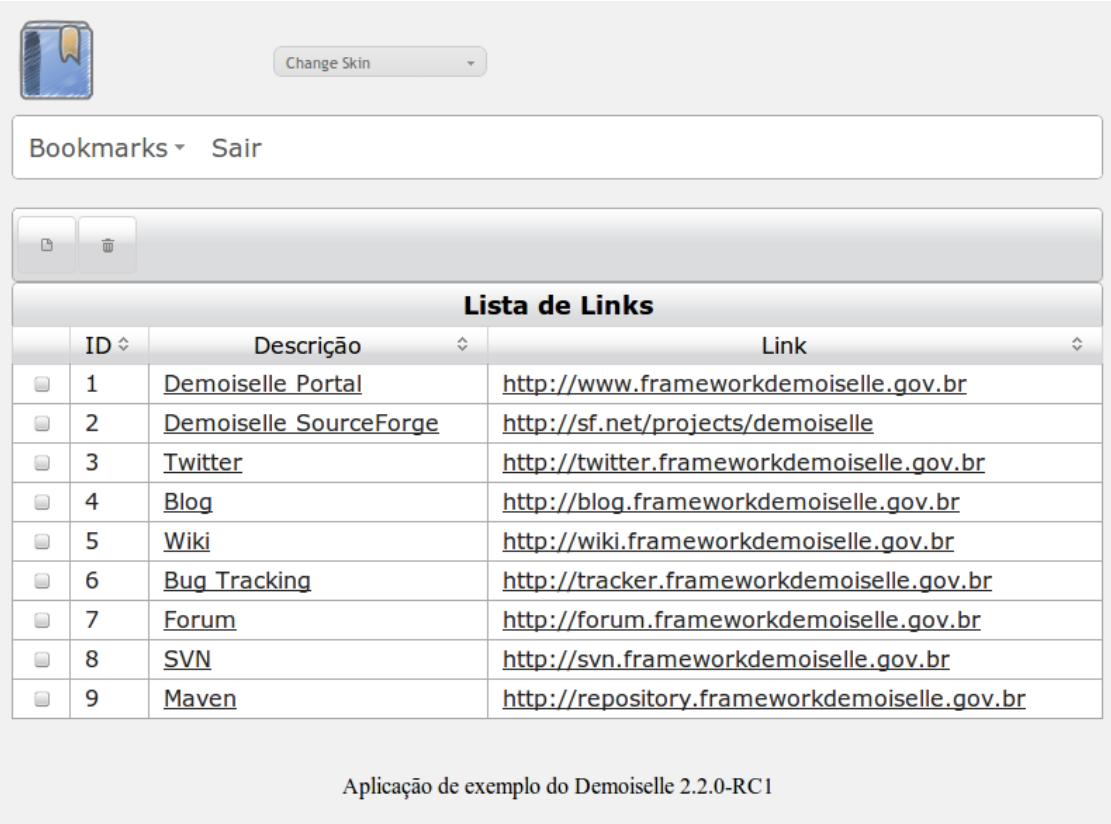
1. Arraste o projeto para o elemento **JBoss** da aba *Servers*;
2. Clique com o botão direito em *JBoss* e selecione **Start**;
3. Aguarde o servidor iniciar;

4. Acesse o endereço *http://localhost:8080/bookmark*.

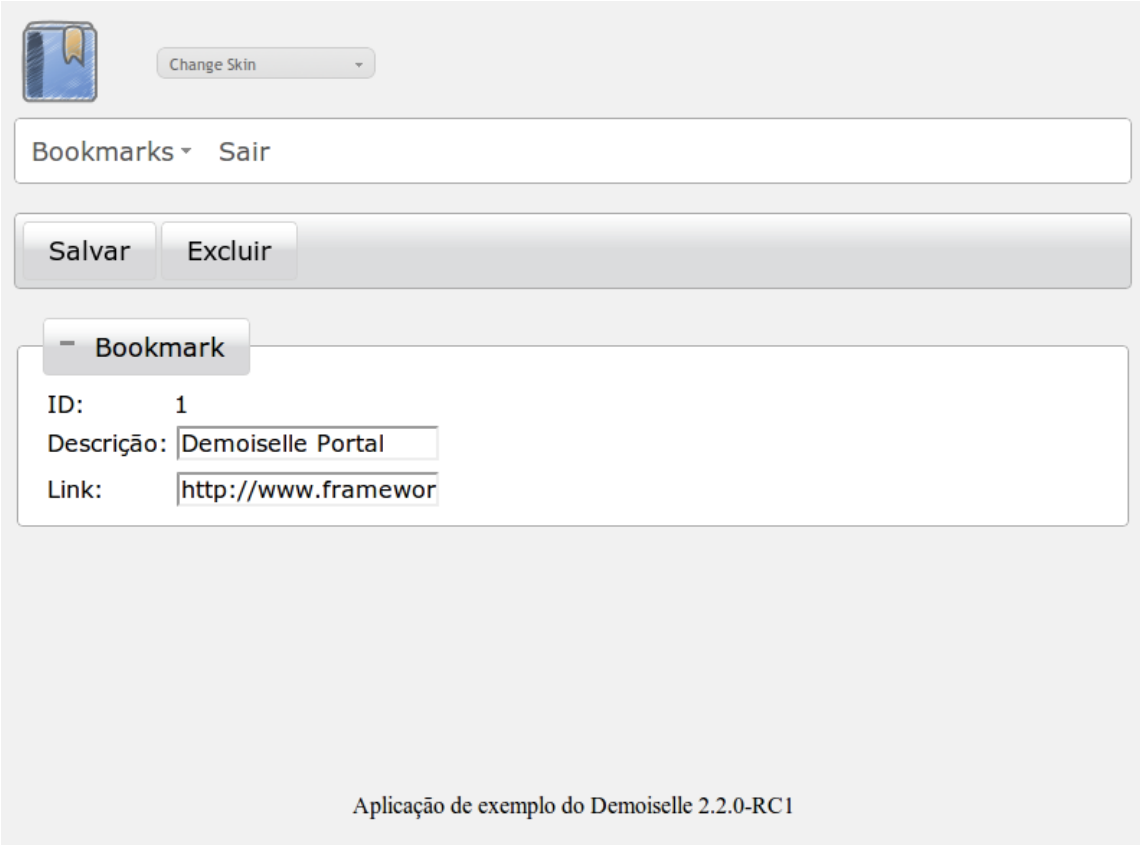
A aplicação é composta por uma tela inicial, listagem, edição e cadastro de novos de links:



Tela inicial



Tela de listagem



Tela de cadastro/edição

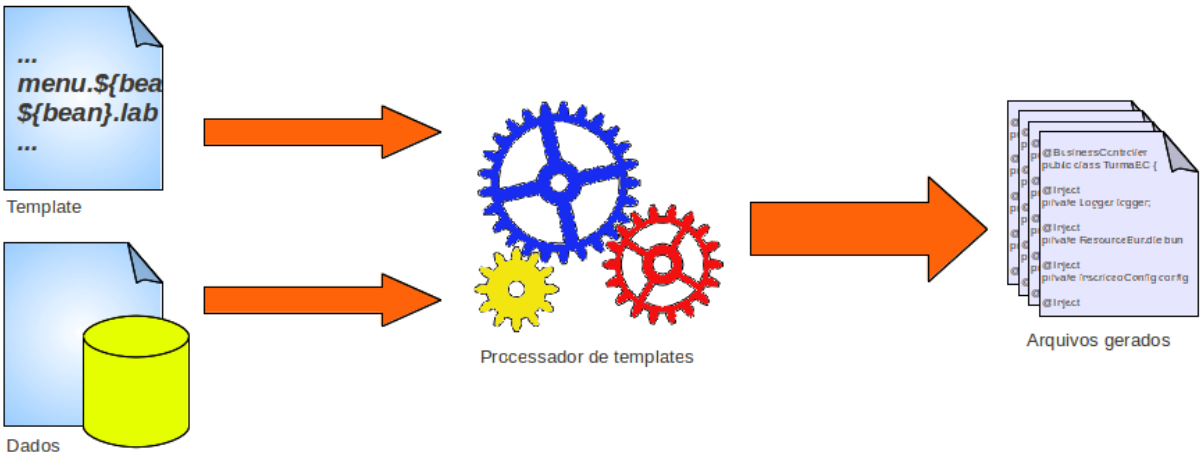
Tomando como base o conhecimento adquirido neste curso, analise todo o código-fonte e arquivos de configuração da aplicação. Você será capaz de entender!

Confira o vídeo de demonstração clicando no link abaixo:

http://www.frameworkdemoiselle.gov.br/documentacaodoprojeto/manuais-e-tutoriais/tutorial-da-versao-2-2-3-0/videos/modulo-8-video-4

Nimble

Durante o desenvolvimento de um projeto nos deparamos com tarefas repetitivas, como criar CRUD por exemplo. Nestes casos o que fazemos é “copiar e colar” algo existente e adaptá-lo. É um trabalho chato, cansativo e suscetível a erros. Para facilitar as coisas, nasceu o Nimble: é um processador automatizado que gera páginas, classes, arquivos de configuração, etc, com base em *templates* facilmente adaptáveis.



Nos templates são definidos estruturas de pastas, arquivos e scripts de transformação utilizando linguagens como Velocity, Groovy ou FreeMarker. A transformação pode ser parametrizada com variáveis previamente definidas. O Nimble provê alguns *templates*, que você pode tomar como base para criar o seu.

Para saber mais sobre o Demoiselle Nimble acesse http://demoiselle.sourceforge.net/tools/nimble/. Lá você aprenderá como instalar e utilizar esta poderosa ferramenta.

Nos links abaixo demonstramos alguns recursos:

Criando uma aplicação WEB com JPA e JSF (Primefaces):
http://www.frameworkdemoiselle.gov.br/documentacaodoprojeto/manuais-e-tutoriais/tutorial-da-versao-2-2-3-0/videos/modulo-8-video-5

Criando uma aplicação WEB com JPA e JSF para Mobile (Primefaces):
http://www.frameworkdemoiselle.gov.br/documentacaodoprojeto/manuais-e-tutoriais/tutorial-da-versao-2-2-3-0/videos/modulo-8-video-6

Testando a aplicação WEB JSF para Mobile com Android no VirtualBox:
http://www.frameworkdemoiselle.gov.br/documentacaodoprojeto/manuais-e-tutoriais/tutorial-da-versao-2-2-3-0/videos/modulo-8-video-7

Quero mais

O Demoiselle não termina por aqui. Agora que você conhece o básico, que tal aprofundar nos detalhes? Segue algumas indicações para aprimorar o conhecimento:

- **Guia de Referência do Framework**. Aprenda detalhes sobre cada uma das funcionalidades do Framework (Core + Extensions).
- **Guia de Referência dos Componentes**. Quer conhecer melhor os componentes do Demoiselle? Este é o material mais adequado.
- **Blog do Demoiselle.** Leia artigos técnicos sem compromisso com a formalidade que tratam assuntos bem específicos. Você tem experiências para compartilhar? Seu artigo pode parar aqui!

- Vídeos no YouTube. Assista aos vídeos práticos e teóricos sobre o Demoiselle.
- Aplicação de exemplo. Baixe e aprenda com o código-fonte da aplicação Contactlist que explora diversos recursos avançados.
- Lista de Usuários. Quer tirar dúvidas e ajudar outras pessoas? Cadastre-se na lista de usuários!
- Bug Tracker. Descobriu algum erro? Tem sugestões de melhoria? Registre um chamado e torne o projeto cada vez melhor.
- Portal Institucional. Fique por dentro das notícias sobre o Demoiselle.

Retrospectiva

Neste módulo experimentamos a funcionalidade de manipulação de mensagens do Framework. Vimos diversas facilidades que a extensão JSF oferece. Criamos uma aplicação *web* rapidamente utilizando o arquétipo demoiselle-jsf-jpa. Conhecemos o gerador de código baseado em *templates*, o Demoiselle Nimble. Durante o curso experimentamos diversos recursos que o Demoiselle oferece para facilitar o desenvolvimento de aplicações JEE 6. Não pare por aqui e explore os *links* indicados na sessão “Quero mais”.

Aguardamos você na lista de usuários com dúvidas e colaborações para tornar o projeto cada vez melhor!