Lista de Bibliotecas para Java

Porque JAVA?

O paradigma de orientação a objetos, em complemento, facilita a correspondência do modelo de classes do capítulo x com a modelagem das classes da aplicação. Por esse conjunto de razões, optou-se pela linguagem Java. Além de ser orientada a objetos, Java vem se tornando uma das promessas para o futuro da computação baseada em Internet. Java é a linguagem de desenvolvimento da Internet, e foi criada pela Sun Microsystems [SUN 96], como:

- É portável, ou seja, o mesmo código objeto gerado pela compilação pode ser executado em diferentes sistemas operacionais sem demandar uma nova compilação.
- Oferece recursos para acesso de repositórios de dados na Internet, uma vez que Java é baseada no protocolo HTTP.
- Possui acesso a banco de dados através de uma API padrão que é independente do sistema de banco de dados usado.
- É orientada a objetos, ou seja, facilita a reutilização de código e aumenta o nível de correspondência entre as classes do modelo e as entidades reais do sistema. Isso facilita muito o desenvolvimento de aplicações complexas.

Java é uma linguagem interpretada, apesar de possuir um código que é uma combinação de instruções compiladas e interpretadas. De fato, apenas 20% do código Java é interpretado pelo browser. A segurança de Java e a sua capacidade de rodar em diferentes plataformas são devidas ao fato de a fase final de compilação ser executada localmente na estação cliente. O programador Java primeiro compila o fonte Java e gera o bytecode, usando para isso um compilador Java (JDK, Visual J++, Symantec Visual Café, Java Workshop, Jbuilder etc).

O bytecode é um código binário neutro e por isso independente de plataforma. Entretanto, o bytecode não é completo até que seja combinado com um ambiente Java de runtime, que geralmente é o browser web. Como cada ambiente Java de execução é necessariamente dependente de plataforma, o bytecode pode ser interpretado e o produto final é capaz de ser executado naquela plataforma.

Um programa escrito em linguagem Java pode ser de dois tipos: um applet ou uma aplicação. Os applets são programas executados a partir de uma página HTML, e por isso podem facilmente ser disponibilizados na Internet. As classes que compõem o applet são armazenadas no servidor www e são transferidas para o computador do usuário no momento do acesso a página HTML.

Uma vez transferido, o applet inicia a sua execução localmente. Nenhum recurso adicional é necessário localmente para a execução de um applet, uma vez que o browser já provê a máquina virtual responsável pela interpretação do código Java. Ao contrário, as aplicações Java não rodam a partir de páginas HTML, mas sim de maneira isolada através de interpretadores Java. Para rodar uma aplicação, o usuário necessita possuir a classe do programa e o interpretador Java armazenados localmente.

As bibliotecas de Java

Assim como qualquer outra linguagem de programação, o estudo de Java limita-se não só ao estudo das suas estruturas de controle e conjunto de palvras reservadas mas também ao estudo das usas bibliotecas de classes. A API da linguagem Java disponibiliza oito bibliotecas de classes para utilização no desenvolvimento de applets e aplicações. As bibiotecas da linguagem Java são chamadas pacotes (*packages*), e são as seguintes:

java.lang: A biblioteca Java de uso geral

A biblioteca java.lang provê as classes que formam o núcleo da linguagem Java e da máquina virtual. Por exemplo, as classes Object, String, e Thread, que são usadas por quase todos os programas Java são definidas neste pacote. Outros exemplos de classes do pacote java.lang são as classes que definem as exceções e erros que a Máquina Virtual de Java é capaz de tratar.

O pacote java.lang provê, além da definição das classes, os métodos que acompanham cada uma dessas classes. Todos os tipos primitivos da linguagem Java e seus métodos são definidos por java.lang, como a classe Integer que provê objetos que contém valores inteiros e o seu método Interger.toString() para representar um valor inteiro em forma de string. O pacote java.lang é importado automaticamente e todos os programas Java, não necessitando que o desenvolvedor faça isso.

java.io: A biblioteca de I/O

A biblioteca java.io fornece um conjunto de classes de filas de entrada e saída de dados usadas para ler e escrever dados em arquivos ou outras fontes de entrada e saída. As filas de entrada e saída de Java são orientadas a bytes, e as classes definidas por esta biblioteca podem ser usadas na implementação de funcionalidades mais sofisticadas de leitura e gravação.

java.util: A biblioteca de utilidades diversas

A biblioteca java.util contém uma coleção de classes de utilidades. Ela também inclui classes que fornecem estruturas de dados genéricas como dicionários, hashtables, pilhas e vetores, assim como classes para manipulação de strings (classe StringTokenizer, para interpretação de tokens), e utilidades para de calendário e datas (classe Date). É nesta classe que são definidas a interface Observer e a classe Observable, que permite que um objeto notifique outro objeto quando ele sofre uma alteração.

java.net: A biblioteca de rede da linguagem Java

A biblioteca java.net contém classes para acesso a rede, que incluem as classes que representam uma URL e uma conexão URL, a classe que implementa uma conexão através de sockets, e a classe que representa um endereço Internet. Através das classes disponíves em java.net, a operação de leitura de um arquivo remoto possui a mesma complexidade da operação de leitura de um arquivo local.

java.awt: A biblioteca de definição de interface gráfica

A biblioteca java.awt (acrônimo para Abstract Window Toolkit) provê os elementos padrão de interface de usuário como botões, listas de seleção, menus e áreas de texto. Ela também inclui containers (componentes da interface que agrupam outros componentes) e componentes de alto nível, como janelas de diálogo para abertura e escrita de arquivos. A AWT contém duas outras bibliotecas: java.awt.image e java.awt.peer.

java.awt.image: A biblioteca AWT para tratamento de imagens

A biblioteca java.awt.image contém classes para processamento de imagens. Estas classes podem ser usadas por programas que necessitam criar ou manipular imagens e cores.

java.awt.peer: A biblioteca AWT para tratamento de

A biblioteca java.awt.peer possui interfaces usadas para mapear os componentes AWT para as suas implementações dependentes do sistema de janelas usado. Essas classes são usadas quando se necessita desenvolver programas Java nos quais os componentes da AWT possuam características específicas de uma plataforma. Um exemplo dito é a tradução dos componentes básicos da AWT para os elementos de interface (widgets) do ambiente operacional Motif.

java.applet: A biblioteca de definição de applets

A biblioteca java.applet contém classes para criação de applets, que são executados através de páginas HTML.
Através da utilização deste conjunto básico de bibliotecas é possível a criação de novas bibliotecas e novos pacotes que desempenham funções mais específicas do sistema que se pretende desenvolver. Um exemplo disto é o que foi feito no desenvolvimento do sistema GDOC, onde as bibliotecas de acesso a rede e acesso a banco de dados foram combinadas e especializadas de uma forma a permitir a implementação de classes de acesso a documentos.

Acesso a Banco de Dados Através de Java: as classes JDBC

JDBC é uma API independente da plataforma de banco de dados que facilita o desenvolvimento de programas Java independentes de banco de dados. JDBC é uma camada abstrata baseada em especificações estabelecidas para acesso a banco de dados. Os drivers JDBC são acessados pelo programa Java através do JDBC Driver Manager, que é a implementação de uma classe Java usada pelos solicitantes do serviço JDBC (programas Java) e pelos provedores de serviço (JDBC Drivers).

Os drivers JDBC são classificados em quatro tipos ou formatos, classificados em relação ao seu método de acesso ao banco de dados. O tipos mais simples de drivers JDBC são representados por uma API genérica que executa chamadas ODBC para estabelecer acesso independente de banco de dados. Outras categorias de drivers usam APIs nativas que executam chamadas de rotinas de bibliotecas específicas fornecidas pelo fabricante do banco de dados.

Já os drivers do terceiros tipo são formados por protocolos parcialmente escritos em Java que traduzem as chamadas JDBC para um protocolo independente do sistema de banco de dados que é posteriormente convertido para o protocolo do banco de dados. Por último, os drivers do formato 4, que foram usados na implementação do acesso a base de dados, utilizam um protocolo nativo totalmente escrito em Java que converte as chamadas de serviço de banco de dados para o protocolo de rede usado pelo sistema de banco de dados usado. Isto permite a comunicação direta entre a estação cliente e o servidor de banco de dados e é a solução mais eficiente para acesso em Intranet. Uma vez que muitos desses protocolos são proprietários, os fabricantes dos bancos de dados são a própria fonte para esse estilo de driver JDBC, e muitos deles já possuem produtos com essas características, como por exemplo o FastForward, para MS SQL Server e Sybase, e o JSQL para Oracle.

Protocolo de Acesso JDBC

As vantagens deste tipo de driver são que ele não necessita de instalações adicionais nos clientes e possui um desempenho superior. Todas as bibliotecas de acesso ao banco de dados necessárias a aplicação são encontradas na forma de classes Java localizadas no servidor da aplicação.

O desempenho do driver, neste caso, é unicamente dependente do desempenho do protocolo de banco de dados escrito em Java pelo fabricante do banco de dados.

Desvantagens do protocolo HTTP

HTTP não é sensível ao formato dos arquivos que transporta, cada arquivo possui o mesmo tratamento. HTTP é um protocolo stateless, ou seja, ele não sabe onde o usuário está no ciclo de processamento. Este controle deve, portanto, ser executado externamente por meio de programas.

O protocolo HTTP é muito simples e foi criado com a intenção de executar tarefas muito básicas. Além disso, ele possui uma ineficiência inerente pelo fato de cada interação resultar em uma nova conexão. Por outrol lado, o protocolo HTTP é largamente usado, principalmente pela popularidade da Internet e dos web browsers. Devido a sua arquitetura, HTTP facilita a centralização de recursos como objetos Java e

documentos HTML.

Os componentes são basedos em padrões abertos (protocolos Internet, Java, SQL). Isto assegura a portabilidade do sistema que é independente de plataforma.

O browser web, neste caso, é considerado como a peça de software essencial da rede de computadores. Assim, é razoável esperar que muitos usuários no futuro serão acquainted com a utilização de web browsers. Isso reduz o tempo para introduzir interfaces de aplicação a novos usuários.

Outras bibliotecas Java

Biblioteca	**Descrição**
BrowserLauncher2	Agiliza a integração entre uma aplicação Java e o *Desktop*, permitindo abrir o *browser* definido no sistema e direccioná-lo para um endereço à escolha.
db4o	Motor de bases de dados para objectos. Permite a criação de aplicações que recorrem a bases de dados de objectos em vez de bases de dados relacionais e pode ser usada como servidor ou como biblioteca incluída na aplicação.
flying-saucer	Usando XML ou XHTML como input, o *flying saucer* permite apresentar o resultado como uma imagem, um componente Swing ou um PDF(através do uso de iText).
HSQLDB	Motor de bases de dados relacionais, 100% Java, leve e simples de usar. Suporta o modo servidor e o modo embutido.
iText	Biblioteca que permite a criação de vários documentos, especialmente em formato PDF.
JCalendar	Componente visual, JavaBean, que permite a criação de um calendário, oferecendo elementos de escolha de dias, meses, anos, horas, etc.
jFreeChart	Biblioteca para a criação de gráficos. Gera gráficos de elevada qualidade e é licenciada sob a licença LGPL.
JExcelApi	API que permite a utilização, escrita, leitura e modificação, de folhas de cálculo criadas pelo MS Excel
JGoodies Forms	*Framework* que facilita a criação de interface gráficas.
JGoodies Looks	*Look-and-Feel* que replica o aspecto do Windows e que pode ser

	usado em qualquer plataforma.
Meval	Biblioteca com classes para avaliação de expressões matemáticas.
Rhino	Implementação de um motor de JavaScript, totalmente escrito em Java, e que pode ser embutido em aplicações, dotando as mesmas de capacidades de *scripting*.