

EDSON GONÇALVES

DESENVOLVENDO APLICAÇÕES WEB COM NETBEANS IDE 6



CM EDITORA
CIÊNCIA MODERNA

AGRADECIMENTOS

Primeiramente gostaria de agradecer os inúmeros e-mails de leitores elogiando a primeira edição deste livro e também os que criticaram, pedindo mais detalhes a respeito de determinados assuntos.

Também agradeço o apoio dado pela Editora Ciência Moderna para o desenvolvimento desta segunda edição.

Um agradecimento especial ao apoio do JUG Leader e Founding Java Champion, Daniel deOliveira, do DFJUG.

INTRODUÇÃO

Mais uma versão e, novamente, o NetBeans amadurece e surpreende com novas características que tornam o desenvolvimento mais agradável e ágil.

A versão 6 desta fantástica IDE não poderia ser diferente de seus lançamentos anteriores, onde vemos que a intenção é tornar a ferramenta uma plataforma de desenvolvimento para várias linguagens.

Esta nova versão, além de aproveitar melhor as facilidades incorporadas nas especificações Java EE 5, conta também com as características que vem mudando o mundo do desenvolvimento Web, como o framework Rails da linguagem Ruby. Mais ainda, é possível, através do uso de JRuby, rodar aplicações Ruby on Rails (RoR) sobre uma Java Virtual Machine, agregando as facilidades do desenvolvimento com RoR, aliados a estabilidade e integração com sistemas Java rodando sobre os servidores de aplicações. Além do Ruby, a linguagem PHP, tida como plugin adicional na versão 6.0, também ganhou seu lugar na IDE definitivamente na versão 6.1, ainda em desenvolvimento no momento em que este livro é escrito.

Com um número maior de linguagens e frameworks suportados, o desenvolvedor pode contar com assistentes que se integram tanto para o desenvolvimento de aplicações Java, como também na integração com o poderoso framework Rails (RoR) para o mesmo princípio.

Outra novidade é a incorporação do editor visual ao NetBeans IDE 6.x, tido antes como um pacote separado, para geração de aplicações Web que utilizam o framework JavaServer Faces. Seu nome foi rebatizado para Visual Web JavaServer Faces Applications e é mais uma das diversas facilidades que o desenvolvedor Java conta na construção de aplicações Web.

Para este livro, nesta nova edição, cerca de 50% do material foi reescrito e ampliado. O leitor agora tem em suas mãos informações que vão desde a construção de aplicações Web básicas, contendo páginas JSP e Servlets, como também o foco na utilização de JavaServer Faces, incluindo Facelets. O uso de JPA (Java Persistence API) e EJB 3 foram adicionados, tendo em vista os leitores que estão aproveitando as facilidades que ambos incorporam no desenvolvimento de aplicações Web.

O uso de Ruby on Rails foi adicionado, incluindo o JRuby on Rails para programadores Java, focado na versão 2.0.2 do framework. Neste caso, não só um CRUD é feito através da IDE, mas também há muitas explicações sobre sua arquitetura e um exemplo com relacionamento.

A grande novidade neste livro está no aprimoramento dos capítulos sobre Visual Web JavaServer Faces. Para aqueles que desejam trabalhar com esta ferramenta, fora elaborado em dois capítulos um aplicativo completo, utilizando seus principais componentes com acesso a dados. E para aqueles que desejam utilizar JPA e Hibernate com Visual Web JavaServer Faces, um capítulo especial fora dedicado a este assunto, incluindo o uso de Spring.

Por fim, há no CD-ROM como brinde, diversos Capítulos Extras que contém o trabalho com Struts, criando um CRUD completo, a utilização do novíssimo plugin iReport for NetBeans, integrando-se a ferramenta e facilitando a criação de relatórios JasperReports e dois estudos de caso completos, envolvendo o Visual Web JSF, incluindo o uso de DAO genérico, injeção de dependências com Spring Framework e a JPA com o Hibernate como provider.

QUEM DEVE LER ESTE LIVRO?

Este livro foi escrito para desenvolvedores com pouca ou nenhuma experiência na utilização do NetBeans IDE. Embora sejam apresentados alguns conceitos sobre as tecnologias Java, para a criação de aplicações Web, é de suma importância que o leitor tenha conhecimento de lógica e da estrutura da linguagem Java. O mesmo vale para os desenvolvedores que desejam trabalhar com aplicações Ruby on Rails, ao qual é necessário um prévio conhecimento de Ruby, facilitando assim sua compreensão.

É desejável também um conhecimento sobre o desenvolvimento Web com a linguagem Java, tais como páginas JSP ou Servlets, assim como acesso e utilização de um banco de dados.

ANTES DE COMEÇAR

Em algumas partes desse livro você encontrará um símbolo, que o ajudará a entender o código proposto e desenvolvido, mostrado a seguir:

... - Indica que acima ou abaixo contém mais código, mas que não está sendo exibido por não ter sido alterado e que o mesmo pode ser acessado pelo CD-ROM, em anexo ao livro.

OS SOFTWARES REQUERIDOS

Os aplicativos criados nesse livro não exigem software proprietário. Portanto ao longo dos capítulos você não só aprenderá como usá-los como também onde encontrá-los na Internet, caso precise de uma atualização.

Esse livro não está focado especialmente em um sistema operacional, portanto a sua escolha é livre nesse ponto.

HARDWARE REQUERIDO

Uma boa configuração de hardware se faz necessário para trabalhar com aplicações escritas em Java. Um computador para rodar bem o NetBeans na versão 6.0 deve ter as seguintes configurações para uma confortável utilização, segundo o autor:

Processador: Pentium 4 ou similar (recomendo um Dual Core ou Core 2 Duo)

Memória: 1 GB de RAM mínimo (recomendo 2GB de RAM)

HD: 10GB de espaço livre

Monitor: 17 polegadas ou superior

Alguns testes foram executados em Pentium 4 com 1 GB de memória, no qual houve certa lentidão na inicialização da IDE, mas não a inviabilidade de seu uso.

O maior problema na utilização da IDE com relação à exigência do Hardware está no trabalho com o Visual Web JavaServer Faces e com servidores de aplicações como o Glassfish.

Para a criação deste livro, um Core 2 Duo com 4 MB de cache e 2GB de RAM foi utilizado.

CÓDIGOS DOS EXEMPLOS CRIADOS NO LIVRO

Todos os códigos dos exemplos criados no livro, bem como a IDE e outros, se encontram no CD-ROM anexo.

Caso não encontre algum exemplo, entre em contato com o autor pelo site <http://www.integrator.com.br>.

VISÃO GERAL DOS CAPÍTULOS

Embora este livro esteja completamente focado no NetBeans IDE, ainda assim você terá ao longo do livro, muitos códigos para desenvolver. Todos os detalhes, em sua maioria, se encontram na íntegra, para que sejam digitados por você mesmo. Em todo caso, dúvidas poderão surgir, o que pode requerer a visão do arquivo em geral. Para este caso, o CD-ROM em anexo possui o projeto com seu nome proposto em livro para ser analisado.

ATENÇÃO: Em caso de erro, é recomendado a visualização dos exemplos contidos no **CD-ROM** anexo ao livro, antes de entrar em contato com o autor.

Com um conteúdo completamente ilustrado, o livro possui diversas imagens, espalhadas por todos os capítulos e sempre com foco em detalhes quando necessário. Em seu longo, dicas são dadas para uma melhor produtividade do que está sendo feito, aproveitando melhor os recursos que a IDE tem a oferecer.

PARTE 1: INTRODUÇÃO: DESENVOLVIMENTO DE APLICAÇÕES WEB COM JAVA

CAPÍTULO 1: OBTENDO E INSTALANDO O NetBEANS IDE 6 – Como obter e instalar a NetBeans IDE na versão 6.x.

CAPÍTULO 2: SERVIDORES DE APLICAÇÕES E SERVLETS – Visão geral, uso e aprofundamento do NetBeans IDE com o desenvolvimento de aplicações Web escritas em Java, utilizando servidores de aplicações, monitoramento e distribuição para produção.

CAPÍTULO 3: DESENVOLVENDO PÁGINAS DINÂMICAS NO NETBEANS - Dedicado ao trabalho com páginas dinâmicas usando Java, envolvendo JSP, JSTL, Custom Tags, as configuração da sua aplicação e o Deployment Descriptor usando o NetBeans IDE.

CAPÍTULO 4: TRABALHANDO COM BANCO DE DADOS – Desenvolvido para o contato inicial com o banco de dados, usando JDBC, em aplicações Web através Servlets e páginas JSP, utilizando NetBeans IDE. Os padrões de desenvolvimento MVC e DAO são apresentados, integrando o JSP e JSTL com acesso a dados através do MySQL.

PARTE 2: JAVA EE5: AVANÇANDO NO DESENVOLVIMENTO DE APLICAÇÕES WEB

CAPÍTULO 5: JAVASERVER FACES – Visão geral e técnica do framework JavaServer Faces trabalhado através do NetBeans IDE, com configurações e acesso a banco de dados e Web 2.0 com Facelets.

CAPÍTULO 6: EJB 3 E JAVA PERSISTENCE API – Apresenta o desenvolvimento de aplicações Enterprise utilizando o NetBeans.

CAPÍTULO 7: O VISUAL WEB JAVASERVER FACES – A primeira parte de um estudo de caso detalhado, ensinando o uso do Visual Web JSF através do desenvolvimento de uma aplicação, focando na etapa visual sem acesso a banco de dados.

CAPÍTULO 8: DESENVOLVENDO NO VISUAL WEB JSF COM BANCO DE DADOS – Continuação do estudo de caso usando Visual Web JSF com acesso a dados, incluindo uma área administrativa.

CAPÍTULO 9: TRABALHANDO COM WEB SERVICES NO NETBEANS IDE – Desenvolve e consome Web Services usando o NetBeans IDE, incluindo a integração com EJB 3, acesso a dados e o uso de Visual Web JavaServer Faces.

CAPÍTULO 10: VISUAL WEB JSF COM JPA, SPRING E HIBERNATE – Finaliza o trabalho com Visual Web JSF integrando um CRUD com Spring 2.5 e Hibernate 3, através do uso de Java Persistence API (JPA).

PARTE 3: DESENVOLVIMENTO COM LINGUAGENS DINÂMICAS E AJAX

CAPÍTULO 11: RAILS 2 COM NETBEANS IDE – Cria um estudo de caso usando o NetBeans como ferramenta para desenvolver aplicações Ruby on Rails.

CAPÍTULO 12: JRUBY ON RAILS – Recria o projeto do Capítulo 11, adicionando as características individuais do JRuby, que roda sobre a Java Virtual Machine, incluindo acesso a dados via JDBC e deploy no Application Server GlassFish.

CAPÍTULO 13: TRABALHANDO COM AJAX NO NETBEANS IDE – Utiliza o NetBeans para trabalhar com AJAX através de plug-ins, integrando frameworks conhecidos como jMaki, GWT e ICEfaces.

APÊNDICE A: Ruby para desenvolvedores Java – Explica de forma comparativa a linguagem Ruby com Java para um suave entendimento.

No CD-ROM

CAPÍTULO EXTRA 1: TRABALHANDO COM TOMCAT 5.5 – Introduz ao uso do Tomcat 5.5 utilizando o NetBeans IDE.

CAPÍTULO EXTRA 2: APLICAÇÕES WEB COM ACESSO A DADOS SEM PADRÃO – Indicado para iniciantes com baixa experiência em aplicações Web Java, ensina a acessar dados via JDBC diretamente através de scriptlets sem o padrão DAO.

CAPÍTULO EXTRA 3: STRUTS - Visão geral e técnica do framework Struts trabalhado através do NetBeans IDE, com configurações e acesso a banco de dados.

CAPÍTULO EXTRA 4: DESENVOLVENDO RELATÓRIOS COM NETBEANS IDE – Utilização do plug-in iReport for NetBeans para construir relatórios visualmente na IDE.

CAPÍTULO EXTRA 5: Estudo de caso completo com Visual Web JSF – Continua a aplicação criada no livro através dos capítulos 7 e 8, criando todos os relacionamentos e acesso ao banco de dados, incluindo o uso de novos componentes.

CAPÍTULO EXTRA 6: Estudo de caso completo com Visual Web JSF, Spring e Hibernate utilizando JPA – Criação da mesma aplicação gerada através dos capítulos 7, 8 e Extra 5, criando todos os relacionamentos e acesso ao banco de dados, utilizando DAO genérico, Spring framework e Hibernate com JPA, incluindo o uso de novos componentes.

APÊNDICE B: O MySQL – Explica o MySQL mais detalhadamente para desenvolvedores que não o conhecem.

SUMÁRIO

PARTE 1 - INTRODUÇÃO: DESENVOLVIMENTO DE APLICAÇÕES WEB COM JAVA	1
 Capítulo 1 - Obtendo e instalando o NetBeans IDE 6	3
Os pacotes.....	5
O pacote sem instalador e o JDK requerido	6
A instalação	6
A desinstalação	13
 Capítulo 2 - Servidores de Aplicações e Servlets	15
Criando um projeto	16
Visão geral do NetBeans IDE.....	20
Explorando seu projeto	21
Desenvolvendo Servlets	25
Como alterar o navegador no NetBeans.....	34
Entendendo como funciona um Servlet.....	36
Servidores de Aplicações Web	39
Monitorando transações HTTP.....	51
Distribuindo sua aplicação em arquivos WAR	54
 Capítulo 3 - Desenvolvendo páginas dinâmicas no NetBeans.....	57
Trabalhando com páginas JSP.....	57
Um pouco mais sobre o Deployment Descriptor.....	59
A estrutura de JavaServer Pages.....	61

Diretivas	61
O controle de erros configurado através da IDE	63
Recebendo dados de um formulário com JSP.....	68
O auto-completar do editor.....	71
Rodando uma página ou Servlet como inicial.....	72
Objetos implícitos	73
Criando JavaBeans	75
Outros atalhos do Editor de Códigos do NetBeans IDE.....	82
Utilizando JSTL em suas páginas	87
Desenvolvendo tags customizadas	111
Dinamizando Tag Files	116
 Capítulo 4 - Trabalhando com Banco de Dados	 119
Introdução ao JDBC.....	119
MySQL e o JDBC.....	120
A instalação e utilização do MySQL	120
Comandos básicos de utilização do MySQL	122
Acessando o banco de dados MySQL	122
O comando CREATE.....	123
O comando USE	124
Criando tabelas.....	124
O comando SHOW	125
Configurando usuários	125
Inserindo um registro	126

Baixando o driver JDBC.....	126
Utilizando o driver JDBC no NetBeans.....	127
As APIs JDBC	134
Os tipos de dados no Java e na SQL.....	137
Utilizando o Design Query	140
Utilizando padrões de desenvolvimento.....	141
O que é MVC?.....	141
O Padrão DAO (Data Access Object)	145
Pool de conexões.....	186
O aperfeiçoamento.....	193
 PARTE 2 - JAVA EE5: AVANÇANDO NO DESENVOLVIMENTO DE APLICAÇÕES WEB.....	 195
 Capítulo 5 - JavaServer Faces	 197
Um projeto JavaServer Faces.....	198
Conhecendo melhor o JavaServer Faces.....	213
As tags padrões de JavaServer Faces	215
Criando um exemplo utilizando banco de dados e JSF.....	222
Personalizando mensagens padrão do JavaServer Faces.....	238
Facelets e Web 2.0	241
Instalando um plugin com suporte a Facelets	242
Criando um CRUD nos padrões de Facelets.....	247

Capítulo 6 - EJB 3 e Java Persistence API	259
Criando um projeto Java EE 5	260
Seu primeiro EJB 3.....	263
Session Bean.....	269
As interfaces EJB	271
EJB 3 com acesso a dados	272
Utilizando JavaServer Faces para acessar o EJB.....	284
 Capítulo 7 - O Visual Web JavaServer Faces	 291
Criando uma aplicação	293
Definindo o layout da página principal	296
A página de contato.....	302
Criando a primeira navegação	314
 Capítulo 8 - Desenvolvendo com Visual Web JSF usando banco de dados	 317
O acesso a banco de dados	317
Desenvolvendo uma área administrativa	326
O acesso a área administrativa	346
Assegurando o acesso a área administrativa.....	355
Alterando as mensagens da sua aplicação	362
Adicionando o sistema de pesquisa no site.....	363
Adicionando Código a SessionBean1	366
 Capítulo 9 - Trabalhando com Web Services no NetBeans IDE	 371
Web Services	371

Entendendo a estrutura do documento WSDL	381
Consumindo o Web Service criado	386
Um Web Service mais complexo	390
Acessando o Web Service com Visual Web JSF.....	395
Criando um Data Provider	396
 Capítulo 10 - Visual Web JSF com JPA, Spring e Hibernate	 401
A aplicação que será construída.....	401
O Hibernate	402
Onde baixar a última versão	403
O Spring.....	404
O plugin do Spring Framework para o NetBeans.....	405
Criando o projeto Visual Web JSF com Spring Framework.....	406
Criando o DAO genérico	410
Configurando o Spring através de applicationContext.xml	415
Configurando o Spring no deployment descriptor	417
Criando a classe que controlará o CRUD	418
Configurando o Spring para trabalhar com JSF	419
Configurando o arquivo persistence.xml	422
O Log4j	424
Alterando a classe SessionBean1	427
Configurando o componente Table na página	429
Adicionando os métodos a Page1.java	430

PARTE 3 - DESENVOLVIMENTO COM LINGUAGENS DINÂMICAS E AJAX 439

Capítulo 11 - Rails 2 com NetBeans IDE 441

O que é Ruby?..... 442

O que é Ruby on Rails?..... 442

Onde baixar o Ruby 442

Configurando o Ruby no NetBeans IDE 6.0 444

Desenvolvendo com Ruby on Rails 444

A Camada Modelo..... 457

A Camada Controle 460

A Camada Apresentação..... 462

Adicionando relacionamentos..... 463

Mais sobre Ruby on Rails 474

Capítulo 12 - JRuby on Rails 475

O que é JRuby?..... 475

Baixando e instalando a última versão do JRuby..... 476

Configurando o JRuby no NetBeans 477

Instalando os Ruby Gems no NetBeans 478

Criando um projeto JRuby on Rails 481

Colocando sua aplicação Rails no Application Server 484

Capítulo 13 - Trabalhando com AJAX no NetBeans IDE 491

AJAX..... 491

Utilizando a tecnologia jMaki..... 495

Criando um projeto utilizando jMaki	499
Mas o que é JSON?.....	501
Ajax com GWT	511
Utilizando Ajax com Visual Web JSF	525
Outros frameworks AJAX.....	530
 Apêndice A - Ruby para desenvolvedores Java	 531
Recursos do Ruby	531
Desenvolvendo com Ruby no NetBeans IDE.....	532
Conhecendo o básico sobre Ruby	534
 Bibliografia	 579
 Capítulo Extra 1 – Trabalhando com Tomcat 5.5	 CD-ROM
 Capítulo Extra 2 – Aplicações Web com acesso a dados sem padrão	 CD-ROM
 Capítulo Extra 3 – Struts	 CD-ROM
 Capítulo Extra 4 – Desenvolvendo relatórios com NetBeans IDE	 CD-ROM
 Capítulo Extra 5 – Estudo de caso completo com Visual Web JSF	 CD-ROM
 Capítulo Extra 6 – Estudo de caso completo com Visual Web JSF, Spring e Hibernate utilizando JPA	 CD-ROM
 Apêndice B – O MySQL	 CD-ROM

PARTE 1



INTRODUÇÃO: DESENVOLVIMENTO DE APLICAÇÕES WEB COM JAVA



CAPÍTULO 1

OBTENDO E INSTALANDO O

NETBEANS IDE 6

O NetBeans é uma IDE criada em Java Swing e portanto, depende da Java Virtual Machine (JVM) instalada em sua máquina.

Além disso, você verá que há várias opções da IDE, onde cada uma contém módulos de instalação diferentes para cada necessidade do desenvolvedor.

Neste Capítulo você aprenderá a obter e instalar o NetBeans IDE na versão 6.x.

OBSERVAÇÃO: Apesar de abordar onde obter o NetBeans IDE, o CD-ROM anexo contém todos os arquivos que estão sendo ilustrados para a instalação.

BAIXANDO O NETBEANS NA INTERNET

Existem dois sites que oficialmente dão ao usuário a possibilidade de obter o NetBeans na versão 6.0. Um é o site oficial da própria IDE, que se encontra no endereço **<http://www.netbeans.org>**. O outro site é o oficial do Java, da própria Sun Microsystems, criadora da linguagem e principal mantenedora do NetBeans. O site neste caso é **<http://java.sun.com>**.

Em ambos os casos, a obtenção da IDE está correta.



FIGURA 1.1 – SITE OFICIAL DO NETBEANS IDE

Assim que você entra no site oficial do NetBeans, há um grande botão escrito **Download NetBeans IDE 6.0.**

Dando um clique neste botão, você irá até a página de downloads da IDE.

OBSERVAÇÃO: Você pode notar que existe ao lado do botão Download NetBeans IDE 6.0 um ícone de uma mídia CD/DVD escrito em seu rótulo Get Free DVD. Caso você queira, basta pedir o DVD pelo site que eles o enviarão sem custo algum. O tempo necessário para recebê-lo depende de sua localização.

ATENÇÃO: No momento em que este livro está sendo escrito, ainda não existe disponível uma versão traduzida em nosso idioma.

OS PACOTES

Ao clicar no botão Download NetBeans IDE 6.0, você será levado à área de download, contendo os diversos pacotes que a IDE atualmente oferece.

HOME / Download

NetBeans IDE 6.0 Download

6.0 | [Development](#) | [Archive](#)

Email address (optional):

Language: English Platform: Windows 2000/XP/Vista

Subscribe to newsletters: ☒ Monthly ☐ Weekly

☒ NetBeans can contact me at this address

NetBeans IDE Download Bundles

NetBeans Packs	Web & Java EE	Mobility	Java SE	Ruby	C/C++	All
Base IDE	•	•	•	•	•	•
Java SE	•	•	•			•
Web & Java EE	•					•
Mobility		•				•
UML						•
SOA						•
Ruby				•		•
C/C++					•	•
Bundled Servers						
GlassFish V2	•					•
Apache Tomcat 6.0.14	•					•
	Download Free, 96 MB	Download Free, 58 MB	Download Free, 21 MB	Download Free, 19 MB	Download Free, 11 MB	Download Free, 169 MB

FIGURA 1.2 – NETBEANS PACKS

Se você for desenvolver para a Web, como é o caso, selecione a opção **Web & Java EE** para uma opção enxuta ou **All** para todas as soluções.

Como pretendemos trabalhar também com Ruby on Rails, a opção **All** é mais atrativa, embora você possa instalar os pacotes separadamente após a instalação.

O PACOTE SEM INSTALADOR E O JDK REQUERIDO

Abaixo das opções mostradas anteriormente, você tem o link para o NetBeans compactado sem instalador (**zip file format**) e o link para o JDK requerido (**download the JDK here**).

* You can add or remove packs later using the IDE's Plugin Manager (Tools | Plugins).
JDK 6 or JDK 5.0 is required for installing and running the NetBeans IDE. You can [download the JDK here](#).
You can also download the NetBeans IDE as part of the [Java EE 5 Tools Bundle](#).

NetBeans source code and binary builds without bundled runtimes are also available in [zip file format](#).
See also [instructions on how to build the IDE from sources](#).

FIGURA 1.3 – A OPÇÃO COMPACTADA SEM ASSISTENTE

Sem o assistente, você tem uma listagem dos arquivos compactados no formato **.zip**, onde o maior é o **All**.

A INSTALAÇÃO

Este livro está baseado em um pacote independente da sua escolha, mas focado no ambiente Web.

O servidor de aplicações oficialmente suportado pelo NetBeans é o GlassFish V2 e o container Web é o Tomcat 6, ambos compatíveis com a versão Java EE 5. O JBoss possui suporte a EJB 3, tanto na versão 4 como na 5, mas que deve ser instalado até o momento separadamente e incluso logo após (veremos isso mais adiante).

NO WINDOWS

A instalação do NetBeans é tranqüila, quando utilizamos assistente, como a maioria dos programas existentes para este sistema operacional. Para o exemplo, a instalação do pacote contendo todos os aplicativos será usado (**All**). Assim que baixar o programa (ou pegá-lo no CD em anexo) dê um duplo clique sobre o arquivo que iniciará o processo de instalação.

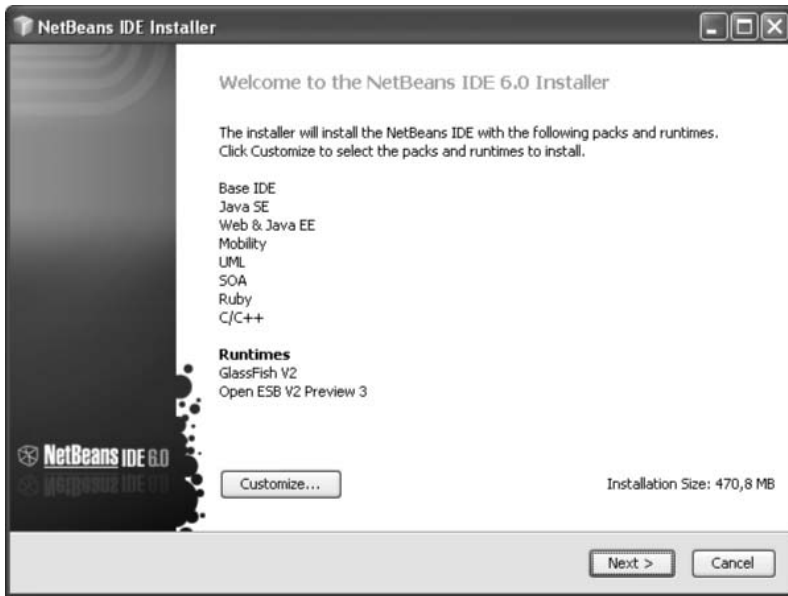


FIGURA 1.4 – INÍCIO DO ASSISTENTE DE INSTALAÇÃO

No botão **Customize**, marque ou desmarque os itens que não deseja instalar. Como o Tomcat 6 não está selecionado por padrão, caso não o tenha em sua máquina, selecione-o. Pacotes como C/C++ e Mobility são desnecessários para o nosso trabalho. Instale-os apenas se você for usar. Cada pacote a mais implicará em maior consumo de memória e portanto uma maior lentidão em seu uso em máquinas menos poderosas.

Confirme a customização e clique logo após no botão **Next** para prosseguir à segunda parte do assistente.



FIGURA 1.5 – CUSTOMIZANDO A INSTALAÇÃO

No assistente de instalação, leia a licença se não a conhecer e aceite marcando a opção **I accept the terms in the license agreement**. Para prosseguir, clique em **Next**.

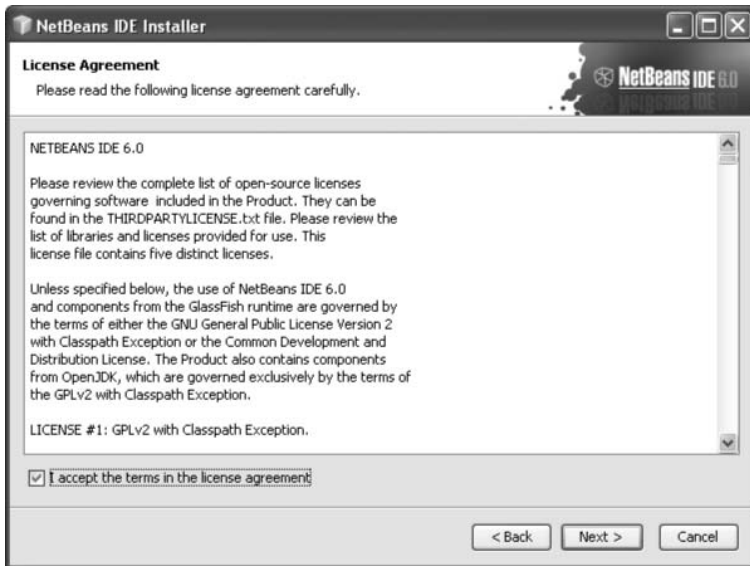


FIGURA 1.6 – TERMOS DE LICENÇA

Especifique um diretório vazio dentro do qual será instalado o NetBeans IDE. Esta instalação da IDE não alterará as configurações de outras instalações do NetBeans, caso você as tenha. Isso ocorre porque a IDE cria automaticamente um novo diretório de usuário quando é aberto. Para modificar o local da instalação, no campo **Install the NetBeans IDE to** digite ou clique no botão **Browse** e selecione.

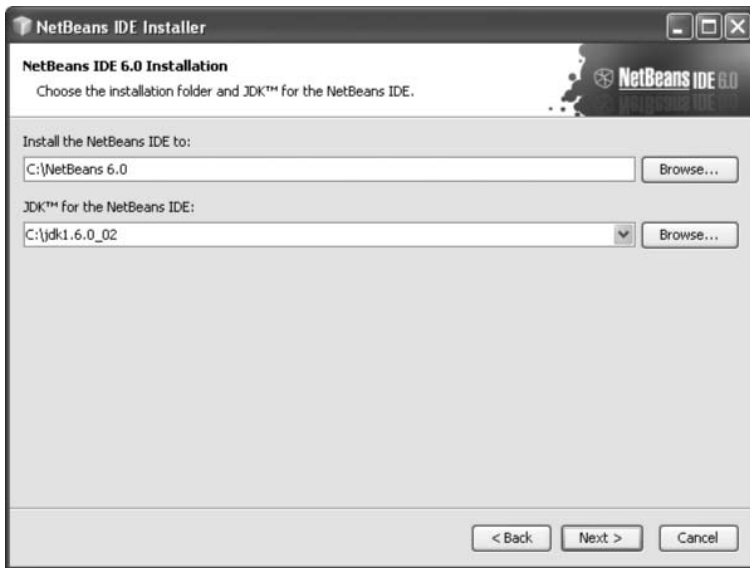


FIGURA 1.7 – DETERMINANDO O LOCAL DA INSTALAÇÃO E O JDK ENCONTRADO

Caso você tenha mais de um JDK instalado em sua máquina, este aparecerá em uma lista, no qual você deverá escolher o compatível com a IDE. No caso somente poderá ser as versões Java SE 5 ou 6. Clique no botão **Next**.

Na etapa seguinte você define o local da instalação do GlassFish (Install GlassFish to), o JDK e o usuário e senhas administrativas. Mais abaixo existe as portas para rodar o GlassFish em sua máquina.

ATENÇÃO: Caso mantenha como está, observe que a senha padrão é adminadmin.

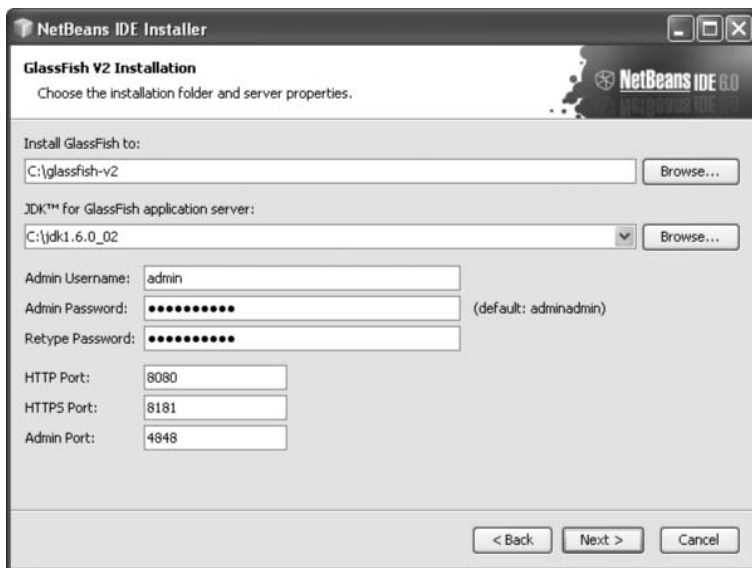


FIGURA 1.8 – CONFIGURAÇÃO DO GLASSFISH V2

A senha do usuário administrativo será guardada em um arquivo chamado **.asadminpass**. Este arquivo se encontra no diretório do usuário de seu sistema operacional.

Se em sua instalação houver também o Tomcat, você pode alterar o seu local de instalação no campo **Install Apache Tomcat to**.

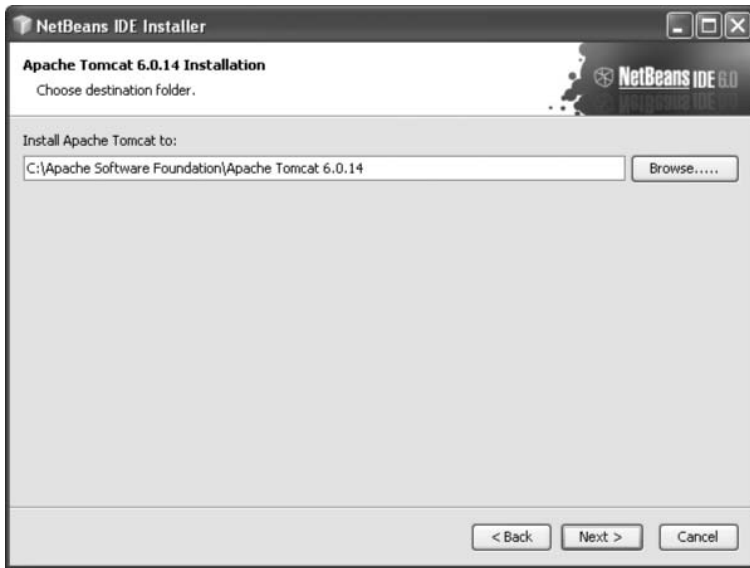


FIGURA 1.9 – DEFININDO O LOCAL DE INSTALAÇÃO PARA O TOMCAT 6

OBSERVAÇÃO: O servidor GlassFish e o container Tomcat não terão suas telas apresentadas caso você tenha optado por não instalá-los.

Por fim, veja os itens que serão instalados e clique no botão **Install**.

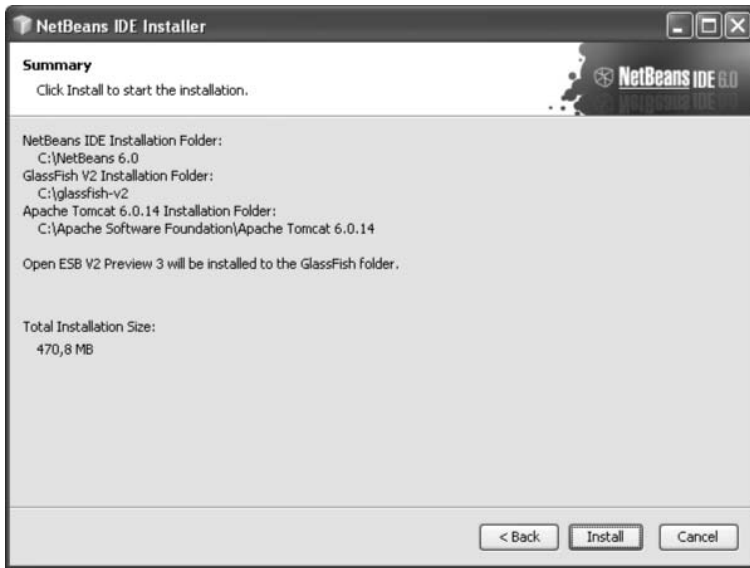


FIGURA 1.10 – CONFIRMAÇÃO PARA A INSTALAÇÃO

Aguarde a instalação até o botão **Finish**.

NO LINUX

A versão para Linux é ligeiramente diferente do Windows. O arquivo vem em formato binário, com assistente também, o que facilita sua instalação.

Para iniciar o processo de instalação, vá até o diretório onde se encontra o arquivo do qual baixou.

Dê a permissão necessária para que seja possível a execução do arquivo binário:

```
shell# chmod +x netbeans-6.0-linux.sh
```

Vale lembrar que a versão descrita se trata da versão mais atual no momento em que este livro está sendo escrito.

Para iniciar a instalação, execute o comando como demonstrado:

shell# ./netbeans-6.0-linux.bin

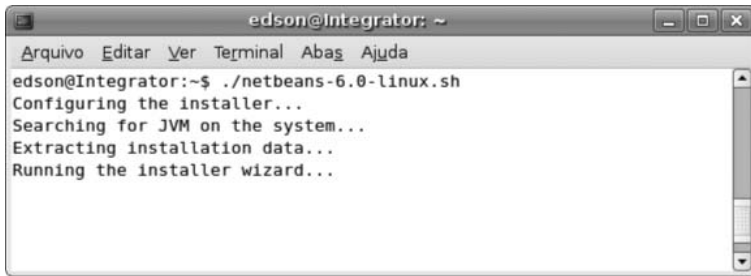


FIGURA 1.11 – INICIANDO A INSTALAÇÃO NO LINUX UBUNTU

As demais instruções são similares ao sistema operacional Windows.

A DESINSTALAÇÃO

DESINSTALANDO O NETBEANS NO WINDOWS XP

Para desinstalar a IDE no Windows, vá ao menu **Iniciar** do sistema e selecione o **Painel de controle**. Dê um duplo clique no ícone **Adicionar ou remover programas**.

Selecione o NetBeans IDE 6.0 na lista e clique em **Remover**.

Surgirá o desinstalador do NetBeans. Basta confirmar manter selecionados os servidores adicionados à IDE e clicar no botão **Uninstall**.

DESINSTALANDO O NETBEANS NO LINUX

Para desinstalar a IDE no Linux, vá ao diretório de instalação do NetBeans, geralmente **netbeans-6.0**, através do terminal. Digite a sequência a seguir:

```
shell# ./uninstall.sh
```

O mesmo que ocorre com o Windows ocorrerá com o Linux. Confirme no botão **Uninstall** para remover a IDE.

CAPÍTULO 2

SERVIDORES DE APLICAÇÕES E SERVLETS

Deste capítulo em diante você será levado a compreender a IDE com relação ao desenvolvimento de aplicações escritas para a Web. O seu fundamento criando e utilizando Servlets, trabalhará com os servidores de aplicações Web para rodar seus códigos desenvolvidos, monitorará sua aplicação e aprenderá como é distribuída para um servidor de produção.

Os tópicos apresentados neste capítulo serão:

- Criação de um novo projeto
- Visão geral do NetBeans IDE
- Explorando seu projeto
- Desenvolvendo Servlets
- Entendendo como funciona um Servlet
- Servidores de Aplicações Web
- Monitorando transações HTTP
- Distribuindo sua aplicação em arquivos WAR

CRIANDO UM PROJETO

Como o livro está focado em desenvolvimento Web, o seu primeiro projeto será feito para construir aplicações Web. Neste caso, mais especificamente para o desenvolvimento de um Servlet.

Servlets e JSP, assim como JavaServer Faces são tecnologias desenvolvidas pela Sun Microsystems para a construção de aplicações Web.

Para criar seu primeiro projeto Web, vá ao menu **File** e clique em **New Project**. Alternativamente, na janela **Projects**, você pode dar um clique com o botão direito do mouse e selecionar a opção **New Project (Ctrl+Shift+N)** no menu de contexto.

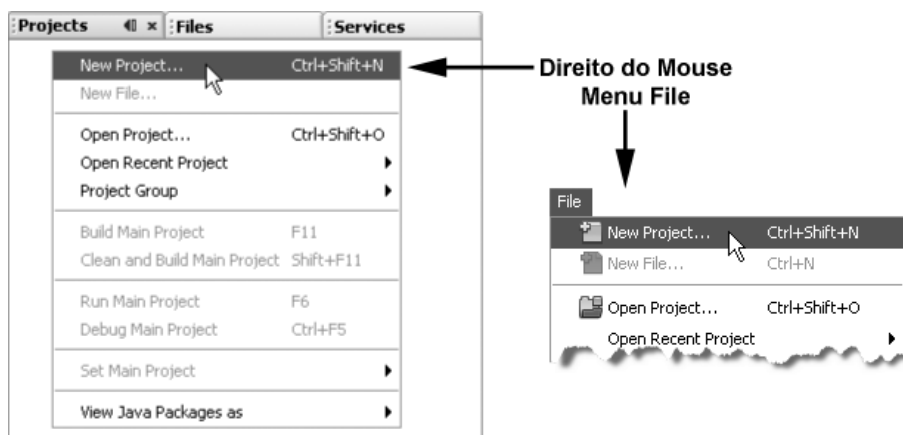


FIGURA 2.1 – SELECIONANDO A OPÇÃO NEW PROJECT

A caixa de diálogo **New Project** surgirá. O desenvolvimento de projetos para aplicações Web se encontra na categoria (Categories) **Web**. Como a aplicação trata-se de um desenvolvimento sem informações anteriores, ou seja, limpo, você seleciona **Web Application** em **Projects**. Após selecionar a opção em **Projects** clique no botão **Next**.

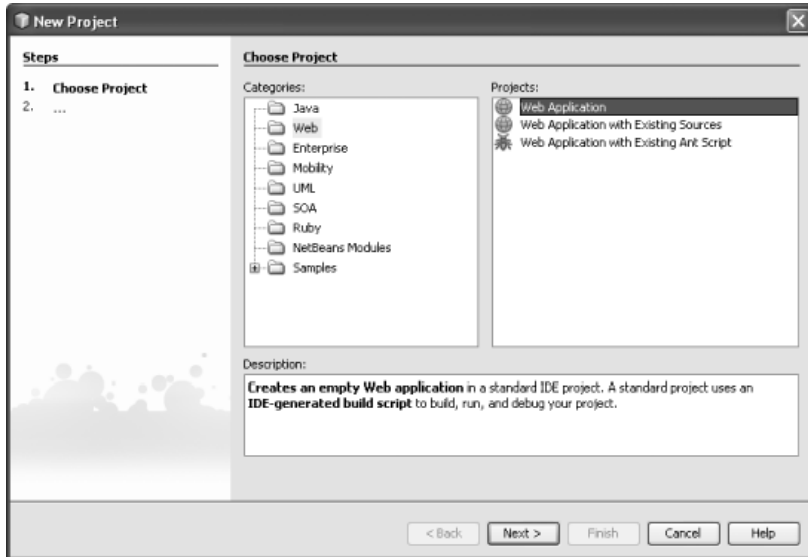


FIGURA 2.2 – SELEÇÃO DO ITEM WEB APPLICATION EM PROJECTS

Na segunda etapa do assistente, você possui várias opções para a construção do seu projeto.

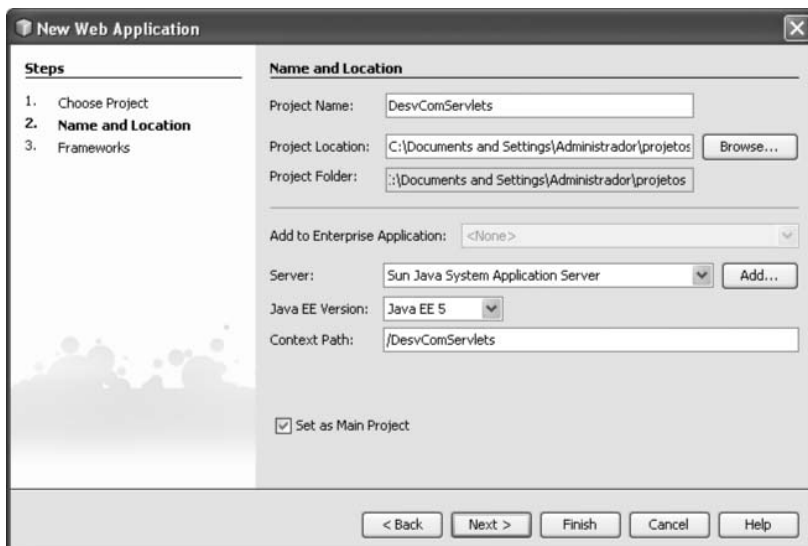


FIGURA 2.3 – NOME E LOCALIZAÇÃO DO PROJETO

Em **Project Name** você digita o nome do seu projeto. No caso do livro é **DesvComServlets**.

Em **Project Location** o NetBeans coloca seus projetos geralmente no diretório do usuário do seu sistema operacional. Para mudar a localização, clique no botão **Browse** e selecione ou crie um novo diretório para salvar seus projetos.

Perceba que o seu projeto gerará um novo diretório, e este é mostrado em **Project Folder**.

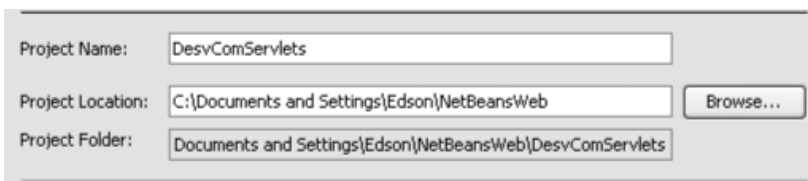


FIGURA 2.4 – DETALHE DO PASSO NAME AND LOCATION

Na parte inferior da caixa de diálogo você tem como principais pontos o Servidor, que pode ser selecionado em **Server**. Os servidores possuem características próprias, embora, no geral, sejam parecidos em suas administrações, desde que sigam rigorosamente regras impostas pela Sun Microsystems. Um exemplo de servidores que trabalhem com Java EE na versão 5 é o Sun Java System Application Server, GlassFish, Geronimo e etc.

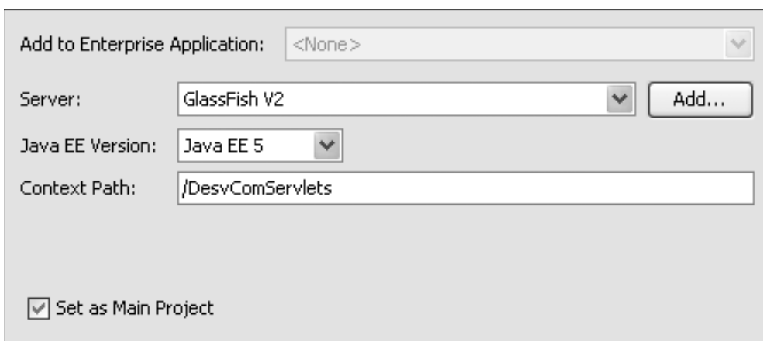


FIGURA 2.5 – DETALHE DO SERVIDOR, VERSÃO E CAMINHO DO CONTEXTO

Caso você precise trabalhar com a versão 1.4, basta selecionar em **Java EE Version** a opção **J2EE 1.4**.

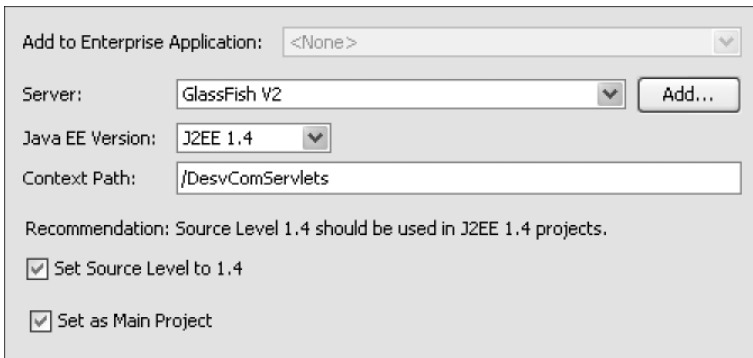


FIGURA 2.6 – SELEÇÃO DO JAVA EE VERSION INFERIOR A 5

Outra forma seria clicar no botão **Add** e adicionar um novo servidor (ou container) para uma versão inferior a 5.

No caso do livro, vamos utilizar o **GlassFish V2** como **Server** para o primeiro projeto. E com a versão de **Java EE 5** selecionada em **Java EE Version**.

Em **Context Path** você possui o nome do contexto de acesso ao projeto, o que na realidade será a sua aplicação.

Em caso de utilizar a versão J2EE 1.4, haverá a opção **Set Source Level to ...**, onde você possui o nível dos arquivos desenvolvidos no projeto. Em alguns casos (principalmente no uso de annotations) você deve desmarcar esta opção.

A opção **Set as Main Project** indica qual é o projeto que será compilado toda vez que você mandar rodar para exibir no browser.

A terceira etapa seria para a seleção de um framework, no qual veremos mais adiante. Para o momento, apenas clique no botão **Finish** para completar as configurações e criar o projeto.

VISÃO GERAL DO NETBEANS IDE

O NetBeans possui muitos menus, ferramentas e janelas que o auxiliam no desenvolvimento de uma aplicação Web. A seguir a **Figura 2.7** que demonstra os principais componentes que existem na IDE quando você cria um projeto.

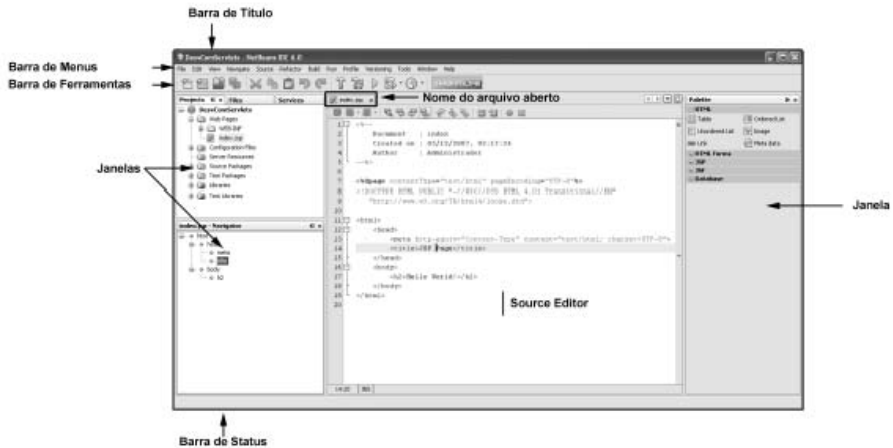


FIGURA 2.7 – CARACTERÍSTICAS GERAIS DO NETBEANS IDE 6.0

- **BARRA DE TÍTULO:** A barra de título sempre exibe o nome do projeto.
- **BARRA DE MENUS:** Esta é a barra de menu global, que lhe permite fazer todas as tarefas gerais. As opções disponíveis em cada menu também mudarão dependendo do que estiver selecionado.
- **BARRA DE FERRAMENTAS:** Esta é a barra de ferramentas global da IDE que também lhe permite executar tarefas gerais e tarefas específicas para itens selecionados.
- **BARRA DE STATUS:** Esta linha exibe tipicamente informações que dependem da tarefa você está executando atualmente.
- **SOURCE EDITOR:** É exatamente o que o seu nome diz: é um painel para editar documentos. Aqui você escreve seus códigos Java.
- **JANELAS:** Mostram grupos de objetos relacionados, propriedades, componentes e até mesmo a saída da execução de um código Java.

EXPLORANDO SEU PROJETO

A janela **Projects** será com certeza a que você mais vai utilizar, após o **Source Editor**.

Nesta janela você possui diversos atalhos, com o menu de contexto, que podem ser muito úteis, passando desde a execução de um projeto como até mesmo adicionar novas bibliotecas, criar novos arquivos, alterar seus nomes e etc.

Na construção de uma aplicação Web, você possui diversos elementos. Estes elementos estão dispostos em forma de diretórios que podem ser expandidos clicando em seus nós. O diretório **Web Pages** representa o diretório principal (raiz) de uma aplicação Web normal. Dentro deste nó você encontra o diretório **WEB-INF**, que contém o arquivo obrigatório **web.xml**, conhecido como **deployment descriptor**.

Junto a este diretório, há também um arquivo inicial, chamado de **index.jsp**, que é aberto inicialmente assim que concluído o projeto. Veremos isto mais adiante.

Expandindo o nó de **Configuration Files** você encontra os arquivos **MANIFEST.MF**, **sun-web.xml** e **web.xml**. Evidentemente pode haver mais arquivos, pois neste diretório se encontram todos os arquivos referentes à configuração da aplicação Web que você está desenvolvendo.

Em **Source Packages** você encontra pacotes e classes Java e Servlets, criados no processo de desenvolvimento da sua aplicação.

Para testes unitários você tem **Test Packages**.

As bibliotecas utilizadas no seu projeto se encontram em **Libraries**. Em **Test Libraries** você adiciona as bibliotecas necessárias para criar os testes necessários para a sua aplicação (por padrão, já estão contidas as bibliotecas do JUnit 3.8.2 e 4.1).

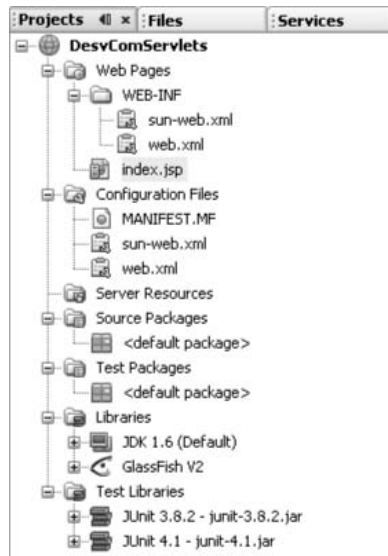


FIGURA 2.8 – DIRETÓRIOS EXPANDIDOS DO PROJETO NA JANELA PROJECTS

Esta estrutura utiliza o padrão **Java BluePrints**.

No caso de uma aplicação criada para trabalhar com o Tomcat, por exemplo, o padrão **Jakarta** entra em ação.

A Apache Jakarta fornece diretrizes de como estruturar suas aplicações Web para assegurar que elas trabalhem corretamente com o servidor Tomcat. Quando você cria um projeto na IDE e seleciona o Tomcat, esta estrutura é respeitada.

A seguir você tem a imagem da janela **Projects** com a estrutura de sua aplicação utilizando a **Apache Jakarta** para um projeto com o container **Servlet Tomcat 6**:

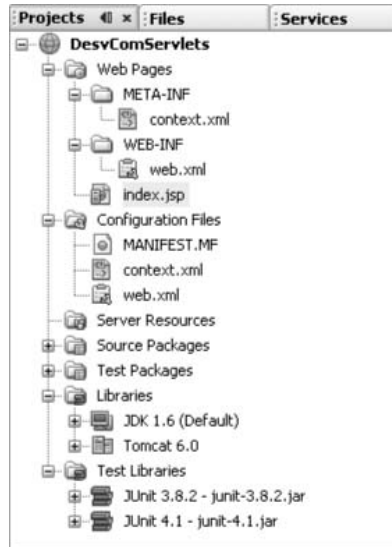


FIGURA 2.9 – ESTRUTURA DO PROJETO UTILIZADO PELO TOMCAT

Clicando na janela **Files**, ao lado de **Projects**, você vê a estrutura do seu projeto como se encontra no sistema operacional.

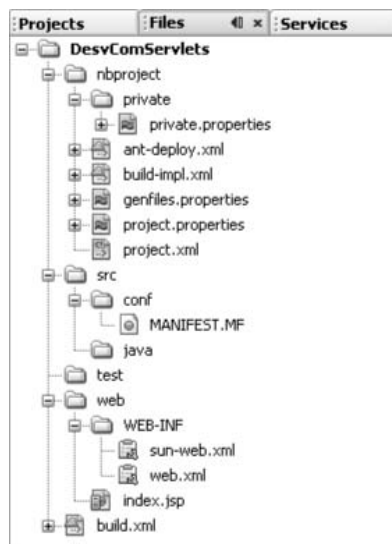


FIGURA 2.10 – ARQUIVOS ENCONTRADOS EM UM PROJETO NO NETBEANS IDE 6.0

A **Tabela 2.1** a seguir compara a estrutura de um projeto Web criado no NetBeans e a estrutura real da aplicação no GlassFish ou Tomcat.

TABELA 2.1 – COMPARAÇÃO DA APLICAÇÃO REAL VERSUS NO NETBEANS

CONTEÚDO	REPRESENTAÇÃO NA JANELA PROJECTS	REPRESENTAÇÃO NA JANELA FILES	NO ARQUIVO WAR
Páginas Web	Nó Web Pages	Diretório web	Na raiz do arquivo
Arquivos Java não compilados, servlets, entre outros	Nó Source Packages	Diretório src	Diretório <i>classes</i> dentro de WEB-INF
testes unitários	Nó Test Packages	Diretório test	N/A
deployment descriptor (web.xml)	Nó Configuration Files	Diretório WEB-INF em web	Diretório WEB-INF
GlassFish(sun-web.xml)	Nó Configuration Files	Diretório WEB-INF em web	Diretório WEB-INF
Arquivo de configuração do Tomcat e GlassFish V2 ou superior (context configuration) (context.xml)	Nó Configuration Files	Diretório META-INF em web	Diretório META-INF
Bibliotecas	Nó Libraries	Diretório lib em web/WEB-INF	Diretório WEB-INF/lib
Testes	Nó Test Libraries	Diretório test	N/A
Metadata do projeto incluindo script build	Propriedade do projeto na caixa de diálogo Project Properties, que pode ser aberta com um clique com o direito do mouse sobre o projeto, selecionando a opção Properties no menu de contexto.		

ATENÇÃO: Nem todos os itens apresentados na **Tabela 2.1** aparecem em um projeto utilizando o Application Server GlassFish.

DESENVOLVENDO SERVLETS

Como você pôde ver, o NetBeans criou na conclusão do projeto, um arquivo JSP (JavaServer Pages).

O que você vai fazer agora é um Servlet, que receberá de um formulário vindo deste arquivo criado na conclusão do projeto, para imprimir um determinado valor, que no seu caso, será seu nome.

Vá ao menu **File** e clique em **New File**. Alternativamente você pode clicar com o direito do mouse e selecionar, no menu de contexto, em **New**, o item em **Servlet**. Caso Servlet não esteja aparecendo, clique em **New File**. O atalho para um novo arquivo é **Ctrl + N**.

Na caixa de diálogo **New File** selecione em **Categories** o item **Web** e em **File Types** o item **Servlet**. Clique no botão **Next** para prosseguir.

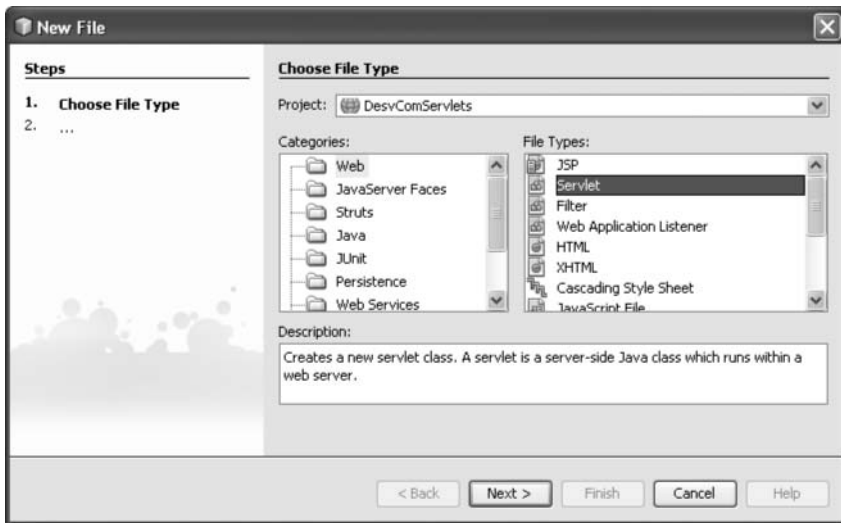


FIGURA 2.11 – CRIANDO UM SERVLET

Na segunda etapa do assistente, em **New Servlet**, digite **MeuPrimeiroServlet** em **Class Name**. Toda classe Java, assim como Servlets, devem ter um pacote. Como você ainda não fez nenhum pacote, digite em **Package** o nome do seu pacote. Caso queira seguir o livro, seria **br.com.integrator**. Perceba que o assistente lhe mostra a localização da classe Servlet que será gerada, em **Source Packages**, bem como onde será criado, em **Created File**. Clique no botão **Next** para a terceira e última etapa deste assistente.

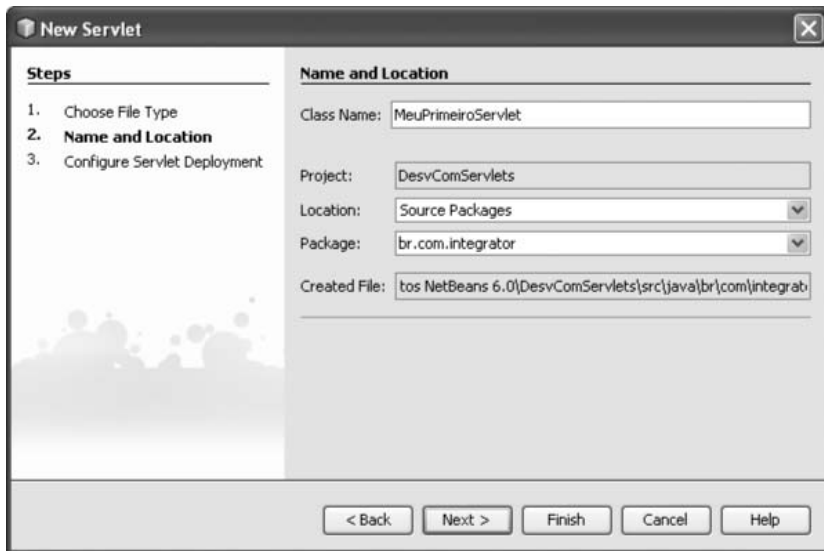


FIGURA 2.12 – DEFININDO UM NOME E UM PACOTE PARA O SERVLET

Na última etapa, mantenha como está. Esta etapa existe para que você possa configurar o caminho do seu Servlet na chamada pela URL através de **URL Pattern(s)**. Se você possui alguma experiência com aplicações Web construídas em Java, sabe que isto pode ser mudado, assim como o seu **Name**. Estas informações deverão ser gravadas no deployment descriptor (**Add information to deployment descriptor**), portanto, mantenha esta opção selecionada. Clique no botão **Finish** para completar o assistente.

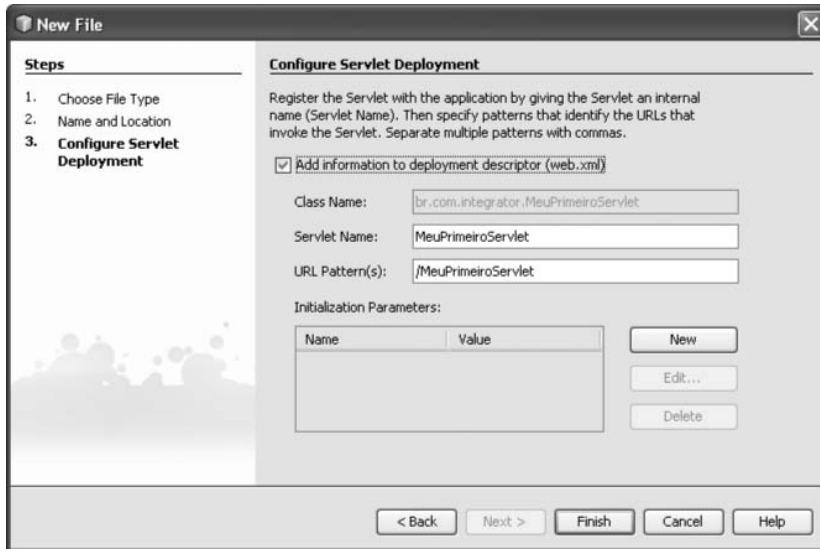


FIGURA 2.13 – ÚLTIMA ETAPA PARA A CRIAÇÃO DO SERVLET

Perceba que o NetBeans criou para você um arquivo, baseado em seu template, com parte do desenvolvimento de um Servlet.

Vamos alterá-lo para que você o veja funcionando e então explicarei melhor como tudo funciona.

No Servlet pré-construído, altere como mostrado na **Listagem 2.1** a seguir, em destaque:

LISTAGEM 2.1 – CÓDIGO DO ARQUIVO MEUPRIMEIROSERVLET.JAVA

...

```
protected void processRequest(HttpServletRequest request,
    HttpServletResponse response)
    throws ServletException, IOException {
    String nome = request.getParameter("nome");
    response.setContentType("text/html; charset=ISO-8859-1");
```

```

        PrintWriter out = response.getWriter( );
        // /* TODO output your page here
        out.println("<html>");
        out.println("<head>");
        out.println("<title>Meu nome é “+nome+”</title>");
        out.println("</head>");
        out.println("<body>");
        out.println("<h1>Meu nome é “ + nome + ”</h1>");
        out.println("</body>");
        out.println("</html>");
        // */
        out.close( );
    }
    ...

```

Os demais itens permanecerão exatamente como estão.

No arquivo **index.jsp**, você irá adicionar um formulário HTML. Para fazer isso, você possui duas formas: digitando ou utilizando a janela **Palette** como auxiliar. Vou ensinar pela janela **Palette** para simplificar sua digitação.

- Primeiramente, apague todo o conteúdo existente por entre as tags HTML **<body> ... </body>**.
- Mantenha o cursor por entre estas tags. Vá a janela **Palette** e dê um duplo clique em **Form**, na categoria **HTML Forms**. Alternativamente você pode arrastar desta janela para o Editor, por entre as tags HTML mencionadas.

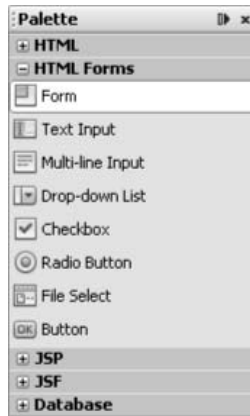


FIGURA 2.14 – O ITEM FORM SELECIONADO

- Na caixa de diálogo **Insert Form** digite em **Action** o nome do seu Servlet, que no caso é **MeuPrimeiroServlet**. Selecione em **Method** o método **POST** de envio de dados e em **Name** você pode dar um nome ao seu formulário, alternativamente. Confirme a caixa de diálogo clicando no botão **OK**.

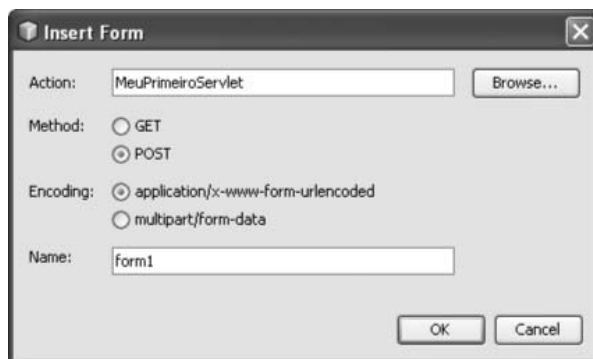


FIGURA 2.15 – A CAIXA DE DIÁLOGO INSERT FORM COM AS CONFIGURAÇÕES PREENCHIDAS

Perceba que o NetBeans construiu o formulário, através das tags HTML `<form />` no seu código, por entre as tags `<body />`.

- Coloque seu cursor por entre as tags `<form> ... </form>`. Novamente na janela **Palette**, na categoria **HTML Forms**, dê um duplo clique no componente **Text Input**.

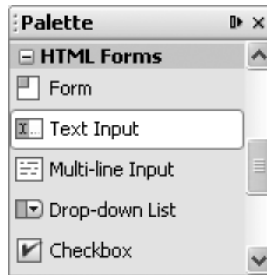


FIGURA 2.16 – COMPONENTE TEXT INPUT SELECIONADO

- Na caixa de diálogo **Insert Text Input** digite nome em **Name**. Mantenha **text** selecionado em **Type** e coloque **25** para a largura da sua caixa de texto, em **Width**. Confirme a caixa de diálogo clicando no botão **OK**.

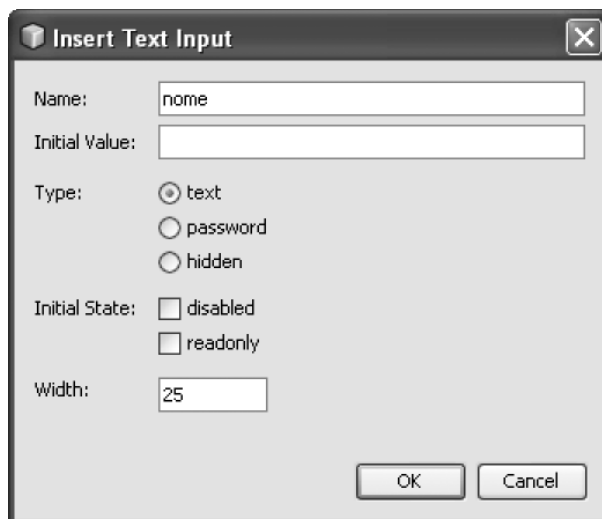


FIGURA 2.17 – CAIXA DE DIÁLOGO INSERT TEXT INPUT COM O CAMPO NOME CONFIGURADO

- Resta agora apenas adicionar um botão de envio (submissão) do formulário. Vá à janela **Palette** novamente, e dê um duplo clique no componente **Button**, ainda na categoria **HTML Forms**.

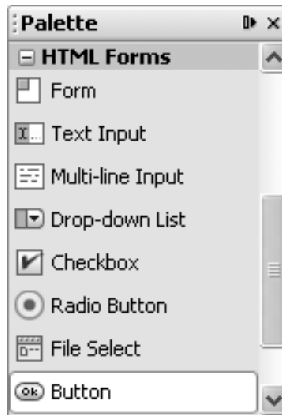


FIGURA 2.18 – COMPONENTE BUTTON EM DESTAQUE

- Na caixa de diálogo **Insert Button** digite **Enviar** em **Label** e mantenha selecionada a opção **submit** em **Type**. Dê um nome ao seu botão em **Name** e confirme a caixa de diálogo no botão **OK**.

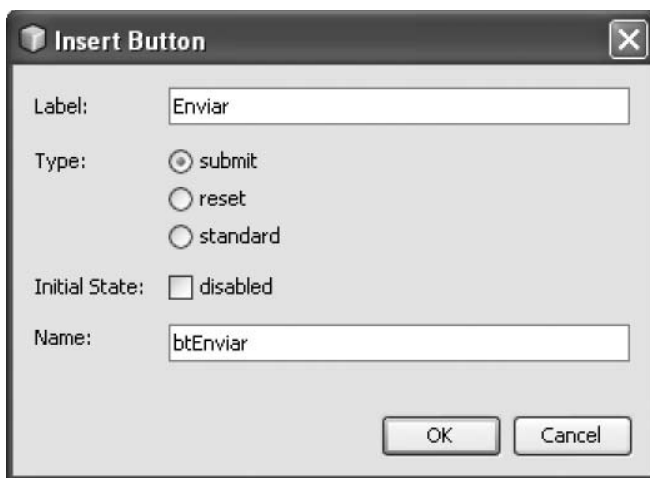


FIGURA 2.19 – BOTÃO ENVIAR CONFIGURADO NA CAIXA DE DIÁLOGO INSERT BUTTON

Neste ponto, o NetBeans deve ter criado todo o código necessário do formulário HTML para enviar seu nome ao Servlet desenvolvido.

Com o formulário construído e o Servlet alterado, vamos compilar e rodar a aplicação criada, de uma só vez.

Para compilar e rodar, você tem mais do que duas formas, na qual você pode escolher entre:

1. Clique no menu **Run** e selecione **Run Main Project**.

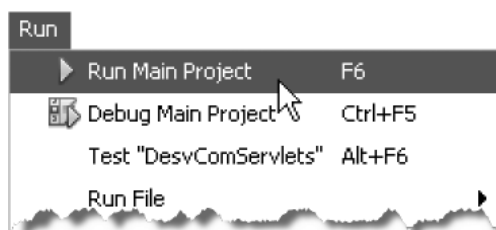


FIGURA 2.20 – RUN MAIN PROJECT DO MENU RUN

2. Com o botão direito do mouse sobre o projeto, na janela **Projects** selecione no menu de contexto o item **Run**. Este modo de rodar seu projeto também é muito usado quando queremos um projeto que não seja o padrão do compilador e executor.

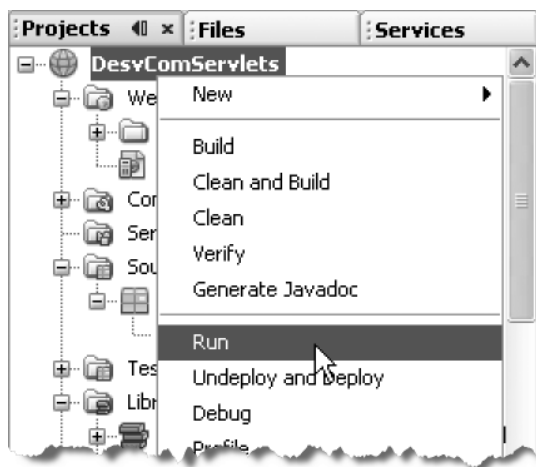


FIGURA 2.21 – DETALHE NA SELEÇÃO RUN DO MENU DE CONTEXTO NA JANELA PROJECTS

3. A terceira forma é clicando sobre o botão **Run Main Project**, na barra de ferramentas **Build**.



FIGURA 2.22 – BOTÃO RUN MAIN PROJECT PRESSIONADO NA BARRA DE FERRAMENTAS BUILD

4. A quarta forma se encontra no atalho **F6**, que pode ser visto no menu **Run** quando você o abre, em **Run Main Project**.

Ao começar a rodar a sua aplicação, note a janela **Output** que surgiu logo abaixo de sua IDE, acima da barra de status. Ela contém três janelas. A primeira é a exibição da compilação da sua aplicação Java (ferramenta Ant), a segunda é o Java DB Database Process, que exibe a inicialização do banco de dados derby, embutido a IDE. O log do servidor de aplicações ou Container Servlet é a terceira a janela, contendo no caso o nome GlassFish V2, juntamente com botões de controle. Veremos isto com detalhes mais adiante.

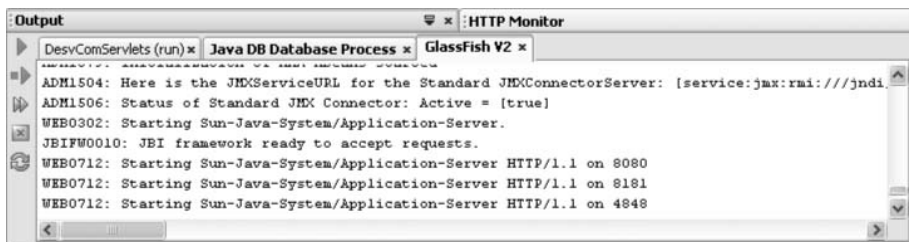


FIGURA 2.23 – SAÍDAS ENCONTRADAS NA EXECUÇÃO DE UMA APLICAÇÃO WEB

O navegador abrirá. O NetBeans usa o navegador padrão do seu sistema operacional para abrir a página principal, no caso o **index.jsp** primeiramente. No formulário, digite seu nome e clique no botão **Enviar**. O Servlet criado no exemplo, receberá esta informação e a imprimirá na tela, gerando um HTML.

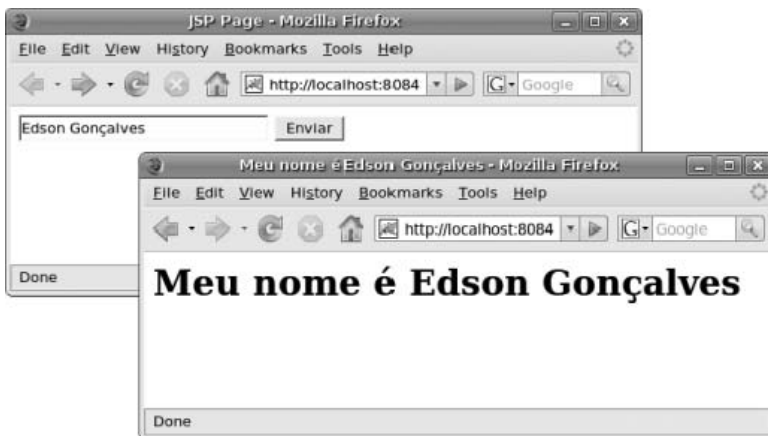


FIGURA 2.24 – FORMULÁRIO DE ENVIO E O RESULTADO IMPRESSO PELO SERVLET APÓS RECEBER OS DADOS

COMO ALTERAR O NAVEGADOR NO NETBEANS

Caso você queira alterar o navegador do NetBeans, vá ao menu **Tools** e clique em **Options**.

Na caixa de diálogo **Options**, em **General**, você seleciona em **Web Browser** o navegador de sua preferência para a exibição de suas páginas.

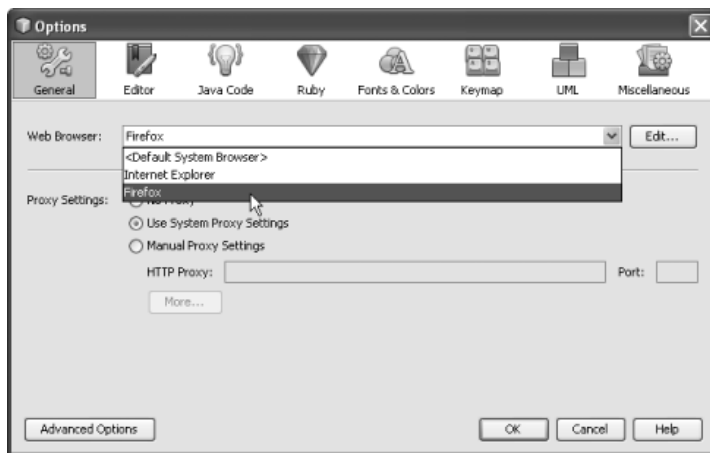


FIGURA 2.25 – SELECIONANDO UM NOVO WEB BROWSER COMO PADRÃO

Caso você queira adicionar um novo browser, que não esteja na lista, clique no botão **Edit**. Na caixa de diálogo **Web Browsers** clique no botão **Add**. Digite em **Name** o nome referente ao navegador que está adicionando, selecione em **Process** o executável deste browser e coloque em **Arguments** o argumento necessário para que a página da sua aplicação seja aberta quando o mesmo for chamado.

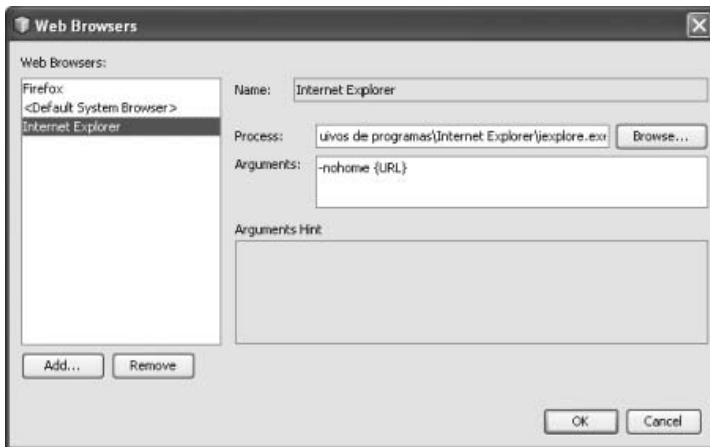


FIGURA 2.26 – ADICIONANDO OU REMOVENDO UM WEB BROWSER

Ao preencher todos os itens necessários clique no botão **OK**.

Caso haja necessidade de configurar algum Proxy, em **Proxy Settings** você pode definir de três formas:

- **No Proxy** – Sem Proxy.
- **Use System Proxy Settings** – O padrão. Ele utiliza as configurações padrão do sistema para a configuração do Proxy.
- **Manual Proxy Settings** – Neste caso, você deve configurar manualmente o Proxy a ser utilizado.

ENTENDENDO COMO FUNCIONA UM SERVLET

Agora que você já entende como se cria um projeto e um Servlet na IDE, basta apenas explicar como ele funciona e o que pode ser feito.

Vamos começar pela sua localização, na IDE. Expanda o nó de **Source Packages**, na janela **Projects** e você encontrará um pacote, que no exemplo dado será **br.com.integrator**, ou o nome que você deu, na geração do Servlet. Expanda o nó deste pacote e você verá o Servlet criado.



FIGURA 2.27 – O SERVLET CRIADO EM SOURCE PACKAGES

Observe que o Servlet possui a extensão **.java**. Portanto, Servlets são classes Java, desenvolvidas de acordo com uma estrutura bem definida e que, quando instaladas junto a um Servidor que implemente um Servidor de Aplicações ou Servlet Container (um servidor que permita a execução de Servlets), podem tratar requisições recebidas de clientes.

Se você analisar o código, que foi gerado pela IDE, notará que contém dois métodos importantes: **doGet()** e **doPost()**. Ambos chamam **processRequest()**, onde você alterou o código para receber os dados vindos do formulário.

Estes métodos possuem a habilidade de resgatar informações enviadas pelo usuário tanto pelo método **GET**, como pelo método **POST**. Na geração do formulário **HTML**, você criou a tag **<form/>** selecionando em **METHOD** o item **POST**, o que, portanto, é resgatado por este Servlet. Embora existam outros protocolos de envio, **POST** e **GET** são os mais usados.

Olhando atentamente a classe Servlet criada, notará que ela estende a classe **javax.servlet.http.HttpServlet**, uma classe abstrata que estende a **javax.servlet.GenericServlet**. A criação de Servlets exige as classes do pacote **javax.servlet** e **javax.servlet.http**, que pertencem a API Servlet do Java, que faz parte do Java EE. Perceba então que há dois **imports** de pacotes nesta classe.

A saída de informações, para serem impressas no HTML de retorno ao usuário, é feito pelo método **println**, de **java.io.PrintWriter**. Há também a importação desta biblioteca no Servlet criado pela IDE.

Para compilar um Servlet manualmente, você precisaria incluir no classpath o arquivo **servlet-api.jar**, que fica no diretório **lib** do Tomcat, o que o NetBeans faz para você, através do arquivo **Ant** pré-configurado **build.xml** (este arquivo só é visível através da janela **Files**). O **Ant** é uma ferramenta que o NetBeans usa para rodar comando relacionados ao projeto. Se você não conhece Ant e não possui nenhum interesse em conhecer, saiba que este pode ser ignorado em questão quase que completamente. **Ant** foi desenvolvido pela **Apache Software Foundation** para automatizar rotinas de desenvolvimento como compilar, testar e empacotar sua aplicação.

Expandindo o nó de **Configuration Files**, na janela **Projects**, você encontra o deployment descriptor, como já foi dito, chamado de **web.xml**. Este **deployment descriptor** é usado por Servlets Containers como o Tomcat e servidores de aplicação como o GlassFish. O GlassFish, utilizado no exemplo, utiliza também o **Sun Deployment Descriptor**, chamado de **sun-web.xml**.

O arquivo descritor de contexto (Deployment Descriptor) é o padrão de uma aplicação Web, segundo a especificação Java Servlet/Java EE, em um arquivo chamado de **web.xml**.

Em um Deployment Descriptor padrão, você verá as seguintes informações nele contidas, que detêm as configurações específicas da aplicação:

- Informações textuais de título (elemento **<display-name />**, nome para exibição no Manager) e comentário da descrição (**<description />**) do contexto, úteis para identificação e documentação.
- O elemento **<servlet />** indica o nome do Servlet bem como sua classe.

No elemento **<servlet-mapping />** você mapeia o Servlet para que seja melhor acessível no navegador.

Como ele possui muitos elementos, estes são separados por botões, pela IDE. Note que temos inicialmente **General**, para informações gerais, como o tempo de duração de uma sessão, **Servlets**, onde são mapeados os servlets da sua aplicação, **Pages** que indica o arquivo inicial, quando em produção no Servidor e **XML**, onde você pode visualizar toda a estrutura do arquivo web.xml.

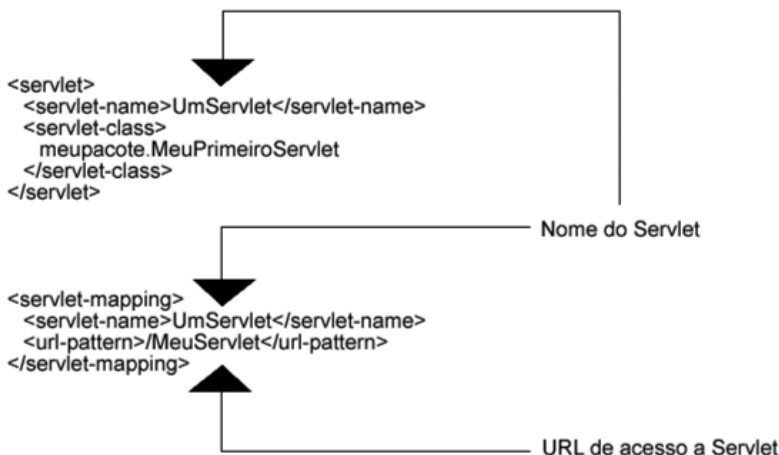


FIGURA 2.28 – O DEPLOYMENT DESCRIPTOR ABERTO PELO NETBEANS

Para compreender um pouco mais sobre os Servlets, vamos mapear um. Quando você criou o Servlet, você tinha como opção de digitação na construção pelo assistente o **URL Pattern(s)**, lembra?

Pois é, também possuía um nome para o Servlet. Estes são configurações que o deployment descriptor precisa para saber qual Servlet utilizar, mesmo porque, colocar o caminho do pacote mais o nome do Servlet não é muito inteligente.

O GLASSFISH E O TOMCAT

O GlassFish é um servidor de aplicações Web open source, baseado no Sun Java System Application Server Platform Edition, sendo 100% compatível com as especificações Java EE 5 robusto, de padrão aberto, mantido pelo Projeto GlassFish, parte da comunidade OpenJava EE.

Poderoso, este servidor contém itens que um Servlet Container como o Tomcat não possui, incluindo suporte a EJB (Enterprise JavaBeans) e JMS (Java Message Service).

O Tomcat tem suas origens no início da tecnologia Servlet. A Sun Microsystems criou o primeiro Servlet Container, o Java Web Server, para demonstrar a tecnologia, mas não era um servidor robusto, para uso na Web como se necessitava. Ao mesmo tempo, o Apache Software Foundation (ASF) criou JServ, um servlet engine que integrava com o servidor Web Apache.

A versão Tomcat 6.x é a atual e é a RI de especificações Servlet 2.5 e JSP 2.1, pertencentes ao Java EE 5.

TRABALHANDO COM OS SERVIDORES PELO NETBEANS

Na janela **Services**, expanda o nó do item **Servers**. Veja os servidores que existem configurados neste local.

Você pode iniciar o **GlassFish V2** ou o **Tomcat**, clicando com o direito do mouse sobre ele e selecionando **Start** no menu de contexto.

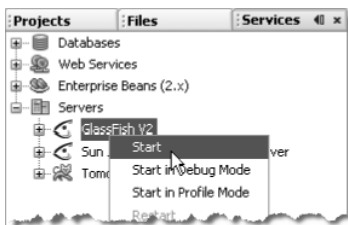


FIGURA 2.30 – INICIANDO O GLASSFISH



FIGURA 2.37 – VISUALIZANDO OS WEB APPLICATIONS PELO ADMIN CONSOLE

Retornando ao NetBeans, com o uso do direito sobre **GlassFish**, no menu de contexto, selecione **Refresh** para refletir a alteração no servidor.

ADMINISTRANDO APLICAÇÕES PELO TOMCAT FORA DO NETBEANS

Para acessar o administrador do Tomcat, abaixo da versão 6, **Administration Tool**, clique com o direito do mouse sobre o servidor, no NetBeans, janela **Services**, e selecione no menu de contexto o item **View Admin Console**.

ATENÇÃO: No momento em que este livro é escrito, não havia aplicação Web administrativa do Tomcat 6.0.14 liberada pela Fundação Apache. Nos extras do livro, contido no CD-ROM anexo, você encontra um capítulo inteiro ensinando a trabalhar com a versão Admin do Tomcat 5.5.

Outra maneira de administrar as aplicações Web através do Tomcat é pelo **Manager**. O **Manager** possui uma estrutura mais simples, pois apenas administra as aplicações para o básico como **deploy**, **undeploy**, **reload** da aplicação e etc.

Para isso, no seu browser, seguindo a porta padrão, você digita:

http://localhost:8084/manager/html

Ao que surgirá uma caixa de diálogo pedindo o usuário e a senha. Após o preenchimento, você entrará no manager.

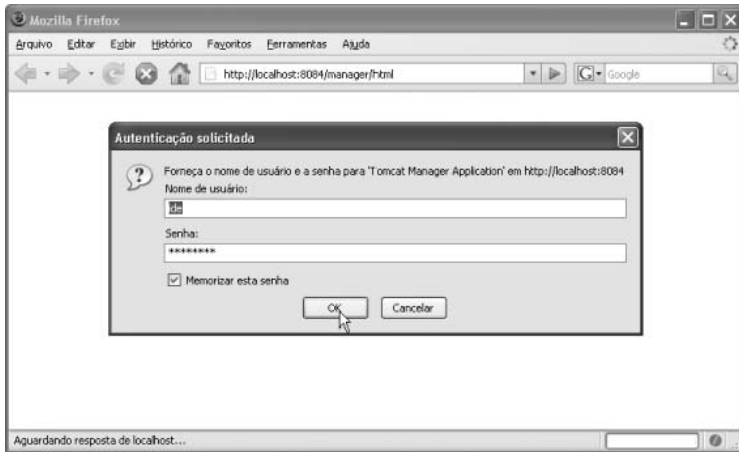


FIGURA 2.38 – ACESSANDO O MANAGER PELO BROWSER

O Manager é dividido em duas sessões principais: **Applications** e **Deploy**. Através de **Applications** você pode dar desde Start, Stop e Reload em sua aplicação, como até mesmo fazer o undeploy, simplesmente com um clique. Ele também exibe as Sessões em execução na aplicação, além de poder modificar o tempo de sessão e expirar com o botão **Expire sessions**. Ou seja, se houver um navegador visualizando uma de suas aplicações, ele conta uma sessão aberta. Clicando em **Path** você abre a aplicação no navegador.

Com isso, você tem a seguinte compatibilidade quanto aos servidores de aplicação:

- Sun Java System Application Server
- GlassFish V1 e V2
- Tomcat 5.x e 6
- JBoss 4 e 5
- BEA WebLogic Server 9 e 10
- IBM WebSphere Application Server V6.0 e 6.1 (NetBeans 6.1)
- Sailfin V1 (NetBeans 6.1)

Segundo as recomendações da equipe do NetBeans.org, o uso de versões do JBoss, BEA WebLogic ou qualquer outra não seja especificada nesta lista, pode alterar o funcionamento da IDE de forma imprevisível, uma vez que as interfaces externas não são controladas pela equipe do projeto.

MONITORANDO TRANSAÇÕES HTTP

O NetBeans fornece um Monitor HTTP embutido para ajudá-lo a isolar problemas com a fluência de dados de páginas JSP e execução de Servlets em um servidor Web. Quando a IDE é configurada com um container ou application server, ou é feito um deploy em uma aplicação web com o Monitor HTTP do NetBeans habilitado, este registrará todos os pedidos feitos neste container web. Para cada pedido HTTP que é processado pelo servidor, o Monitor HTTP registra não só o pedido, mas também registros com informações de estados mantidos no container Web.

Por usar o Monitor HTTP, você pode analisar pedidos HTTP feitos pelo método GET e POST para uma futura análise. Você também pode editar estes pedidos armazenados e também pode revê-los, em um replay.

Os pedidos são armazenados até que você encerre a IDE. Você também pode salva-los de forma que estejam disponíveis em sessões posteriores da IDE.

CAPÍTULO 4

TRABALHANDO COM

BANCO DE DADOS

O trabalho com banco de dados utilizando o NetBeans se desenvolveu ao longo das últimas versões. O usuário pode criar desde suas tabelas como até mesmo instruções SQL de forma visual. Este capítulo tratará do uso de banco de dados em aplicações Web com Servlets e páginas JSP, utilizando o NetBeans IDE.

Ao longo deste capítulo será apresentado:

- A instalação, configuração e acesso ao banco de dados MySQL;
- Como utilizar o NetBeans IDE para criar e modificar tabelas e instruções SQL;
- O trabalho com os padrões MVC e DAO em suas aplicações;
- A utilização de refactor do código e do histórico do NetBeans;
- Como criar testes usando JUnit;
- A criação de páginas JSP com uso de JSTL para acessar dados via JDBC;

INTRODUÇÃO AO JDBC

JDBC é uma API incluída dentro da linguagem Java para o acesso a banco de dados. Consiste em um conjunto de classes e interfaces escritas em Java que oferecem uma completa API para a programação com banco de dados, por tanto é uma solução 100% Java.

JDBC é uma especificação formada por uma coleção de interfaces e classes abstratas, que devem implementar todos os fabricantes de drivers que queiram realizar uma implementação de seu driver 100% Java e compatível com JDBC.

Devido ao JDBC ser escrito completamente em Java também passa a ter a vantagem de ser independente de plataforma. Sendo assim, não será necessário escrever um programa para cada tipo de banco de dados, uma mesma aplicação escrita utilizando JDBC poderá trabalhar com banco de dados como Oracle, Sybase, SQL Server, MySQL, Firebird, PostgreSQL e etc. Para que isso aconteça, basta alterar o JDBC referente ao banco de dados usado e o seu sistema passará a se comunicar com o banco de dados configurado.

MySQL e o JDBC

Sendo um dos sistemas de gerenciamento de bancos de dados mais usados do mundo, sua velocidade e capacidade de ser multiplataforma só poderiam chamar a atenção de quem desenvolve em Java.

O driver JDBC escolhido para fazer os exemplos neste livro foi o **Connector/J**, suportado oficialmente pela mantenedora do MySQL. Usando o driver Connector/J, todos os tipos de aplicações Java podem acessar um banco de dados e seus dados, desde que seja em MySQL, é claro.

A INSTALAÇÃO E UTILIZAÇÃO DO MySQL

O MySQL tem diferentes formas de instalação quando se trata de sistemas operacionais. No caso do Windows, você pode baixar a última distribuição através do site:

<http://www.mysql.com/downloads>

INSTALANDO NO WINDOWS

Procure pelo formato executável. O arquivo vem compactado no formato **.zip**.

Descompacte e instale. A instalação, como não poderia deixar de ser, é feita por um assistente. Siga os passos até a finalização.

UTILIZANDO O DRIVER JDBC NO NETBEANS

Na janela **Services** expanda o nó do item **Databases**. Clique com o direito do mouse sobre o item **Drivers** e selecione no menu de contexto o único item: **New Driver**.

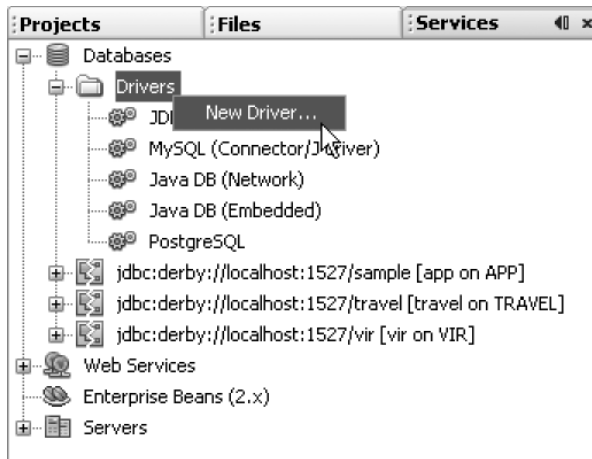


FIGURA 4.1 – MENU DE CONTEXTO PARA ADIÇÃO DE UM NOVO DRIVER

Na caixa de diálogo **New JDBC Driver** clique no botão **Add** e selecione no local onde você descompactou o arquivo **mysql-connector-java-5.1.5-bin.jar**.

Ao selecionar, note que os campos **Driver Class** e **Name** ficam preenchidos. Confirme a caixa de diálogo clicando no botão **OK**.

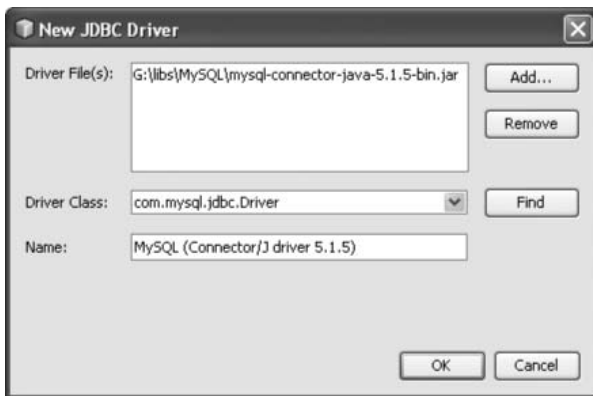


FIGURA 4.2 – CONFIGURAÇÃO DO NOVO DRIVER JDBC

SE CONECTANDO AO BANCO DE DADOS

O NetBeans possibilita a conexão ao banco de dados pela IDE, tornando acessível sua manipulação. Na janela **Services** clique com o direito do mouse sobre o item **Databases** e selecione no menu de contexto o item **New Connection**.



FIGURA 4.3 – CRIANDO UMA NOVA CONEXÃO

Na caixa de diálogo **New Database Connection**, na aba **Basic setting**, selecione em **Name** o driver do MySQL adicionado. No campo **Driver** aparecerá a classe de conexão.

No campo **Database URL** perceba que por parte já está preenchido. Digite o restante como mostrado a seguir:

jdbc:mysql://localhost:3306/livraria

No campo **User Name** use o usuário que você adicionou ao criar o banco de dados e em **Password** a senha.

Marque a opção **Remember password** se você desejar manter a senha durante a execução do NetBeans. Confirme clicando no botão **OK**.

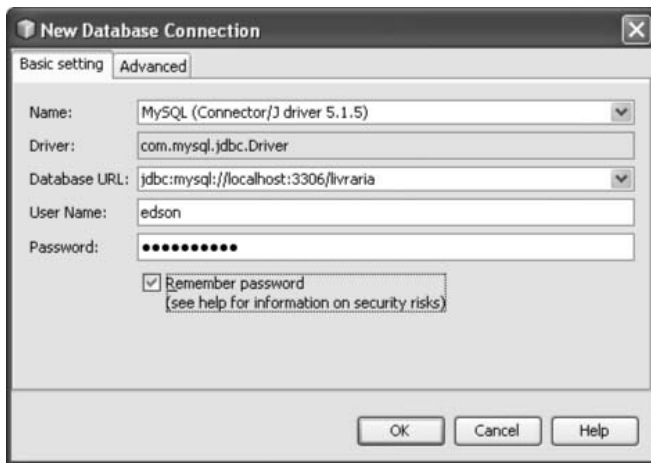


FIGURA 4.4 – BANCO DE DADOS LIVRARIA CONFIGURADO

A caixa de diálogo mudará o foco para a aba **Advanced**, onde no final aparecerá uma mensagem - **Connection established**. Isso significa que a conexão foi estabelecida com sucesso. Confirme novamente no botão **OK**.

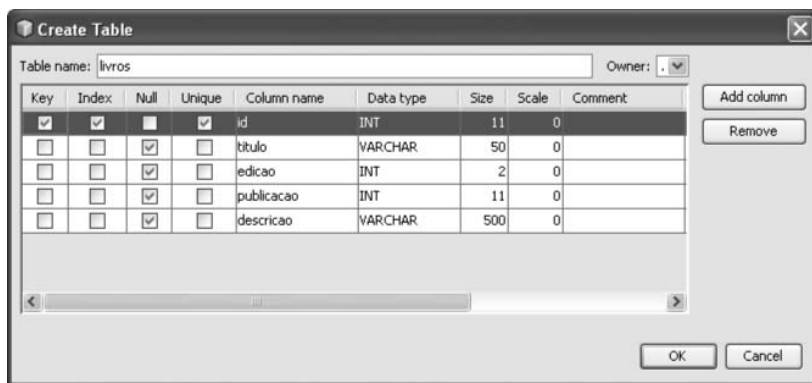


FIGURA 4.8 – CRIAÇÃO DA TABELA LIVROS PELA IDE

Você deve ter percebido que alguns tipos de dados (em **Data type**) não existem como os do MySQL. Por isso, para completar a criação da tabela, você vai colocar os comandos a seguir, em **SQL Command** (menu de contexto, **Execute Command**):

alter table livros modify id int auto_increment;

Para confirmar a execução, clique no botão **Run SQL (Ctrl+Shift+E)**.



FIGURA 4.9 – EXECUTANDO A INSTRUÇÃO PARA MODIFICAR A COLUNA ID

UTILIZANDO O DESIGN QUERY

Na sexta versão do NetBeans, o Design Query foi adicionado nativamente a IDE graças a incorporação do Visual Web JavaServer Faces. Agora, você pode criar instruções SQL visuais pelo Design Query antes de adicionar em suas aplicações. Para acessar o Design Query, clique com o direito do mouse sobre a tabela **livros**. Selecione no menu de contexto o item **Design Query**.

Nesta janela podemos criar instruções SQL para seleção de dados visualmente, o que garante uma velocidade maior no desenvolvimento de queries mais complexas.

Mais adiante veremos com mais detalhes o seu uso.

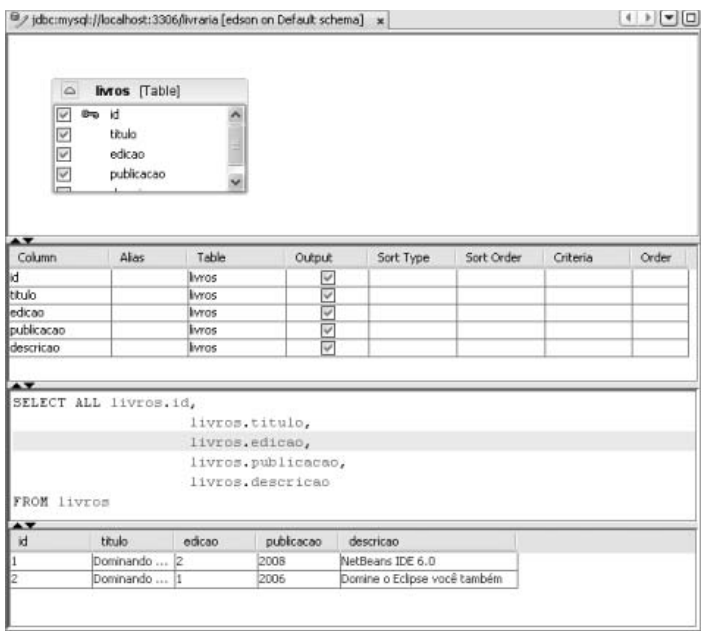


FIGURA 4.12 – DESIGN QUERY

UTILIZANDO PADRÕES DE DESENVOLVIMENTO

Em uma página JSP o correto é conter apenas o mínimo possível de scriptlets. A essa forma de desenvolver aplicações chamamos de Padrões de Desenvolvimento (Design Patterns). Um padrão muito praticado no desenvolvimento de aplicações Web escritas em Java é o Model 2, baseado no paradigma MVC (Model-View-Controller). Quanto ao acesso a banco de dados temos um padrão popularmente usado, chamado de DAO (Data Access Object).

O QUE É MVC?

MVC é um conceito (paradigma) de desenvolvimento e design que tenta separar uma aplicação em três partes distintas. Uma parte, a Model, está relacionada ao trabalho atual que a aplicação administra, outra parte, a View, está relacionada a exibir os dados ou informações dessa aplicação e a terceira parte, Controller, em coordenar os dois anteriores exibindo a interface correta ou executando algum trabalho que a aplicação precisa completar.

A arquitetura MVC foi desenvolvida para ser usada no projeto de interface visual em Smalltalk.

Estas partes são respectivamente:

- **MODEL:** O Model (Modelo) é o objeto que representa os dados do programa. Maneja esses dados e controlam todas suas transformações. Esse modelo não tem conhecimento específico dos controladores (controller) e das apresentações (views), nem sequer contém referência a eles. Portanto, o Model são as classes que trabalham no armazenamento e busca de dados. Por exemplo, um cliente pode ser modelado em uma aplicação, e pode haver vários modos de criar novos clientes ou mudar informações de um relativo cliente.
- **VIEW:** A View (Apresentação) é o que maneja a apresentação visual dos dados representados pelo Model. Em resumo, é a responsável por apresentar os dados resultantes do Model ao usuário. Por exemplo, uma Apresentação poderá ser um local administrativo onde os administradores se logam em uma aplicação. Cada administrador poderá visualizar uma parte do sistema que outro não vê.

- **CONTROLLER:** O Controller (Controlador) é o objeto que responde as ordens executadas pelo usuário, atuando sobre os dados apresentados pelo modelo, decidindo como o Modelo deveria ser alterado ou deveria ser revisto e qual Apresentação deveria ser exibida. Por exemplo, o Controlador recebe um pedido para exibir uma lista de clientes interagindo com o Modelo e entregando uma Apresentação onde esta lista poderá ser exibida.

O modelo MVC é uma forma de desenvolvimento que ajuda na manutenção do sistema, um padrão muito aceito no desenvolvimento de aplicações Java, principalmente no de aplicações escritas para a Web.

A separação lógica da aplicação nestas partes assegura que a camada Modelo não sabe nada praticamente do que é exibido; restringida por representar as partes de componentes do problema que é resolvido pela aplicação. Igualmente, a camada de Apresentação só está relacionada a exibir os dados e não com o implementar lógica de negócios que é controlada pela camada Modelo. O Controlador, como um gerenciador de tráfego, dirige as apresentações a serem exibidas e com as devidas mudanças de dados e recuperações vindas da camada Modelo.

O MODEL 1

A primeira arquitetura, conhecida como Model 1, é muito comum no desenvolvimento de aplicações Web, chamada de page-centric. Esta arquitetura fornece o modo mais fácil de reunir uma aplicação Web. Envolve simplesmente a construção de uma aplicação como um conjunto de páginas JSP.

A sucessão de eventos explicada neste exemplo é simples:

1. O usuário pede uma página de Web—por exemplo, a página principal, index.jsp.
2. O container Servlet executa a lógica contida na página index.jsp como também inclui páginas para que se possa apontar. Esta execução pode incluir a recuperação de dados de um banco de dados ou outras funções que satisfaçam à lógica de negócios. Os JavaBeans fornecem as representações de dados dentro da página JSP.
3. Unido junto à lógica de negócios da página, serão confeccionadas e apresentadas o HTML ao usuário.
4. Como resultado do processo, é construído o HTML final e exibido ao usuário.

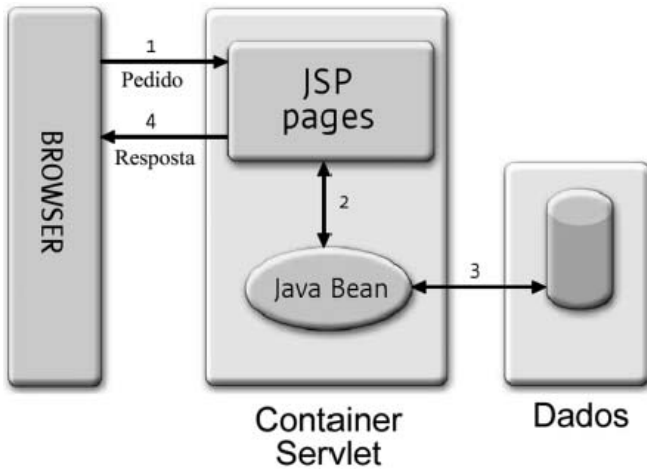


FIGURA 4.13 - ARQUITETURA MODEL 1

O MODEL 2

O Model 1, é indicada para uma aplicação pequena, que contém um limitado número de usuários e possui pouca lógica de negócios, principalmente por ser simples e efetiva. Porém, em uma aplicação mais complexa, onde a lógica de negócios não só é mais detalhada, mas a lógica de exibição necessária também é significativamente grande, uma arquitetura de desenvolvimento baseada no Modelo 1 fará com que seja um tanto bagunçado o montante de códigos desenvolvidos. Quando você coloca a lógica de negócios em um modelo simples de desenvolvimento, uma repetição de código acaba ocorrendo (isso é muito comum no desenvolvimento de outras linguagens de programação Web cujos conceitos de desenvolvimento não estão fortemente agregados). Isso impossibilita uma rápida manutenção e evidentemente, em um crescimento da aplicação, não haverá uma possível extensão. E isso porque não estamos contando com o fator de testes.

Desafiado por estas desvantagens óbvias, os desenvolvedores identificaram uma arquitetura mais sofisticada que usa Servlets e páginas JSP. Esta arquitetura foi batizada de Model 2 (Modelo 2), que está baseada em uma adaptação da arquitetura MVC. Nessa implementação, um Servlet é usado como um *Controlador*, recebendo pedidos do usuário, enquanto efetuando mudanças no *Modelo*, e fornecendo a *Apresentação* ao usuário.

As apresentações ainda implementadas nesta arquitetura usam páginas JSP, mas a lógica que elas contêm é só a de exibir a interface ao usuário. A camada de Modelo foi encapsulada em objetos Java.

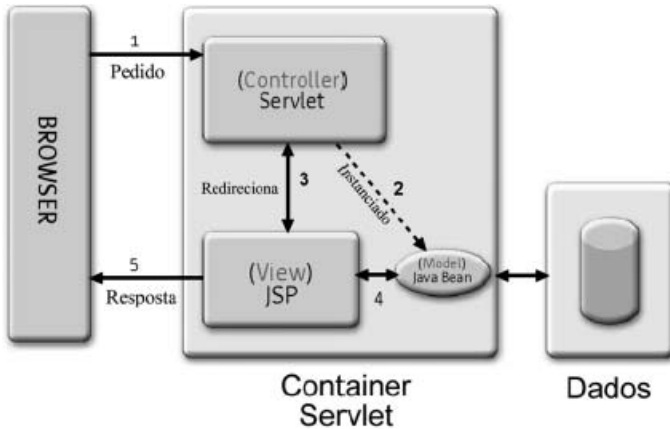


FIGURA 4.14 - ARQUITETURA MODEL 2

A seguir você tem uma explicação do que acontece nesse modelo de desenvolvimento:

1. Um pedido é feito a um Servlet através da URL. No pedido há uma indicação do trabalho a ser executado. Por exemplo, na URL você pode ter algo como: `/Livraria?action=mostrarAutores`, onde **action** representa o trabalho que a camada Controlador deve empreender.
2. A camada *Controlador* recebe o pedido e determina o trabalho baseando-se no pedido. Essa camada executa chamadas à camada *Modelo* que empreende a lógica de negócios exigida.
3. A camada *Modelo* é instruída a fornecer uma lista de objetos de Livros pelo *Controlador*. Este, por sua vez, pode acessar algum tipo de camada persistente, como um banco de dados.
4. Para a camada *Controlador* é proporcionada a lista de objetos de Livros para serem exibidos na camada de *Apresentação*. A camada *Controlador* também determina a apresentação apropriada para fornecer ao usuário. Usando um

despachante de pedidos, o Servlet pode fornecer a lista de objetos Livros à camada de *Apresentação* selecionada (página JSP mais indicada).

5. A camada de *Apresentação* tem uma referência agora aos dados fornecidos e faz a exibição da lista conforme sua lógica definida.

O HTML gerado no resultado desse processo é fornecido em resposta ao usuário.

Embora esta explicação técnica possa parecer complexa ou confusa a um iniciante, utilizar o Model 2 para desenvolvimento é de simples compreensão e traz muita vantagem no momento da manutenção.

O PADRÃO DAO (DATA ACCESS OBJECT)

O padrão DAO (Data Access Object) é o padrão mais utilizado para acesso a dados contidos em um banco de dados.

Sempre que você precisa acessar um banco de dados que está mantendo seu modelo de objetos, é melhor empregar o padrão DAO. O Padrão DAO fornece uma interface independente, no qual você pode usar para persistir objetos de dados. A idéia é colocar todas as funcionalidades encontradas no desenvolvimento de acesso e trabalho com dados em um só local, tornando simples sua manutenção.

Tipicamente um DAO inclui métodos para inserir, selecionar, atualizar e excluir objetos de um banco de dados. Dependendo de como você implementa o padrão DAO, você poderá ter um DAO para cada classe de objetos em sua aplicação ou poderá ter um único DAO que é responsável por todos os seus objetos.

CRIANDO UM PROJETO COM PADRÕES

Crie um novo projeto no NetBeans IDE. Chame-o de **TrabComPadroes**. Adicione as bibliotecas: JDBC **MySQL** e **JSTL**.

A FÁBRICA DE CONEXÃO

Você vai começar a construir a classe de conexão, no qual acessará o banco de dados criado.

Com o direito do mouse sobre o projeto, vá a **New** e clique em **Java Class**. Se preferir, **Ctrl + N** e na caixa de diálogo **New File** selecione **Java Classes** em **Categories** e clique em **Java Class** em **File Types**.

Digite **ConnectionFactory** em **Class Name** e **br.com.integrator.util** em **Package**. Clique no botão **Finish** para confirmar.

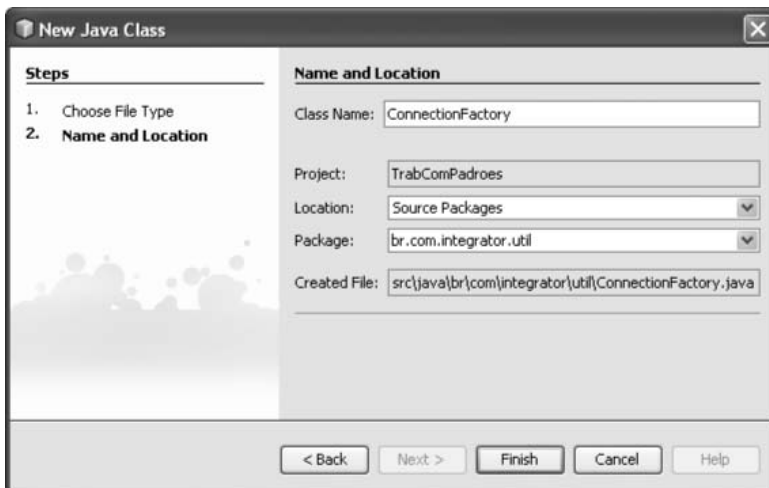


FIGURA 4.17 – CRIAÇÃO DA CLASSE CONNECTIONFACTORY

Observe que ao renomear a classe, o construtor também foi renomeado automaticamente, o que é altamente recomendável utilizar sempre esta ferramenta para tais situações.

TESTANDO SEU DAO

Uma boa prática no desenvolvimento é testar classes para verificar a assertividade e conseqüentemente evitar erros futuros por falhas ocorridas em etapas preliminares. A classe DAO desenvolvida faz o que chamamos de CRUD (Create – Read – Update – Delete), ou seja, ela opera em quatro situações no banco de dados. Estas operações devem ser testadas para saber se estão corretamente funcionais ou se há pequenas falhas não percebidas pelo desenvolvedor.

O exemplo que será feito é simples, apenas para testar a funcionalidade do DAO, sem a intenção de obter uma biblioteca de terceiros não suportada nativamente pelo NetBeans.

Quando você cria um projeto, automaticamente o NetBeans adiciona em sua árvore um diretório chamado de **Test Libraries** contendo duas bibliotecas: **JUnit 3.x** e **JUnit 4.x**. A biblioteca mais moderna possui suporte a annotations (Java SE 5 ou superior e Java EE 5).

JUnit, para quem não conhece, é um framework open source, criado para fazer testes automatizados, chamados de **test cases**.

Este framework foi idealizado por Erich Gamma, um dos autores dos populares padrões de projeto e Kent Beck, pai do XP (Extreme Programming). Para desenvolvê-lo, usaram o conceito de testes unitários (unit tests) em Java criando uma ferramenta chamada JUnit, baseada em SUnit, do SmallTalk.

Para iniciar a criação da classe de testes, selecione a classe **AutorDAOImp** e vá até o menu **Tools** e clique no item **Create JUnit Tests (Ctrl+Shif+U)**. O mesmo pode ser feito com o direito do mouse sobre a classe, na janela **Projects**, e no menu de contexto selecionar o item **Tools** e clicar em **Create JUnit Tests**.

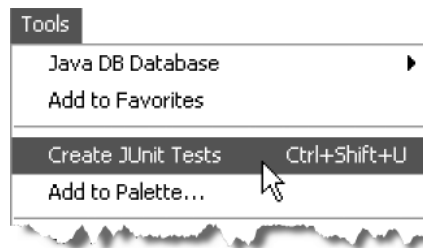


FIGURA 4.30 - SELEÇÃO DA OPÇÃO CREATE JUNIT TESTS NO MENU PRINCIPAL TOOLS

Como o projeto possui duas versões do JUnit, haverá uma pergunta ao criar o teste unitário, para escolha da versão desejada. Para o exemplo, você vai escolher o JUnit 4.x

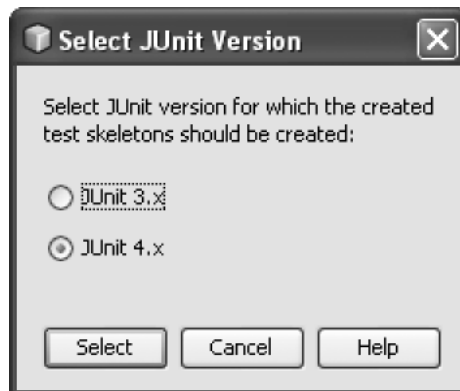


FIGURA 4.31 – SELECIONANDO A VERSÃO DO JUNIT A SER USADA

O NetBeans mantém em Test Libraries apenas a versão escolhida, removendo a anterior. Outra caixa de diálogo surgirá, **Create Tests**, onde no campo **Class Name** haverá o nome da classe e o pacote que será criado. Se desejar, pode alterá-los (o que para o caso seria desperdício de tempo). Em Location é mostrado o local onde será criada a classe, que na janela **Projects** ficará em **Test Packages**.



FIGURA 4.35 – EXIBIÇÃO DOS AUTORES CADASTRADOS

A **Tabela 4.4** a seguir demonstra seus atributos:

TABELA 4.4 - ATRIBUTOS DA ACTION `<fmt:formatDate />`

ATRIBUTOS	DESCRIÇÃO
type	Pode ser time, date ou both. Usado para imprimir somente a hora, data ou ambos.
dateStyle	Pode ser usado short, medium, long ou full (ou default). Usado para imprimir a data.
timeStyle	Pode ser short, medium, long ou full (ou default). Usado para imprimir a hora.
value	Um valor do tipo java.util.Date usado para renderizar a data e a hora.

Para excluir um autor, o Servlet novamente deve ser chamado, enviando dois comandos: **exc** e o **ID** do autor que deseja remover do banco de dados. Para executar essa ação, a query string **cmd=exc&id=ID do autor** é transmitida para o Servlet para que esse execute o determinado.

CADASTRANDO NOVOS AUTORES

Aproveitando o DAO, você vai criar um formulário (**Listagem 4.15**) que irá cadastrar os dados do novo autor que deseja armazenar:

Outro detalhe é que novamente JSTL é usado aqui para simplificar a formatação da data de nascimento do autor.



ID:	1
Nome:	Edson Gonçalves
E-mail:	edson@integrator.com.br
Nascimento (dd/mm/aaaa):	11/04/1978

Atualizar

FIGURA 4.36 – FORMULÁRIO DE ATUALIZAÇÃO PREENCHIDO

POOL DE CONEXÕES

Quando uma aplicação Web acessa um banco de dados remoto, esse acesso pode ser feito por uma conexão JDBC, como visto anteriormente. Tipicamente, uma conexão de JDBC física é estabelecida entre a aplicação cliente e o servidor de banco de dados por uma conexão TCP/IP.

Pool de conexões reduzem expressivamente o tempo de conexões estabelecidas criando uma conexão física no início do sistema.

Quando uma aplicação requerer uma conexão, uma destas conexões físicas é fornecida a esta aplicação. Em um sistema comum, sem o pool de conexão, quando a aplicação termina de usar a conexão, esta a desconecta, como feito anteriormente usando o método **close()**. Porém, no caso de uma conexão física, essa é devolvida somente para o pool de conexões, onde espera o próximo pedido da aplicação para um novo acesso ao banco de dados.

CONFIGURANDO POOL DE CONEXÕES

No NetBeans IDE, você pode também configurar Pool de conexões. Para o exemplo, você vai configurar um no **GlassFish**.

The background of the entire page is a complex, abstract image. It features a dense network of glowing, metallic-looking lines and structures that resemble a futuristic data center or a complex circuit board. The lines are primarily horizontal and vertical, creating a sense of depth and perspective. The overall color palette is dark, with bright highlights from the glowing elements, giving it a high-tech, digital feel.

PARTE 2

JAVA EE5: AVANÇANDO NO DESENVOLVIMENTO DE APLICAÇÕES WEB

CAPÍTULO 5

JAVASERVER FACES

JavaServer Faces é um framework desenvolvido pela Sun Microsystems, e é parte integrante da tecnologia do mundo de Java EE.

O framework JavaServer Faces foi desenhado para facilitar o desenvolvimento de aplicações Web através de componentes de interface de usuário (GUI) e conecta estes componentes a objetos de negócios.

O JavaServer Faces utiliza o paradigma MVC para trabalhar com sua apresentação e navegação de dados. Sua utilização é recomendada pela Sun Microsystems para o desenvolvimento Web com Java na atualidade.

Neste capítulo será apresentado:

- Os princípios de JavaServer Faces no NetBeans IDE;
- Como desenvolver aplicações JSF com acesso a banco de dados;
- Como validar erros e criar navegação por entre as páginas;
- A alteração das mensagens padrões de JSF.

UM PROJETO JAVASERVER FACES

Para que você se acostume com o desenvolvimento de páginas JSF, seu primeiro exemplo será pequeno e simples. Com este exemplo, você vai aprender a base característica do desenvolvimento de JavaServer Faces utilizando o NetBeans IDE.

Inicie a criação de um projeto Web. Digite o nome do seu projeto. Se quiser seguir o livro, chame-o de **PrimProjJSF**. Escolha seu servidor e clique no botão **Next** para prosseguir.

Na terceira etapa, marque a opção **JavaServer Faces** em **Select the frameworks you want to use in your web application**.

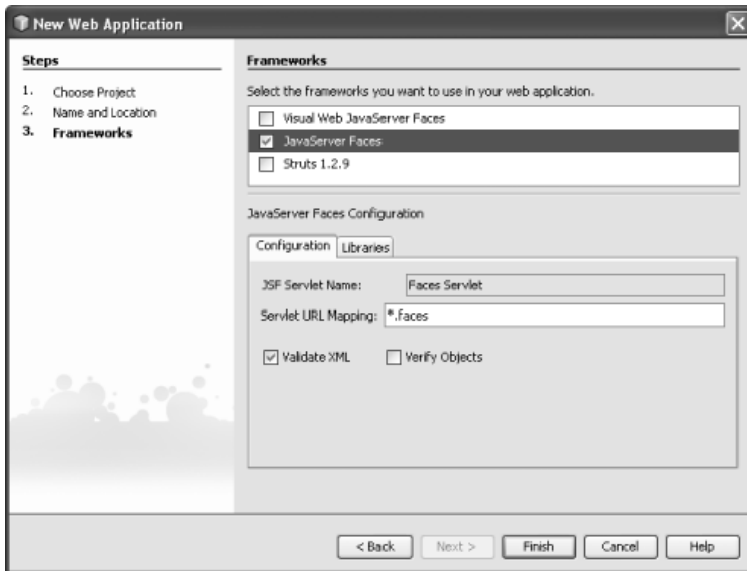


FIGURA 5.1 – CONFIGURANDO A TERCEIRA ETAPA DO ASSISTENTE DE CRIAÇÃO

Em **Servlet URL Mapping** você tem por padrão a configuração **/faces/***, indicando que o acesso às aplicações escritas em JSF serão antecipadas da palavra **faces/**, o que teríamos, por exemplo, um link como o mostrado a seguir:

<http://localhost:8080/PrimProjJSF/faces/pagina.jsp>

Para este caso, você vai mudar para ***.faces**, ou seja, no acesso a suas aplicações escritas em JavaServer Faces, as páginas conterão a extensão **.faces**, como o link de exemplo mostrado a seguir:

<http://localhost:8080/PrimProjJSF/pagina.faces>

Ao terminar a configuração, basta clicar no botão **Finish**. O NetBeans IDE cria duas páginas JSP (`forwardToJSF.jsp` e `welcomeJSF.jsp`), onde a segunda contém um pequeno trecho de JavaServer Faces.

Se você estiver usando o Tomcat, expandindo o nó de **Libraries** você verá que o NetBeans IDE colocou todas as bibliotecas necessárias para o desenvolvimento de JSF. No caso do GlassFish, tais bibliotecas já estão embutidas no servidor, o que não será necessário esta adição.

Em uma aplicação que utilize o framework JavaServer Faces na versão 1.2, há a necessidade de oito arquivos do tipo JAR (bibliotecas) para que sejam acessados pelo servidor ou pela aplicação Web:

Quatro JARs Commons:

1. **commons-beanutils.jar**
2. **commons-collections.jar**
3. **commons-digester.jar**
4. **commons-logging.jar,**

Dois JARs JSF:

1. **jsf-api.jar**
2. **jsf-impl.jar**

Dois JARs JSTL:

1. **jstl.jar**
2. **standard.jar**

Em um ambiente de produção, estas bibliotecas devem estar disponíveis em sua aplicação no diretório **lib**, encontrado em **WEB-INF**, caso o servidor não as disponibilize nativamente.

OBSERVAÇÃO: O JavaServer Faces atualmente está na versão 1.2, suportado apenas pelos servidores Java EE 5. Se o usuário tiver disponível um Servlet Container como o Tomcat versão 5.5, que implementa Java EE 1.4 (J2EE 1.4), será necessária a configuração do JavaServer Faces na versão 1.1 (anterior a atual).

TRABALHANDO COM JAVASERVER FACES

Assim que configurado, você apenas precisa criar um exemplo para começar a entender como funciona o JavaServer Faces.

O primeiro exemplo terá apenas um campo para o envio de nomes. Este exemplo contará com uma validação, para o caso do usuário entrar com um valor inválido, não alfabético, retornando um erro.

Caso retorne o erro, além de manter preenchido o campo digitado, também mostrará uma mensagem, solicitando a alteração.

CRIANDO O JAVA BEAN

O JavaBean mostrado a seguir será o responsável pela comunicação entre a página inicial, que o usuário digitará o nome, em um formulário, e a página que resultará na mensagem de boas vindas, caso esta seja submetida com sucesso.

No NetBeans IDE, crie uma nova classe e chame-a de **NomeBean**. Coloque no pacote **br.com.integrator** e clique no botão **Finish** para confirmar.

```

<body>
  <f:view>
    Olá <h:outputText value="#{NomeBean.nome}"/> <br />
  </f:view>
</body>
</html>

```

A saída do nome, como resultado positivo, vindo do Bean, é feito pela tag **<h:outputText />**.

A página **forwardToJSF.jsp** poderá ser alterada a seu gosto, embora ela esteja pronta com um redirecionamento para **welcomeJSF.jsp**. Rode a aplicação e tente enviar um texto contendo um caractere numérico. Em seguida, corrija e envie os dados corretamente, conforme as regras estabelecidas no Bean.

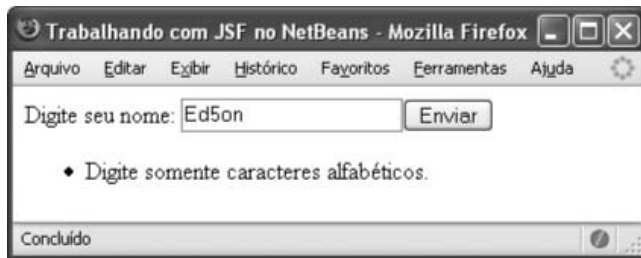


FIGURA 5.10 – ERRO APRESENTADO PELO ENVIO INCORRETO DE DADOS

CONHECENDO MELHOR O JAVASERVER FACES

Agora que você já fez sua primeira aplicação em JSF, é mais fácil de entender os serviços que o framework JSF oferece ao desenvolvedor. Como você pôde ver, o framework JSF é responsável por interagir com o usuário (cliente), e fornece ferramentas para criar uma apresentação visual, a parte lógica e a lógica de negócios de uma aplicação Web. Porém, o escopo de JSF é restrigido à camada de apresentação. A persistência de banco de dados e outras conexões de back-end estão fora do escopo de JSF.

CRIANDO UM EXEMPLO UTILIZANDO BANCO DE DADOS E JSF

A idéia neste exemplo é demonstrar as principais características encontradas em uma página JavaServer Faces acessando um banco de dados. Este exemplo acessará o banco de dados MySQL, executando o famoso CRUD (Create, Read, Update and Delete).

Além disso, o código de acesso ao banco de dados não será necessário refazer, uma vez que ele será o DAO criado no **Capítulo 4** deste livro.

Crie um novo projeto web. Se desejar, chame-o de **UtilJSFComDAO**. Não se esqueça de selecionar o framework JavaServer Faces na terceira etapa do assistente. Quanto ao **Servlet URL Mapping**, você pode alterar ou manter como está. No caso do livro, alterei para ***.jsf**.

Adicione a biblioteca JDBC do MySQL ao projeto. Do projeto que utiliza o padrão DAO, feito no capítulo anterior, copie as classes e seus respectivos pacotes.

O BEAN DE COMUNICAÇÃO COM AS PÁGINAS JSF

Para a comunicação com as páginas JavaServer Faces que serão construídas, um bean será construído, um controlador com o intuito de se comunicar com as classes já existentes.

Para desenvolver este controlador, crie uma classe chamada de **AutorController** em um pacote chamado de **br.com.integrator.controller**. Altere esta classe como mostrado na **Listagem 5.7** a seguir.

LISTAGEM 5.7 – ALTERAÇÃO EM AUTORCONTROLLER.JAVA

...

```
public class AutoresController {
```

```
    private Autor autor;
```

```
    private DataModel model;
```

```

        value="Cadastrar novo Autor"/>
    </h:form>

</f:view>

</body>
...

```

O datagrid desenvolvido utiliza a tag JSF `<h:dataTable/>`, que recebe todos os dados existentes na tabela autores do banco de dados através do método **getTodos()**, da classe `AutorController`, no atributo **value**. Com a tag `<h:commandLink/>` você pode chamar os métodos **editar()** ou **excluir()**.

A tag JSF `<f:convertDateTime/>`, adicionada a tag JSF `<h:outputText/>` é responsável por renderizar corretamente a formatação da data, trazida do banco de dados.

O último link criado pela tag `<h:commandLink/>` está fora do `dataTable`, por não interagir diretamente sobre a mesma, chamando o método **novoAutor()** para criar um novo cadastro de Autores.

O resultado desta página será como mostrado na **Figura 5.13** a seguir:



Mostrar Autores				
ID do Autor	Nome	E-mail	Nascimento	Excluir Autor
1	Edson Gonçalves	edson@integrator.com.br	11/04/1978	Excluir

[Cadastrar novo Autor](#)

FIGURA 5.13 – JAVASERVER FACES DATATABLE RENDERIZADO COM OS DADOS

O resultado é a página com o formulário similar ao mostrado na **Figura 5.14** a seguir:

FIGURA 5.14 – FORMULÁRIO DE CADASTRO E ATUALIZAÇÃO DE DADOS

ATUALIZANDO UM AUTOR CADASTRADO

Na página onde contém todos os autores cadastrados existe na primeira coluna o ID do autor com um link. Este link, gerado pela tag JSF `<h:commandLink/>` contém a chamada ao método **editar()** da classe **AutorController**. Este método, na sua classe de origem, chama outro método, o **getAutorFromEditOrDelete()**, responsável pela captura da linha em questão no **DataModel**. Essa linha é pega pelo método **getRowData()**, explicado anteriormente.

Assim que pego a linha em questão, escolhida na hora do clique sobre o link, o método **editar()** se responsabiliza de preencher o **JavaBean Autor** com os valores captados e envia a String **editar** para que a navegação entre as páginas ocorram.

CONFIGURANDO A NAVEGAÇÃO

A maior dificuldade neste exemplo será na criação da navegação do aplicativo. Não que esta complicação seja por causa do que foi feito, mas sim por poder causar confusão, na construção da navegabilidade das páginas. No arquivo **faces-config.xml** o resultado final da navegação necessária da aplicação será como mostrada a **Figura 5.15** a seguir.

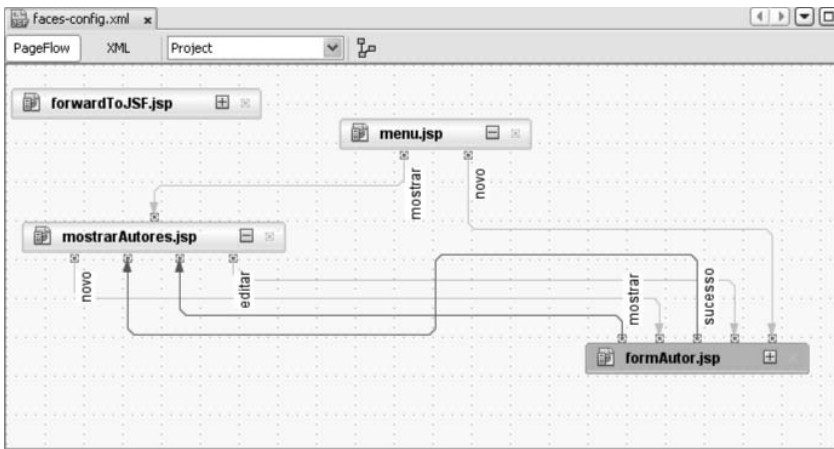


FIGURA 5.15 – NAVEGAÇÃO FINAL CONFIGURADA EM FACES-CONFIG.XML

A seguir você tem a **Listagem 5.12** de como deverá ficar sua navegação no formato XML.

LISTAGEM 5.12 – RESULTADO FINAL DA NAVEGAÇÃO EM FACES-CONFIG.XML

...

```
<!-- Cria um novo cadastro ou edita,
      atraves de mostrarAutores.jsp -->
<navigation-rule>
  <from-view-id>/mostrarAutores.jsp</from-view-id>
  <navigation-case>
    <from-outcome>novo</from-outcome>
    <to-view-id>/formAutor.jsp</to-view-id>
  </navigation-case>
  <navigation-case>
    <from-outcome>editar</from-outcome>
    <to-view-id>/formAutor.jsp</to-view-id>
  </navigation-case>
```

Com adição de CSS, você pode ter uma página similar ao da **Figura 5.24** a seguir:

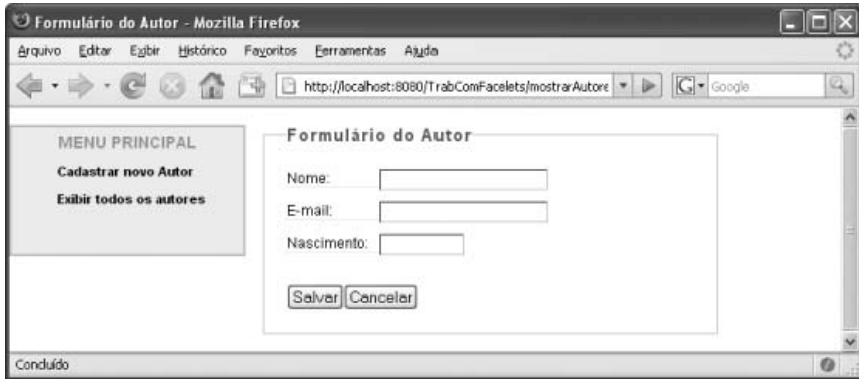


FIGURA 5.24 – FORMULÁRIO UTILIZANDO FACELETS COM FORMATAÇÃO CSS

EXIBINDO OS AUTORES COM FACELETS

Crie uma nova página Facelets, em **Facelets Template Client**. Chame-a de **mostrarAutores.xhtml** e selecione em **Template**, clicando no botão **Browse**, o arquivo **template.xhtml**. Marque **<ui:composition>** em **Generated Root Tag** e clique no botão **Finish**.

Adicione em **<ui:composition/>** as chamadas as bibliotecas JavaServer Faces. Altere os demais itens como mostrado na **Listagem 5.19** a seguir:

LISTAGEM 5.19 – ALTERAÇÕES NO ARQUIVO MOSTRARAUTORES.XHTML

...

```
<ui:define name="titulo">
```

```
    Autores Cadastrados
```

```
</ui:define>
```

```
<ui:define name="conteudo">
```

```
    <div id="autores">
```

```
        <h:dataTable value="#{autorC.todos}"
```

```

</f:facet>
<h:outputText
    value="#{item.nascimento}">
    <f:convertDateTime pattern="dd/MM/yyyy" />
</h:outputText>
</h:column>
<h:column>
    <f:facet name="header">
        <h:outputText value="Excluir Autor"/>
    </f:facet>
    <h:commandLink
        action="#{autorC.excluir}"
        value="Excluir"/>
    </h:column>
</h:dataTable>
</div>
</ui:define>
...

```

O resultado poderá ser algo similar a **Figura 5.25** a seguir:

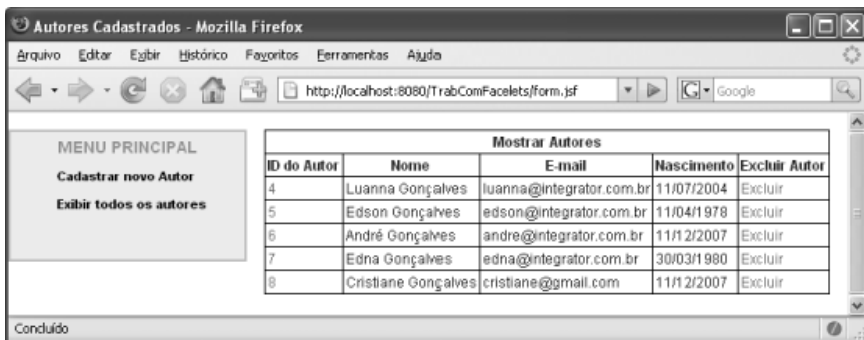


FIGURA 5.25 – PÁGINA MOSTRARAutores.xhtml FORMATADA COM CSS

CAPÍTULO 6

EJB 3 E JAVA PERSISTENCE API

Enterprise JavaBeans, ou somente EJB, é uma das peças-chaves da plataforma Java EE, com uma arquitetura de componentes para aplicações de negócios distribuídos, que roda em um servidor de aplicações. Definido pela Sun Microsystems, as aplicações escritas usando a arquitetura EJB são escalonáveis, portáteis, transacionais e seguras. Além disso, garantem alta disponibilidade e neutralidade do tipo de cliente.

Com a especificação do EJB3, na introdução da JPA (Java Persistence API), ficou muito popular o uso de JPQL, Java Persistence Query Language, onde o mapeamento de objeto/relacional foi padronizado na plataforma Java.

Neste Capítulo será apresentado as facilidades do NetBeans IDE no desenvolvimento com EJB 3, utilizando seus assistentes através de exemplos.

Os seguintes tópicos serão apresentados:

- Desenvolvimento de aplicações Enterprise;
- Gerando EJB 3;
- Como funciona a API de persistência do Java EE 5 (JPA);
- O significado das anotações;
- A linguagem JPQL;
- Acessando EJB através de páginas JavaServer Faces.

CRIANDO UM PROJETO JAVA EE 5

O desenvolvimento de um projeto de Enterprise Application roda em torno do servidor de aplicações GlassFish. Embora o JBoss, no momento em que escrevo este livro, possua suporte oficialmente ao EJB 3, o GlassFish é o mais indicado para uso com a IDE. Inicie criando um novo projeto no NetBeans. Na caixa de diálogo **New Project**, vá em **Enterprise**, em **Categories** e selecione **Enterprise Application**, em **Projects**. Clique em **Next** para prosseguir.

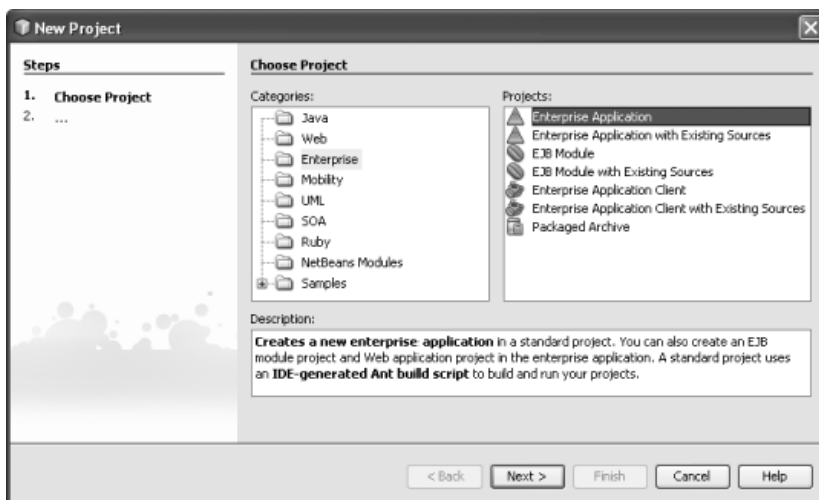


FIGURA 6.1 – INICIANDO UM PROJETO ENTERPRISE APPLICATION

Na segunda etapa, chame o projeto de **EALivraria**. O servidor será o **GlassFish** (por implementar o Java EE 5). Observe que estão selecionadas duas opções: **Create EJB Module** e **Create Web Application Module**.

Clique no botão **Finish** para terminar.

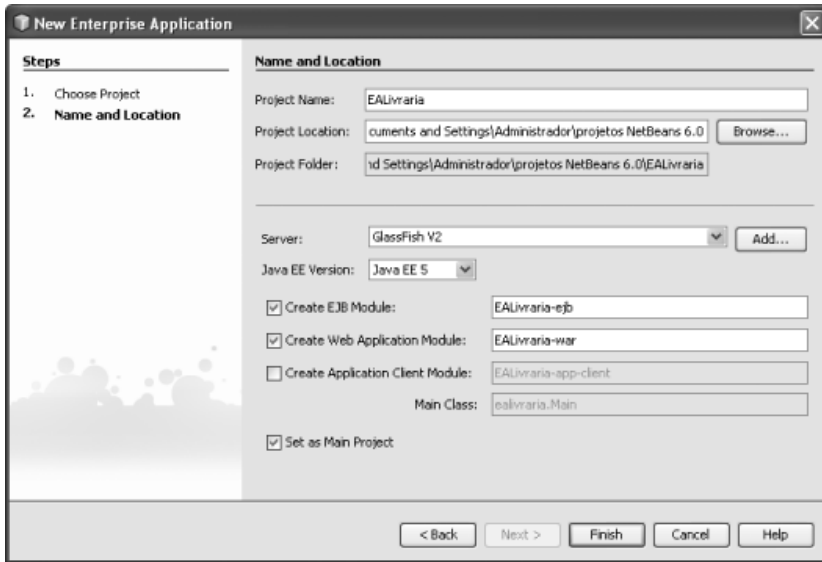


FIGURA 6.2 – DEFINIÇÃO DO NOME E SERVIDOR DO PROJETO

O resultado ao final do assistente é a criação de uma estrutura de três projetos: **EALivraria-ejb**, que gerará o EJB-JAR; o **EALivraria-war**, que gera o WAR e o **EALivraria** que gera o EAR.



FIGURA 6.3 – RESULTADO FINAL DO PROJETO GERADO PELO NETBEANS

ACESSANDO SEU EJB

Clique com o direito do mouse sobre **EALivraria-war** e no menu de contexto selecione **Servlet**, em **New**. Chame este Servlet de **AcessaEJB** e coloque um pacote (br.com.integrator.web).

Após a criação do Servlet, clique com o direito do mouse dentro no Editor, e selecione no menu de contexto o item **Call Enterprise Bean**, em **Enterprise Resources**.



FIGURA 6.7 – CHAMANDO UM ENTERPRISE BEAN PELO MENU DE CONTEXTO

Na caixa de diálogo **Call Enterprise Bean**, selecione seu EJB. Confirme a caixa de diálogo.

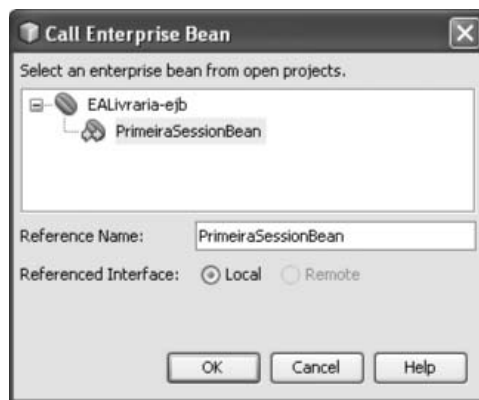


FIGURA 6.8 – INVOCANDO SEU EJB

CRIANDO A SESSION BEAN

O segundo passo a proceder na construção do EJB com acesso a dados será criar uma Session Façade para se comunicar com a Entity Bean desenvolvida.

Crie um novo arquivo e selecione a categoria **Persistence** e o tipo **Session Beans For Entity Classes**. Clique no botão **Next** para prosseguir.

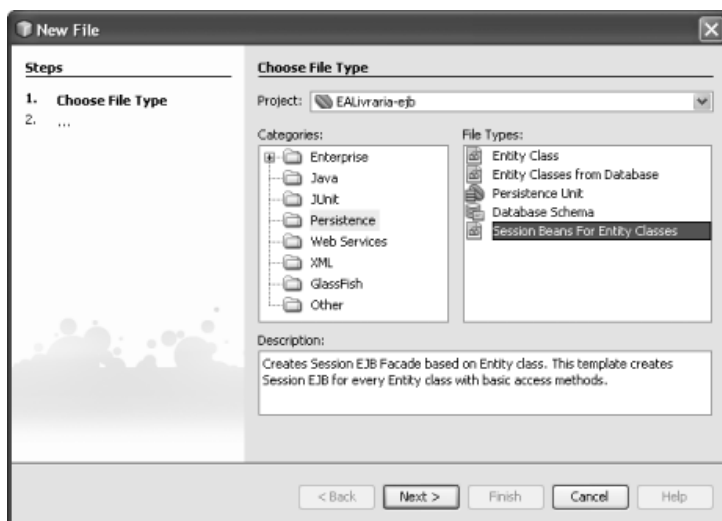


FIGURA 6.15 – SELEÇÃO DE SESSION BEANS FOR ENTITY CLASSES

No segundo passo, selecione a entidade **Autor**, clicando no botão **Add**. Clique no botão **Next**.

A JAVA PERSISTENCE QL

A Java Persistence API (JPA) suporta duas linguagens de consultas (queries) para recuperar entidades e outros dados persistentes do banco de dados. A linguagem primária é a **Java Persistence Query Language (JPQL)**. Esta é uma linguagem de consultas independente de banco de dados e opera no modelo de entidades lógicas, ao contrário do modelo de dados físico. As consultas também podem ser expressas em SQL. No caso dos códigos gerados pelos assistentes do NetBeans, a query criada está na linguagem JPQL.

Observando atentamente a query gerada, no método `findAll()` do CRUD, você verá:

select object(o) from Autor as o

Perceba que é similar ao SQL que conhecemos. A JPQL usa uma sintaxe similar à SQL, onde é possível dar ao desenvolvedor experiente com instruções SQL a vantagem de escrever as queries. A diferença fundamental entre SQL e JPQL está na seleção de uma tabela, onde a entidade do modelo da aplicação é especificada ao invés da tabela propriamente dita.

OBSERVAÇÃO: Na utilização de um container Servlet, como o Tomcat, a criação de instância do gerenciador da persistência é feita pela classe abstrata **EntityManagerFactory**. Essa classe é criada pela anotação **@PersistenceUnit**, que lê as configurações existentes no arquivo de configuração **persistence.xml**.

```
@PersistenceUnit(unitName = "livraria")  
private EntityManagerFactory emf;
```

É evidente que algumas das vantagens existentes em um EJB 3 serão perdidas, mas em um contexto geral, utilizar EJB 3 Persistence (JPA) é similar.

UTILIZANDO JAVASERVER FACES PARA ACESSAR O EJB

Para utilizar o JavaServer Faces para acessar o EJB criado com acesso a banco de dados, o projeto precisa ter as configurações de JavaServer Faces.

Clique com o direito do mouse sobre o projeto **EALivraria-war** e selecione no menu de contexto o item **Properties**. Na caixa de diálogo **Project Properties**, selecione **Frameworks** em **Categories**. Em **Used Frameworks**, clique no botão **Add**. Selecione **JavaServer Faces** e confirme tudo.

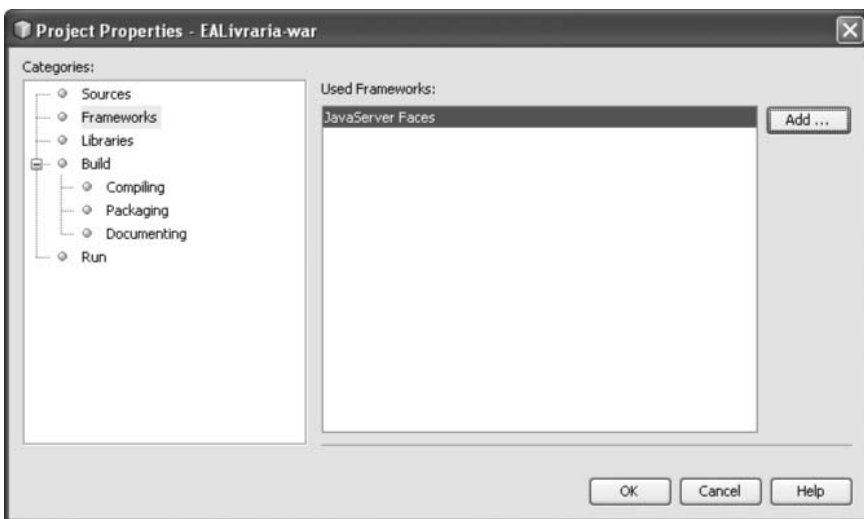


FIGURA 6.18 – ADIÇÃO DO FRAMEWORK JAVASERVER FACES

Abra o projeto **UtilJSFComDAO**, criado no **Capítulo 5**, e copie todas as páginas JSP para **EALivraria-war**. Copie também o arquivo **faces-config.xml** e o pacote **br.com.integrator.controller** com a classe **AutorController**.

RODANDO O EJB 3 PERSISTENCE EM UM CONTAINER J2EE 1.4

Para fazer isso com o NetBeans IDE, crie um projeto normalmente. Escolha as opções padrão para gerar um projeto Web, incluindo seu container J2EE 1.4.

Na segunda etapa do assistente, você deve desmarcar a opção Set Source Level to 1.4. Os demais itens serão similares ao já feito até o momento na criação de um projeto utilizando JavaServer Faces.

O arquivo **persistence.xml** é ligeiramente diferente, pois não possui uma integração completa, como ocorre com **GlassFish**.

Um container servlet como o Tomcat 5.5, por exemplo, também não implementa a versão Java EE 5 e, portanto, precisa da biblioteca do TopLink Essentials (**toplink-essentials.jar** e **toplink-essentials-agent.jar**).

ATENÇÃO: Caso precise saber como usar JNDI no Tomcat 5.5 com o arquivo persistence.xml, no CD-ROM, pegue o capítulo extra que trata do assunto sobre esta versão do container Servlet.

CAPÍTULO 7

O VISUAL WEB JAVASERVER FACES

O Visual Web JavaServer Faces é um editor visual, no estilo WYSIWYG (*What You See Is What You Get*), baseado no **Java Studio Creator IDE**, da Sun Microsystems, criado para desenvolver páginas JavaServer Faces, como já diz seu nome. Isso significa que seu desenvolvimento é baseado em componentes arrastáveis que, com pouco código, o desenvolvedor pode criar páginas totalmente funcionais. Este capítulo foi escrito em forma de estudo de caso, desenvolvendo uma aplicação JavaServer Faces envolvendo o uso de diversos componentes e suas configurações, focando na etapa visual sem acesso a banco de dados.

Ao longo deste capítulo será apresentado:

- Os componentes principais e suas configurações;
- A comunicação entre componentes;
- A geração de navegação por entre páginas através da ligação de componentes;
- Como alterar ou mesmo criar novos estilos CSS para o aplicativo;
- Envio de e-mail com componente;
- Formulários virtuais;

UM PROJETO COM VISUAL WEB JAVASERVER FACES

Inicie a criação de um novo projeto pelo menu de contexto ou tecla **Ctrl + Shift + N**. Na caixa de diálogo **New Project**, vá em **Web**, nas categorias, e selecione **Web Application**, em **Projects**. Clique em **Next** para prosseguir.

Na segunda etapa, defina um nome para o seu projeto, em **Project Name**. Caso queira fazer com um nome idêntico ao usado no livro, coloque **DesComVisualWebApplication**. Selecione o **GlassFish** como servidor e clique no botão **Next** para prosseguir.

Na terceira etapa, em **Frameworks**, selecione **Visual Web JavaServer Faces**. Altere o pacote da aplicação, em **Default Java Package** – para o exemplo fora utilizado **br.com.integrator**. Altere para ***.faces** no campo **Servlet URL Mapping**. Clique no botão **Finish** para completar o assistente.

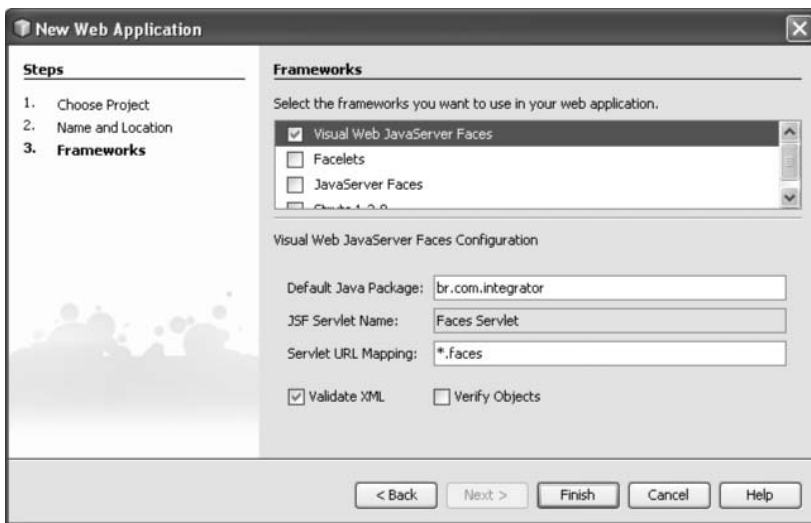


FIGURA 7.1 – SELEÇÃO E CONFIGURAÇÃO DO VISUAL WEB JAVASERVER FACES

VISÃO GERAL DO VISUAL WEB JAVASERVER FACES

Ao finalizar a criação do projeto, o NetBeans IDE abrirá as janelas necessárias para o trabalho com a ferramenta **Visual Web JavaServer Faces**.



FIGURA 7.4 – EXEMPLO DA PÁGINA INICIAL

ADICIONANDO UM TÍTULO E COR DE FUNDO A PÁGINA

Com um clique no editor visual (Design) da página, você pode alterar suas propriedades. Na janela **Properties**, altere a cor de fundo (**Background**) para as cores RGB **232,231,207**.

O título é adicionado a página pela propriedade **Title**. Digite **Aplicações com Visual Web JSF**.

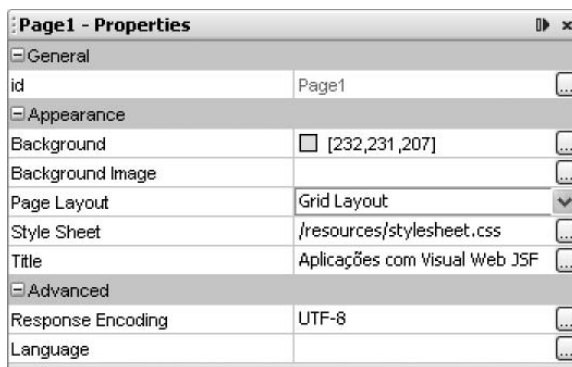


FIGURA 7.5 – PROPRIEDADES DA PÁGINA

Na caixa de diálogo **Configure Virtual Forms**, observe que todos os componentes selecionados que pertencerão ao seu formulário virtual estarão listados logo abaixo do título. Clique no botão **New**. Aparecerá uma linha contendo uma **cor**, um **nome** e mais dois itens. Em **Name** altere para **enviarForm**. Nos itens **Participate** e **Submit** altere para **Yes**, pois os mesmos participarão da submissão do formulário. Confirme clicando no botão **Apply**. Clique no botão **OK** para fechar a caixa de diálogo.

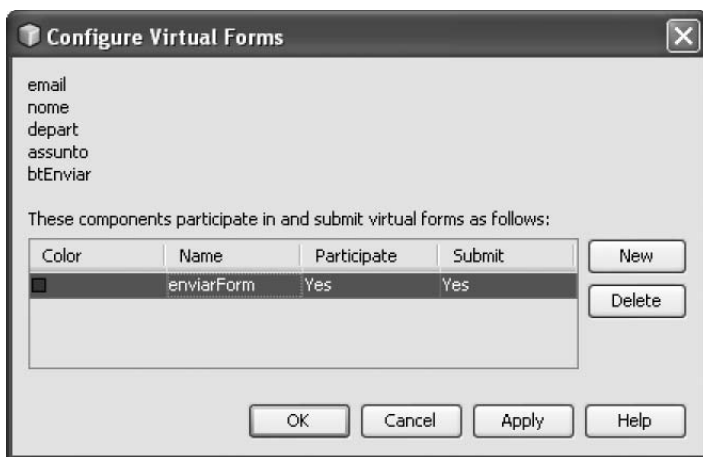


FIGURA 7.20 – VIRTUAL FORM DOS COMPONENTES QUE ENVOLVEM A SUBMISSÃO DO FORMULÁRIO

Selecione o botão **Limpar** e crie um formulário virtual também para ele. Chame de **limparForm**. Em **Submit** altere para **Yes** e confirme.



FIGURA 7.21 – FORMULÁRIOS VIRTUAIS SENDO EXIBIDOS ATRAVÉS DE SHOW VIRTUAL FORMS

CAPÍTULO 8

DESENVOLVENDO COM VISUAL WEB JSF USANDO BANCO DE DADOS

Integrar um banco de dados com uma página desenvolvida usando Visual Web JavaServer Faces é a tarefa mais comum para o desenvolvimento de uma aplicação Web.

Este capítulo continua o estudo de caso iniciado no capítulo anterior, adicionando a interatividade com banco de dados.

Além de completar o desenvolvimento da aplicação, será mostrado:

- Como criar formulários conectados a banco de dados gerando CRUDs;
- O upload de arquivos integrado a um formulário;
- A criação de consultas SQL complexas pelo editor;
- A configuração para paginação de dados;
- A adição de pesquisa ao sistema;
- A geração de segurança através de sessão para navegar na área administrativa.

O ACESSO A BANCO DE DADOS

O banco de dados utilizado será o livraria, o mesmo usado ao longo do livro. O princípio para acesso a dados usando Visual Web JSF é muito similar aos demais já utilizados, uma vez que você deve ter uma conexão criada na IDE para integração direta com os componentes.



FIGURA 8.1 – VINCULANDO A TABELA AO COMPONENTE TABLE

COMPREENDENDO COMO OS DADOS SÃO ACESSADOS

Observe na janela **Navigador** que agora há um componente chamado **livrosDataProvider** e em **SessionBean1**, **livrosRowSet**.

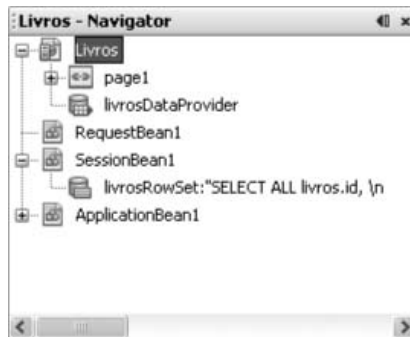


FIGURA 8.2 – NOVOS COMPONENTES NA PÁGINA LIVROS

Todos os componentes da seção Data Provider, encontrados na janela **Palette**, implementam a interface básica **DataProvider**, que fornece um caminho consistente para acesso a dados em um objeto **FieldKeys** usando campos chaves que correspondem aos seus nomes de propriedades. Com a interface **TableDataProvider**, podemos usar o conceito de acesso baseado em cursor (usando a linha em curso) e acesso aleatório (especificado por você através de **FieldKey** ou **RowKey**). No caso, **livrosDataProvider** é um **CachedRowSetDataProvider**, que implementa ambas as interfaces.

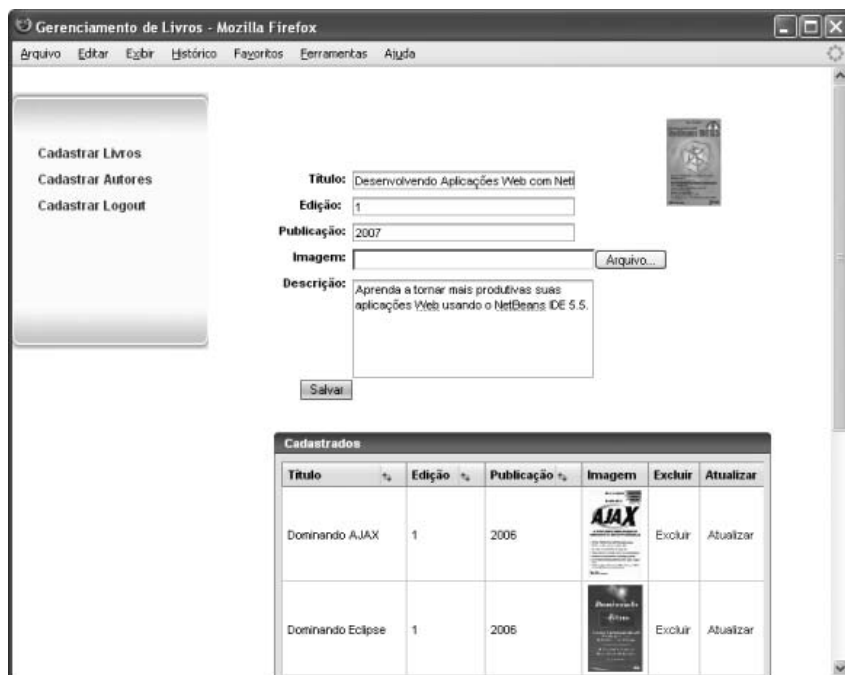


FIGURA 8.16 – RESULTADO FINAL DA PÁGINA LIVRO.JSP

CONFIGURANDO O TAMANHO DO ARQUIVO PARA UPLOAD

Quando é adicionado a uma página o componente **File Upload**, automaticamente se tem as configurações de seu filtro em **web.xml**. O filtro, no caso, utiliza a **classe com.sun.webui.jsf.util.UploadFilter**. Em sua configuração, temos o parâmetro **maxSize** que determina o tamanho máximo permitido do arquivo que será feito upload. A biblioteca usada para upload de arquivos é a **commons-fileupload-1.0.jar**, que pode ser encontrada em **Projects**, em **Libraries**.

CRIANDO UM CADASTRO DE AUTORES

Crie uma página chamada **Autor** e execute os passos relacionados para a geração de um sistema de cadastro de autores. Como diferença, não haverá um upload de imagens (a menos que queira adicionar uma foto, o que implica na alteração da tabela como ocorreu com livros).

Com o direito do mouse sobre o componente **Table** da sua página, selecione **Table Layout** no menu de contexto e configure como a tabela da página **Livros.jsp**.

Na guia **Options** digite em **Title** o texto **Resultados da pesquisa realizada** e em **Empty Data Message** digite **Não há livros encontrados na pesquisa**. Confirme as alterações clicando em **Apply**.



FIGURA 8.35 – APARÊNCIA FINAL DA PÁGINA DE PESQUISAS

Na janela **Outline** expanda **SessionBean1** e dê um duplo clique no componente **livrosPesqRowSet**.

Clique com o direito do mouse sobre a coluna **titulo** e selecione no menu de contexto o único item **Add Query Criteria**.

CAPÍTULO 9

TRABALHANDO COM WEB SERVICES NO NETBEANS IDE

Desde que a palavra Web Service foi pronunciada pela primeira vez ao mundo, todas as linguagens de programação voltadas para a construção de aplicações Web começaram a correr em busca de trabalhar com esta forma de serviço.

Alguns padrões foram estabelecidos e isto facilitou o desenvolvimento de rotinas aplicáveis a IDE's, como o NetBeans.

Neste Capítulo será criado e consumido Web Services através do NetBeans IDE, onde será visto:

- Como criar e consumir um Web Services simples;
- Como alterá-lo de forma eficiente e visual;
- A integração com EJB 3 e acesso a banco de dados;
- Como consumir com Visual Web JavaServer Faces.

WEB SERVICES

Para que você entenda um Web Service, vamos ilustrar uma situação: imagine que você esteja criando um site de comércio eletrônico e que deseja implementar um **programa de afiliados**. Este afiliado como recompensa receberá uma pequena porcentagem da venda ocorrida através de seu vínculo em um produto ou serviço. Muitos sites de comércio eletrônicos fazem isso, criando uma forma de vincular seus produtos e serviços a outros sites.

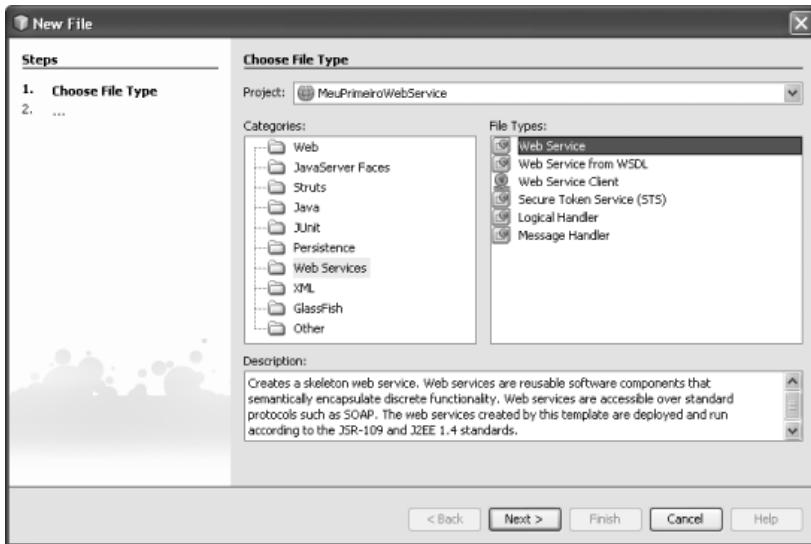


FIGURA 9.1 – CRIANDO UM NOVO WEB SERVICE

Na segunda etapa do assistente, digite **MeuPrimeiroWS** em **Web Service Name**. Em **Package** coloque o nome do seu pacote (no caso do livro: **br.com.integrator**). Clique no botão **Finish** para terminar.

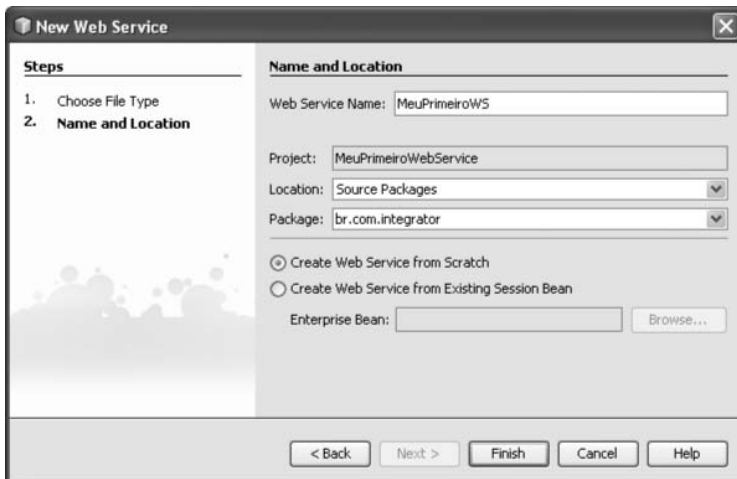


FIGURA 9.2 – DETERMINANDO O NOME E PACOTE DO WEB SERVICE

Observe que uma classe chamada **MeuPrimeiroWS** foi criada. Na janela **Projects**, um diretório novo, chamado de **Web Services** surgiu, abrindo também um editor visual. Clique no botão **Add Operation**.

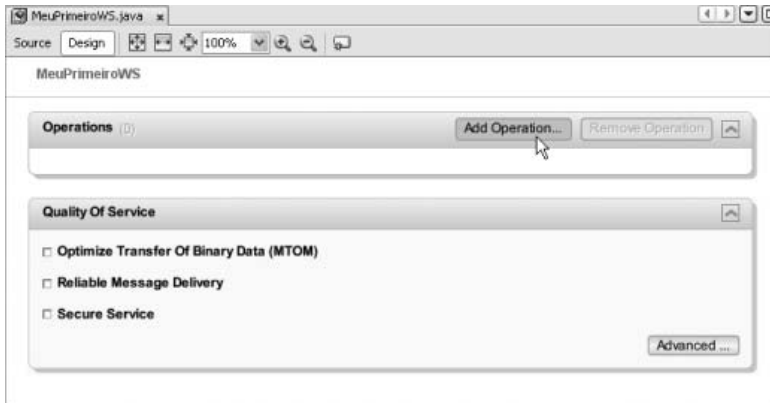


FIGURA 9.3 – EDIÇÃO DO WEB SERVICE VISUAL DO NETBEANS IDE

Na caixa de diálogo **Add Operation**, digite no campo **Name** o nome do método a ser criado, o que no caso será **seuNome**. Em **Return Type** é determinado o tipo do método, onde no exemplo será **java.lang.String**. Clique no botão **Add**, logo abaixo, na guia **Parameters**. Ao surgir uma linha, defina o nome, em **Name**, da variável parâmetro ao qual o método receberá. No caso, fora colocado **nome**. Em **Type** mantenha **java.lang.String**. Clique no botão **OK** para confirmar esta caixa de diálogo.

Do código gerado automaticamente pela IDE, fora acrescentado uma requisição de um campo, para o uso de um formulário logo abaixo e uma condição que evita a chamada do serviço Web antes de submetido um valor. Ao submeter o nome, o Web Service o receberá e retornará com a string devida.

O resultado é similar ao visto na **Figura 9.14** mostrada a seguir, após envio:



FIGURA 9.14 – CONSUMINDO O WEB SERVICE CRIADO POR UMA PÁGINA JSP

UM WEB SERVICE MAIS COMPLEXO

Criar e consumir um Web Service usando o NetBeans é simples, uma vez que ele se propõe a gerar o código bruto de seu desenvolvimento.

O exemplo anterior tratou de trabalhar com apenas uma informação retornada. Neste exemplo agora, você fará um acesso a um banco de dados e uma pesquisa, onde o retorno da pesquisa será feita pelo Web Service, ao qual será consumido por sua aplicação Web.

CRIANDO O PROJETO ENTERPRISE APPLICATION E O ENTITY BEAN

Crie um projeto Enterprise Application e o chame de **WComBancoDados**. Clique com o direito do mouse sobre o EJB (**WComBancoDados-ejb**), vá em **New** e clique no item **Entity Classes from Database**. Em **Database Tables**, escolha em **Data Source** o JNDI criado do banco de dados **livraria**. Selecione a tabela **livros**, em **Available Tables**, clicando em seguida no botão **Add**. Clique no botão **Next** para prosseguir.

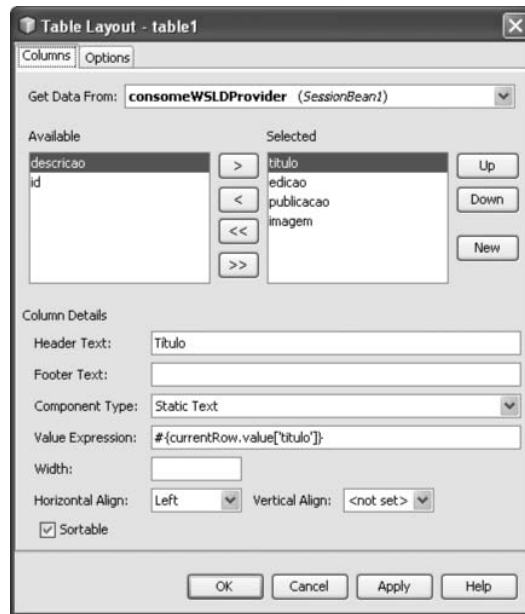
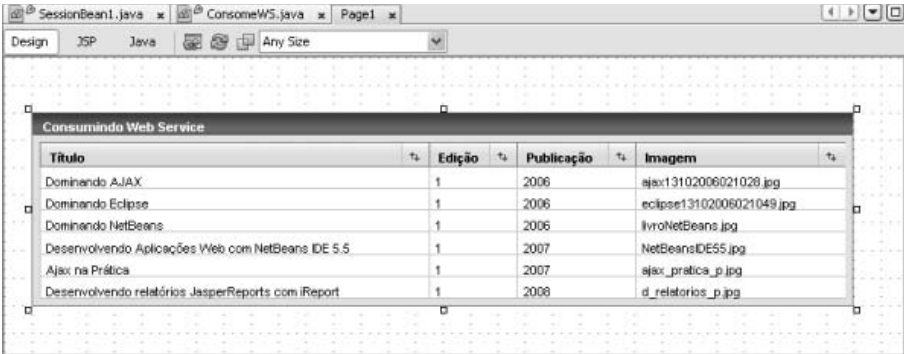


FIGURA 9.22 – UTILIZANDO O DATA PROVIDER CRIADO

Será bem provável que o Visual Web JavaServer Faces preencha no Design os dados oriundos do Web Service, como mostra a **Figura 9.23**.



Título	Edição	Publicação	Imagem
Dominando AJAX	1	2006	ajax13102006021028.jpg
Dominando Eclipse	1	2006	eclipse13102006021049.jpg
Dominando NetBeans	1	2006	livroNetBeans.jpg
Desenvolvendo Aplicações Web com NetBeans IDE 5.5	1	2007	NetBeansIDE55.jpg
Ajax na Prática	1	2007	ajax_pratica_p.jpg
Desenvolvendo relatórios JasperReports com iReport	1	2008	d_relatorios_p.jpg

FIGURA 9.23 – TABLE PREENCHIDO COM DADOS ORIUNDOS DO WEB SERVICE

ATENÇÃO: Não esqueça de ter o servidor rodando para que o Web Service se mantenha funcional.

CAPÍTULO 10

VISUAL WEB JSF COM JPA, SPRING E HIBERNATE

Usar a Java Persistence API (JPA) com Visual Web JSF, no NetBeans IDE, é um caminho alternativo ao acesso padrão usando JDBC. O Visual Web JSF possui, além de uma ferramenta visual para desenvolvimento de páginas JSF poderosa, uma grande flexibilidade, ao qual o desenvolvedor pode adicionar outros frameworks, como o Spring e o Hibernate para trabalhar em conjunto.

Neste capítulo será apresentado como criar um CRUD usando Visual Web Java-Server Faces, através do uso da JPA, com Hibernate e Spring 2.5.

Ao longo do capítulo o leitor aprenderá:

- A instalar e configurar o plugin do Spring Framework;
- A criar um DAO genérico;
- A configurar o arquivo de persistência para trabalhar com Hibernate;
- A configurar o Spring para trabalhar com JPA e com o Visual Web JSF;

A APLICAÇÃO QUE SERÁ CONSTRUÍDA

Para o exemplo proposto, será feita uma aplicação contendo uma página somente, seguindo a aparência similar à utilizada no cadastro de autores do **Capítulo 8**, na área administrativa. A **Figura 10.1** exibe a aparência proposta e seus respectivos componentes.

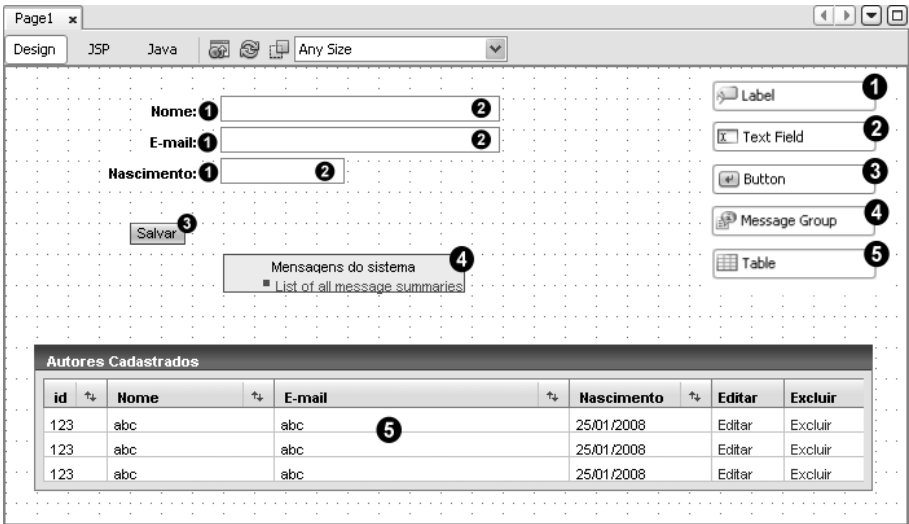


FIGURA 10.1 – EXEMPLO DA APARÊNCIA DA PÁGINA

O HIBERNATE

Hibernate é um projeto audacioso que procura ter uma completa solução para o problema de gerenciamento de dados persistentes em Java. O Hibernate é um framework que se relaciona com o banco de dados, onde este relacionamento é conhecido como mapeamento objeto/relacional (ORM) para Java, deixando o desenvolvedor livre para se concentrar em problemas da lógica do negócio. Sua simplicidade em configuração, dá ao desenvolvedor algumas regras para que sejam seguidas como padrões de desenvolvimento ao escrever sua lógica de negócios e suas classes persistentes. De resto, o Hibernate se integra suavemente ao seu sistema se comunicando com o banco de dados como se fosse diretamente feito por sua aplicação.

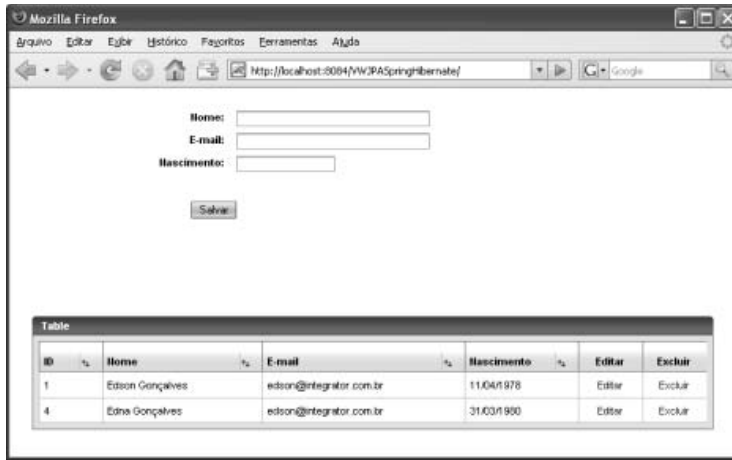


FIGURA 10.15 – EXIBIÇÃO DO PROJETO EM EXECUÇÃO NO NAVEGADOR

ATENÇÃO: No Capítulo Extra 6, encontrado no CD-ROM anexo ao livro, há um projeto completo, como estudo de caso, usando o Spring Framework e Hibernate com JPA, o mesmo criado ao longo dos Capítulos 7, 8 e Extra 5, mas com as técnicas para utilizar o poder destes frameworks em seus componentes.

PARTE 3

DESENVOLVIMENTO
COM LINGUAGENS
DINÂMICAS E AJAX

CAPÍTULO 11

RAILS 2 COM NETBEANS IDE

O NetBeans IDE vem se firmando cada vez mais como uma plataforma de desenvolvimento, possibilitando o uso não somente da linguagem Java e sua gama de frameworks, como também com outras linguagens e seus respectivos frameworks.

Uma das linguagens que foram incorporadas a IDE na versão 6.0 é o Ruby, que também possui um popular framework, chamado de Rails.

Para desenvolvimento com Ruby on Rails, o NetBeans tem se tornado uma das mais indicadas ferramentas do mercado.

Neste capítulo, o leitor será introduzido no trabalho com Ruby on Rails na versão 2.0.2, usando o NetBeans IDE, ao qual será apresentado:

- O que é Ruby on Rails e o que este framework muda no desenvolvimento de aplicações;
- Como criar um projeto e configurar o Ruby e Rails;
- A compreensão da estrutura do framework Rails;
- O ActiveRecord e o que este facilita no trabalho com banco de dados;
- A utilizar o Scaffold;
- A trabalhar com relacionamentos;
- A conhecer o console Rails através do NetBeans IDE;

VISUALIZANDO A VERSÃO DO RUBY NO TERMINAL

Após a instalação, abra o terminal ou prompt de comando e digite a sequência a seguir, ***sem espaço entre os hífens***:

ruby - -version

CONFIGURANDO O RUBY NO NETBEANS IDE 6.0

Como já foi dito anteriormente, no NetBeans IDE 6.0, já existe embebido em seu pacote de instalação o JRuby, que é compatível com o Ruby compilado em C.

Para utilizar o Ruby instalado, caso queira, vá ao menu **Tools** e clique em **Options**.

Na caixa de diálogo **Options**, clique em **Ruby**. Na guia **Platform**, em **Ruby Interpreter**, clique no botão **Browse** e selecione o interpretador que será usado em suas aplicações Ruby on Rails. Se for o **Ruby**, selecione o executável a ser usado.

DESENVOLVENDO COM RUBY ON RAILS

O exemplo que será apresentado envolverá o uso de banco de dados ensinando a criar um CRUD completo.

NOTA: O uso imediato de banco de dados acontecerá em parte devido a extrema facilidade em lidar com Rails, principalmente para um programador Java, que já possui experiência com as complicações que a linguagem costuma trazer.

CRIANDO O PROJETO

Comece por criar um novo projeto na IDE. Selecione **Ruby on Rails Application** da categoria **Ruby** e clique no botão **Next**.

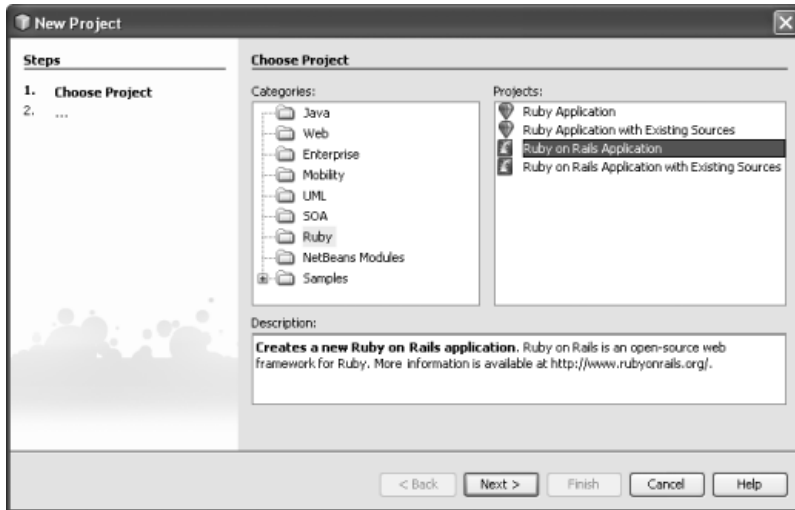


FIGURA 11.1 – CRIANDO UM PROJETO RUBY ON RAILS

A segunda etapa é para definição do nome do projeto. Para o livro o projeto fora chamado de **LivrariaRails**. O banco de dados será o mysql, claro. Se estiver usando o JRuby, há também a opção de ter acesso ao banco de dados via JDBC e adicionar o **Rake**⁴ para criar arquivos WAR. Como este exemplo visa trabalhar diretamente com o Ruby na versão C, estas características serão ignoradas. Clique no botão **Next** para prosseguir.

⁴ Rake: Comando que realiza tarefas descritas em um arquivo chamado **Rakefile**

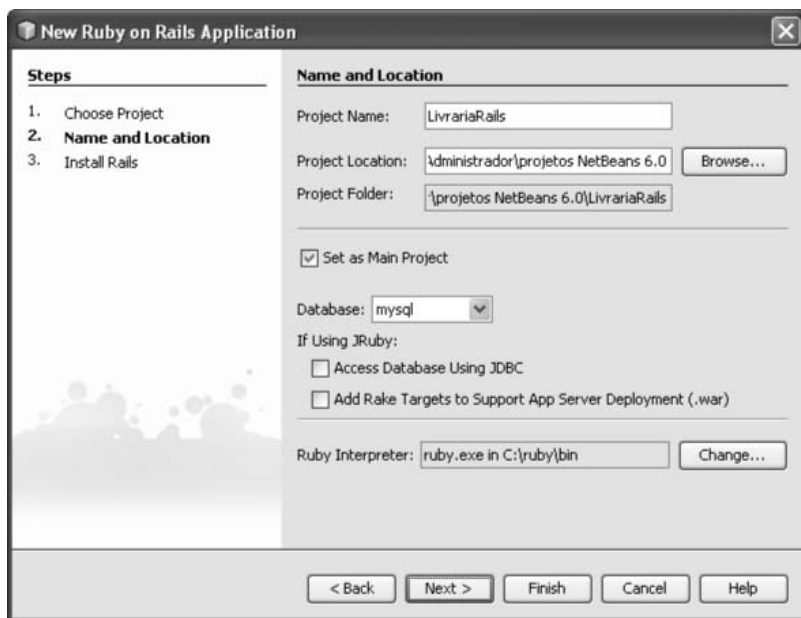


FIGURA 11.2 – ESCOLHA DO NOME DO PROJETO

NOTA: Na criação do projeto há também a possibilidade de mudar o interpretador Ruby. Clique em **Change** e depois de um alerta, a caixa de diálogo **Options** será aberta para modificação.

Na terceira etapa, dê um **Update em Rails**, caso esteja usando uma versão antiga. A caixa de diálogo **Gem(s) Update** surgirá, neste caso, para fazer a atualização. Se tudo estiver correto, clique no botão **Finish** para confirmar.

Observe que há uma saída na janela **Output**, mostrando o trabalho executado por Rails para gerar a árvore de seus componentes.

ATENÇÃO: É provável que seja necessário fazer a atualização o RubyGems. Entre no terminal ou prompt de comando e digite:

```
gem update --system
```

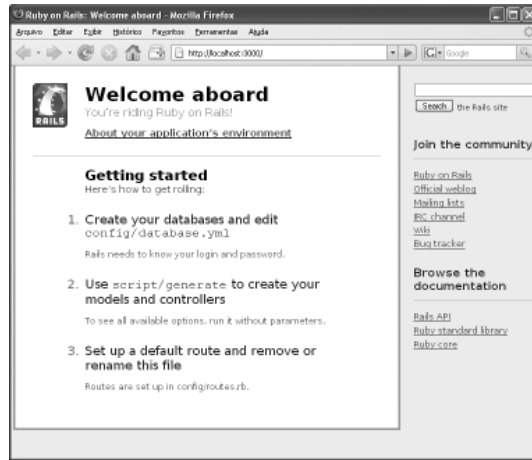


FIGURA 11.9 – PÁGINA WELCOME PADRÃO DO RAILS

Observe que o servidor roda na porta 3000. Para acessar a parte da aplicação gerada, digite o endereço seguinte em seu navegador:

http://localhost:3000/livros

Rails criou o CRUD, sendo que na primeira página você possui uma tabela listando os dados existentes na tabela **livros**.



FIGURA 11.10 – LISTAGEM INICIAL DA TABELA LIVROS

O método **validates_presence_of** é um validador do Rails que verifica se um campo está vazio. Já **validates_numericality_of** valida entrada numérica nos campos. Com **validates_uniqueness_of** o Rails valida o campo para garantir a entrada de um nome único. Muito comum para usuários de um sistema, neste caso fora utilizado para validar o título de cada livro cadastrado.



FIGURA 11.12 – VALIDAÇÕES EM AÇÃO

O método **error_messages_for** recebe todas as mensagens de erro geradas na validação e cria o HTML para exibi-las, usando uma folha de estilo. O padrão para o estilo usado neste caso é o **scaffold.css**, encontrado no diretório **Public > stylesheets** de seu projeto.

LISTAGEM 11.10 - ALTERAÇÃO NO MODEL AUTOR

```
class Autor < ActiveRecord::Base
  belongs_to :livros
end
```

A declaração **belongs_to** diz a Rails que a tabela autores é filha de livros. Isso significa que não deve haver um autor sem que antes haja um livro existente associado.

SOBRE O RELACIONAMENTO USADO

O relacionamento usado para este exemplo foi o conhecido “um-para-muitos”, que pode ser melhor entendido na **Figura 11.14** mostrada a seguir:

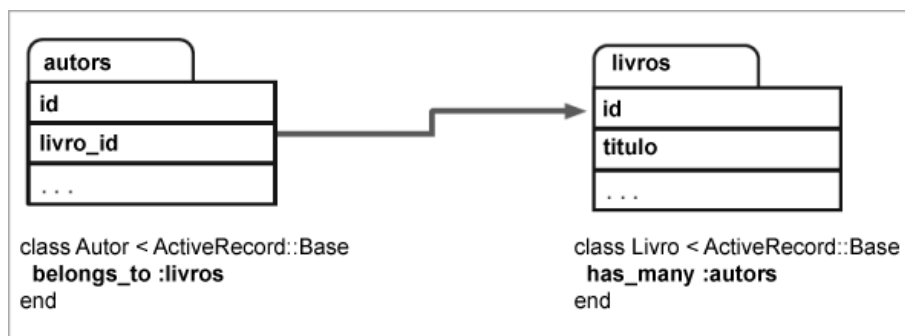


FIGURA 11.14 – RELACIONAMENTO ONE-TO-MANY DO ACTIVE RECORD DE RAILS⁶

Em Active Record, o objeto parente, o que contém uma coleção de objetos filhos, usa **has_many** para declarar sua relação à tabela filho e por sua vez, a tabela filho usa **belongs_to** para indicar seu pai.

NOTA: O relacionamento não está levando em conta uma modelagem mais correta para o caso.

⁶ Imagem inspirada no livro Agile Web Development with Rails – second edition – The Pragmatic Bookshelf

CAPÍTULO 12

JRUBY ON RAILS

A máquina virtual Java, Java Virtual Machine (JVM), também possui sua versão da linguagem Ruby, chamada de JRuby. O NetBeans suporta o trabalho com JRuby como em Ruby, incluindo também em sua instalação uma versão de JRuby para desenvolvimento.

Neste capítulo, o leitor aprenderá a trabalhar com JRuby em conjunto com Rails 2.0.2, usando o NetBeans IDE 6.x, ao qual será apresentado:

- Como configurar o JRuby;
- Como fazer um projeto e configurar o JRuby e Rails;
- A configurar, instalar e desinstalar Ruby Gems;
- Como configurar o JDBC para trabalhar com JRuby;
- A criar arquivos WAR para fazer deploy em um Application Server;

O QUE É JRUBY?

JRuby é a versão Java da linguagem Ruby criada originalmente por Jan Arne Petersen em 2001. Atualmente possui quatro como principais desenvolvedores: Charles Nutter, Thomas Enebo, Ola Bini e Nick Sieger. Percebendo a popularidade do Ruby, devido em grande parte ao impacto do framework Rails, a Sun Microsystems contratou Thomas Enebo e Charles Nutter para trabalhar com JRuby em tempo integral, mantendo-o compatível com a versão C do Ruby e conseqüentemente acompanhando a evolução do Rails em seu uso.

No NetBeans IDE 6.1, os interpretadores para Ruby e JRuby serão configurados separadamente, através do menu **Tools** em **Ruby Platforms**.

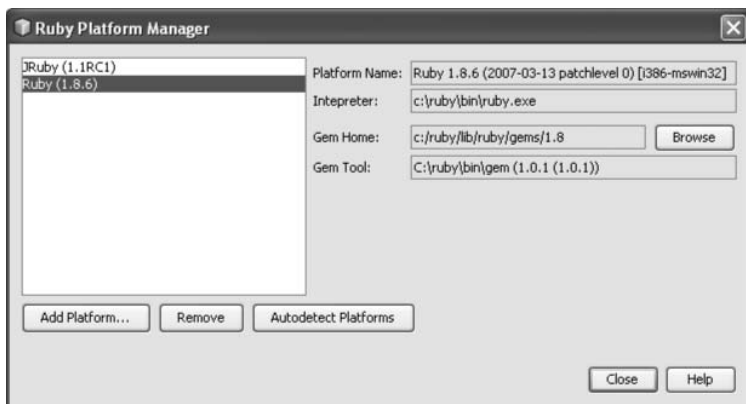


FIGURA 12.1 – OS INTERPRETADORES RUBY E JRUBY CONFIGURADOS NO NETBEANS IDE 6.1

INSTALANDO OS RUBY GEMS NO NETBEANS

Indo ao menu **Tools**, no item **Ruby Gems**, temos acesso aos Gems instalados do Ruby (**Installed**), os que precisam ser atualizados (**Updated**) e os novos que podem ser adicionados (**New Gems**). Ao lado de cada guia há o número correspondente de cada uma das opções que podem e muitas vezes devem ser feitas como operações necessárias para a geração de uma aplicação Rails.

Para instalar um novo Gem, vá a guia **New Gems** e selecione o que deseja adicionar. Clique no botão **Install**. A caixa de diálogo **Gem Installation Settings** surgirá definindo o nome (**Name**) e versão (**Version**). Sempre que estiver usando uma última versão, por exemplo do Rails, deixe em **Version** o item **Latest** selecionado. Mas se este não for o caso, selecione a versão ao qual deseja usar. Ao confirmar o item que será instalado, a caixa de diálogo **Gem Installation** surge mostrando o status do console enquanto o Gem é baixado e instalado. Ao terminar, o botão **Close** é habilitado, para que seja fechada e confirmada a instalação.

ADICIONANDO O ARQUIVO WAR NO GLASSFISH

Inicie o GlassFish no NetBeans através da janela **Services**. Com o direito sobre o GlassFish, selecione **View Admin Console**. Entre no administrador do GlassFish e faça o deploy do arquivo WAR, em **Applications > Web Applications** e clique no botão Deploy e suba o arquivo WAR gerado.

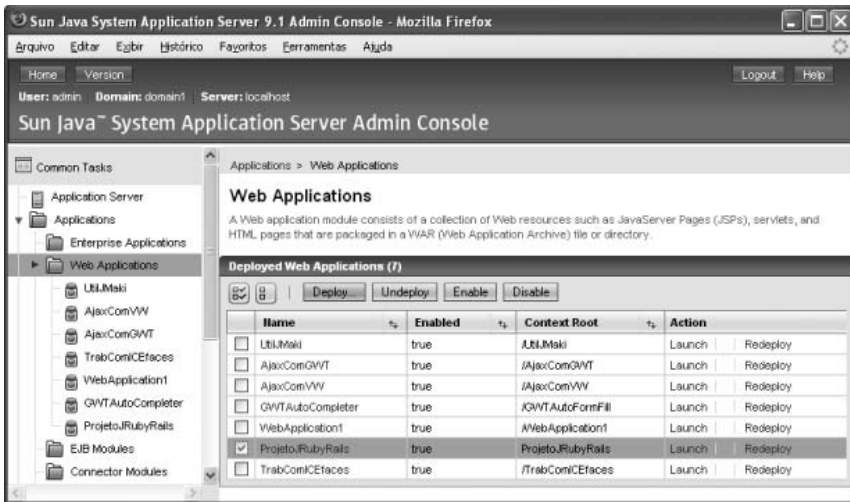


FIGURA 12.14 – DEPLOY DO PROJETO JRUBY ON RAILS CRIADO NO NETBEANS

Para executar a aplicação, dentro do próprio administrador do GlassFish há um link Launch. Clique nele para que seja aberta a aplicação. Caso tenha configurado o arquivo routes.rb e retirado o index.html, você verá imediatamente os livros cadastrados. Do contrário, adicione livros em seu final, na barra de endereços.



FIGURA 12.15 – APLICAÇÃO EXECUTANDO NO GLASSFISH

CAPÍTULO 13

TRABALHANDO COM AJAX

NO NETBEANS IDE

O NetBeans possui bons módulos de desenvolvimento AJAX que podem ser incorporados à ferramenta, tornando mais rápido e prático seu desenvolvimento. Além disso, também há empresas que estão investindo no Visual Web JavaServer Faces, como ICEfaces que possui componentes integrados.

Neste capítulo será apresentado:

- A origem do AJAX e seu significado
- Como o AJAX trabalha
- Como criar um projeto AJAX no NetBeans IDE
- A instalar novos módulos (plug-ins)
- O framework jMaki e seus componentes
- O uso de GWT integrado ao NetBeans
- A utilizar o banco de dados com AJAX
- A integrar o ICEfaces ao Visual Web JSF

AJAX

O termo AJAX surgiu em fevereiro 2005, por Jesse James Garrett de Adaptive Path, LLC, onde publicou um artigo on-line intitulado, "Ajax: A New Approach to Web Applications". O artigo se encontra em inglês e pode ser lido no endereço <http://www.adaptivepath.com/publications/essays/archives/000385.php>.

UTILIZANDO A TECNOLOGIA jMAKI

O jMaki é um framework que oferece uma série dos melhores componentes de vários outros frameworks Ajax para projetos com este fim. Na lista de frameworks usados temos: Dojo, Scriptaculous, Yahoo UI Widgets, Spry, DHTML Goodies, e Google, permitindo ao desenvolvedor Java, trabalhar como se estivesse usando uma tag do JSTL ou JSF. Também possui suporte a outras tecnologias além de Java.

Você pode encontrar este framework no endereço:

<https://ajax.dev.java.net/>

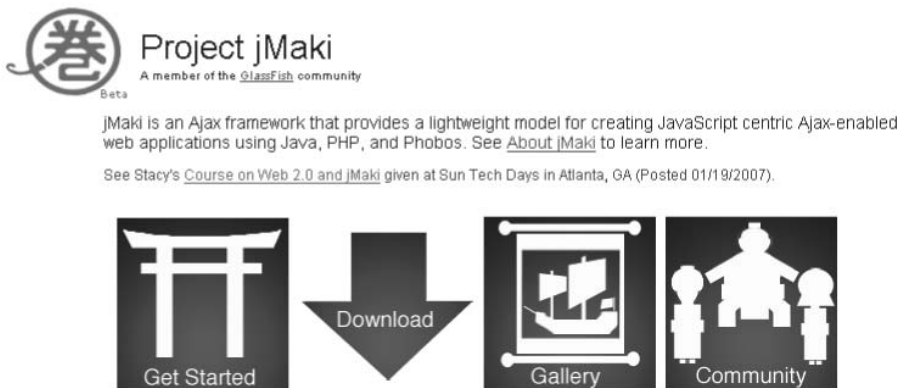


FIGURA 13.2 – PÁGINA PRINCIPAL DO PROJETO

O framework **jMaki** possui uma série de componentes, ao qual a seguir estão listados como organizados:

CLIENT SIDE COMPONENTS

jMaki Layouts: jMaki Layouts fornece um padrão baseado em pontos iniciais para criar suas aplicações Web usando HTML e CSS.

OBSERVAÇÃO: Existe um conjunto de classes no endereço <http://www.json.org/java> para desenvolver o formato JSON utilizando Java. O NetBeans IDE 6.0 possui em suas bibliotecas o JSON para que você possa adicionar em seus projetos.

Uma biblioteca Java também pode ser encontrada no endereço <http://json-lib.sourceforge.net/>.

Na sua página, altere seu componente como mostrado a seguir:

```
<a:widget name="dojo.etable"
    service="LivrosFacade" />
```

O resultado pode ser visto na **Figura 13.11** a seguir:



FIGURA 13.11 – EXIBIÇÃO DA TABELA RENDERIZADA COM OS DADOS DO SERVLET

Na janela **Projects**, expandindo os nós de **Web Pages > resources > dojo > etable**, você encontra o arquivo JavaScript **component.js**, onde reside os métodos que manipulam este componente.

Existem outros componentes que podem ser analisado, onde em suma, trabalham com formatos JSON e que são muito simples de se adaptar.

AJAX COM GWT

O Google Web Toolkit, ou simplesmente GWT, é um framework Ajax que facilita o desenvolvimento por esconder do programador a implementação de código JavaScript. O GWT abstrai o JavaScript a partir de uma biblioteca de classes Java, disponibilizando diversos componentes widgets (componentes visuais). Estes componentes são usados pela própria Google, desenvolvedora do framework, em aplicações já consagradas como GMail e Google Maps.

BAIXANDO O GWT

O projeto do Google Web Toolkit possui o pacote que é encontrado na página do framework. Disponível para os sistemas operacionais Linux, Windows e MacOS, você o encontra no seguinte endereço:

<http://code.google.com/webtoolkit/>

Neste endereço há o link **Download Google Web Toolkit (GWT)**, que o leva ao download do framework, que no momento em que este livro é escrito, se encontra na versão **1.4.61**. O arquivo vem compactado. Descompacte-o em um local desejado.

INSTALANDO O PLUGIN GWT NO NETBEANS

Vá ao menu **Tools** e clique em **Plugins**. Na guia **Available Plugins**, da caixa de diálogo **Plugins**, selecione o item **GWT4NB**. Este plugin possibilita integrar o NetBeans ao GWT de forma simples, integrando a suas aplicações Web. Clique no botão **Install**.

FormPanel é uma classe que gera a tag XHTML <form/> com seus respectivos atributos. O método **setAction()** determina o valor do atributo **action** da tag <form/> e **setMethod()** o atributo **method**. Pode-se definir o método de envio POST através da constante **METHOD_POST** de **FormPanel** ou **GET** através de **METHOD_GET**.

O **FormPanel** é uma subclasse de **SimplePanel**, o que significa que este componente pode conter somente um widget ou painel. A menos que seu formulário possua somente um controle, é necessário colocar os demais componentes em um painel e depois adicioná-lo ao **FormPanel**. Para simplificar seu desenvolvimento, foi usado no exemplo um **VerticalPanel**, onde o resultado é visto na **Figura 13.18**.

OS EVENTOS DE FORMPANEL

O **FormPanel** permite registrar um ouvinte de eventos, permitindo escrever um manipulador que os codifique. Os dois eventos para o ouvinte são a submissão do formulário, através do método **onSubmit**, e a conclusão da submissão, através de **onSubmitComplete**.

No exemplo, somente **onSubmit** fora utilizado para definir se um dos campos de entrada possui algum valor. Se este estiver vazio, aparece um alerta, seguido do foco no campo (**setFocus(true)**) e cancelamento da submissão (**setCancelled(true)**).

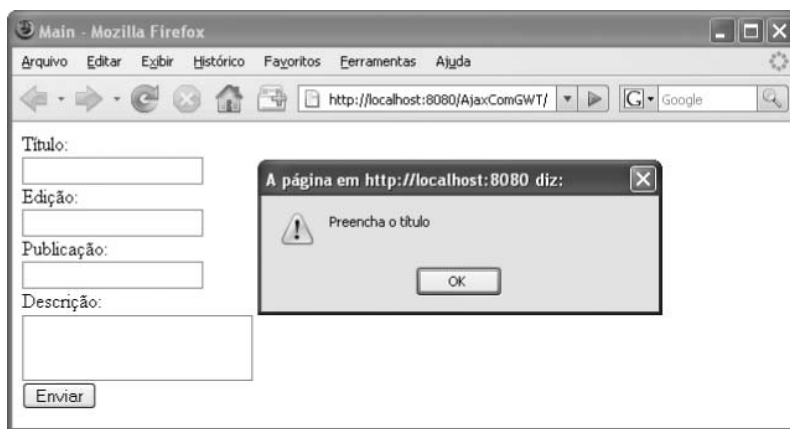


FIGURA 13.18 – FORMULÁRIO GERADO POR FORMPANEL

UTILIZANDO AJAX COM VISUAL WEB JSF

O Visual Web JavaServer Faces possui também seus próprios componentes Ajax. O que atualmente possui constante atualização é o projeto ICEfaces.

O PROJETO ICEFACES

O projeto Open Source ICEfaces é mantido pela empresa ICEsoft Technologies Inc., que contribui constantemente com a comunidade e que atualmente é o que possui mais componentes Ajax integrados com o Visual Web JSF.

Para obter este framework, vá ao site <http://www.icefaces.org> e clique em **Download**.

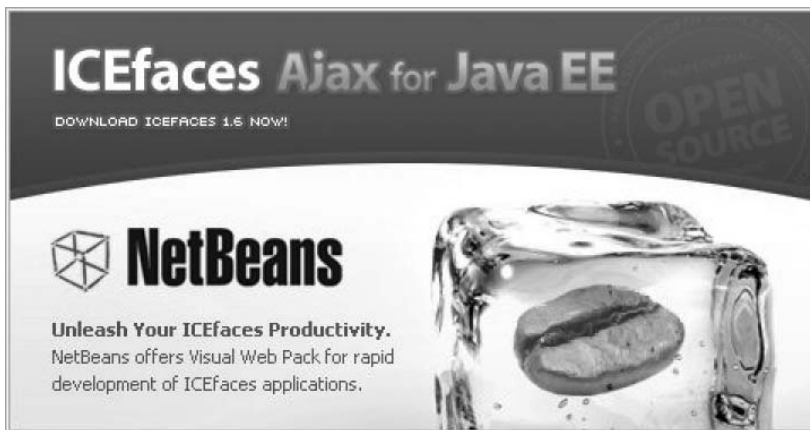


FIGURA 13.21 – BANNER DE ENTRADA DO PROJETO ICEFACES

É necessário fazer login para baixá-lo. Para o NetBeans IDE, existem as ferramentas de integração:

- **ICEFACES-NETBEANS6-IDE-v3.0.0.1.ZIP** - que se integra ao NetBeans.
- **ICEFACES-1.6.2-LIBS-NETBEANS6.ZIP** – as bibliotecas do ICEFaces.

Ao descompactar estes arquivos, perceba que há por entre eles um arquivo com extensão **.nbm**.

APÊNDICE A

RUBY PARA DESENVOLVEDORES

JAVA

Embora a linguagem Ruby não seja nova, somente há pouco tempo os holofotes do desenvolvimento Web se voltaram para esta linguagem devido ao reconhecimento de seu framework, chamado de Rails. Além da espantosa velocidade com que uma aplicação pode ser feita com a dupla, Ruby on Rails, Ruby é uma linguagem orientada a objetos, possuidora de uma sintaxe amigável e simples. O conjunto destes elementos trouxe ao desenvolvedor a abertura de um mundo de novas possibilidades, onde a simplicidade é a palavra de ordem.

Neste Apêndice você conhecerá o básico sobre a linguagem Ruby, focada nos exemplos com relação a Java, em conjunto com o NetBeans IDE.

RECURSOS DO RUBY

O Ruby possui muitos recursos similares a muitas linguagens, inclusive a Java, o que explica sua adoção por muitos desenvolvedores desta linguagem. Entre seus principais recursos temos:

- Tratamento de Exceções;
- É uma linguagem orientada a objetos (não 100%), mas melhor que Java e similar a mãe das linguagens orientadas a objetos SmallTalk. Isso significa que um número é a extensão de uma classe;
- Não precisa ser compilada após uma alteração. Você a modifica e executa.
- Closures (funções criadas dentro de outras funções, que referenciam o ambiente da função externa mesmo depois de ter saído de escopo) e com bindings de variáveis;