Globalcode®
THE DEVELOPERS COMPANY

Material para treinamento na plataforma NetBeans com ênfase em desenvolvimento de plug-ins

Globalcode – The Developers Company
www.globalcode.com.br

Sun Microsystems
www.sun.com

NetBeans USA
www.netbeans.org

SouJava
www.soujava.org.br

Sobre o curso

Este curso foi desenvolvido pela Globalcode apoiando-se em materiais publicados no site www.netbeans.org e também com colaborações diversas de outros técnicos.

O principal objetivo deste treinamento é fornecer um conhecimento suficiente para o participante desenvolver aplicativos com o NetBeans, tendo sua maior ênfase no desenvolvimento de plug-ins.

Este material foi desenvolvido como parte do projeto “Desafio NetBeans” promovido pela Globalcode, Sun Microsystems e SouJava. Poderá ser utilizado para cursos diversos, porém não é permitida a venda ou publicação deste material.

Fornecendo treinamento com este material

Caso você queira replicar o treinamento fornecido pela Globalcode, sugerimos as seguintes cargas e distribuição de matérias:

Tópico	Horas
Introdução a Plataforma NetBeans	4 horas
Swing com NetBeans	4 horas
Desenvolvimento avançado de plug-ins	8 horas
Workshop de desenvolvimento de plug-ins	4 horas
TOTAL	20 horas

Vale ressaltar a ênfase no desenvolvimento de plug-ins em função do projeto “Desafio NetBeans”, portanto o participante que já tem alguma vivência em NetBeans poderá aproveitar da segunda aula em diante, para quem já conhece bem Swing e Swing com NetBeans poderá aproveitar a parte de desenvolvimento de plug-ins.

Lembramos também que sendo um material público ele pode receber a sua colaboração, portanto você é bem-vindo para nos ajudar a desenvolvê-lo!

Nota para Instrutor

. Sugerimos que você inicie o treinamento fazendo esta introdução sobre o que será ensinada e também apresentando na lousa o planejamento de aulas. Será útil para os desenvolvedores mais experientes em NetBeans não desanimarem no começo, ou até mesmo dispensar alguma aula que ele já tenha domínio.Ressalte a idéia que este treinamento tem dois diferentes objetivos: nivelar conceitos básicos de NetBeans / Swing e ensinar o desenvolvimento de plug-ins.

. Ao terminar a apresentação do curso, faça sua própria apresentação e solicite aos participantes que se apresentem e comentem suas experiências com Java e IDE’s diversos.

1 Introdução à Plataforma NetBeans

Na visão popular, o NetBeans é um ambiente de desenvolvimento Java open-source concorrente de diversos outros ambientes. Tipicamente chamamos de IDE (Interface Development Environment) essas ferramentas que nos auxiliam para escrever software de forma mais produtiva.

O NetBeans vem evoluindo com passos largos a cada versão. Atualmente encontramos diversas facilidades que vem agradando e atraindo cada vez mais desenvolvedores de todo o mundo:

- Um excente editor de telas Swing;
- Suporte completo e robusto para Java Enterprise Edition;
- Plugins para UML, desenvolvimento remote em equipes, profiling, suporte para Micro Edition, etc.;
- Interface amigável com CVS;
- Debuging apurado de aplicativos e componentes como JSPs e EJBs;
- Suporte nativo para desenvolvimento e consumo de WebServices;
- Funcionalidades para refactoring de código.

Por tais motivos, aconselhamos que você utilize o NetBeans sem necessariamente abandonar seu IDE preferido. Mas não deixe de aproveitar as qualidades para desenvolvimento produtivo com NetBeans.

1.1 Plataforma Vs. IDE

A plataforma NetBeans representa um conjunto de serviços de infraestrutura comum a qualquer aplicativo desktop. A plataforma NetBeans oferece suporte para manipulação de menus, posicionamento de janelas, gestão de configurações entre outros.

Qualquer software que se desenvolve com Swing pode ser "abraçado" pela plataforma NetBeans.

Existem diversos softwares que foram escritos com a plataforma NetBeans, inclusive IDEs para outras linguagens. Porém o software mais popular e utilizado é o NetBeans IDE; este é justamente o IDE que utilizamos e utilizaremos aqui no curso. O NetBeans IDE é representado pela plataforma NetBeans com um conjunto de plugins que ofecerem o suporte a linguagem Java: wizard, colorido do código, debuging, suporte a RDBMS entre outros. Portanto podemos resumir que:

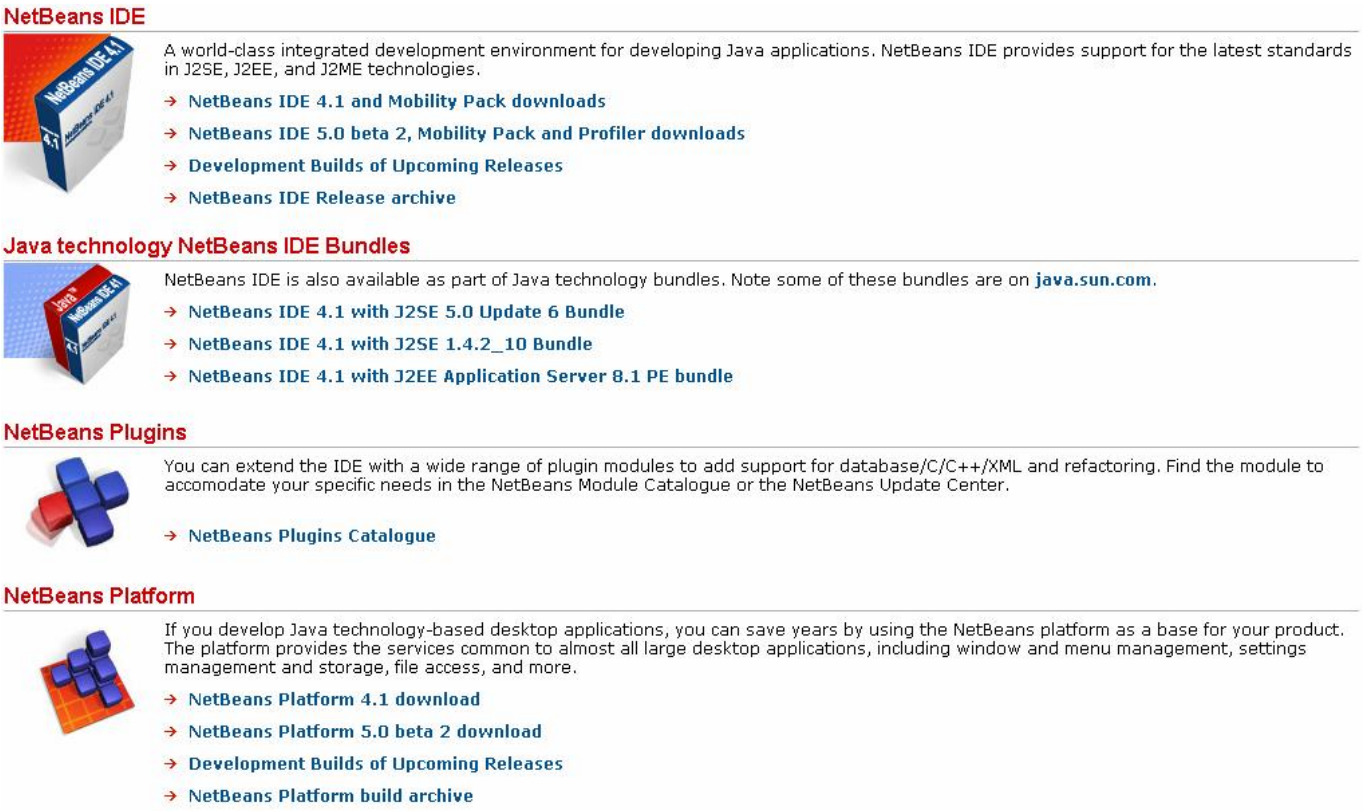
- NetBeans Platform é um núcleo básico de serviços para aplicativos desktop.
- NetBeans IDE é o NetBeans Platform com diversos plugins para desenvolvimento Java.

Anotações

1.2 Download e instalação

O download pode ser feito a partir do site http://www.netbeans.org e a instação é a mais trivial possível: clique em next até o finish...

Agora quando chegar à página de download saberá a diferença entre IDE e Platform:



Anotações

1.3 Resumo dos recursos nativos da versão 5.0

Uma das vantagens de trabalharmos com NetBeans é que diversas funcionalidades são nativas do IDE, ou seja, você não precisa baixar o NetBeans e depois dúzias de plug-ins, apenas uns dois ou três. Esta é uma relação dos recursos que você encontra na versão default para download:

- Editor Swing imbatível;
- Suporte completo para J2EE e também Java Enterprise Edition 5, incluindo JSPs, Servlets, WebServices, Custom Tags, Tag Files, EJB 3.0 e Listeners;
- Suporte para diversos servidores: Sun, JBoss, Tomcat, WebLogic;
- Suporte (na vs. 5 básico) para Struts e JavaServer Faces;
- Code completion;
- Refactoring de código;
- Integração com CVS;
- Integração com RDBMS e wizard para criação de entity beans através de bottom-up ou top-down;
- Suporte para design patterns Java EE;
- Wizards para criação de plug-ins;
- Wizards para internacionalizar um aplicativo;
- Abreviação para digitação de código;
- Suporte a JUnit;
- Debuging local e remoto;
- Integração total com Ant;

Anotações

1.4 Plugins úteis para NetBeans

Além dos recursos nativos, destacamos também alguns plug-ins específicos de excelente qualidade e distribuídos pelo próprio grupo NetBeans. Para uma lista completa de plug-ins, acesse a URL http://www.netbeans.org/catalogue/index.html.

1.4.1 NetBeans Mobility Pack

Certamente se você trabalha ou tem interesse na área de desenvolvimento para celulares e outros dispositivos ficará entusiasmado ao ver este pacote que adiciona funcionalidade no NetBeans. Basta fazer o download e ao instalar você deve informar onde você instalou o NetBeans IDE.

1.4.2 NetBeans Profiler

O profiler é uma ferramenta para análise de performance de código excelente para tuning de código e identificação de gargalo.

1.4.3 NetBeans Collaboration

Este plug-in adiciona a capacidade para você desenvolver em equipes com desenvolvedores remotos. Através de um instant messanger você pode fazer chat e também escrever código “a quatro mãos”.

Anotações

1.5 Comparativo entre ferramentas

Vamos fazer uma analisa mais neutra possível, mesmo tratando apenas de NetBeans neste treinamento. Caso discorde de alguma pontuação sinta-se a vontade para discuti-la em sala-de-aula com o instrutor e a turma.

Item	NetBeans	Eclipse	JDeveloper
Escrever código	razoável	**excelente**	fraco
Java 5	**excelente**	excelente	fraco
Suporte para Enterprise Edition: JSPs Servlet	**excelente**	c/ plug-ins	excelente
Suporte para Enterprise Edition: Tags e Filtros	**excelente**	c/ plug-ins	fraco
Suporte para Enterprise Edition: EJB	**excelente**	c/ plug-ins	excelente
Suporte para Enterprise Edition 5	**excelente**	excelente	?
Suporte para ME	**excelente**	c/ plug-ins	?
WebServices	**excelente**	c/ plug-ins	razoável
Debug de código	**excelente**	excelente	excelente
JavaServer Faces	fraco	c/ plug-ins	**excelente**
Struts	fraco	c/ plug-ins	**excelente**
Refactoring	fraco	**excelente**	razoável
CVS	razoável	**excelente**	razoável
HTML e CSS	razoável	c/ plug-ins	**excelente**
Servidores J2EE	**excelente**	c/ plug-ins	razoável
Swing	**excelente**	c/ plug-ins	razoável
Quantidade de plug-ins	fraco	**excelente**	fraco
Suporte para UML	c/ plug-ins	c/ plug-ins	**excelente**
Performance	razoável	**razoável**	fraco
Estabilidade	**excelente**	razoável (win)	razoável
Conjunto sem plug-ins	**excelente**	razoável	excelente
Upgrades e correções de bugs	**excelente**	excelente	fraco

Anotações

Nível de atualização com o mercado e IDE	**excelente**	excelente	razoável
open-source	sim	sim	não
gratuito	sim	sim	sim
Integração com Banco de dados	**excelente**	c/ plug-ins	razoável
Organização de janelas e look-and-feel	excelente	**excelente**	razoável
Tempo de startup*	30’	**15 seg.**	44 seg.
Memória*	171MB	**47MB**	219MB

*Tempos medidos no Windows XP com P4 e 768MB de memória.

Lembramos mais uma vez que o Eclipse é um ambiente de desenvolvimento “limpo”, ou seja, vem com pouquíssimas funcionalidades por padrão necessitando a instalação de plug-ins específicos. A vantagem acaba sendo o tempo de startup e consumo de memória, porém instalando plug-ins estes números tendem a aumentar.

É possível configurar tanto o JDeveloper quanto o NetBeans para utilizar uma quantidade menor de plug-ins aumentando seu tempo de startup de diminuindo o consumo de memória.

Anotações

1.6 Criando projetos com NetBeans

Um projeto no NetBeans é uma estrutura de diretórios que contém um arquivo Ant build.xml, ou seja, o NetBeans não trabalha com arquivos de configuração específicos, apenas configuração do script Ant. Em compensação o NetBeans está fortemente vinculado com o Apache Ant, o que não é nada ruim em função da altíssima qualidade da ferramenta Ant e ampla adesão da comunidade Java, incluindo o próprio James Gosling que em entrevista exclusiva com técnicos da Globalcode no JavaOne 2003, confessou ser a ferramenta que mais utiliza na plataforma Java.

Se um dizer quiser abandonar o NetBeans, tenha certeza que o processo de migração será simples, pois os IDEs / ambientes mais populares do mercado também dão suporte ao Ant. Para cada projeto criado ou integrado ao NetBeans você encontrará um diretório chamado nbproject que apesar do nome, contém apenas arquivos de configuração que alimentam o script Ant do NetBeans.

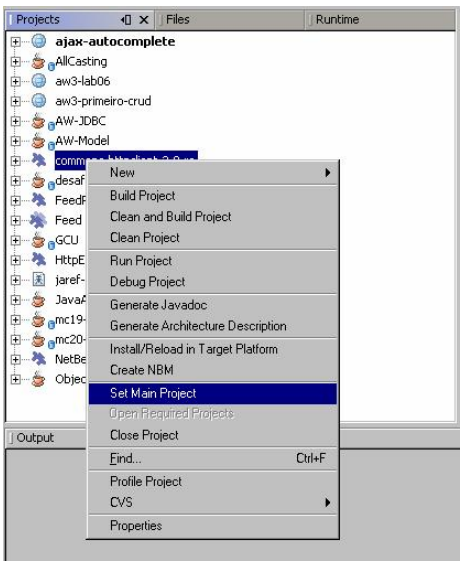
Para começarmos a trabalhar com o NetBeans devemos aprender a criar projetos frente a diferentes situações:

1. Um projeto totalmente novo: esta é a situação mais simples;
2. Um projeto a partir de um código-fonte existente;
3. Um projeto a partir de um código-fonte existente com build.xml próprio.

Vamos neste capítulo explorar como tratar cada uma destas situações para que você consiga utilizar o NetBeans para novos projetos e também para projetos existentes, onde você até mesmo utiliza outro IDE como Eclipse. Vale lembrar que é perfeitamente possível trabalhar **simultaneamente** com ambos os IDE's apontando para o mesmo projeto!

Um detalhe que lembramos desde já é que o NetBeans pode manipular múltiplos projetos simultâneamente, porém você deverá selecionar um deles como "Main Project". O efetito do projeto ser marcado como Main Project é simples: quando clicarmos em Run, Debug, Deploy ou qualquer outra ação, o NetBeans executa tal ação no Main Project.

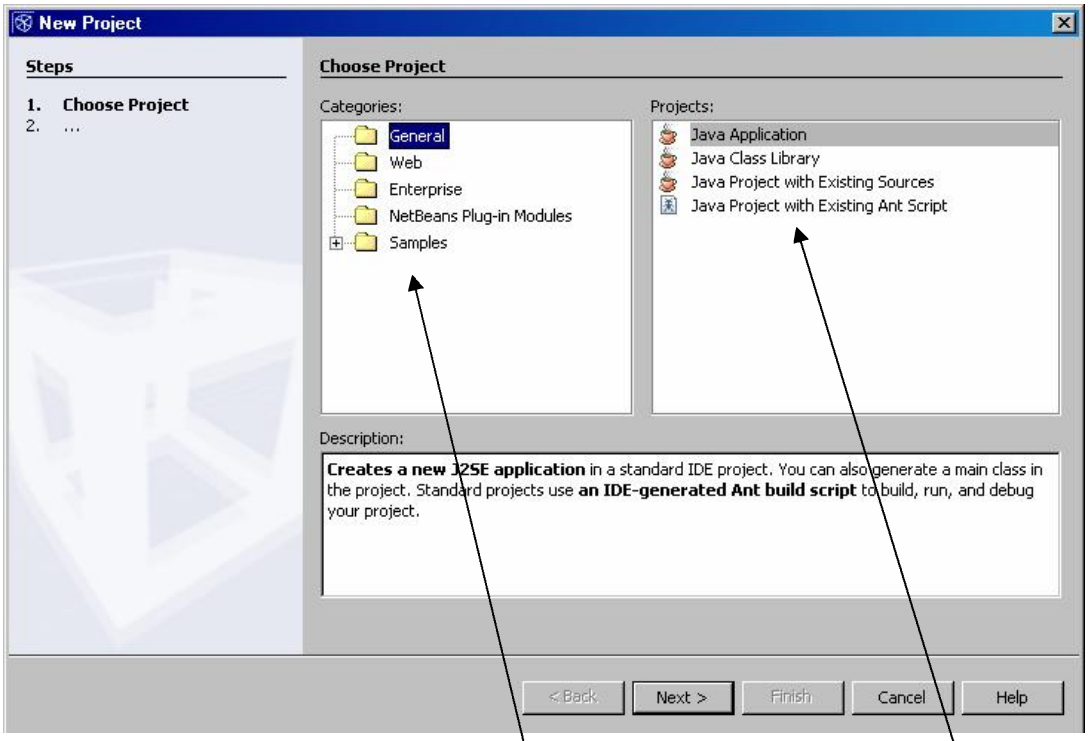
Para selecionar um projeto como Main Project, clique com o botão direito no projeto e selecione Set Main Project, o Main Project sempre está destacado em negrito para se diferenciar dos demais.



Anotações

1.6.1 Criando um projeto novo

Conforme comentado, esta é a situação mais simples de todas; para criarmos um novo projeto devemos seguir os tradicionais passos de clicar em File -> New Project, e o seguinte diálogo será disparado:



Podemos observar que temos uma árvore de categorias e ao lado um template / wizard para a criação de um determinado tipo de projeto. As categorias de projeto são:

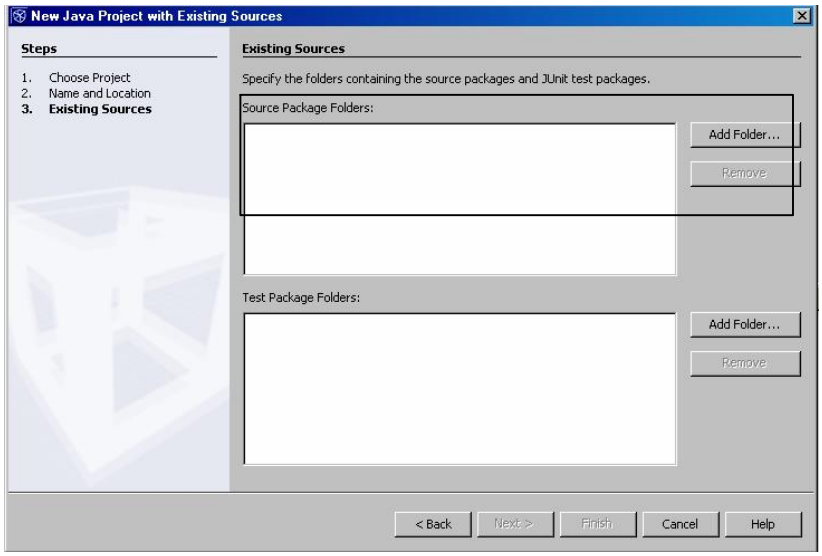
1. General: para projetos desktop, bibliotecas genéricas e aplicativos stand-alone com script ant pré-existente;
2. Web: para projetos novos, com código-fonte ou script ant pré-existente. Nesta categoria o NetBeans trabalhará em conjunto com um servidor Java Enterprise Edition;
3. Enterprise: para projetos de EJBs;
4. NetBeans Plug-in Modules: para criação de plug-ins;
5. Samples: excelentes exemplos aplicativos, incluindo exemplos Web com AJAX e Faces, um plug-in, etc.

Anotações

Para podermos criar um projeto stand-alone e totalmente novo, selecione **General -> Java Application**. Um segundo diálogo surgira para você entrar com o nome do projeto e também o folder onde os arquivos serão armazenados. O NetBeans criará um script Ant build.xml para você automaticamente.

1.6.2 Criando um projeto com código-fonte pré-existente

Neste caso ao clicar em **Project -> New Project**, selecione **General -> Java Project with Existing Sources**, escolha o nome e odiretório do projeto, sendo que o não precisa ser o diretório fonte, posteriormente você vai selecionar qual(is) sub-diretório(s) contém código-fonte em arquivos .java.



Veja no diálogo ao lado os botões que permitem adicionar os fontes e também os pacotes que eventualmente contenham testes JUnit (Test Package Folders).

O NetBeans também criará um script Ant para seu projeto automaticamente.

Anotações

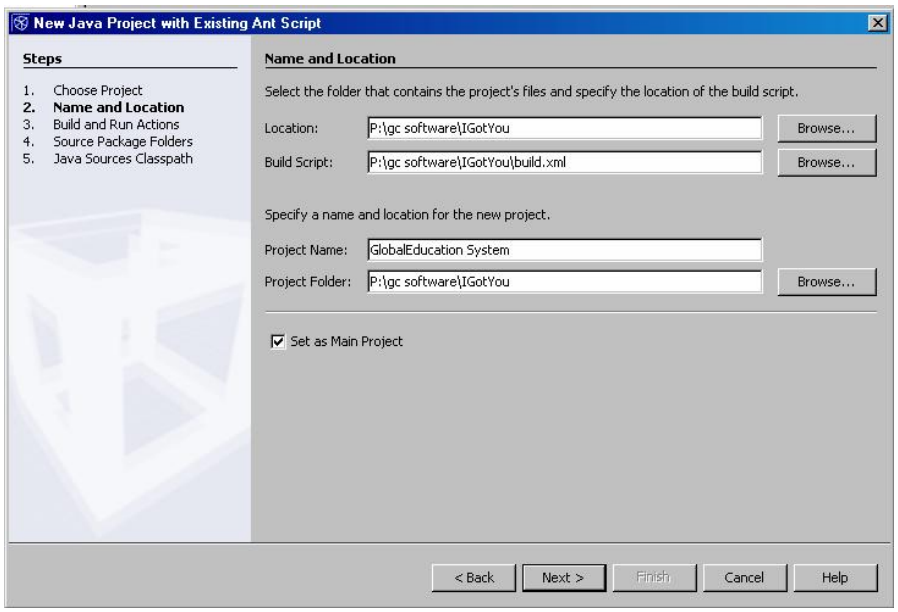
1.6.3 Criando projetos com build.xml e código-fonte pré-existentes

Esta situação é muito comum principalmente quando falamos em desenvolvimento para Web, que concentra muitos usuários da ferramenta de building da Apache chamada Ant.

Antes de mais nada se você nunca ouviu falar sobre Ant, é importante saber que trata-se de uma linguagem baseada em scripts, que são escritos com tags XML; com tais comandos representados por tags, podemos agilizar e automatizar o processo de compilação, empacotamento em jar, execução de testes de unidade, deployment, inserção de massa de testes no banco, envio de e-mail, e muito mais.

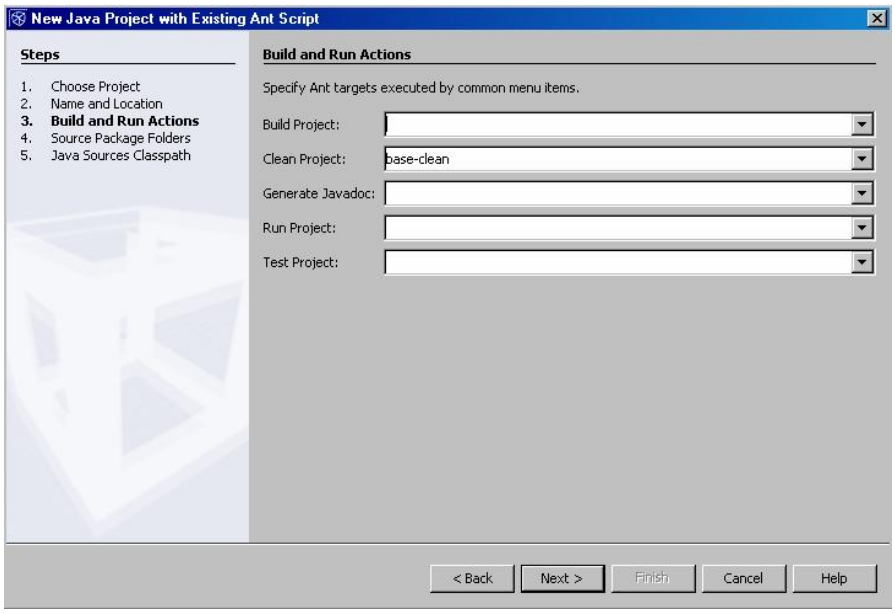
Ant é como programar um .bat ou .sh com shell script, mas conta com a vantagem de ter um interpretador Java que provê a independência de plataforma. Não é possível copiar um .bat e executa-lo em Linux, mas o mesmo script build.xml Ant pode ser executado no Windows, Linux, Unix, Apple, etc.

Para criar seu projeto baseando em código-fonte e build.xml pré-existentes, clique em **File -> New Project**, selecione **General -> Java Project with Existing Ant Script**, surgirá o seguinte diálogo para você selecionar o nome, folder onde está armazenado e também você deverá selecionar o script Ant e seu respectivo folder de localização. Veja a seguir:



Anotações

Em seguida o NetBeans apresentará um diálogo para você mapear as targets do seu script com o NetBeans:



Dica: se você utilizar os nomes nas targets como run, clean, build, javadoc e test o NetBeans será capaz de fazer o mapeamento automaticamente!

1.6.4 Abrindo e fechando projetos

Abrir um projeto no NetBeans significa abrir um diretório que contém um script Ant e o diretório com os arquivos de configuração do script, chamado nbproject. Você pode abrir e fechar os projetos a qualquer momento.

Anotações

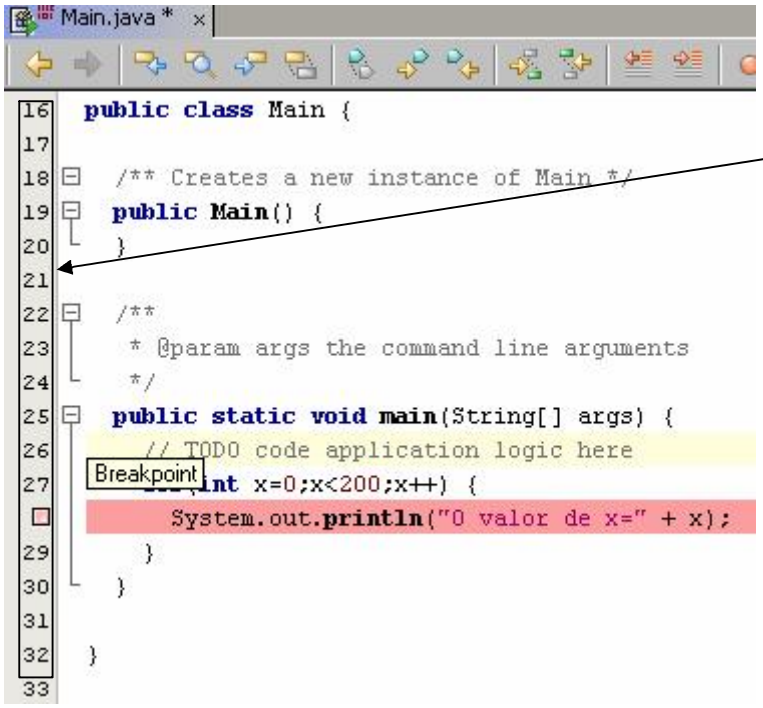
1.7 Debuging de aplicativos

Uma das grandes funcionalidades do NetBeans é para depuração de aplicativos locais ou que rodam em servidores / enterprise. O NetBeans acumula grande tempo de experiência e desenvolvimento dessas features que atualmente se encontram extremamente maduras.

Ao depurar um aplicativo com NetBeans, você será capaz de:

1. Executar um código passo-a-passo;
2. Observar valores de variáveis e até mesmo altera-las;
3. Observar as threads do seu aplicativo;
4. Executar JSP's e depurar completamente JSP's sem a necessidade de manipular o servlet de background do JSP;
5. Entre outras capacidades.

Para depurar um aplicativo você pode começar adicionando Breakpoints no código-fonte. Um breakpoint é representado por uma linha de código, que antes de ser executada pausa o aplicativo para o IDE poder depurá-lo. Existem três formas principais para adicionar um breakpoint:



1. No editor de código-fonte Java, posicione o cursor na linha e aperte CONTROL + F8;
2. Podemos também clicar em cima do número da linha que queremos adicionar o breakpoint;
3. Ou recorrer ao menu Run -> Toggle Breakpoint;

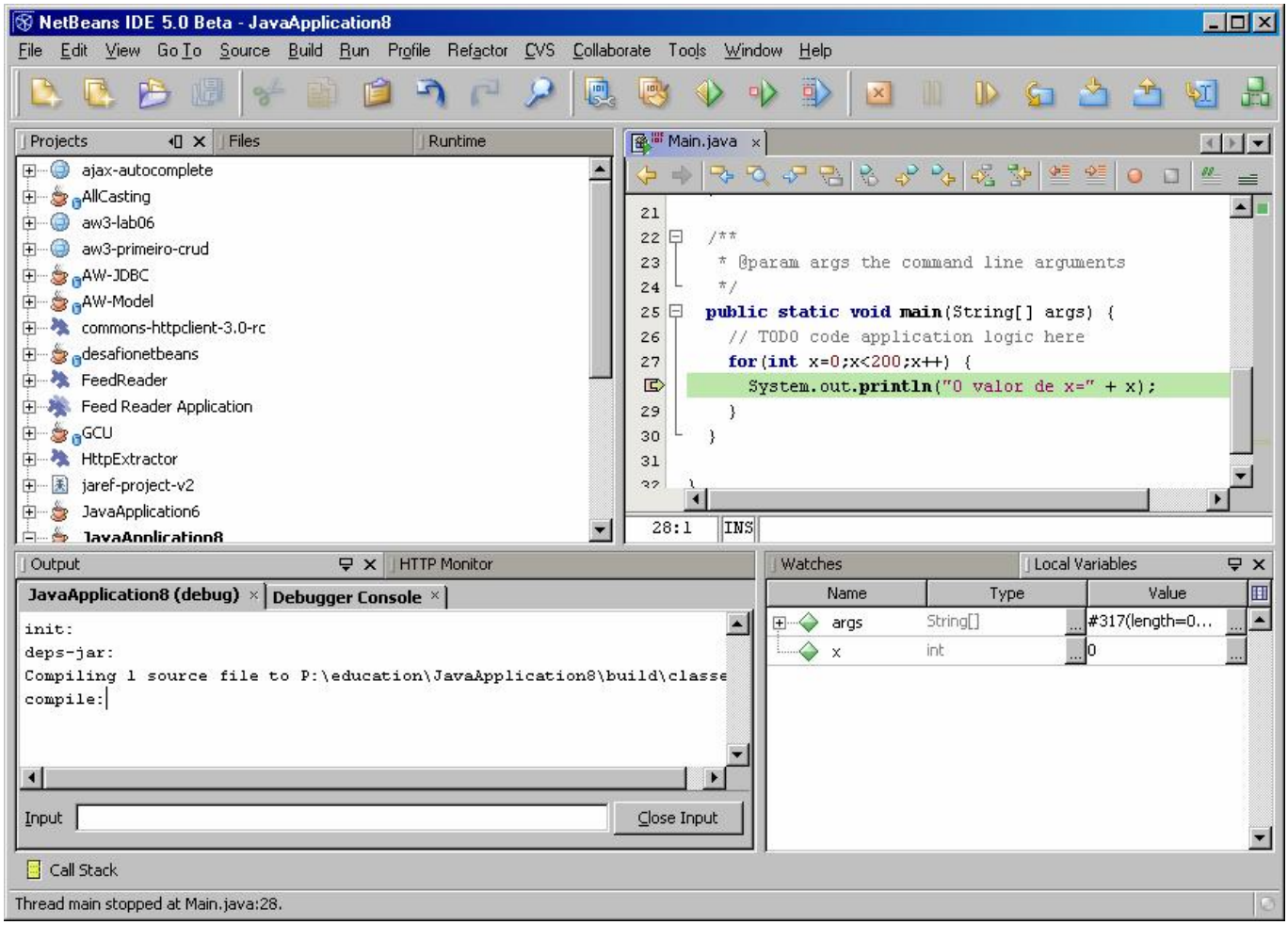
A linha com breakpoint ficará avermelhada. Ao iniciar o aplicativo em modo de debug (basicamente uma configuração VM), a linha com breakpoint ocasionará em uma pausa na execução do aplicativo, para que o IDE possa depurá-lo.

Para iniciar o processo de depuração do nosso aplicativo podemos apertar F5 ou então clicar no botão superior Debug, que fica ao lado do Run. O NetBeans vai iniciar a máquina virtual Java em modo de debug, de forma que quando ela encontrar um breakpoint no código, ela pausa sua execução e enviar uma mensagem para o depurador, neste caso o NetBeans.

Anotações

Vale observar que existe uma arquitetura de debug internamente na máquina virtual eu garante que um aplicativo qualquer, em qualquer máquina virtual possa ser depurado com diferentes ferramentas como Eclipse, JDeveloper, NetBeans e até mesmo o pouco conhecido comando de linha jdb.

Ao executar o aplicativo em modo de depuração o NetBeans vai replanejar sua área de trabalho para mostrar informações sobre variáveis, threads, stack, classes carregadas e muito mais:



O aplicativo vai pausar na linha com breakpoint tornando-a com fundo verde, você poderá correr o código passo-a-passo executando "**Step Into**" e "**Step Over**". A diferença entre Into e Over é que se a linha tratar de uma chamada a um método o Into entrará no código-fonte do método, enquanto o Over simplesmente chamará o método sem mostrar o código passo-a-passo. Para Step Over aperte F8 para Step Into aperte F7.

A partir deste ponto você poderá explora demais características de debug do NetBeans aplicando esta mesma técnica para JSP's, Servlets, WebServices, Swing etc.

Anotações

1.8 Utilizando servidores CVS

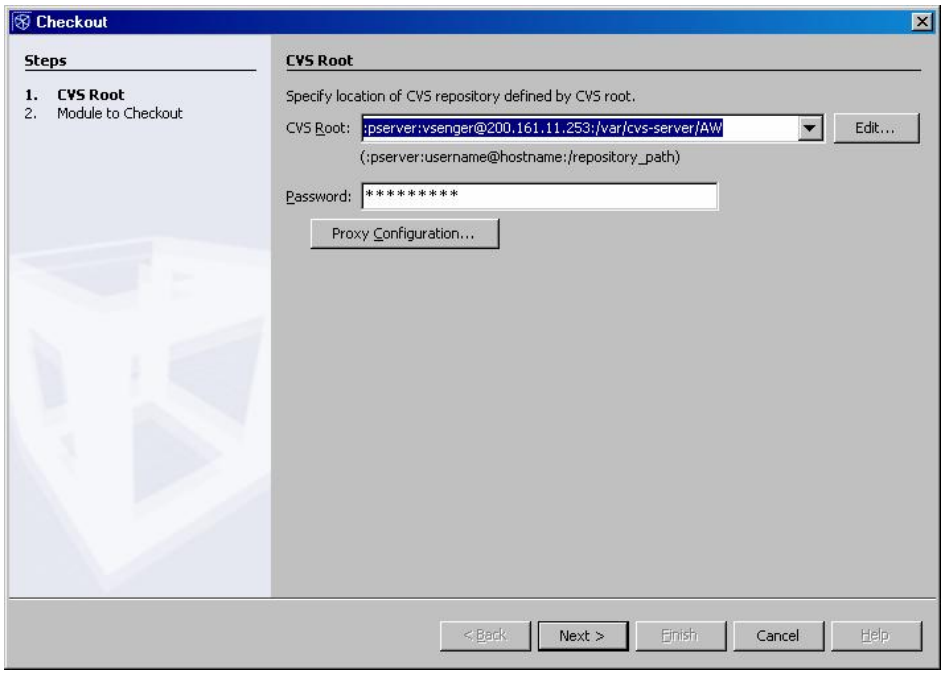
O NetBeans mudou bastante seu suporte para servidores CVS na versão 5 e agora viabilizou o uso deste recurso de integração. Faremos uma abordagem bem prática, porém antes vamos revisar alguns termos de CVS para quem ainda não teve oportunidade de utilizá-lo:

Termo	Descrição
Checkout	Operação que um programador faz para baixar uma cópia de código-fonte de um servidor CVS. Quando fazemos o checkout, obtemos uma cópia local completa do projeto, se você danificar ou perder algum arquivo, poderá baixar novamente do servidor.
Commit	Operação que um programador faz para gravar no servidor as alterações que ele fez no código-fonte local.
Update	Operação para atualizar a sua cópia local com eventuais atualizações do servidor, ou seja, você vai “re-sincronizar” sua cópia local com o que há de mais novo no servidor.
Diff	Operação para verificar diferenças entre arquivos que estão em conflito
Conflict	Você faz o checkout de um projeto e modifica um suposto arquivo Logger.java, porém antes de você efetuar o commit, um segundo desenvolvedor faz checkout e também modifica o arquivo Logger.java, efetuando a operação de commit antes que você. Neste caso dois desenvolvedores alteraram o mesmo fonte com diferentes alterações e existe uma necessidade de resolver este conflito fazendo o merge das alterações. O IDE vai ajuda-lo fortemente para isso.

Agora que vimos a terminologia e principais conceitos do CVS, vamos analisar como baixar um projeto de um servidor CVS:

Anotações

Clique no menu **CVS -> Checkout**, o seguinte diálogo surgirá:

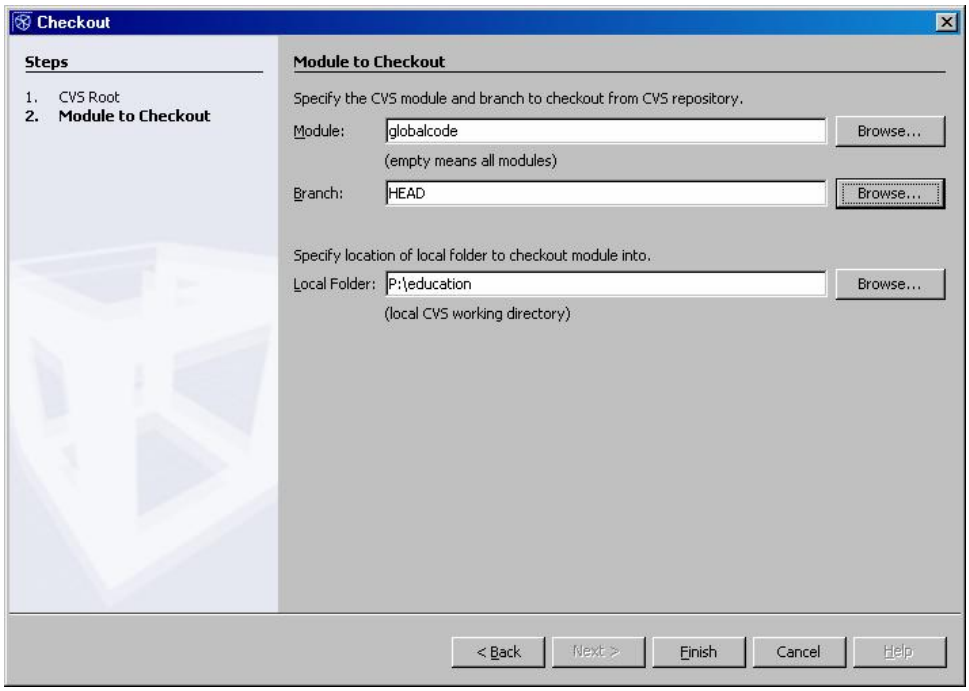


Você deve indicar uma String que representará o CVS Root: **:pserver:vsenger@cvs.dev.java.net:/cvs**

Parâmetro	Descrição
pserver	Mecanismo de autenticação / password
vsenger	Login do usuário
cvs.dev.java.net	Servidor
/cvs	Repositório CVS
Password	Senha do usuário no servidor CVS

Anotações

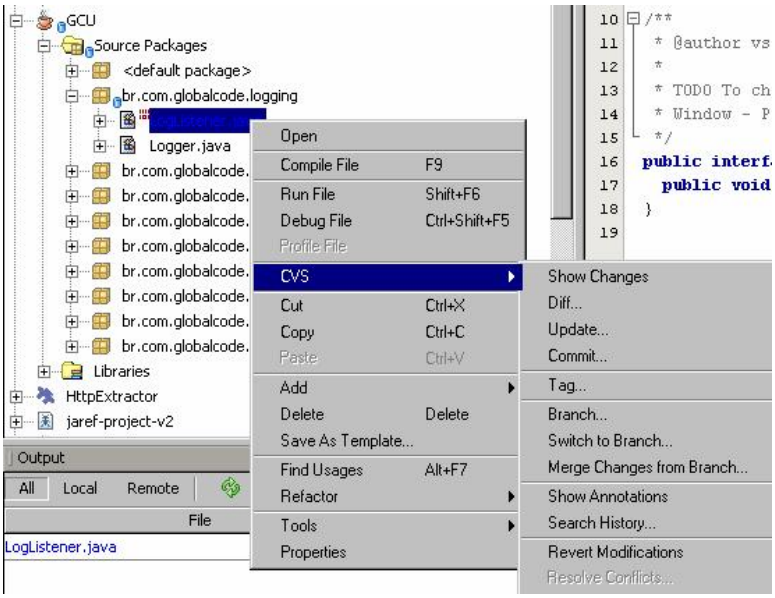
Em seguida o seguinte diálogo surgirá:



Nele podemos escolher o módulo CVS e também o branch, além do folder que desejamos receber os arquivos do CVS. Basta clicar em finish que o NetBeans iniciará o processo de Checkout do código a partir do servidor informado!

Pelo fato do projeto ter sido baixado de um servidor CVS, ao alterar um arquivo, o NetBeans mudará sua cor para azul, indicando que você deve fazer o commit do que alterou e em vermelho o que estiver com conflito.

Para navegar nas demais operações CVS, basta clicar com o botão direito do mouse em um projeto, pacote de classes ou um arquivo específico, ou seja, as operações CVS podem ser feitas em todos os níveis.



Anotações

1.9 Laboratório

Tarefa	Concluída
Inicie o NetBeans na sua máquina.	
Crie um projeto novo, sem se basear em código-fonte pré-existente.	
Escreva um pequeno programa, dentro do método main ou então Swing e execute-o.	
Utilize os recursos de Debug do NetBeans para executar seu programa passo-a-passo	
Agora vamos utilizar os recursos de CVS do NetBeans. Informe-se com o instrutor sobre o endereço CVS válido da sala-de-aula em questão e siga os passos documentados neste material para efetuar o checkout de um projeto.	

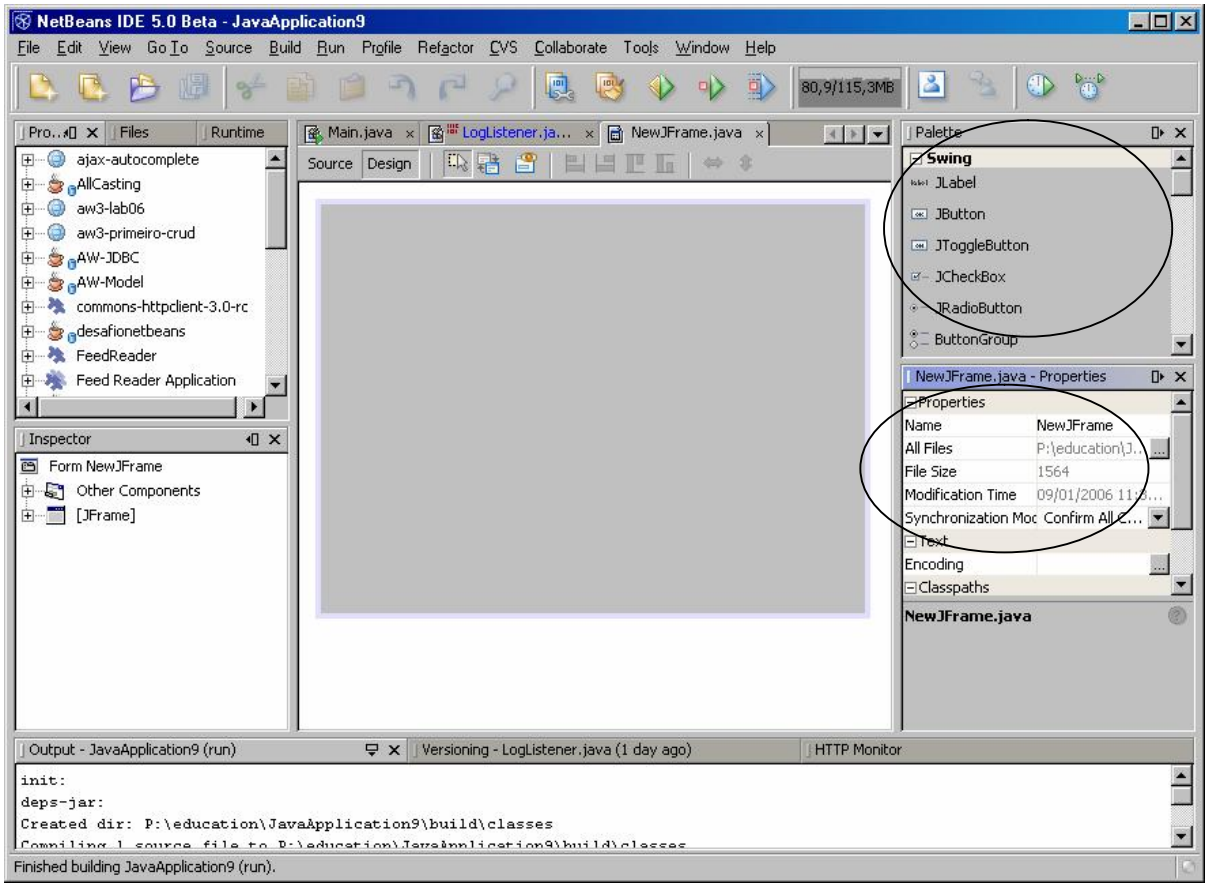
Nota para instrutor: este laboratório deverá consumir no máximo 30 minutos para não comprometer a agenda oficial.

Anotações

1.10 NetBeans e Aplicativos Desktop

Uma vez que temos o capítulo seguinte, exclusivamente para este assunto, vamos apenas apresentar passo-a-passo, como construir uma aplicativo Desktop Hello World, utilizando Swing e o editor de interfaces gráficas do NetBeans.

1. Clique em **File -> New Project** e escolha **General -> Java Application** como tipo do projeto;
2. Digite o nome e folder de localização do projeto;
3. Desabilite a opção "**Create Main Class**" e clique em **Finish**;
4. Agora vamos criar uma tela Swing, clique em **File -> New File** e escolha **Java GUI Forms – Jframe Form**;
5. Um diálogo será apresentado para você digitar o nome da classe e do pacote; digite-os e clique em **Finish**.
6. Como resultado desta operação o NetBeans apresentará o Frame Swing no seu editor visual. Caso já tenha trabalhado com Delphi ou Visual Basic, terá maior facilidade de uso do NetBeans pois diversas características são similares.
7. Para desenhar sua tela basta selecionar componentes da paleta e efetuar drag-and-drop. Ao terminar clique em Run para executar seu aplicativo.



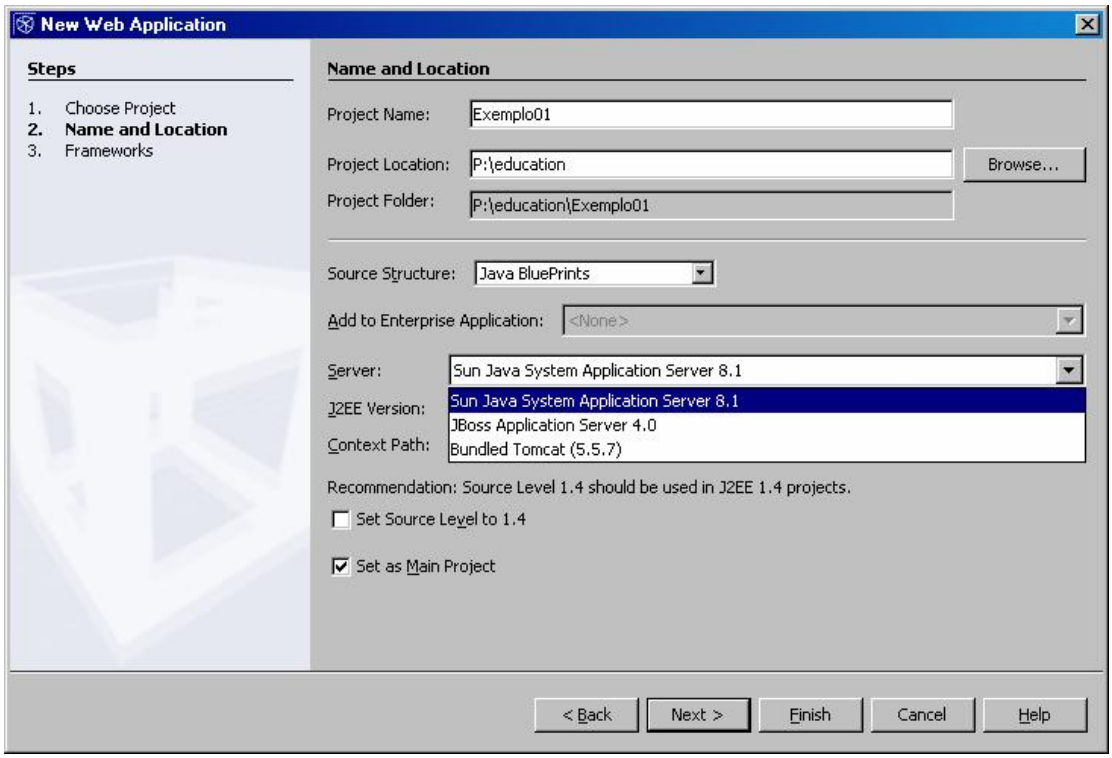
Anotações

1.11 NetBeans e Aplicativos Enterprise

Mudanças efetivas finalmente aconteceram na área Enterprise Edition e NetBeans. Principalmente no Brasil, temos um mercado extremamente consumidor de aplicativos Enterprise para Web, portanto essas mudanças com certeza refletirão em uma maior adoção do NetBeans.

Neste tópico vamos aprender como criar um projeto Web Struts com o NetBeans chegando até no debug de JSPs.

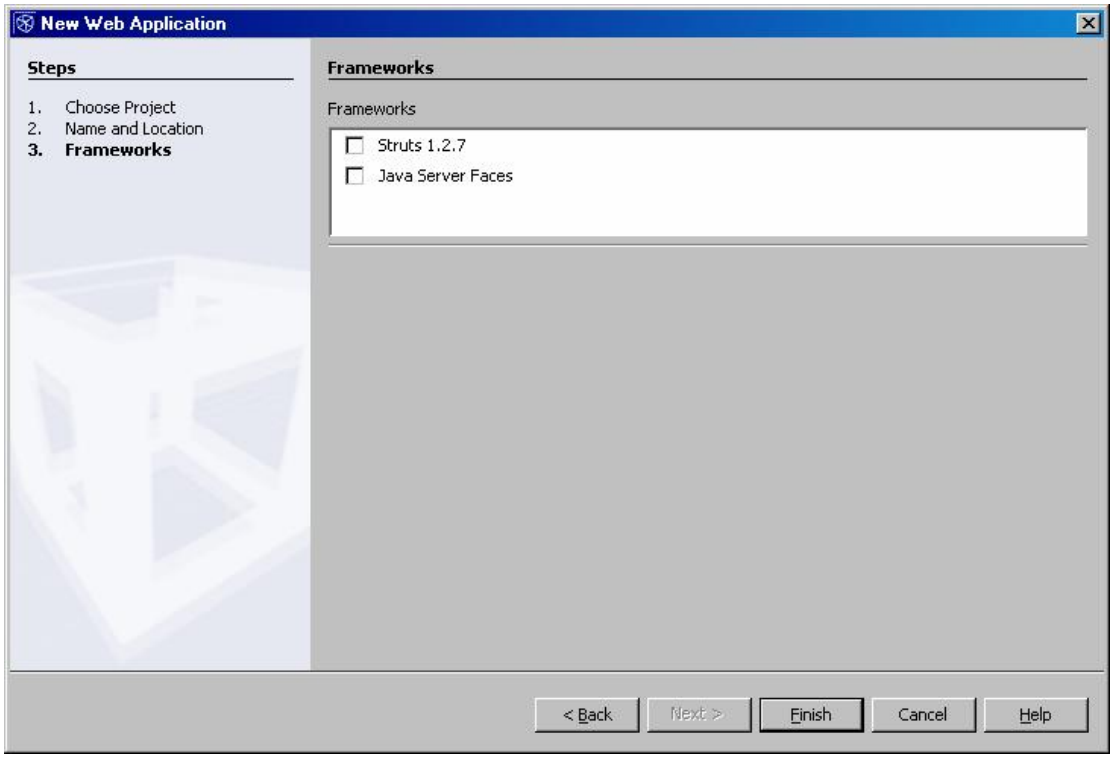
1. Clique em **File -> New Project** e escolha **Web -> Web Application** como tipo do projeto:



Neste diálogo podemos configurar o servidor de aplicativos, o context path do aplicativo e também a estrutura de organização de arquivos quie pode ser Java BluePrints ou Jakarta. Escolha o Tomcat como servidor.

Anotações

2. Em seguida um diálogo será apresentado questionando quanto ao uso de frameworks:



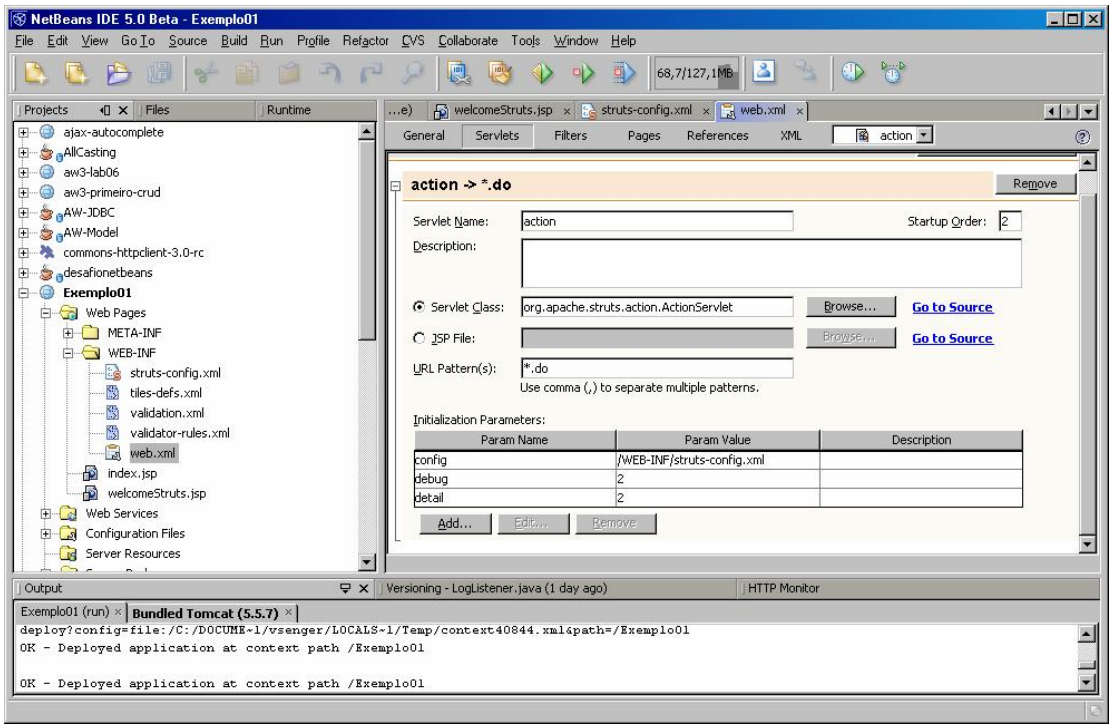
Escolha o Struts 1.2.7 e clique em finish. Como resultado você terá um aplicativo Web de exemplo com todos os arquivos e deployment descriptors pré-configurados. Basta clicar em Run para acionar a página welcomeStruts.jsp.

Dica: você pode instalar o StrutsConsole (James Holmes) como um plug-in para melhorar ainda mais o suporte a Struts e configuração de struts-config.xml.

2. Clique em Run e o NetBeans vai compilar, empacotar, efetuar o deployment no Tomcat, inicializará o Tomcat e abrirá um browser com a página inicial, quanto esforço poupado!

Anotações

3. Agora você pode navegar nas telas de configuração de web.xml conforme imagem abaixo:



4. Repara que o NetBeans tem uma paleta chamada Runtime. Nesta janela podemos iniciar e parar o application server, configurar e instalar ou desinstalar aplicativos.
5. Experipemente colocar um breakpoint no JSP e solicite Debug ao invés de Run. O debug de JSPs é transparente e funciona completamente igual ao debug de um aplicativo stand-alone.

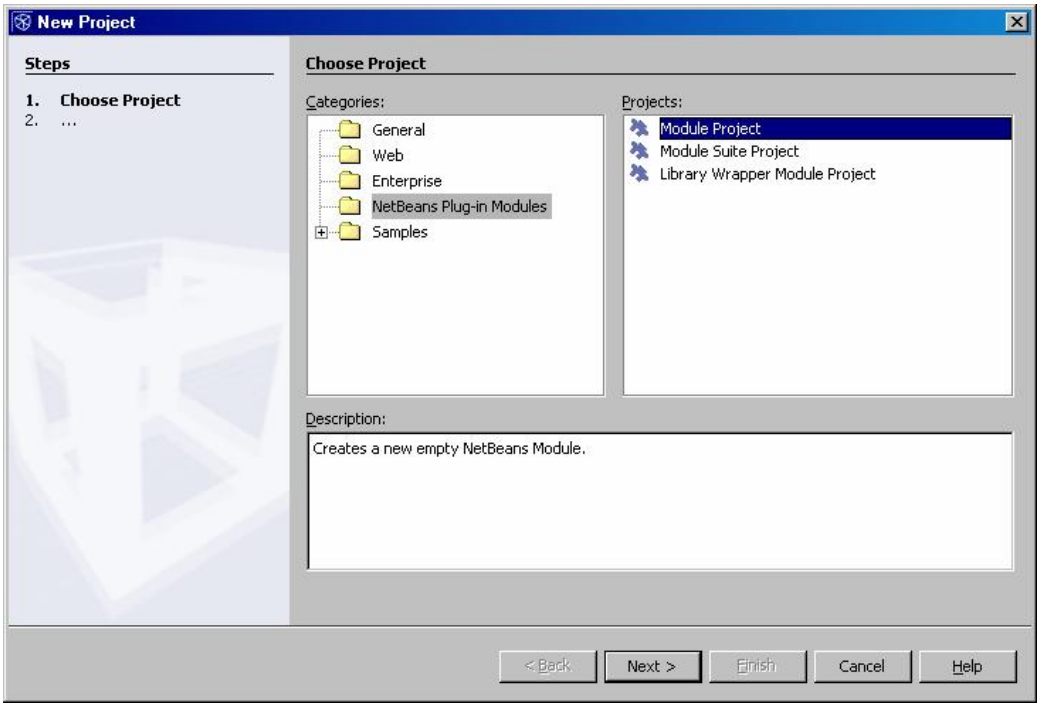
Anotações

1.12 NetBeans e Desenvolvimento de plug-ins

Uma novidade na versão 5 do NetBeans é o suporte e wizards para a criação de plug-ins para o próprio NetBeans. Através destas features você pode criar um plug-in totalmente novo ou pode transformar um aplicativo pré-existente em um plug-in. Como este material reserva também um capítulo só para desenvolvimento de plug-ins, mostraremos os primeiros passos apenas para que você estabeleça o primeiro contato para desenvolver plug-ins.

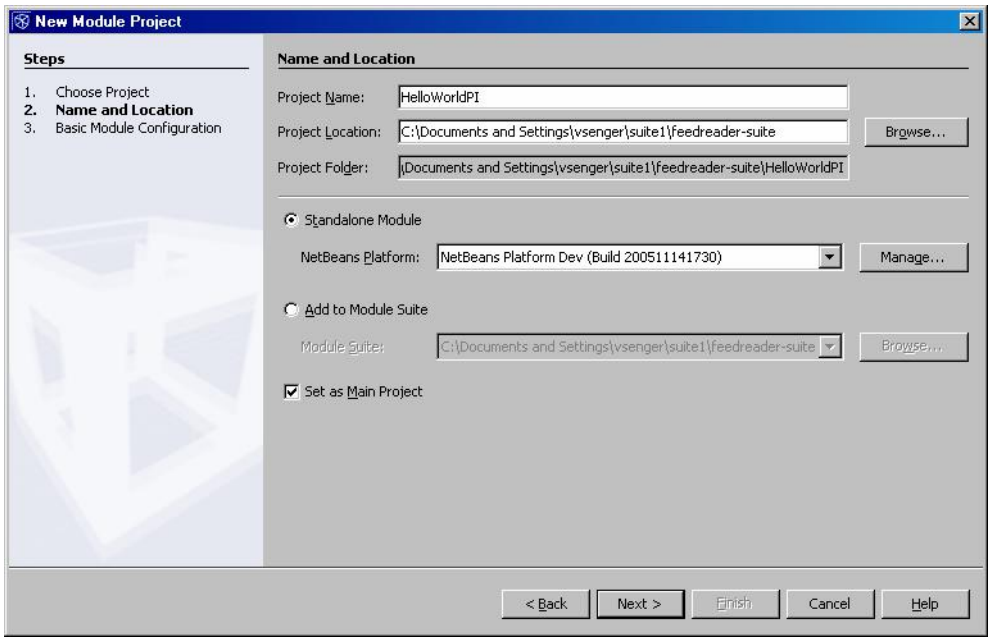
Para criar um plug-in Hello World, siga os seguintes passos:

1. Clique em **File -> New Project** e selecione **NetBeans Plug-in Modules -> Module Project**.



Anotações

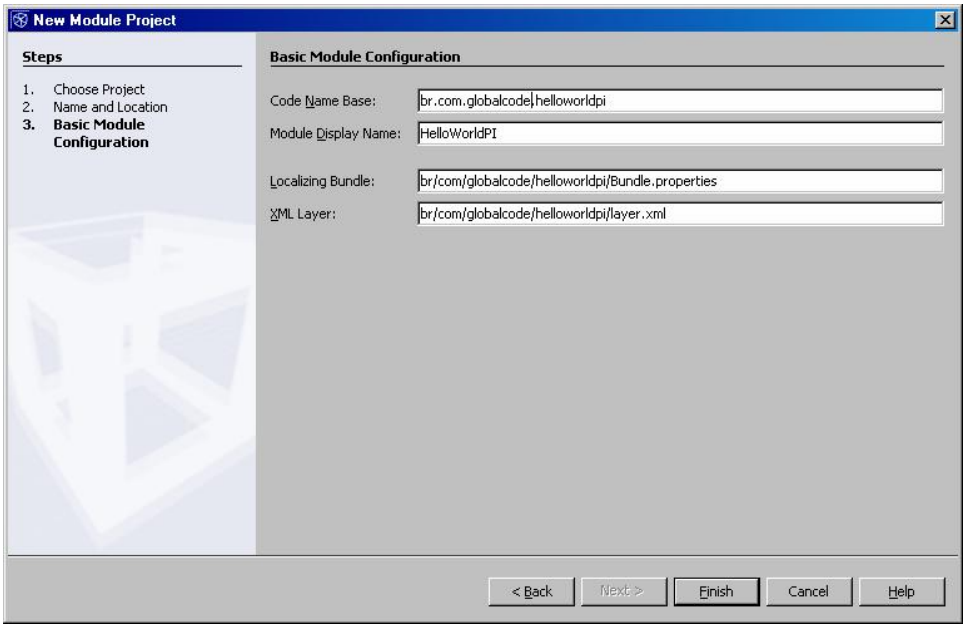
2. Escolha o nome do projeto e selecione a opção **Stand-alone Module**:



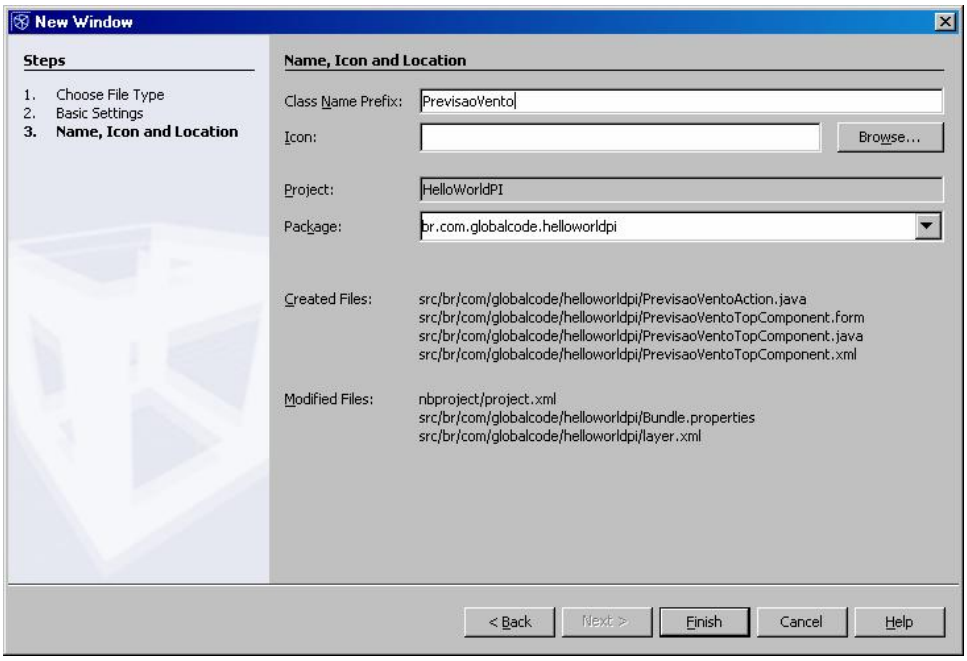
,

Anotações

3. Neste diálogo o NetBeans perguntará sobre o nome de pacotes, o nome para display do plug-in, um arquivo resource bundle para internacionalizar o plug-in e também o “deployment descriptor”. Preencha apenas o nome de pacotes que deseja utilizar o display name do seu plug-in e mantenha o restante com valores default:



4. O projeto será criado e agora vamos adicionar uma nova janela ao ambiente NetBeans. Clique em **File -> New File** e selecione **NetBeans Module Development -> Window Component**. Um diálogo perguntará sobre a posição da janela dentro do NetBeans, selecione **Explorer**. O seguinte diálogo será apresentado para você colocar o prefixo de nome de classe, selecionar opcionalmente um ícone e pacote de classes:



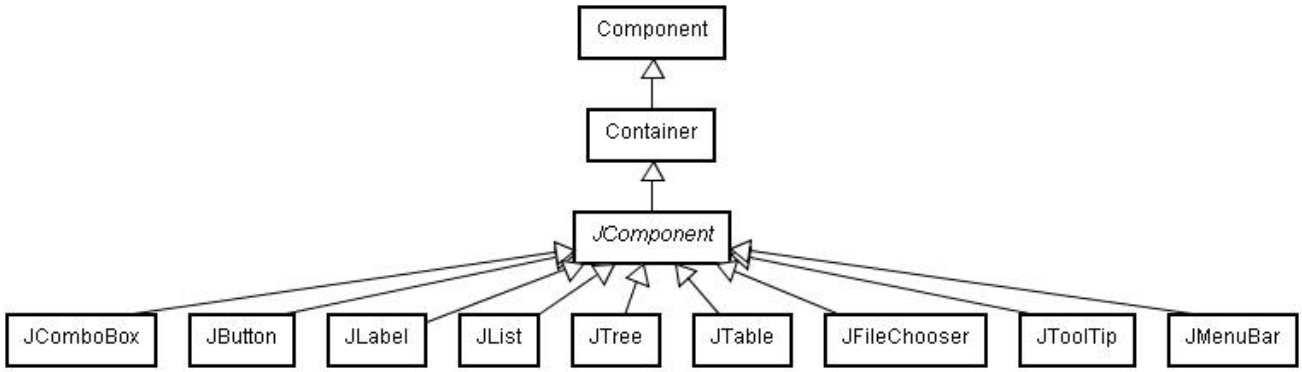
Anotações

2 Interfaces gráficas com Java e NetBeans

2.1 Swing

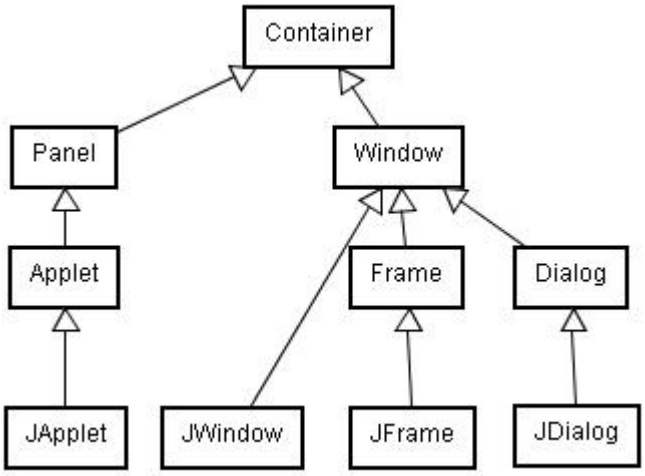
Swing é a API mais recente para desenvolvimento de interfaces gráficas baseada em AWT, mas muito mais rica em componentes.
Esta API não depende de código nativo para representar botões, janelas e outros componentes gráficos, sendo totalmente independente de plataforma e desenhando seus próprios componentes.

A classe `JComponent` é a superclasse de todos os componentes que não são containers de outros, sendo derivada de `Component`, como podemos observar no seguinte diagrama de classes:



2.1.1 Containers

Vejamos o diagrama abaixo:



Anotações

2.1.2 Criando uma janela simples

Exemplo: FrameSwing1.java

```
package br.com.globalcode.swing;

import javax.swing.*;

public class FrameSwing1 extends JFrame {

    public FrameSwing1() {
        super("Janela Swing");
        setSize(275, 100);
        show();
    }

    public static void main(String args[]) {
        FrameSwing1 t = new FrameSwing1();
    }
            }
```

Importante:

Esta janela possui comportamento padrão `HIDE_ON_CLOSE` no botão fechar (X no canto superior direito), ou seja, quando clicamos, a janela é "escondida", mas o processo continua sendo executado. Recomendamos configurar o comportamento de fechamento default, utilizando o seguinte método da classe `JFrame`, que provoca o término da execução.:

```
setDefaultCloseOperation(EXIT_ON_CLOSE);
```

Exemplo: FrameSwing2.java

```
package br.com.globalcode.swing;
import javax.swing.*;

public class FrameSwing2 extends JFrame {

  public FrameSwing2() {
    super("Janela Swing");
    // Utilizamos a constante declarada na classe JFrame para definir
    // o comportamento padrao no fechamento da janela
    setDefaultCloseOperation(EXIT_ON_CLOSE);
    setSize(275, 100);
    show();
  }
  public static void main(String args[]) {
    FrameSwing2 t = new FrameSwing2();
  }
}
```

Anotações

2.1.3 Adicionando um componente ao container

A manipulação e adição de componentes no `JFrame` é um pouco diferente da que vimos em AWT, pois nunca manipulamos diretamente o `JFrame`. Devemos sempre obter uma referência para o único `Container` contido no `JFrame`, obtido através da chamada ao método `getContentPane()`.

No exemplo abaixo, iremos utilizar o `JLabel`, um dos componentes mais simples e que representa o texto em uma única linha, não podendo ser editado pelo usuário.

Um `JLabel` pode ser construído da seguinte forma:

```
JLabel label1 = new JLabel("Texto do jlabel");
```

Exemplo: FrameSwingComLabel.java

```
package br.com.globalcode.swing;

import java.awt.Container;

import javax.swing.*;

public class FrameSwingComLabel extends JFrame {

    public FrameSwingComLabel() {

        super("Janela Swing");

        // Utilizamos a constante declarada na classe
        // JFrame para definir comportamento padrao
        // no fechamento da janela
        setDefaultCloseOperation(EXIT_ON_CLOSE);

        JLabel label = new JLabel("Texto do JLabel");

        Container c = this.getContentPane();
        c.add(label);

        setSize(275, 100);
        show();
    }

    public static void main(String args[]) {
        FrameSwingComLabel t = new FrameSwingComLabel();
    }
}
```

Anotações

2.2 Componentes Swing

2.2.1 javax.swing.JButton

Representa um botão com rótulo.

Construtores

- `JButton(String texto)`: o valor da variável texto será apresentado no botão;
- `JButton(Icon icone)`: o ícone será apresentado no botão.

Observação: Uma das formas de criarmos um ícone é a partir da classe `ImageIcon`, que pode ser construída a partir de uma `String` representando o nome da imagem.
Exemplo: `ImageIcon icone = new ImageIcon("icone.gif")`
Para mais detalhes consulte o Javadoc.

Métodos

- `void setMnemonic(int mnemonic)` : Podemos associar atalhos aos botões utilizando este método, que recebe um `int` como parâmetro, representado pelas constantes definidas na classe `KeyEvent`.

Exemplos de constantes:

`KeyEvent.VK_A` => Atalho para A
`KeyEvent.VK_B` => Atalho para B
`KeyEvent.VK_1` => Atalho para 1

Anotações

Exemplo: TesteJButton.java

```
package br.com.globalcode.swing;
import java.awt.*;
import javax.swing.*;
import java.awt.event.KeyEvent;

public class TesteJButton extends JFrame {

    public TesteJButton() {
        super("Teste JButton");
        setDefaultCloseOperation(EXIT_ON_CLOSE);
        Container c = getContentPane();

        JButton botaoOk = new JButton("Ok");
        botaoOk.setMnemonic(KeyEvent.VK_O);
        c.add(botaoOk);

        setSize(200, 50);
        show();
    }
    public static void main(String args[]) {
        TesteJButton t = new TesteJButton();
    }
}
```

Anotações

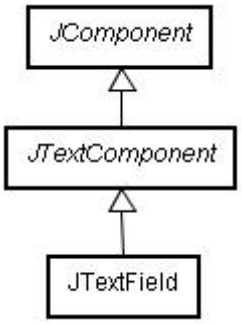
Exemplo: TesteJButtonComIcone.java


```
package br.com.globalcode.swing;

import java.awt.*;
import javax.swing.*;

public class TesteJButtonComIcone extends JFrame {

    public TesteJButtonComIcone() {
        super("Teste JButton");
        setDefaultCloseOperation(EXIT_ON_CLOSE);

        Container c = getContentPane();

        // Criacao de um icone com uma imagem
        ImageIcon iconeBotao = new ImageIcon("duke.gif");
        // Criacao de um botao com o icone iconeBotao
        JButton botaoIcone = new JButton(iconeBotao);
        // Alteramos a cor de fundo do botao para ficar compativel com a imagem
        botaoIcone.setBackground(Color.WHITE);

        c.add(botaoIcone);
        setSize(80, 150);
        show();
    }

    public static void main(String args[]) {
        TesteJButtonComIcone t = new TesteJButtonComIcone();
    }
}
```

Anotações

2.2.2 javax.swing.JTextField

Esta classe representa um campo de texto digitável, usualmente empregado para campos de cadastro de uma única linha.



Exemplo: TesteJTextField.java

```
package br.com.globalcode.swing;

import java.awt.*;
import javax.swing.*;

public class TesteJTextField extends JFrame {

    public TesteJTextField() {
      super("Teste JTextField");
      this.setDefaultCloseOperation(EXIT_ON_CLOSE);
      Container c = getContentPane();

      JTextField textField = new JTextField();
      c.add(textField);

      setSize(200, 50);
      show();
    }

    public static void main(String args[]) {
      TesteJTextField t = new TesteJTextField();
    }
}
```

Métodos para manipulação do conteúdo de um JTextField

- `String getText()`: retorna o valor do texto contido no `JTextField`;
- `void setText(String texto)`: atribui o valor da variável texto ao `JTextField`;
- `void setEditable(boolean editable)`: habilita ou desabilita o componente de texto. Este método é herdado da classe `JTextComponent`.

Anotações

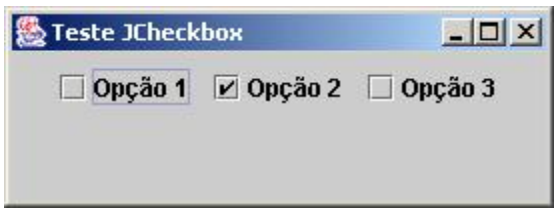
2.2.3 javax.swing.JCheckbox

Um `JCheckBox` é um componente gráfico com dois possíveis estados: “selecionado" (`true`) ou "não selecionado"(`false`).

Alguns construtores

- `JCheckBox(String texto);`
- `JCheckBox(Icon icone);`
- `JCheckBox(String icone, boolean selected);`
- `JCheckBox(Icon icone, boolean selected);`

Exemplo:TesteJCheckbox.java




```
package br.com.globalcode.swing;

import java.awt.*;
import javax.swing.*;

public class TesteJCheckbox extends JFrame {

    public TesteJCheckbox() {
        super("Teste JCheckbox");
        setDefaultCloseOperation(EXIT_ON_CLOSE);

        Container c = getContentPane();

        JPanel pOpcoes = montaPanelOpcoes();
        c.add(pOpcoes);

        setSize(275, 100);
        show();
    }

    public JPanel montaPanelOpcoes() {
        JPanel p = new JPanel();
        p.add(new JCheckbox("Opção 1"));
        // Podemos indicar qual o item selecionado
        p.add(new JCheckbox("Opção 2", true));
        p.add(new JCheckbox("Opção 3"));
        return p;
    }
    public static void main(String args[]) {
        TesteJCheckbox t = new TesteJCheckbox();
    }
}
```

Anotações

2.2.4 javax.swing.JComboBox

Construtores

- `JComboBox();`
- `JComboBox(Object[] itens);`
- `JComboBox(Vector itens).`

Principais métodos

- `void addItem(Object o)`: adiciona um item na ultima posição;
- `Object getSelectedItem()`: retorna o item selecionado;
- `void insertItemAt(Object item, int posicao)`: insere um objeto na posição especificada;
- `Object getItemAt(int posição)`: retorna o item que estiver na posição especificada ou `null` se ele não existir;
- `void removeAllItens()`: remove todos os itens do `JComboBox;`
- `void removeItemAt(int posicao)`: remove o item que estiver na posição especificada;
- `void setEnabled(boolean habilitado)`: habilita ou não o `JComboBox;`
- `void setSelectedItem(Object item)`: configura qual será o item selecionado;
- `void setSelectedIndex(int posicao)`: configura qual será o item selecionado através da sua posição no `JComboBox`.

O componente `JComboBox` apresenta uma lista com scroll de itens, podendo ser configurado para que cada elemento possa ser editável.

É possível construirmos um `JComboBox` passando como parâmetro um `Vector`, contendo os elementos que queremos exibir no `JComboBox;` o texto exibido será o resultado da chamada ao método `toString` de cada componente.

Neste exemplo, estamos adicionando objetos do tipo `String` ao `JComboBox` através do método `addItem(String)`

Anotações

Exemplo: TesteJComboBox.java

```
package br.com.globalcode.swing;

import java.awt.*;
import javax.swing.*;

public class TesteJComboBox extends JFrame {

    public TesteJComboBox() {
        super("Teste JComboBox");
        setDefaultCloseOperation(EXIT_ON_CLOSE);
        Container c = getContentPane();

        JComboBox seletorBancos = new JComboBox();
        seletorBancos.addItem("Itau");
        seletorBancos.addItem("Bradesco");
        seletorBancos.addItem("Globalcode Bank");

        c.add(seletorBancos);
        setSize(275, 50);
        show();
    }
    public static void main(String args[]) {
        TesteJComboBox t = new TesteJComboBox();
    }

}
```

Anotações

2.2.5 javax.swing.JList

O componente `JList` apresenta uma lista com scroll de itens, podendo ser configurado para que o usuário possa selecionar apenas um ou múltiplos itens. È possível adicionar qualquer componente. Construímos um `JList` passando como parâmetro um `Vector`, contendo os elementos que queremos exibir, o texto exibido será o resultado da chamada ao método `toString` de cada componente.

Construtores

- `JList();`
- `JList(Object[] itens);`
- `JList(Vector itens).`

Principais métodos

- `int getSelectedIndex()`: retorna o índice do primeiro item selecionado;
- `int[] getSelectedIndices()`: retorna um array de inteiros contendo os indices dos itens selecionados;
- `Object getSelectedValue()`: retorna o primeiro item selecionado;
- `Object[] getSelectedValues()`: retorna um array de objetos contendo os itens selecionados.

Vamos utilizar a classe `Cliente` definida abaixo para adicionar no componente `JList`.

Exemplo: Cliente.java

```
package br.com.globalcode.beans;

public class Cliente {

    private String nome;
    private int id;
    private String telefone;

    public Cliente(String nome, int id, String telefone) {
        this.setNome(nome);
        this.setId(id);
        this.setTelefone(telefone);
    }
    // Estamos omitindo os getters e setters

    public String toString() {
        return "[ " + id + " ] " + nome;
    }
}
```

Anotações

Exemplo: TesteJList.java

```
package br.com.globalcode.swing;

import java.awt.*;
import java.util.Vector;
import javax.swing.*;
import br.com.globalcode.beans.Cliente;

public class TesteJList extends JFrame {

    public TesteJList() {
        super("Teste JList");
        setDefaultCloseOperation(EXIT_ON_CLOSE);
        Container c = getContentPane();

        Vector clientes = new Vector();
        Cliente c1 = new Cliente("Waldir", 1, "213213");
        Cliente c2 = new Cliente("Ana Paula", 2, "1211213");
        Cliente c3 = new Cliente("Ricardo", 3, "45454545");
        clientes.add(c1);
        clientes.add(c2);
        clientes.add(c3);

        JList listClientes = new JList(clientes);
        c.add(listClientes);

        setSize(275, 100);
        show();
    }
    public static void main(String args[]) {
        TesteJList t = new TesteJList();
    }
}
```

Anotações

2.2.6 Construindo Menus

javax.swing.JMenuBar

A classe `JMenuBar` representa uma barra de menu de um `JFrame`. Para associá-la a um `JFrame`, o método `setJMenuBar` da classe `JFrame` deve ser chamado, passando o próprio `JMenuBar` como parâmetro.

Uma barra de menu pode ter atalhos para seus itens. Cada menu pode ter uma instância de `MenuShortcut`. A classe `JMenuBar` possui diversos métodos para gerenciar atalhos de teclado, semelhantemente ao que existe a classe `Jbutton`.

javax.swing.JMenu

Um objeto da classe `JMenu` representa um item de um componente de menu, alocado em uma barra (`JMenuBar`).

javax.swing.JMenuItem

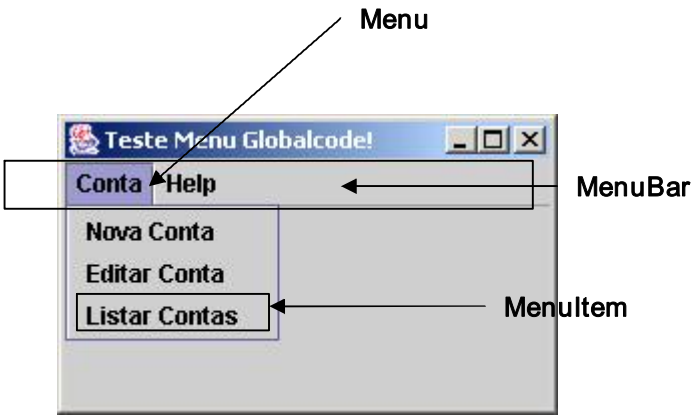
Todos os itens de um menu devem pertencer à classe `JMenuItem` ou uma de suas subclasses.

Exemplo: TesteJMenu.java

```
package br.com.globalcode.swing;
import javax.swing.*;

public class TesteJMenu extends JFrame {

    public TesteJMenu() {
        super("Teste Menu Globalcode!");
        JMenuBar menu = this.montaMenu();
        setJMenuBar(menu);
        setSize(250, 150);
        show();
    }

    public JMenuBar montaMenu() {

        JMenuBar barraMenu = new JMenuBar();
        // Montando o menu Conta
        JMenu menuConta = new JMenu("Conta");
        JMenuItem item1menuConta = new JMenuItem("Nova Conta");
        JMenuItem item2menuConta = new JMenuItem("Editar Conta");
        JMenuItem item3menuConta = new JMenuItem("Listar Contas");
        menuConta.add(item1menuConta);
        menuConta.add(item2menuConta);
        menuConta.add(item3menuConta);
```

Anotações

```
        // Montando o menu Help
        JMenu menuHelp = new JMenu("Help");
        JMenuItem item1menuHelp = new JMenuItem("Globalcode Bank Help");
        JMenuItem item2menuHelp = new JMenuItem("Problemas comuns");
        JMenuItem item3menuHelp = new JMenuItem("Ajuda Online");

        menuHelp.add(item1menuHelp);
        menuHelp.add(item2menuHelp);
        menuHelp.add(item3menuHelp);

        // Adicionando os menus Conta e Help a barra de Menus
        barraMenu.add(menuConta);
        barraMenu.add(menuHelp);

        return barraMenu;
    }

    public static void main(String args[]) {
        TesteJMenu t = new TesteJMenu();
    }
}
```



Anotações

2.3 Gerenciadores de Layout

Para organização das disposições dos componentes dentro das janelas, contamos com o conceito de classes de Gerenciamento de Layouts. Para cada container (`Panel/JPane`, `Dialog/JDialog`, `Frame/JFrame` etc.) podemos escolher um tipo de organizador de layout diferente , conforme a necessidade.

Para controlarmos o tamanho e posição dos nossos componentes utilizamos um dos métodos a seguir: `setLocation, setSize e setBounds.`

No entanto, com o uso dos gerenciadores de layout, estaremos criando um programa dependente de plataforma, pois cada sistema operacional pode trabalhar com servidores gráficos diferentes que terão outras fontes de letra e noções de dimensionamento.

Os organizadores(LayoutManager) ficarão responsáveis pelo alinhamento e posicionamento dos componentes na interface gráfica, podemos controlar os componentes manualmente, eliminando-o:

```
JPanel p1=new JPanel();
p1.setLayout(null);
```

As cinco implementações de LayoutManager são:

- `FlowLayout;`
- `BorderLayout;`
- `GridLayout;`
- `CardLayout;`
- `GridBagLayout;`

O tipo default para `Frame` e `Dialog` é `BorderLayout` e para `Panel` e `Applet` é `FlowLayout`.
Também é possível criarmos uma subclasse da interface `LayoutManager` para gerar nosso organizador de layout.

2.3.1 java.awt.FlowLayout

Quando usamos o `FlowLayout` os componentes são adicionados da esquerda para a direita da área disponível dentro do container.
O alinhamento default dentro da área é centralizado, sendo possível modificá-lo.

Algumas constantes para alinhamento

- `final static int CENTER;`
- `final static int LEFT;`
- `final static int RIGHT;`

Anotações

Construtores

- `FlowLayout();`
- `FlowLayout(int align)`: Normalmente utiliza-se alguma das constantes para alinhamento;
- `FlowLayout(int align, int hgap, int vgap)`: Indica qual o tipo de alinhamento desejado além dos espaçamentos (em pixels) do topo da tela, tanto em termos verticais como em termos horizontais.

Exemplo: TesteFlowLayout.java


```
package br.com.globalcode.swing;

import java.awt.Container;
import java.awt.FlowLayout;
import javax.swing.*;
import javax.swing.JPanel;
import javax.swing.JButton;

public class TesteFlowLayout extends JFrame {

    public TesteFlowLayout() {
        super("Teste FlowLayout Globalcode!");
        Container c = getContentPane();

        JPanel pBotoes = montaJPanelBotoes();
        c.add(pBotoes);
        setSize(275, 100);
        show();
    }

    public JPanel montaJPanelBotoes() {
        JPanel p = new JPanel();
        p.setLayout(new FlowLayout(FlowLayout.CENTER));
        p.add(new JButton("Salvar"));
        p.add(new JButton("Excluir"));
        return p;
    }
    public static void main(String args[]) {
        TesteFlowLayout t = new TesteFlowLayout();
    }
}
```

Anotações

2.3.2 java.awt.GridLayout

Este gerenciador de layout divide a área disponível em linhas e colunas, num esquema semelhante a uma planilha de cálculos.
Os componentes são adicionados da esquerda para direita e de cima para baixo, na ordem de chamada do método `add`.

O único inconveniente é que todas as células têm o mesmo tamanho, ou seja, se quisermos adicionar um pequeno texto e uma grande área de texto, dentro do `GridLayout`, ambos terão o mesmo tamanho, como poderemos observar no exemplo abaixo.

Exemplo: TesteGridLayout.java

```
package br.com.globalcode.swing;

import java.awt.Container;
import java.awt.GridLayout;
import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.JTextField;
import javax.swing.JLabel;

public class TesteGridLayout extends JFrame {

    public TesteGridLayout() {
        super("Teste Grid Layout Globalcode!");
        Container c = getContentPane();
        JPanel pLogin = montaJPanelLogin();
        c.add(pLogin);
        this.setSize(400, 150);
        this.show();
    }

    public JPanel montaJPanelLogin() {
        JPanel p = new JPanel();
        JTextField tfNumeroConta = new JTextField("", 10);
        JTextField tfAgencia = new JTextField("", 10);
        JTextField tfBanco = new JTextField("", 10);
        JTextField tfLimite = new JTextField("", 10);
        JTextField tfCorrentista = new JTextField("", 30);

        p.setLayout(new GridLayout(5, 2));
```

Anotações

```
        p.add(new JLabel("Numero da conta"));
        p.add(tfNumeroConta);
        p.add(new JLabel("Agencia"));
        p.add(tfAgencia);
        p.add(new JLabel("Banco"));
        p.add(tfBanco);
        p.add(new JLabel("Limite"));
        p.add(tfLimite);
```

```
        p.add(new JLabel("Correntista"));
        p.add(tfCorrentista);
```

```
        return p;
    }

    public static void main(String args[]) {
        TesteGridLayout t = new TesteGridLayout();
    }
}
```

Anotações

2.3.3 java.awt.BorderLayout

Organizador padrão para frames onde os componentes são adicionados em cinco áreas determinadas e definidas através de constantes numéricas:

- `final static String NORTH` : NORTE;
- `final static String SOUTH` : SUL;
- `final static String EAST`: LESTE;
- `final static String WEST`: OESTE;
- `final static String CENTER`: CENTRO.

Construtores

- `BorderLayout ();`
- `BorderLayout (int hgap, int vgap).`

Exemplo: TesteBorderLayout.java


```
package br.com.globalcode.swing;

import java.awt.BorderLayout;
import java.awt.FlowLayout;
import java.awt.Container;
import java.awt.GridLayout;

import javax.swing.*;

public class TesteBorderLayout extends JFrame {

    public TesteBorderLayout() {
        super("Teste Border Layout Globalcode!");
        Container c = getContentPane();
        setSize(400, 200);

        c.setLayout(new BorderLayout());

        JPanel pBotoes = montaJPanelBotoes();
        JPanel pLogin = montaJPanelLogin();

        c.add(new JLabel("Border Layout na posição NORTH"), BorderLayout.NORTH);
        c.add(pBotoes, BorderLayout.SOUTH);
        c.add(pLogin, BorderLayout.CENTER);

        show();
    }
```

Anotações

```
    public JPanel montaJPanelBotoes() {
        JPanel p = new JPanel();
        p.setLayout(new FlowLayout(FlowLayout.CENTER));
        JButton bSalvar = new JButton("Salvar");
        p.add(bSalvar);
        JButton bExcluir = new JButton("Excluir");
        p.add(bExcluir);
        return p;
    }

    public JPanel montaJPanelLogin() {
        JPanel p = new JPanel();
        p.setLayout(new GridLayout(5, 2));
        JTextField tfNumeroConta = new JTextField("", 10);
        JTextField tfAgencia = new JTextField("", 10);
        JTextField tfBanco = new JTextField("", 10);
        JTextField tfLimite = new JTextField("", 10);
        JTextField tfCorrentista = new JTextField("", 30);
        p.add(new JLabel("Numero da conta"));
        p.add(tfNumeroConta);
        p.add(new JLabel("Agencia"));
        p.add(tfAgencia);
        p.add(new JLabel("Banco"));
        p.add(tfBanco);
        p.add(new JLabel("Limite"));
        p.add(tfLimite);
        p.add(new JLabel("Correntista"));
        p.add(tfCorrentista);
        return p;
    }

    public static void main(String args[]) {
        TesteBorderLayout t = new TesteBorderLayout();
    }
}
```

Anotações

2.3.4 FreeLayout

O NetBeans 5 ganhou diversas facilidades adicionais para o desenvolvimento de uma interface gráfica. Uma das questões polêmicas da API Swing são os gerenciadores de layout, pois ao mesmo tempo que trazem indiscutíveis benefícios para o desenvolvimento multi-plataforma, acabou tornando o desenvolvimento de uma tela um pouco burocrático.

O NetBeans 4 já contava com um recurso chamado GridBagLayoutEditor que facilita bastante as configurações daquela dúzia de parâmetros de dimensionamento, margem, mescla de células etc. O que dizem é que GridBagLayout é fácil, depois que você aprende... E realmente uma tela bem feita com GridBagLayout conquista um usuário.

Agora para ficar mais fácil ainda o grupo NetBeans desenvolveu o Matisse, que conta com um novo organizador de layout, ainda específico do NetBeans porém será agregado na plataforma padrão no futuro. Trata-se do FreeLayout, um gerenciador que permite que você desenvolva uma tela com “mãos livres”, como NullLaytout; e o gerenciador vai te ajudando com recursos do tipo SnapToGrid, sugerindo a posição ideal do componente, seguindo melhores práticas e padrão de desenvolvimento de telas com semântica amplamente aceita.

Anotações

2.4 Eventos

Eventos são ações que um usuário executa na interface gráfica. Temos 3 componentes muito importantes no tratamento de eventos:

- o objeto que representa o evento;
- o gerador do evento;
- aquele que trata o evento gerado.

Vejamos a descrição abaixo:

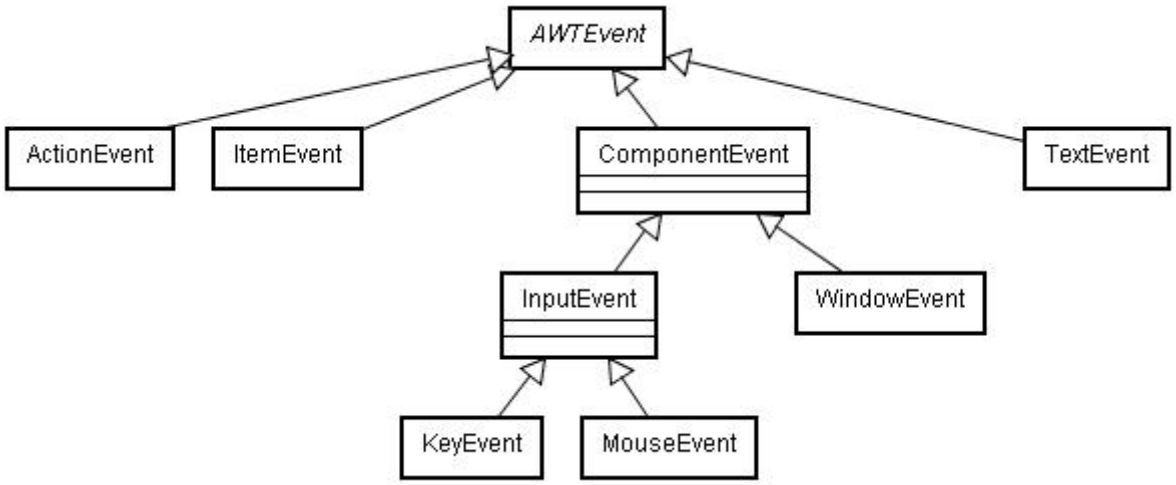
1. Events

Objeto que descreve o evento gerado. Por exemplo, um click de mouse, uma tecla pressionada, ou uma minimização de uma janela.

Existem diversos tipos de eventos; veja alguns exemplos:

- `MouseEvent`: eventos de mouse;
- `KeyEvent`: eventos de teclado;
- `WindowEvent`: eventos de janela;

Diagrama UML representando a hierarquia de algumas classes que representam eventos:



Anotações

2. Event Sources

Gerador do evento: botões, combo boxes, janelas, listas, entre outros.

3. Event Handlers

Métodos que recebem um objeto evento e executam as funcionalidades definidas pelo evento. Normalmente são programados dentro de classes chamadas listeners.
Todo evento é enviado para o componente o que gerou, mas cada componente pode propagá-lo para uma ou mais classes registradas como Listeners.

Anotações

2.4.1 Listeners

Listeners contém event handlers (métodos) que recebem e processam o evento. Desta maneira, o gerenciador de eventos pode ser um objeto separado do componente.

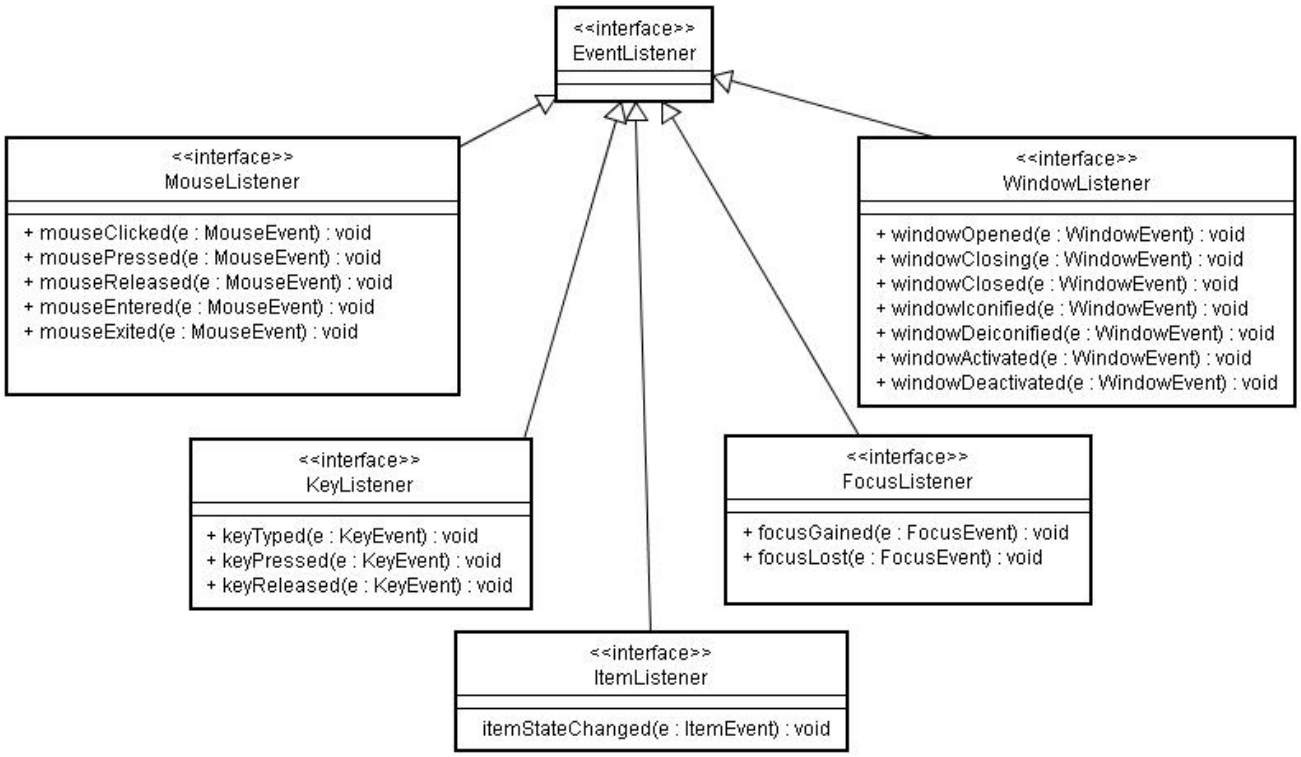
Geralmente, listeners são criados implementando interfaces. Por exemplo, se quisermos criar um listener para "ouvir" eventos de mouse (`MouseEvent`), devemos criar uma classe que implemente a interface `MouseListener`. Analogamente há diversas interfaces que especificam listeners.

Vejamos alguns exemplos:

- `MouseListener;`
- `KeyListener;`
- `WindowListener;`
- `ActionListener.`

Cada componente pode disparar eventos de uma ou mais interfaces.

Hierarquia de algumas interfaces que representam listeners:



A classe `Component`, superclasse de praticamente todos os componentes AWT e Swing, define diversos métodos para associar um listener ao nosso componente, possibilitando programar a tarefa que deverá ser realizada quando os eventos esperados acontecerem.

Anotações

Alguns métodos definidos pela classe Component são:

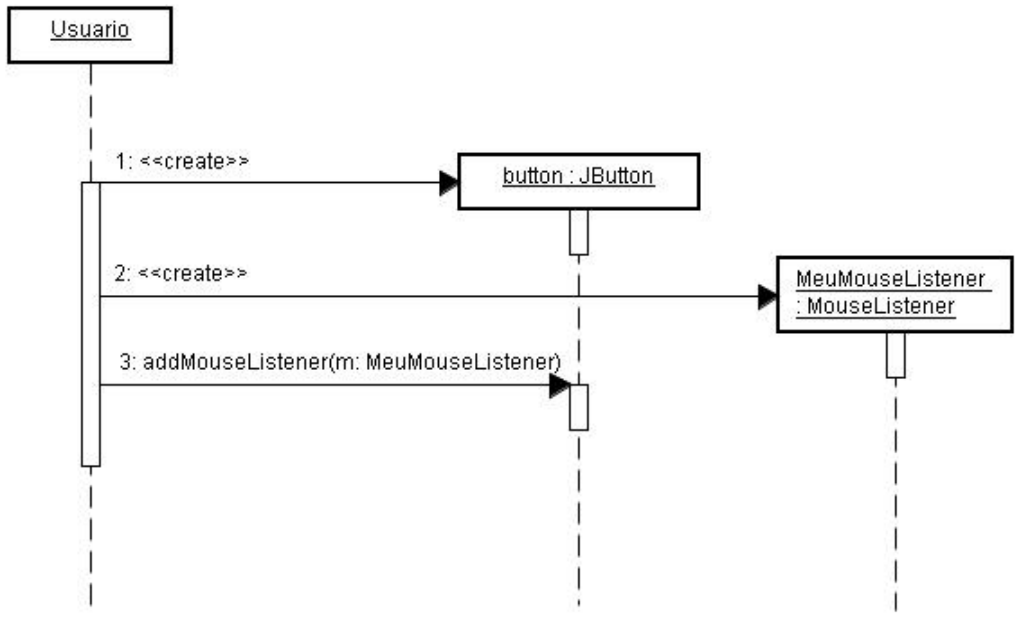
- `addKeyListener(KeyListener listener)`: método para associar um `KeyListener` a um componente;
- `addMouseListener(MouseListener listener)`: método para associar um `MouseListener` a um componente.

Alguns métodos da classe AbstractButton, que é a superclasse de JButton e JMenuItem

- `addActionListener(ActionListener listener)`: método para adicionar um `ActionListener` a um `JButton` ou `JMenuItem`(ou qualquer outra classe derivada de `AbstractButton`).
- `addItemListener(ItemListener listener)`: método para associar um `ItemListener` para um `JButton` ou `JMenu` (ou qualquer outra classe derivada de `AbstractButton`).

Imprimir um texto na console quando um botão for clicado:

- Instancia um objeto cuja classe implementa `MouseListener`, que é a interface que especifica os eventos de mouse, no método adequado iremos programar a impressão na console, que no nosso caso é a operação desejada.
- Instanciar um objeto desta classe.
- Associar este objeto ao botão.



Quando um usuário clica em um botão com o mouse, o método respectivo de `MouseListener` será executado.
A implementação desta interface pode ocorrer de diversas formas, como se segue.

Anotações

2.4.2 Uma classe externa implementa a interface java.awt.XXXListener

Temos diversas técnicas de orientação a objetos no Java para implementarmos interfaces e classes de eventos. Uma delas é quando a interface listener é implementada por uma classe independente do `JFrame`.
Neste exemplo veremos a interface `MouseListener`, que contém cinco métodos:

```
public void mousePressed  (MouseEvent e);
public void mouseReleased (MouseEvent e);
public void mouseEntered  (MouseEvent e);
public void mouseExited   (MouseEvent e);
public void mouseClicked  (MouseEvent e);
```

No exemplo abaixo, criamos uma classe chamada `EventosMouse` que implementa todos os métodos da interface `MouseListener`.
A implementação é bem simples e apenas imprime uma mensagem na console através da chamada ao `System.out.println()`.
Depois disto, associaremos uma instância de `EventosMouse` ao botão cujos "cliques" queremos monitorar ou reagir.

Exemplo: EventosMouse.java

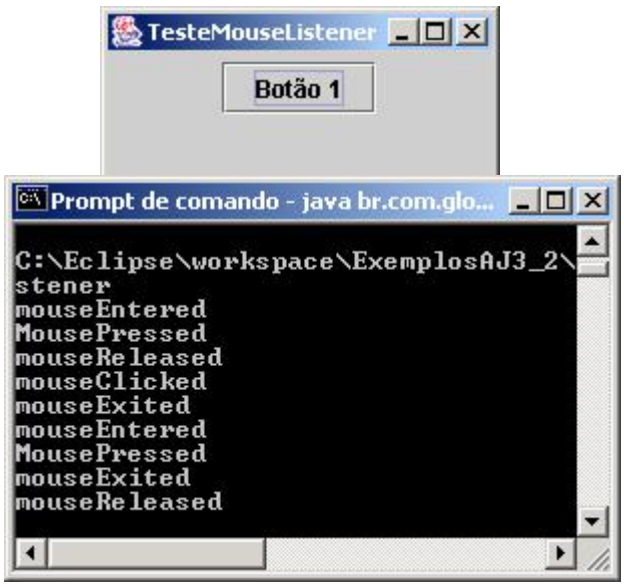
```
package br.com.globalcode.gui.eventos;
import java.awt.event.MouseEvent;
import java.awt.event.MouseListener;

public class EventosMouse implements MouseListener{
     /**************************************************************************
      * Implementamos, na mesma classe, os metodos da interface MouseListener
      *************************************************************************/
    public void mousePressed(MouseEvent e) {
      System.out.println("MousePressed ");
    }
    public void mouseReleased(MouseEvent e) {
      System.out.println("mouseReleased ");
    }
    public void mouseEntered(MouseEvent e) {
      System.out.println("mouseEntered ");
    }
    public void mouseExited(MouseEvent e) {
      System.out.println("mouseExited ");
    }

    public void mouseClicked(MouseEvent e) {
      System.out.println("mouseClicked ");
    }
}
```

Anotações

Apresentamos agora a classe `TesteMouseListener`, que tem um `JPanel` com um botão. O objetivo é monitorar todos os eventos possíveis que são gerados com o mouse sobre o botão, assim associaremos um `MouseListener` ao botão, (implementação da interface `MouseListener`) que é uma instância da classe `EventosMouse`.

Exemplo: TesteMouseListener.java

```
package br.com.globalcode.swing;

import java.awt.Container;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JPanel;
import br.com.globalcode.gui.eventos.*;

public class TesteMouseListener extends
JFrame {

  public TesteMouseListener(String
tituloJanela) {
    super(tituloJanela);
    Container c = getContentPane();

    JPanel p1 = new JPanel();
    JButton b1 = new JButton("Botão 1");
    p1.add(b1);

    c.add(p1);
```




```
    EventosMouse listener = new EventosMouse();
    b1.addMouseListener(listener);

    setSize(200, 200);
    show();
  }

  public static void main(String args[]){
    TesteMouseListener frame = new TesteMouseListener("TesteMouseListener");
  }
}
```

Os métodos do EventosMouse são executados nos seguintes casos:

- mouseEntered: quando o mouse passa sobre o desenho do botão;
- mousePresed: quando pressionamos o botão do mouse sobre o JButton;
- mouseChecked: quando pressionamos e soltamos um dos botões do mouse rapidamente sobre o JButton;
- mouseRelease: quando soltamos o botão do mouse fora do JButton.

Anotações

2.4.3 A própria classe implementa a interface java.awt.event.XXXListener

Também podemos fazer com que a própria classe que estende `JFrame` implemente a interface `MouseListener`, ou qualquer outro `XXXListener`.

No exemplo anterior, não houve manipulação de outros componentes do painel, no entanto, pode ser necessário ler ou alterar valores de caixas de texto, botões, listas e outros componentes. Neste caso, é importante deixar os componentes a serem manipulados como atributos da classe, para que estejam acessíveis no método que irá manipulá-los, e não como variáveis locais declaradas nos métodos.

O exemplo abaixo é uma continuação do exemplo anterior, mas agora adicionamos um `JLabel` contendo um texto que irá ser modificado de acordo com os eventos do mouse sobre o botão.

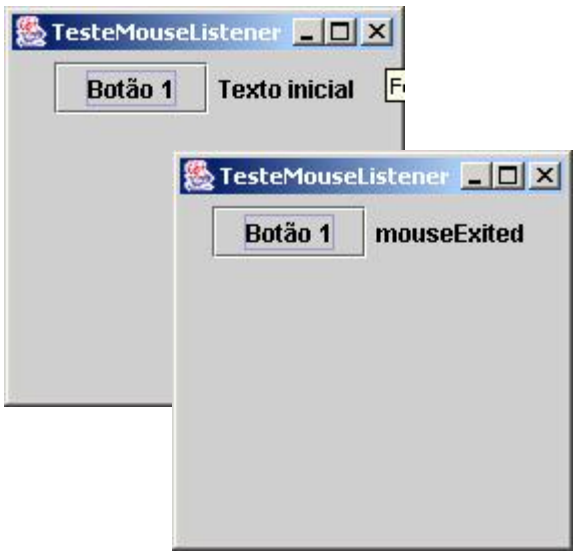
Agora nos métodos como `mouseClicked`, `mouseReleased`, vamos alterar o valor do label `lTexto`, que como foi declarado como atributo da classe.

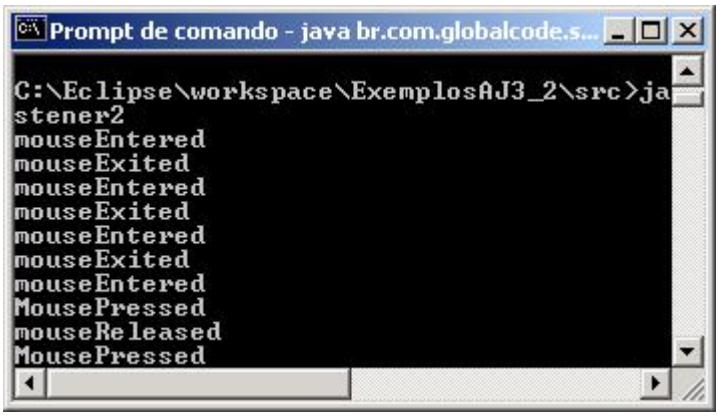
Exemplo: TesteMouseListener2.java alterando componentes do JPanel

```
package br.com.globalcode.swing;

import java.awt.Container;
import java.awt.event.MouseEvent;
import java.awt.event.MouseListener;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;

public class TesteMouseListener2 extends JFrame implements MouseListener {

    private JLabel lTexto = new JLabel("Texto inicial");

    public TesteMouseListener2(String tituloJanela) {

        super(tituloJanela);
        Container c = getContentPane();

        JPanel p1 = new JPanel();

        JButton b1 = new JButton("Botão 1");
        p1.add(b1);
        p1.add(lTexto);
        c.add(p1);

        // this pois a interface esta implementada aqui mesmo
        b1.addMouseListener(this);
        setSize(200, 200);
        show();
    }
```

Anotações

```
    /**************************************************************************
     * Implementamos, na mesma classe, os metodos da interface MouseListener
     *************************************************************************/
    public void mousePressed(MouseEvent e) {
        lTexto.setText("MousePressed ");
        System.out.println("MousePressed ");
    }
    public void mouseReleased(MouseEvent e) {
        lTexto.setText("mouseReleased ");
        System.out.println("mouseReleased");
    }
    public void mouseEntered(MouseEvent e) {
        lTexto.setText("mouseEntered ");
        System.out.println("mouseEntered ");
    }
    public void mouseExited(MouseEvent e) {
        lTexto.setText("mouseExited ");
        System.out.println("mouseExited ");
    }
    public void mouseClicked(MouseEvent e) {
        lTexto.setText("mouseClicked ");
        System.out.println("mouseClicked ");
    }

    public static void main(String args[]) {
        TesteMouseListener2 frame = new TesteMouseListener2("TesteMouseListener");
    }
}
```





Anotações

2.4.4 Implementando o listener como uma classe interna (Inner class)

Outra abordagem bastante comum é a utilização de Inner classes, ou seja, classes declaradas dentro da própria classe.

A classe interna (inner class) tem acesso irrestrito a todos os atributos da classe que a contém, é chamada classe externa (Outer class).

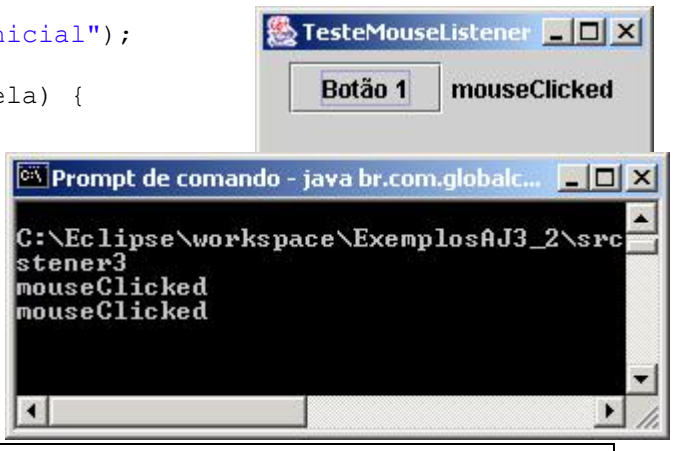
Sintaxe

```
class ExemploInnerClass {
    class InnerClass{

    }
}
```

Agora criaremos uma inner class para a classe `TesteMouseListener` mas, ao invés de implementarmos diretamente a interface `MouseListener`, estenderemos a classe `MouseAdapter` que, por sua vez, implementa a interface `MouseListener`. Desta forma, não precisamos implementar todos os métodos em branco, sobrescrevendo apenas os métodos desejados.

Exemplo: TesteMouseListener3.java

```
package br.com.globalcode.swing;

import java.awt.Container;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;

public class TesteMouseListener3 extends JFrame {

    private JLabel lTexto = new JLabel("Texto inicial");

    public TesteMouseListener3(String tituloJanela) {
        super(tituloJanela);

        Container c = getContentPane();

        JPanel p1 = new JPanel();

        JButton b1 = new JButton("Botão 1");
        p1.add(b1);
        p1.add(lTexto);
        c.add(p1);

        EventosMouse listener = new EventosMouse();
        b1.addMouseListener(listener);
        setSize(200, 200);
```



Anotações

```
        show();
    }

    /*************************************************************************
     * Estendemos MouseAdapter e desta forma nao e necessario implementarmos
     * todos os seus metodos, apenas sobrescrevemos os necessarios.
     ************************************************************************/
    public class EventosMouse extends MouseAdapter {

        public void mouseClicked(MouseEvent e) {
            lTexto.setText("mouseClicked ");
            System.out.println("mouseClicked ");
        }
    }

    public static void main(String args[]) {
        TesteMouseListener3 frame = new TesteMouseListener3("TesteMouseListener");
    }
}
```

2.4.5 Implementando o listener como uma classe anônima

Outra possível estratégia para implementação de listeners é a utilização de classes anônimas, que não possuem um nome e são inteiramente declaradas dentro da passagem de parâmetro do método `addXXXListener`.

Exemplo de Classe Anônima:

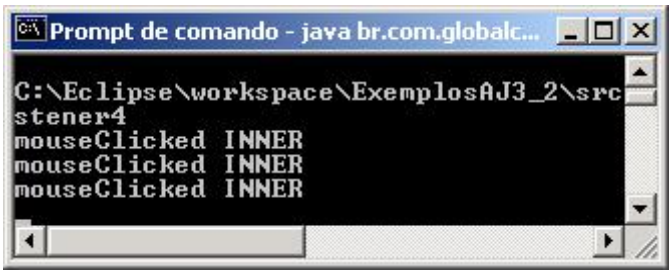
```
    addMouseListener(new MouseAdapter() {
        public void mouseClicked(MouseEvent e) {
            lTexto.setText("mouseClicked ");
            System.out.println("mouseClicked ");
        }
     }
    );
```

Exemplo: TesteMouseListener4.java

```
package br.com.globalcode.swing;

import java.awt.Container;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;

public class TesteMouseListener4 extends JFrame {

    private JLabel lTexto = new JLabel("Texto inicial");

    public TesteMouseListener4(String tituloJanela) {

        super(tituloJanela);
        Container c = getContentPane();
```

Anotações

```
        JPanel p1 = new JPanel();
        JButton b1 = new JButton("Botão 1");

        p1.add(b1);
        p1.add(lTexto);
        c.add(p1);

        b1.addMouseListener(new MouseAdapter() {

            public void mouseClicked(MouseEvent e) {
                lTexto.setText("mouseClicked INNER");
                System.out.println("mouseClicked INNER ");
            }
        });

        setSize(200, 200);
        show();
    }

    public static void main(String args[]) {
        TesteMouseListener4 frame = new TesteMouseListener4("TesteMouseListener");
    }
}
```



Anotações

2.4.6 A interface java.awt.event.ActionListener

Alguns objetos não são sensíveis a ações do mouse através de `MouseListener`s, como é o caso de `JComboBox`. Nestas situações utiliza-se a interface `ActionListener` (mais genérica) e que responde a eventos através de um único método: `actionPerformed`.

A interface `ActionListener` também responde a eventos como ALT+tecla de atalho nos botões que, evidentemente, não são "gerenciados" através do `MouseListener`.

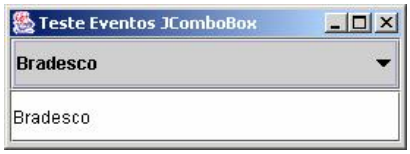
Veja um exemplo de utilização de `ActionListener` a seguir, onde queremos que, ao selecionarmos um banco, seu nome apareça no `JTextField`.

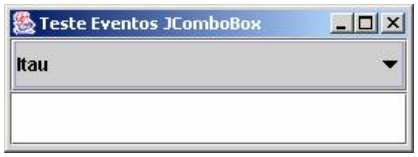
Exemplo: TesteEventosJComboBox.java

```
package br.com.globalcode.swing;

import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.*;

public class TesteEventosJComboBox extends JFrame {

    // o seletorBancos sera manipulado no handler actionPerfomed e por isto e
    // necessario  declara-lo como atributo da classe.
    private JComboBox seletorBancos = new JComboBox();
    private JTextField tfBanco = new JTextField();

    public static void main(String args[]) {
        TesteEventosJComboBox t = new TesteEventosJComboBox();
    }

    public TesteEventosJComboBox() {
        super("Teste Eventos JComboBox");
        setDefaultCloseOperation(EXIT_ON_CLOSE);
        Container c = getContentPane();
        c.setLayout(new GridLayout(2, 1));
        seletorBancos.addItem("Itau");
        seletorBancos.addItem("Bradesco");
        seletorBancos.addItem("Globalcode Bank");
        seletorBancos.addActionListener(new SeletorBancosHandler());
        c.add(seletorBancos);
        c.add(tfBanco);
        setSize(275, 100);
        show();
    }
```

Anotações

```
    class SeletorBancosHandler implements ActionListener {

        public void actionPerformed(ActionEvent e) {
          System.out.println("Item selecionado:"+seletorBancos.getSelectedItem());
          tfBanco.setText(seletorBancos.getSelectedItem().toString());
        }
    }
}
```





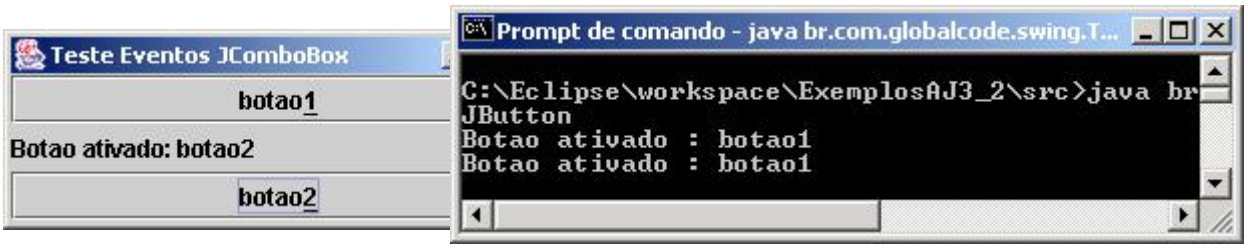
Em seguida, um exemplo de utilização de `ActionListener`, onde o método `actionPerformed` será executado quando um botão for ativado pelo atalho configurado, ou então, selecionando o botão através da utilização da tecla `<TAB>` e pressionado com a tecla `<ESPAÇO>`.

Exemplo: TesteEventosJButton.java

```
package br.com.globalcode.swing;

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class TesteEventosJButton extends JFrame {

    private JButton botao1 = new JButton("botao1");
    private JButton botao2 = new JButton("botao2");
    private JLabel lTexto = new JLabel("ação");

    public TesteEventosJButton() {

        super("Teste Eventos JComboBox");
        setDefaultCloseOperation(EXIT_ON_CLOSE);
        Container c = getContentPane();
        c.setLayout(new GridLayout(3, 1));

        botao1.setMnemonic(KeyEvent.VK_1);
        botao2.setMnemonic(KeyEvent.VK_2);

        botao1.addActionListener(new BotaoHandler());
        botao2.addActionListener(new BotaoHandler());

        c.add(botao1, BorderLayout.NORTH);
        c.add(lTexto, BorderLayout.CENTER);
        c.add(botao2, BorderLayout.SOUTH);
        setSize(275, 100);
        show();
```

Anotações

```
    }

    class BotaoHandler implements ActionListener {

        public void actionPerformed(ActionEvent e) {
            System.out.println("Botao ativado : " + e.getActionCommand());
            lTexto.setText("Botao ativado: " + e.getActionCommand());
        }
    }

    public static void main(String args[]) {
        TesteEventosJButton t = new TesteEventosJButton();
    }
}
```



Anotações

2.5 Componentes avançados

Neste tópico, iremos estudar alguns componentes avançados de Swing bastante interessantes. Contudo, não detalharemos sobre cada um deles focando apenas em mostrar a sua utilização.

2.5.1 A classe javax.swing.JFileChooser

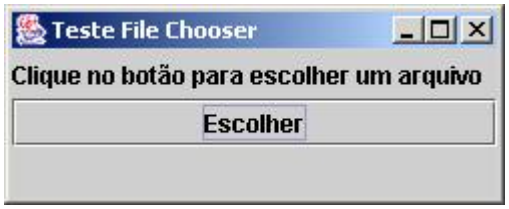
Este componente é utilizado para abrir uma janela de seleção de arquivos. Existem diversas constantes da classe, que devem ser utilizadas para obter as operações realizadas pelo usuário para configurar o componente.

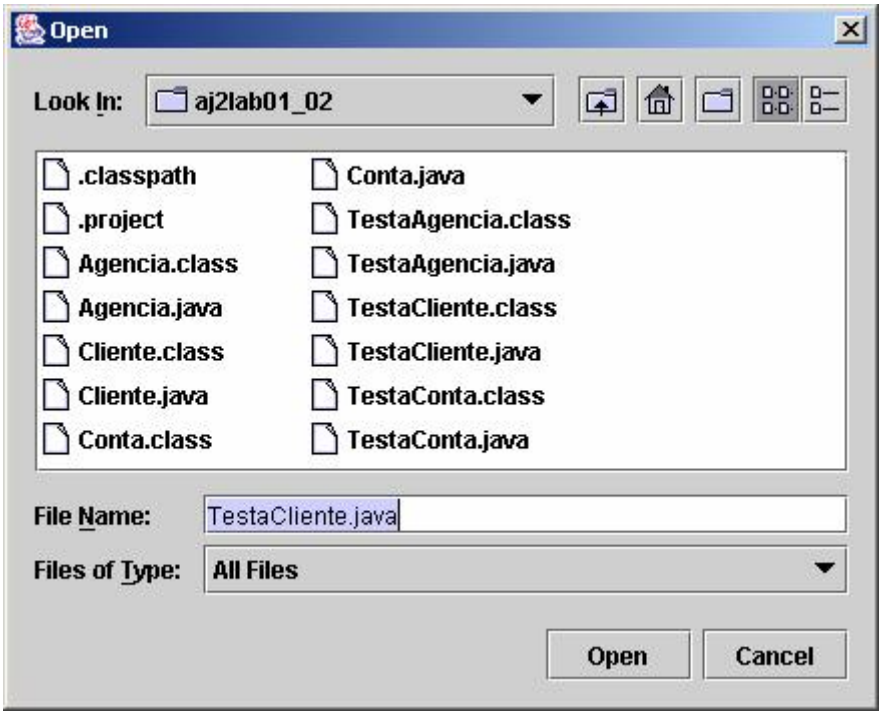
Exemplo: TesteFileChooser.java

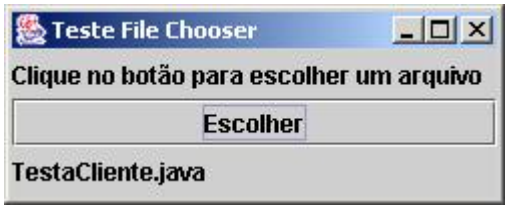
```
package br.com.globalcode.swing.components;

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class TesteFileChooser extends JFrame {

    private JFileChooser fileChooser = new JFileChooser();
    private JButton botaoOk = new JButton("Escolher");
    private JLabel lNomeArquivo = new JLabel("");
    private JFrame eu = this;

    public TesteFileChooser() {
        super("Teste File Chooser");
        setDefaultCloseOperation(EXIT_ON_CLOSE);

        Container c = getContentPane();
        c.setLayout(new GridLayout(3, 1));
        String msg = "Clique no botao para escolher um arquivo";
        JLabel lDigiteNomeArquivo = new JLabel(msg);
        c.add(lDigiteNomeArquivo);
        c.add(botaoOk);
        c.add(lNomeArquivo);

        botaoOk.addActionListener(new AbrirArquivoHandler());
        setSize(250, 100);
        show();
    }

    class AbrirArquivoHandler implements ActionListener {
```

Anotações

```
        public void actionPerformed(ActionEvent e) {
            // Indica que a janela de escolha de arquivo sera aberta
            int opcao = fileChooser.showOpenDialog(eu);
            //    Se a opcao for OPEN
            if (opcao == JFileChooser.APPROVE_OPTION) {
                String arquivoSelecionado=fileChooser.getSelectedFile().getName();
                lNomeArquivo.setText(arquivoSelecionado);
            }
            //    Se a opcao for CANCEL
            else if (opcao == JFileChooser.CANCEL_OPTION) {
                lNomeArquivo.setText("Operação cancelada");
            }
        }
    }
    public static void main(String args[]) {
        TesteFileChooser t = new TesteFileChooser();
    }
}
```







Anotações

2.5.2 javax.swing.JTabbedPane

```
package br.com.globalcode.swing.components;

import java.awt.event.KeyEvent;
import javax.swing.*;

public class TesteTabbedPane extends JFrame {

    public TesteTabbedPane() {
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        JTabbedPane tabbedPane = new JTabbedPane();

        JPanel panel1 = new JPanel();
        panel1.add(new JLabel("Panel1"));
        JPanel panel2 = new JPanel();
        panel2.add(new JLabel("Panel2"));
        JPanel panel3 = new JPanel();
        panel3.add(new JLabel("Panel3"));
        JPanel panel4 = new JPanel();
        panel4.add(new JLabel("Panel4"));

        tabbedPane.addTab("Tab 1", panel1);
        tabbedPane.setMnemonicAt(0, KeyEvent.VK_1);

        tabbedPane.addTab("Tab 2", panel2);
        tabbedPane.setMnemonicAt(1, KeyEvent.VK_2);

        tabbedPane.addTab("Tab 3", panel3);
        tabbedPane.setMnemonicAt(2, KeyEvent.VK_3);

        tabbedPane.addTab("Tab 4", panel4);
        tabbedPane.setMnemonicAt(3, KeyEvent.VK_4);

        this.getContentPane().add(tabbedPane);
        this.setSize(300, 300);
        this.show();
    }
    public static void main(String[] args) {
        TesteTabbedPane tp = new TesteTabbedPane();
    }
}
```

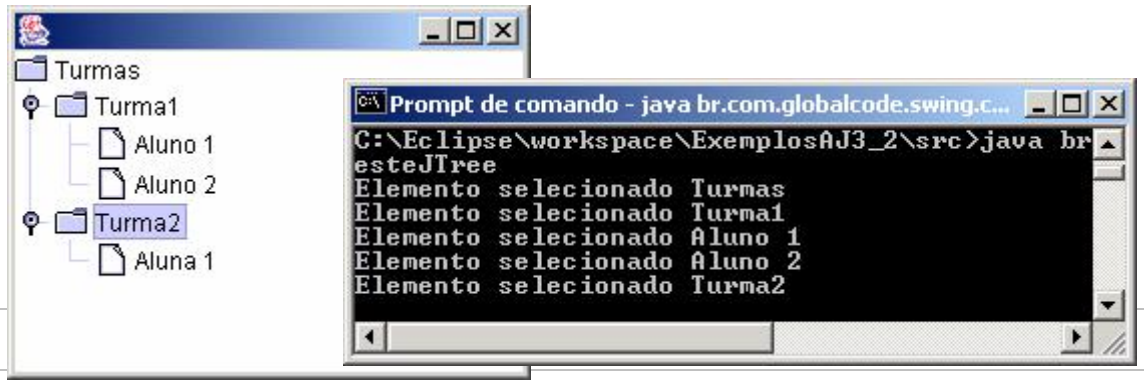
Anotações

2.5.3 javax.swing.JTree

Este componente é utilizado para representar uma estrutura na forma de árvore. Cada nó da árvore, seja um nó ou uma ramificação, é representado por um objeto da classe `DefaultMutableTreeNode`.

Exemplo: br.com.globalcode.swing.components.TesteJTree

```
package br.com.globalcode.swing.components;
import javax.swing.*;
import javax.swing.event.*;
import javax.swing.tree.DefaultMutableTreeNode;

public class TesteJTree extends JFrame {

    private JTree tree = null;

    public TesteJTree() {
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        DefaultMutableTreeNode turmas = new DefaultMutableTreeNode("Turmas");
        tree = new JTree(turmas);
        DefaultMutableTreeNode turma1 = new DefaultMutableTreeNode("Turma1");
        DefaultMutableTreeNode aluno1 = new DefaultMutableTreeNode("Aluno 1");
        DefaultMutableTreeNode aluno2 = new DefaultMutableTreeNode("Aluno 2");
        DefaultMutableTreeNode turma2 = new DefaultMutableTreeNode("Turma2");
        DefaultMutableTreeNode aluna1 = new DefaultMutableTreeNode("Aluna 1");
        turmas.add(turma1);
        turma1.add(aluno1);
        turma1.add(aluno2);
        turmas.add(turma2);
        turma2.add(aluna1);
        tree.addTreeSelectionListener(new TreeHandler());
        getContentPane().add(tree);
        setSize(300, 300);
        show();
    }
    class TreeHandler implements TreeSelectionListener {
        public void valueChanged(TreeSelectionEvent e) {
            DefaultMutableTreeNode node = (DefaultMutableTreeNode) tree
                    .getLastSelectedPathComponent();
            System.out.println("Elemento selecionado " + node.toString());
        }
    }
    public static void main(String[] args) {
        TesteJTree tp = new TesteJTree();
    }
}
```



Anotações

2.5.4 Laboratório

Objetivo:

Atividade	*OK*
Comece o desenvolvimento de uma interface gráfica para uma suposta tela de manutenção de clientes, com os campos ID, CNPJ, Razão Social, Endereço completo e telefones.	
Procure utilizar o FreeLayout para desenvolver esta tela.	
Caso termine em tempo e queira dar inicio ao desenvolvimento de uma tela para futuramente transformar em um plug-in no NetBeans.	

Anotações

3 Desenvolvimento de plug-ins

O desenvolvimento de plug-ins para o NetBeans 5 ficou muito mais simples pois agora contamos com diversos wizards e templates de plug-ins para aumentar sua produtividade.

Você pode imaginar plug-ins para diferentes situações:

- Controle de qualidade de código em tempo real: você coloca suas regras e o plug-in analisa o código on-line para dar dicas de qualidade;
- Controle de horas gastas no projeto: imagine como ficariam os gerentes se soubessem quanto tempo gastamos em cima de cada arquivo. Não dá pra imaginar...
- Caso utilize a plataforma NetBeans como sua infraestrutura de client application, cada plug-in representará um módulo do seu aplicativo, trazendo a vantagem da imposição e visualização da modularidade e depêndencia de módulos do seu sistema. Portanto podemos desenvolver um plug-in que represente um módulo de controle financeiro, outro de controle de telefonemas, folha de pagamento etc.

A área de desenvolvimento de plug-ins para IDEs é muito interessante pois torna o desenvolvedor / empresa capazes de customizarem o IDE alterando e adicionando features importantes para seus padrões de desenvolvimento.

Vamos neste capítulo explorar os recursos de plug-ins embutidos na versão 5 do IDE.

Anotações

3.1 Gerenciando plataformas e plug-ins

Antes de começarmos efetivamente o desenvolvimento de plug-ins é importante conhecermos como e onde vamos instalar e testar nosso plug-in.

Conforme foi apresentado no capítulo de Introdução à NetBeans, sabemos que:

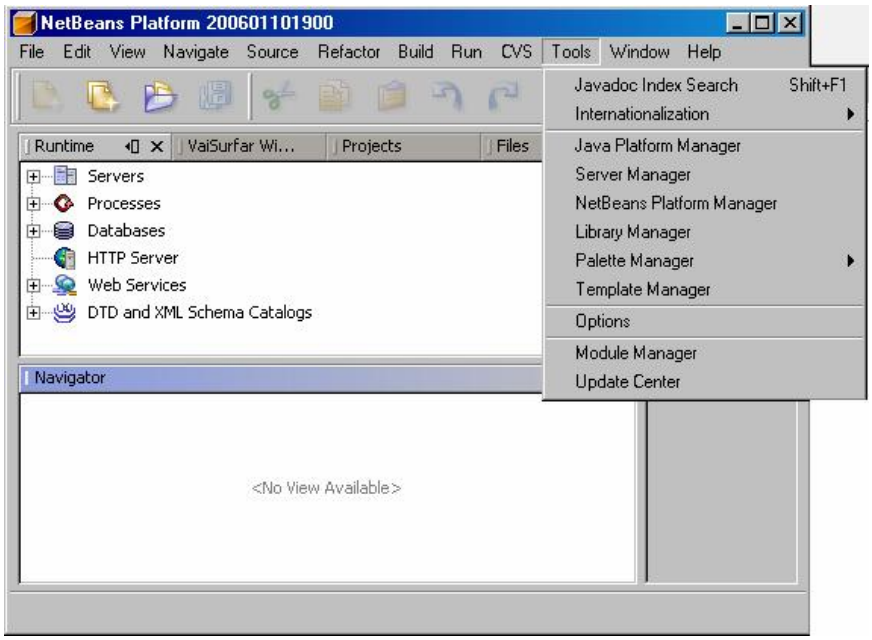
NetBeans Platform: é o núcleo básico que oferece serviços e infraestrutura para aplicativos desktop.
NetBeans IDE: é um aplicativo criado com NetBeans Platform para prover produtividade no desenvolvimento Java, portanto é um IDE Java. Tecnicamente é o NetBeans Platform com diversos plug-ins para suporte a Java Java EE e periféricos.

Você pode trabalhar com diferentes e múltiplas instancias de NetBeans quando desenvolve plug-ins; ou você pode instalar o plug-in diretamente no NetBeans que você está desenvolvendo o plug-in.

Por padrão os plug-ins são instalados e executados em uma segunda instancia de NetBeans que aberta por ele mesmo, isso é bom pois caso seu plug-in cause algum problema no NetBeans, causará em uma segunda instancia e você poderá corrigi-lo através da primeira, em compensação o consumo de máquina é bem maior, pois você passa a manter dois NetBeans na memória no lugar de um só.

Em breve vamos apresentar como instalar o plug-in na mesma instancia ou abrindo uma instancia nova do NetBeans. Por ora, vamos apenas aprender a gerenciar múltiplos tipos de NetBeans e também como ligar / desligar módulos de cada um deles.

3.1.1 Gerenciando plataformas NetBeans



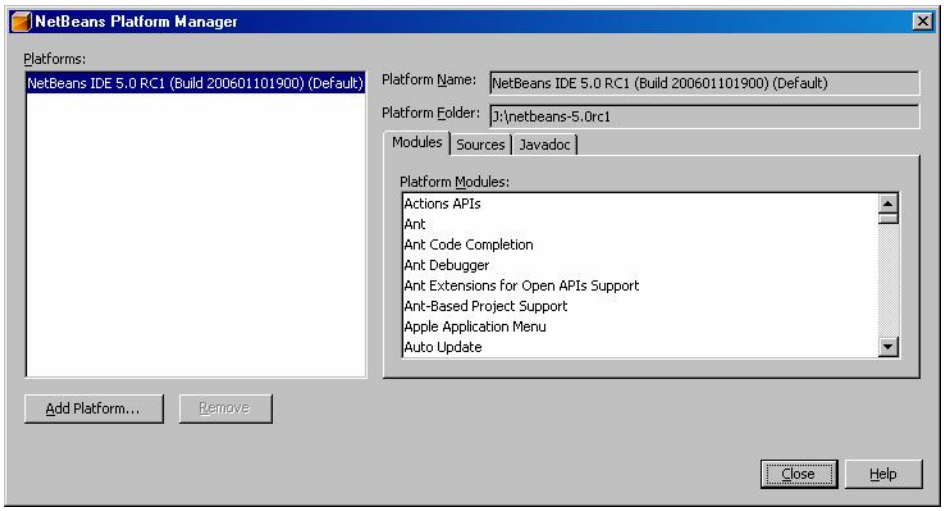
O menu **Tools -> NetBeans Platform Manager**, permite que você configure diferentes NetBeans que você tenha na sua máquina.

Para fazer um teste, vamos duplicar o diretório de instalação do NetBeans para podermos reconfigurar e trabalhar em cima de uma cópia isolada do NetBeans.

Vamos então criar uma cópia do diretório e clicar em **Tools - > NetBeans Platform Manager**.

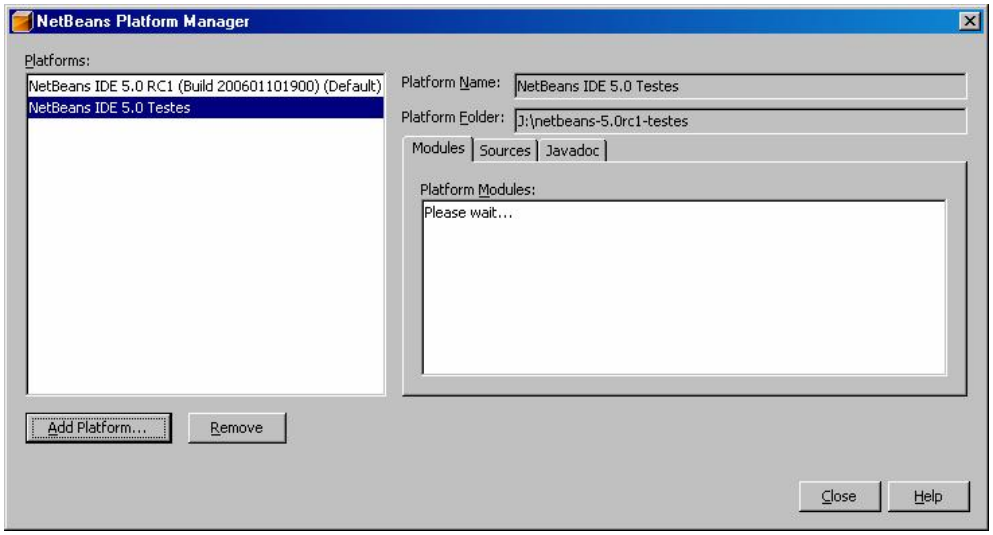
Anotações

O seguinte diálogo será apresentado:



A plataforma que temos é justamento o NetBeans IDE que estamos trabalhando, e podemos verificar os módulos instados. Vamos agora clicar em **Add Platform**.

O próximo passo será indicar o diretório onde você criou a cópia do seu NetBeans e também você deve dar um nome para esta cópia da plataforma NetBeans. Ao concluir você verá duas plataforma configuradas no diálogo **Platform Manager**:

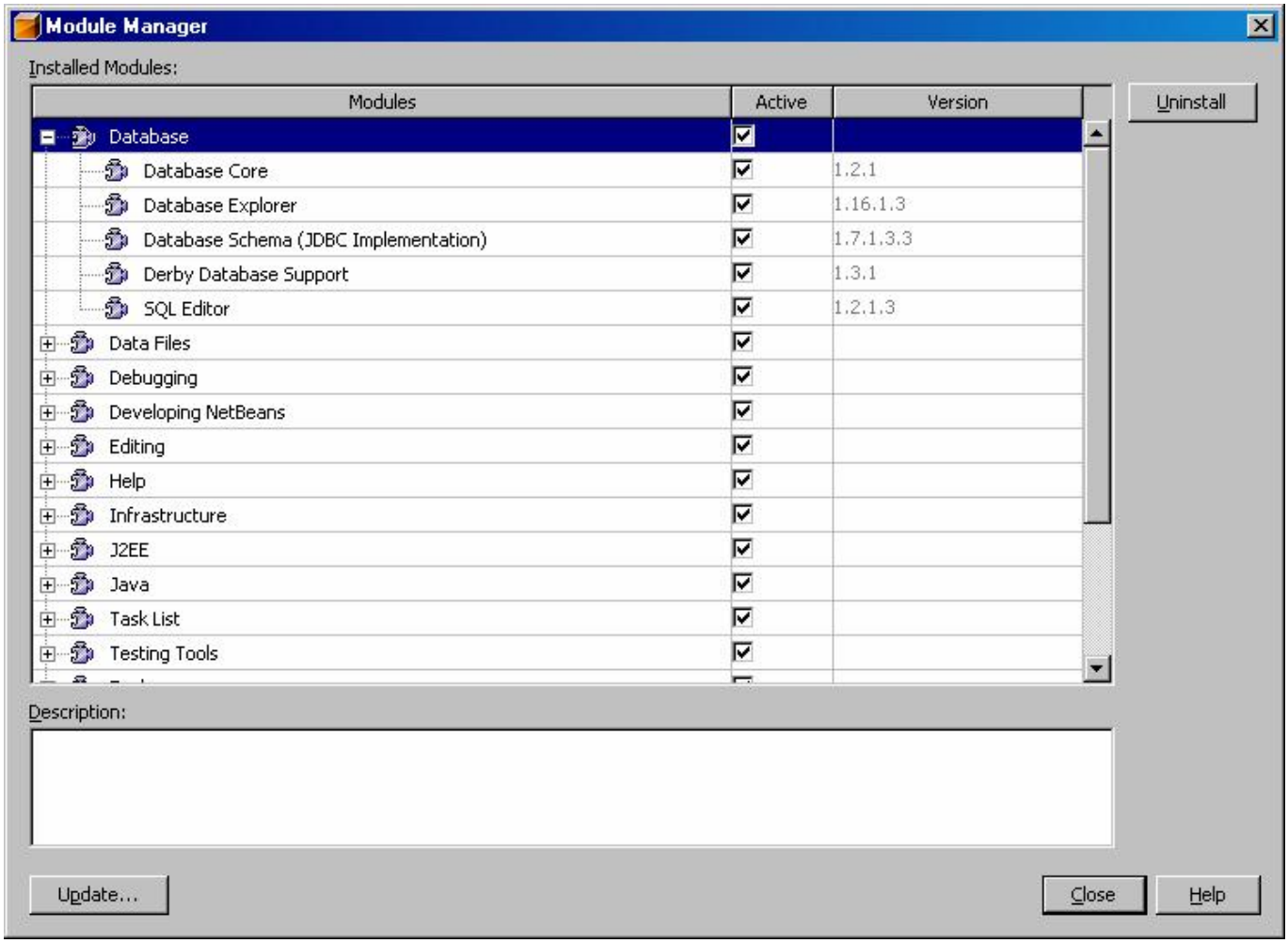


Agora que aprendemos a configurar diferentes plataformas / NetBeans, vamos aprender como instalar / deinstalar, ligar e desligar módulos dentro do NetBeans.

Anotações

3.1.2 Gerenciando plug-ins / módulos das plataformas

No menu **Tools -> Module Manager** você pode gerenciar os plug-ins / módulos do NetBeans. Vejamos o diálogo bastante simples para este propósito:



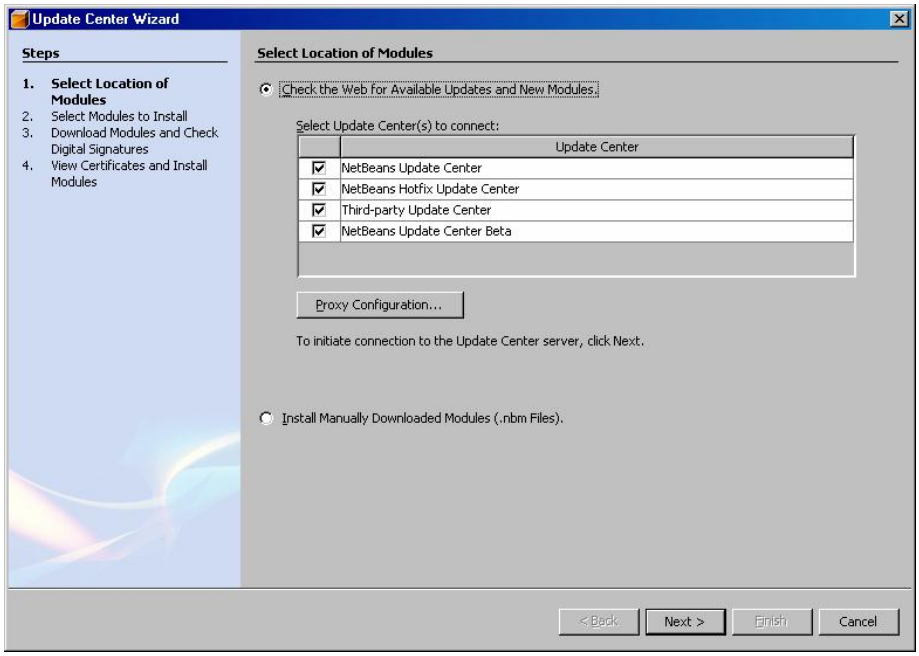
Se você tem problemas de desempenho ou simplesmente não programa Enterprise Edition, você pode desativar os módulos relacionados com J2EE para poupar memória. Portanto, é possível manter diferentes NetBeans, com diferentes módulos instalados para diversos propósitos.

Caso tenha uma máquina performática, provavelmente não terá uma necessidade grande de customizar os módulos.

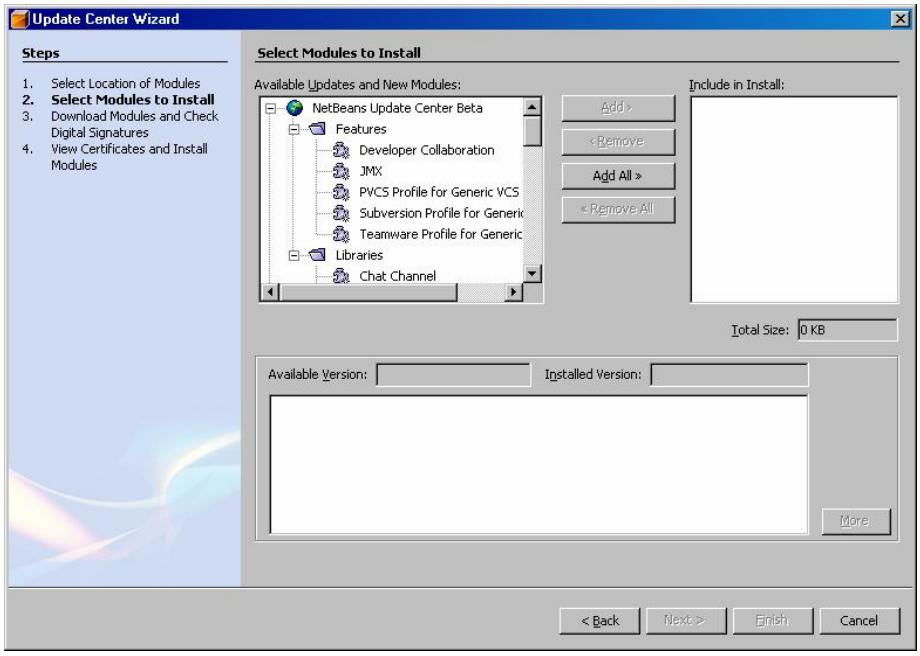
Anotações

3.1.3 Central de Updates NetBeans

O NetBeans conta também com um recurso para atualização, instalação de novas features e também correção d bugs chamado de Update Center. No menu **Tools**, selecione **Update Center**:



O NetBeans vai perguntar onde deseja buscar por atualização ou novos plug-ins, selecione todos as opções. Em seguida vai conectar nos centros e apresentará uma lista de recursos disponíveis para instalação.



Selecione o "Developer Collaboration" para instalar um plug-in interessantíssimo para desenvolvimento em equipes.

Você deve estar de acordo com os contratos de licença que serão apresentados e então o NetBeans iniciará o processo de download e instalação do plug-in.

Esta é uma maneira fácil para você manter seu IDE atualizado e buscar por novas features.

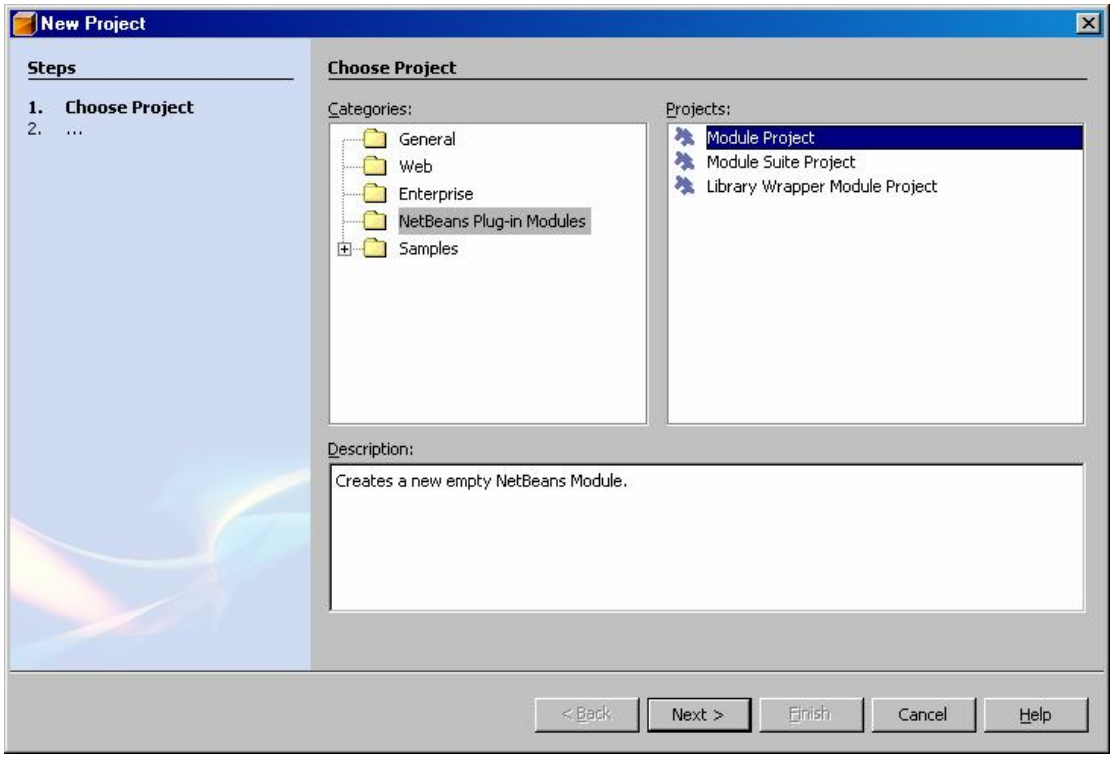
Anotações

3.2 Recursos e conceitos fundamentais para criação de plug-ins

É importante conhecermos alguns conceitos e também terminologias de plug-ins no NetBeans que encontraremos frequentemente nos diálogos e wizards de construção de plug-ins.

O NetBeans gerencia os módulos, menus, janelas, configurações diversas e objetos através de um repositório central chamado de **System FileSystem**. Trata-se de uma coleção de objetos que representam itens de menu, janelas, módulos, ponteiros para arquivos físicos, etc. Ao criar um plug-in, você deverá escrever um arquivo XML para registro do plug-in no FileSystem do NetBeans, este arquivo é chamado layer.xml. Estudaremos maiores detalhes sobre este arquivo quando conveniente.

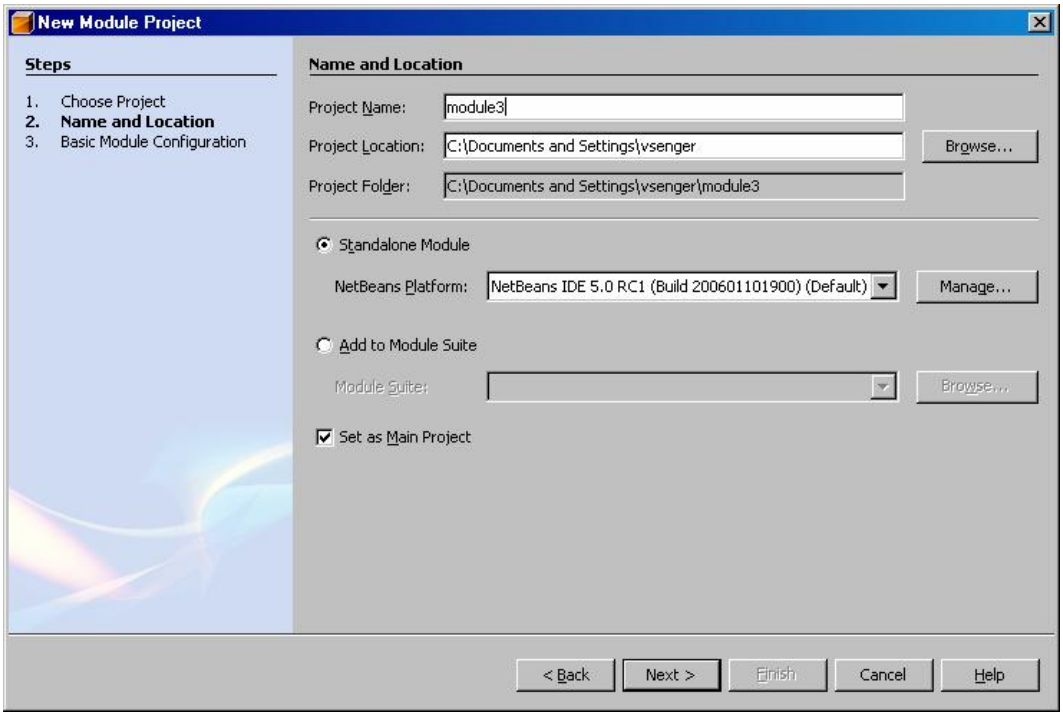
Quando criamos um projeto da categoria plug-ins, o NetBeans apresenta três principais templates:



Module Project: para desenvolver um plug-in individual como janelas, itens de menu, wizard, etc.
Module Suite Project: cria um "guarda-chuva" de plug-ins que funcionam com dependência.
Library Wrapper Module Project: permite que você crie um módulo que representa um ou mais jar files com biblioteca que um ou mais plug-in venha a precisar.

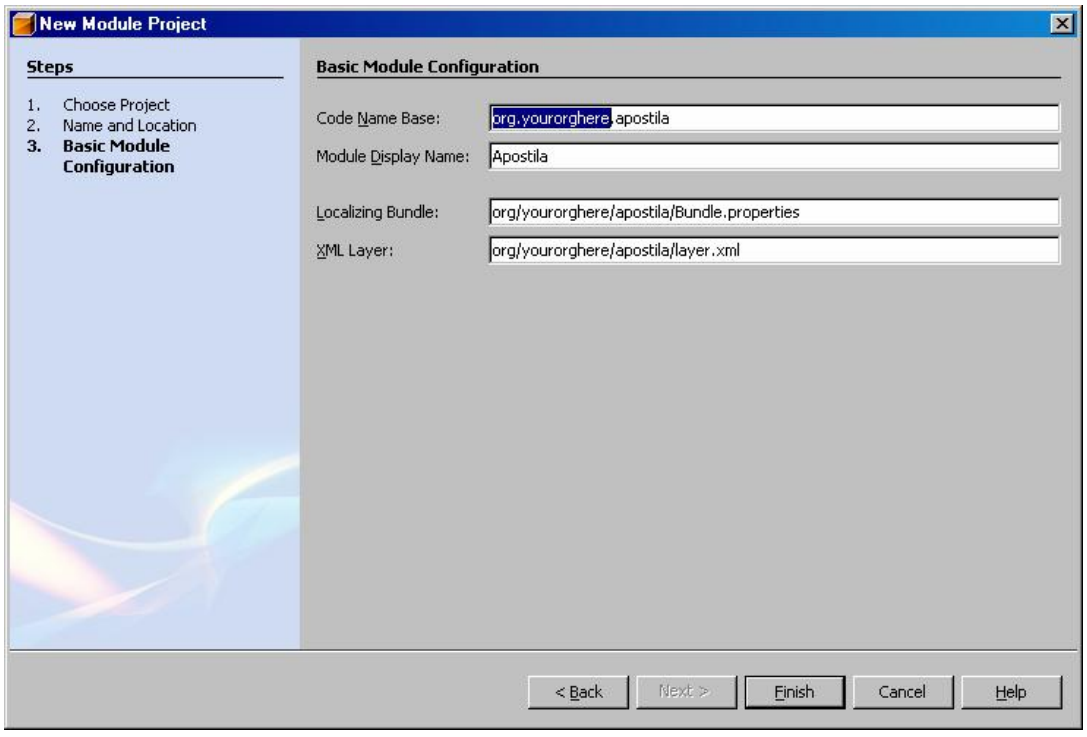
Para começarmos a aprender as terminologias e conceitos básicos, vamos acompanhar os diálogos que são apresentados ao criarmos um "Module Project". Portanto selecione **File -> New Project**, **NetBeans Plug-in Modules -> Module Project**, para visualizar os diálogos que apresentaremos a seguir.

Anotações



Ao criar um plug-in module project o NetBeans vai perguntar o nome do projeto, sua localização e também se é um **Standalone Module** ou se pertence a um **Module Suite**, ou seja, se é um plug-in independente ou um plug-in que fará parte de um suíte de plug-ins.

Sendo um Standalone Module, o NetBeans vai perguntar sobre a plataforma que deseja instalar o plug-in ao testa-lo. Em seguida um diálogo vai perguntar sobre o prefixo e pacote das classes do plug-in:



Além dos nomes de classe e módulo, você deve indicar um arquivo que contenha os textos do plug-in (resource bundle) e também XML chamado de layer que descreve características de registro do plug-in no NetBeans.

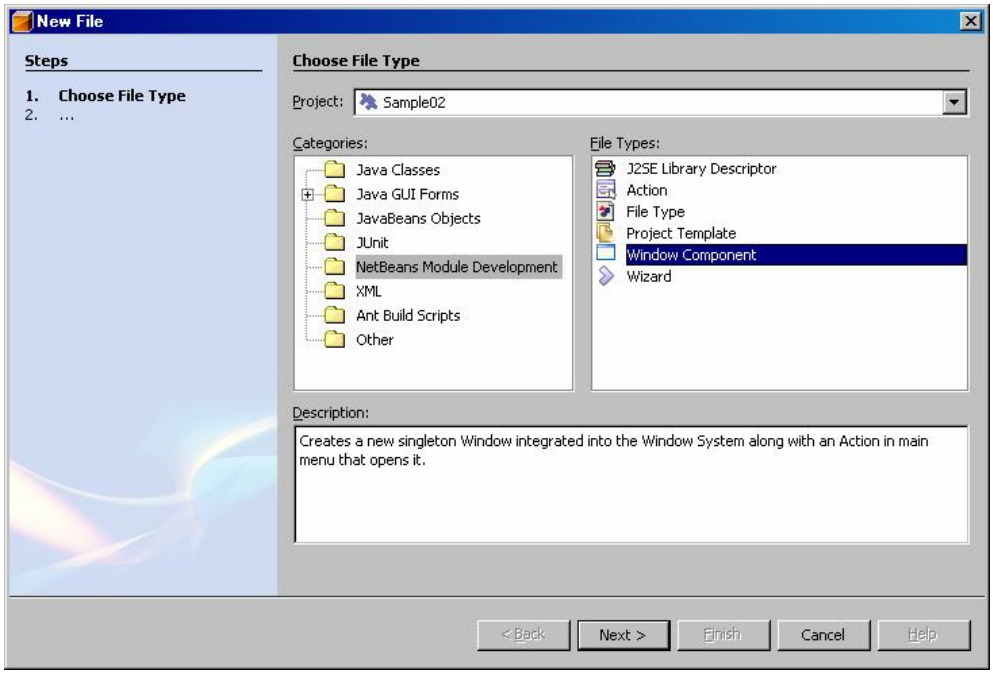
Clicando em finish você terá um projeto pronto para desenvolver seu plug-in.

Anotações

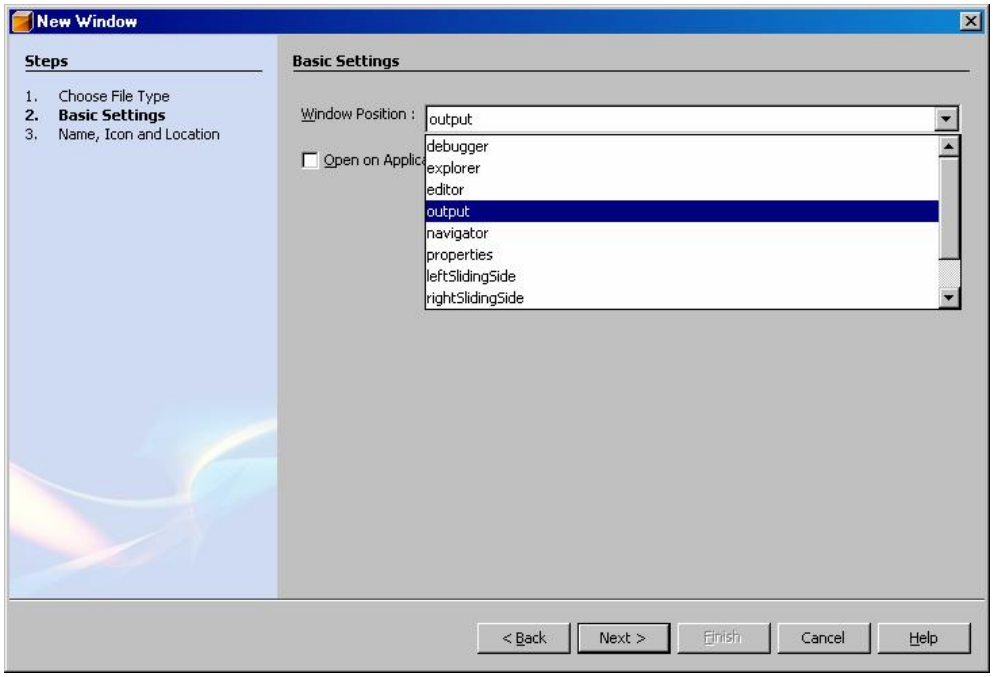
3.3 Plug-in para adicionar uma janela no IDE

Com o projeto para plug-in criado, você pode começar o desenvolvimento de novos elementos para o NetBeans. Vamos neste tópico aprender a criar uma nova janela para o IDE.

Clique em **File -> New File** e escolha **NetBeans Module Development -> Window Component**:

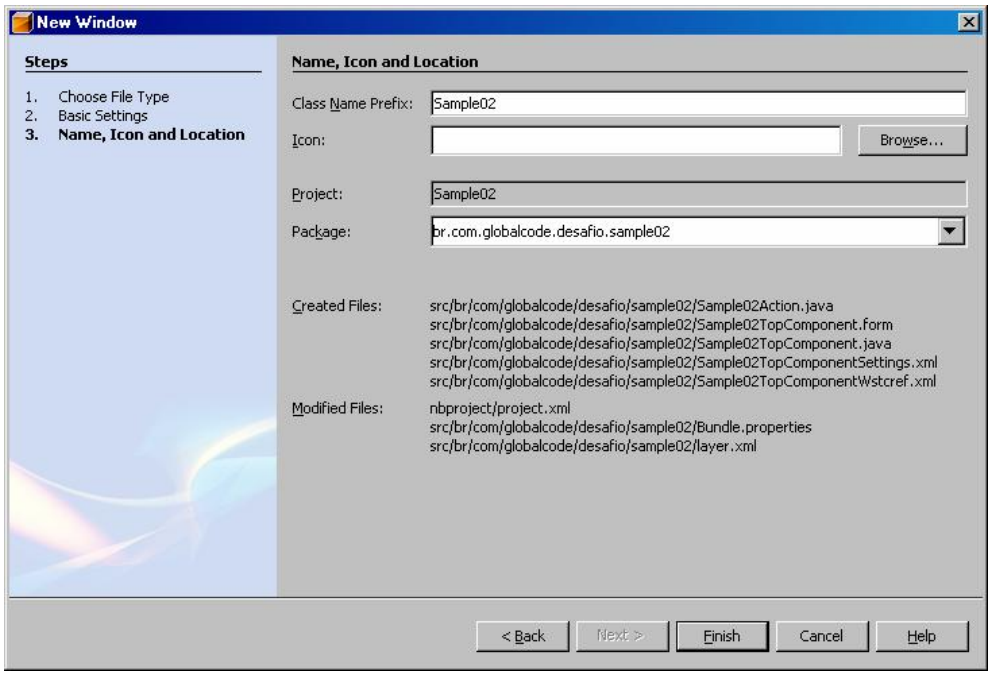


Escolha o grupo que sua janela vai ficar posicionada e também se deseja que a janela seja aberta por default na inicialização do NetBeans:

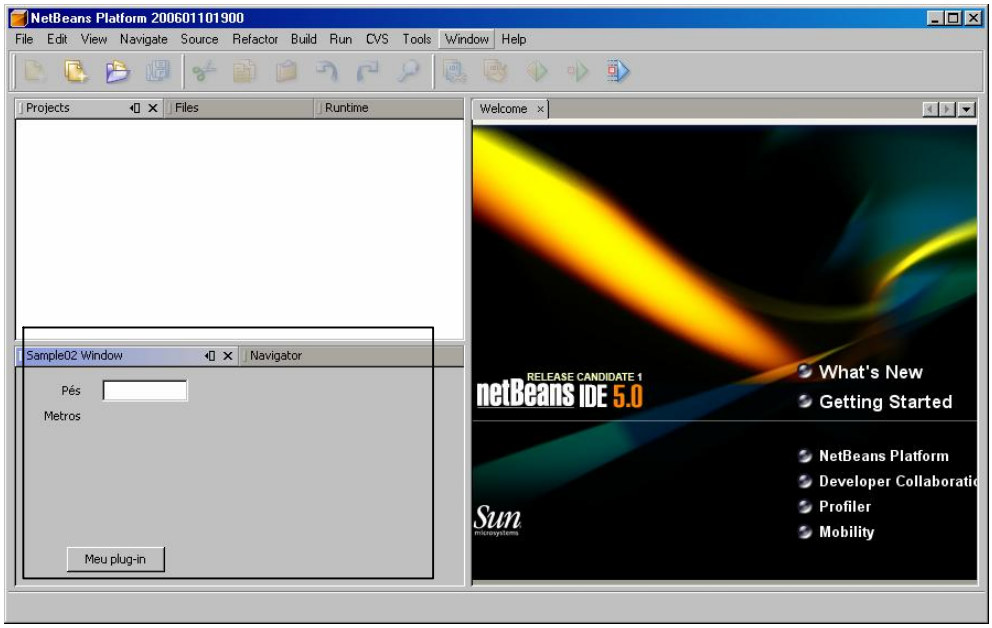


Anotações

Agora coloque o nome do prefixo de classe e também o pacote de classes que d\eseja criar o plug-in e clique em finish:



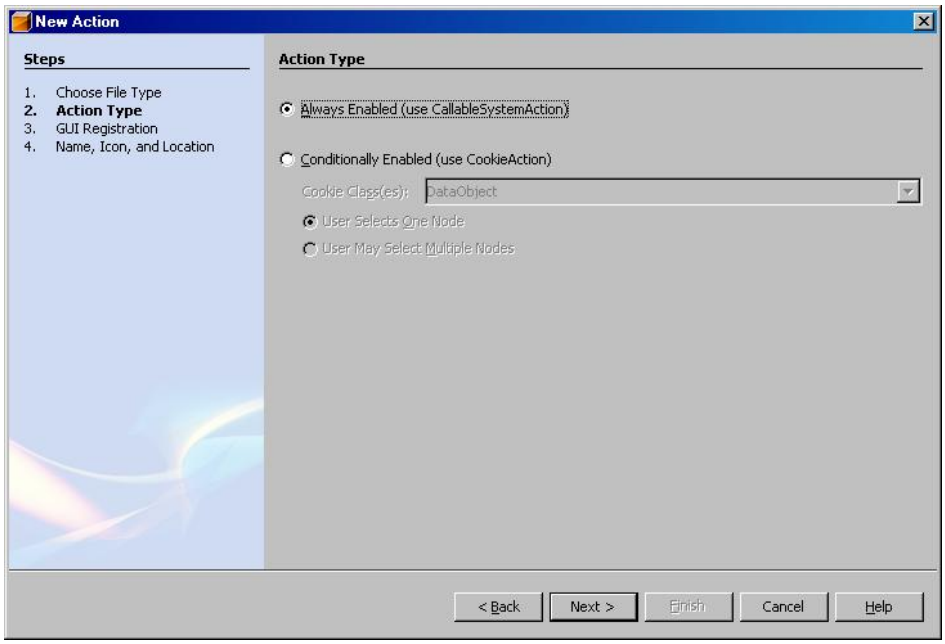
Pronto! Agora você pode começar a desenvolver seu plug-in adicionando componentes no Panel. Clique em Run e o NetBeans vai compilar, empacotar e instalar o plug-in em uma segunda instancia do NetBeans. Neste NetBeans você verá no menu Window o plug-in que você criou:



Anotações

3.4 Plug-in para adicionar um item de menu

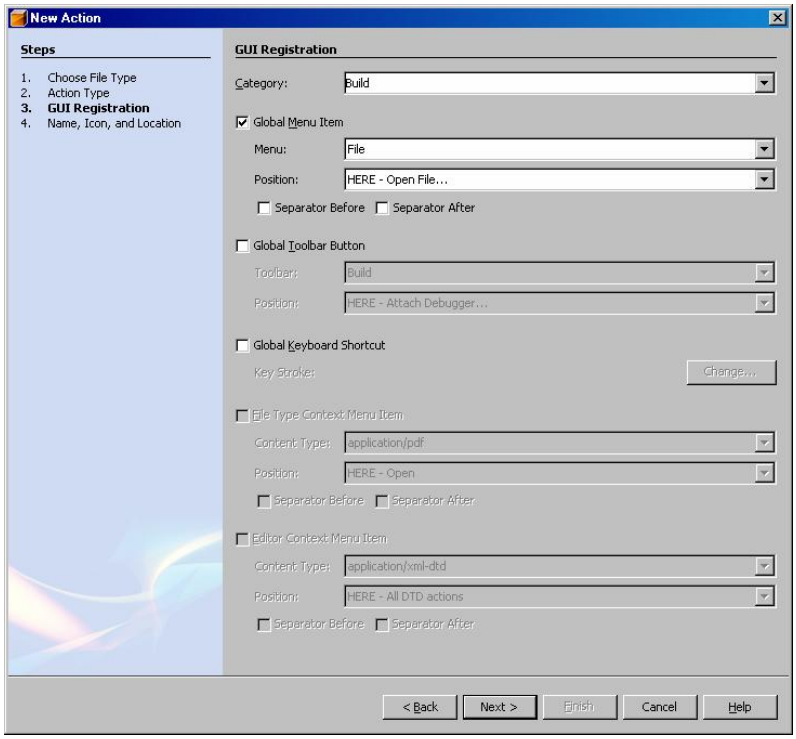
Para adicionar um item no menu, clique em File -> New File e em seguida escolha **NetBeans Module Development -> Action**. O NetBeans permite que você crie um item que fica permanentemente no menu ou então um elemento condicional. Vamos por enquanto utilizar o permanente escolhendo **Always Enabled**:



Escolha neste próximo diálogo o local onde deseja adicionar o item de menu.

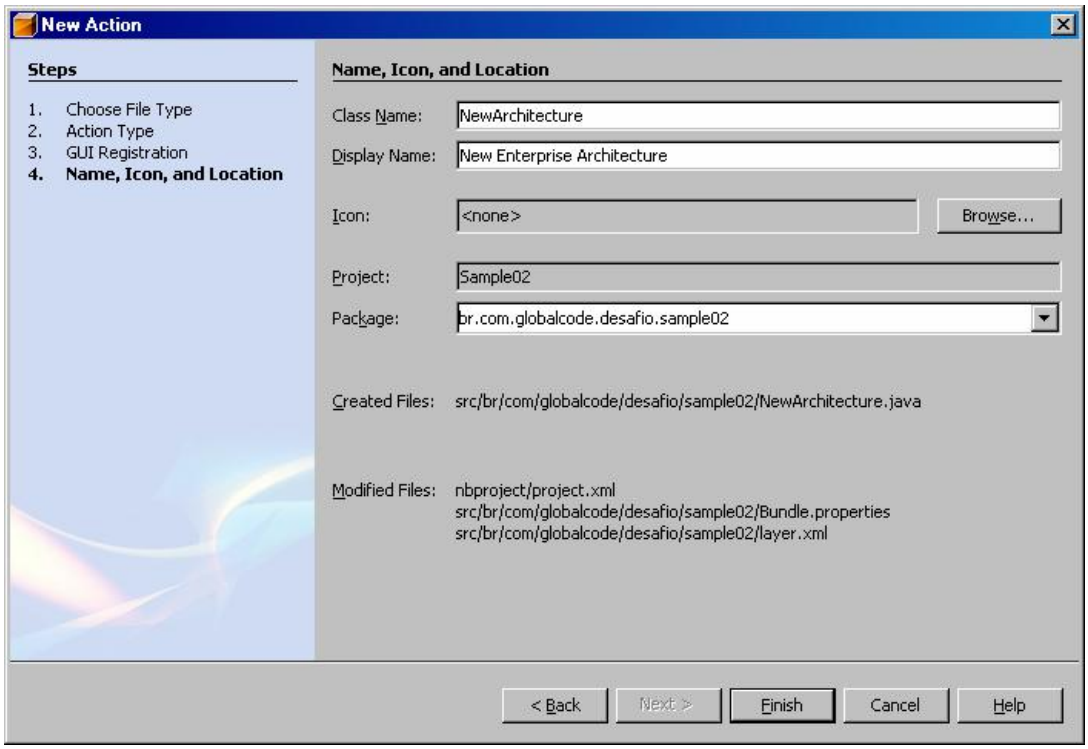
Você pode selecionar também para ter um ícone no toolbar para esta mesma ação.

Essas escolhas refletirão em diferentes configurações no arquivo layer.xml.

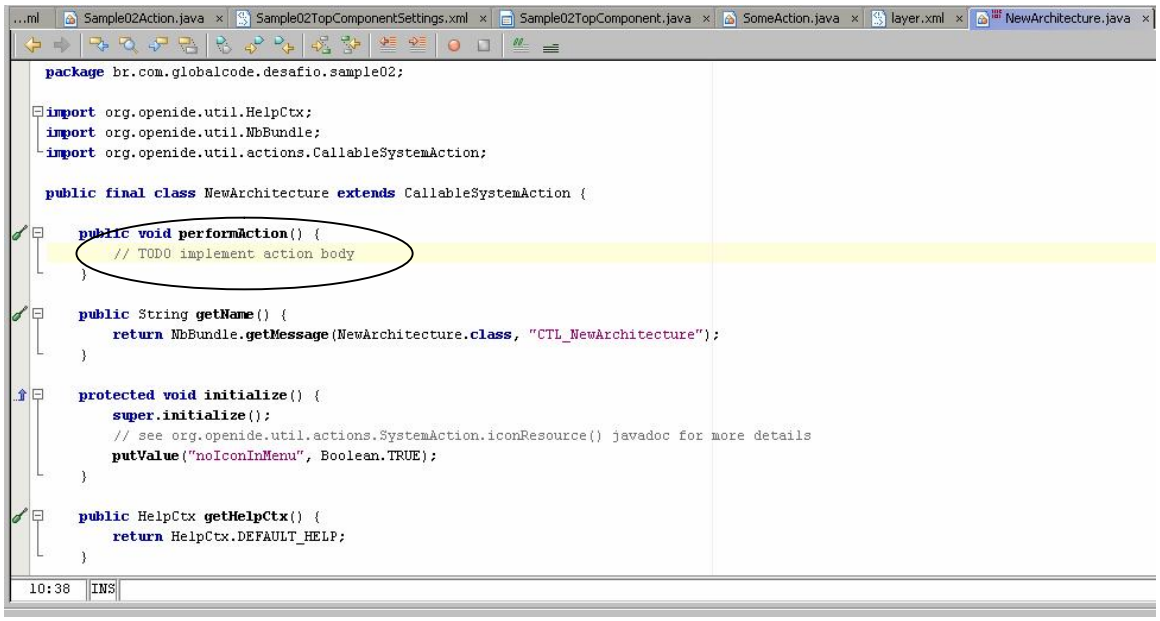


Anotações

Agora entre com o nome da classe que responderá ao clique no seu item do menu e clique em finish.



Agora o NetBeans já vai abrir a classe no local correto onde você deverá programar a resposta à ação:



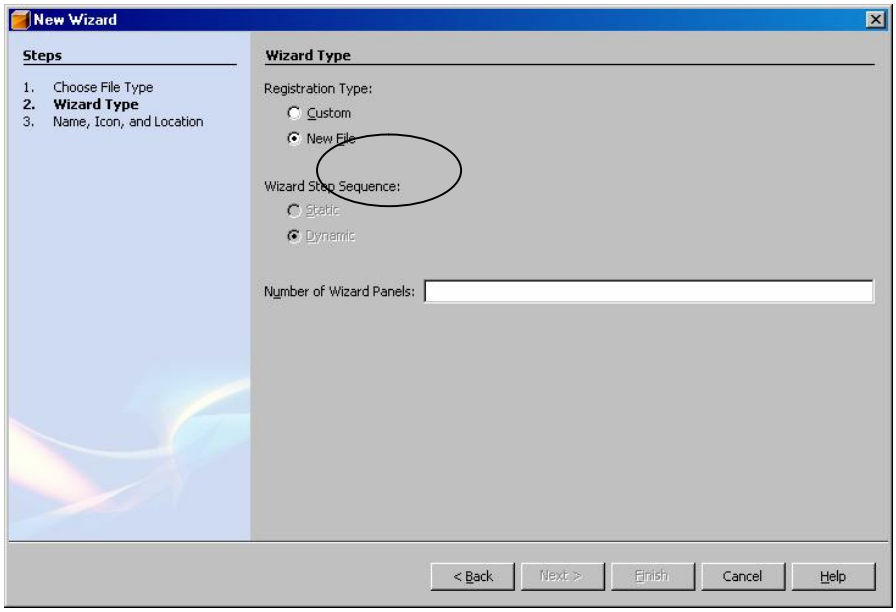
Programe qualquer código, somente para efeito de teste e em seguida clique em Run para testar seu plug-in.

Anotações

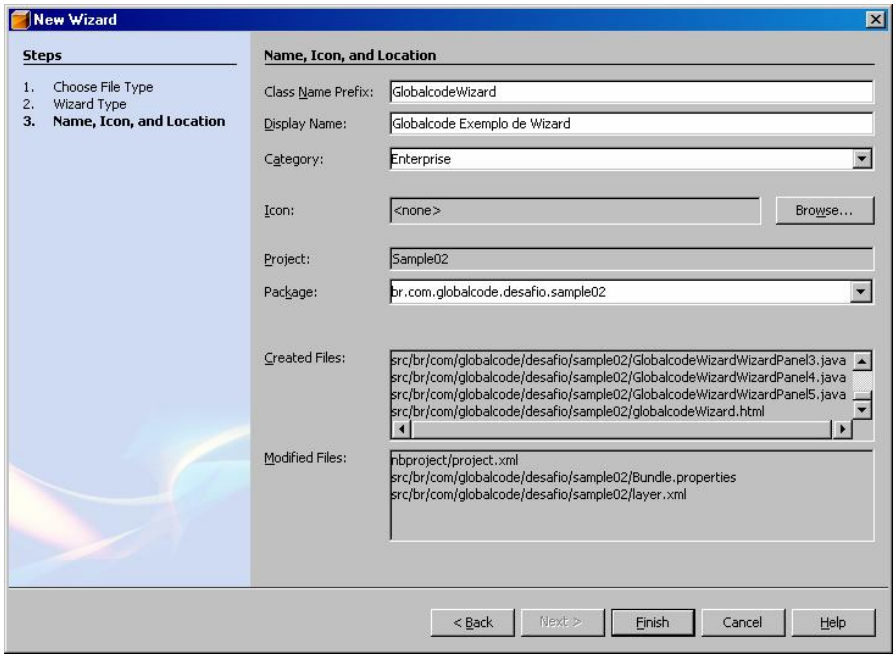
3.5 Criando um Wizard customizado

O NetBeans tem um excelente Wizard para criação de Wizards... Este recurso é excelente caso tenha a idéia / necessidade de um plug-in com um wizard para a criação de um arquivo, manipulação do projeto, entre outros.

Selecione **File -> New File** e em seguida **NetBeans Module Development -> Wizard**. Para simplificar este exemplo, escolha New File e coloque 5 paineis para o Wizard, cada painel representará uma etapa:



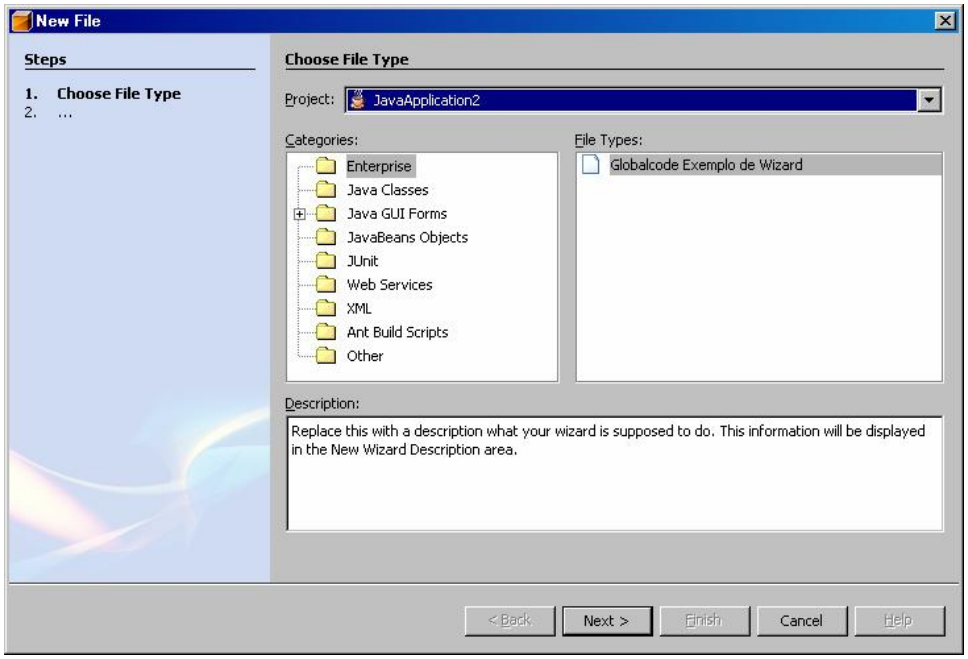
Neste próximo diálogo, coloque o prefixo do nome das classes que serão criadas pelo NetBeans, o nome de apresentação e também a categoria deste wizard e clique em finish:



Anotações

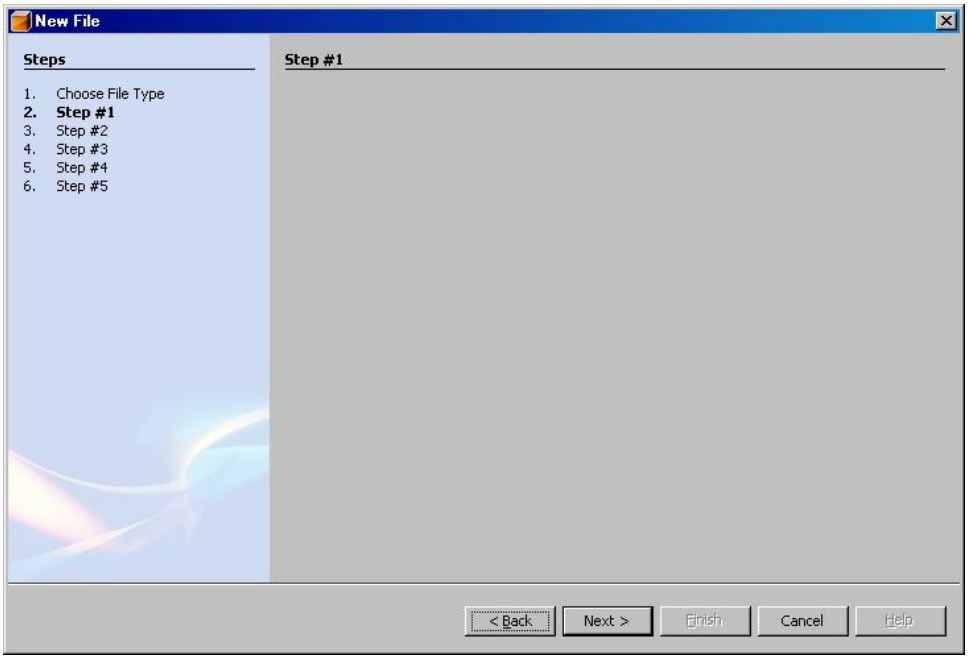
Como resultado o NetBeans vai criar n classes com o prefixo indicado acrescido do nome VisualPanel1. No nosso caso GlobalcodeWizardVisualPanel1, GlobalcodeWizardVisualPanel2, GlobalcodeWizardVisualPanel3 etc. Você pode desenvolver a programação visual que desejar em cada painel; lembramos também que existe um método chamado getName() que deve retornar uma String contendo o nome da etapa.

Desenvolva os painéis do seu plug-in e clique em Run para testa-lo:



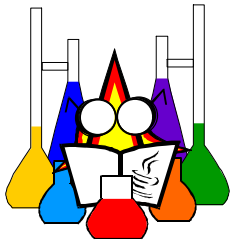
Pronto!

Agora você tem todo o template para poder desenvolver seu próprio wizard com gerenciamento do NetBeans.



Anotações

3.6 Laboratório



Objetivo:

Atividade	*OK*
Desenvolva 3 plug-ins em projetos diferentes, sendo: - Um para criar uma janela customizada; - Um para adicionar um elemento no menu que dispare uma acao qualquer; - Um Wizard;	

Anotações

Anotações