Desenvolvimento Desktop

Aula 15 - Arquivos JAR

Apresentação

Nesta aula, abordaremos um assunto muito importante em Java. Trata-se do empacotamento de todos os arquivos de uma aplicação em um único arquivo com a extensão **JAR** ou **Java ARchive. JAR** é um arquivo compactado usado para distribuir um conjunto de classes Java. É usado para armazenar classes compiladas e metadados associados que podem constituir um programa. Arquivos **.jar** podem ser criados e extraídos usando o utilitário **jar** da JDK.

Vídeo 01 - Apresentação

Objetivos

Ao final desta aula, você será capaz de:

- Descrever um arquivo **.jar**.
- Empacotar todos os arquivos de sua aplicação em apenas um arquivo **.jar**.
- Criar um arquivo **jar** no Prompt de Comando e extrair um arquivo qualquer de um pacote **.jar**.
- Criar um arquivo **.jar** no NetBeans.
- Executar um arquivo **.jar** no Prompt de Comando e no Windows Explorer.
- Entendendo os arquivos **.JAR**

Entendendo os Arquivos .JAR

Um arquivo **.jar** possui um arquivo manifesto localizado no caminho META-INF/MANIFEST.MF. As entradas do arquivo manifesto determinam como o arquivo **.jar** será usado. Arquivos **.jar** que têm a intenção de serem executáveis terão uma de suas classes especificadas, como a classe principal, e assim eles poderão ser executados como qualquer outro arquivo executável.

Para criar um arquivo **.jar**, basta que você tenha instalado o **JDK** e que o seu path (caminho) esteja corretamente configurado. Se não tiver, o **JDK** basta acessar o site da Oracle e fazer o download apropriado e instalá-lo. Depois de instalar o **JDK**, configure o path. Caso você não se lembre mais como criar um path para o **JDK**, aqui vai um pequeno resumo:

No Windows/Xp/7, clique em **Meu computador** com o botão direito, escolha **Propriedades** (no Windows 7, selecione 'Configurações avançadas do Sistema' em seguida), selecione a aba **Avançado** e clique em **Variáveis de ambiente**. Aí você poderá definir variáveis que serão usadas por todos os usuários ou apenas por você. Escolha uma das opções e clique em **Nova**. Em **Nome da Variável**, preencha como **JAVA_HOME** e, em **Valor da variável**, coloque o diretório de instalação do **JDK** (ex: 'C:\Java\jdk1.7.0', sem as aspas). Agora, selecione a variável **path**, se existir, clique em **editar** e adicione ao final dessa variável o caminho do **JDK**. Por exemplo:

C:\Program Files (x86)\Java\jdk1.6.0_20\bin

Se a variável não existir, ou se você não tiver permissão para alterá-la, basta criar uma variável na parte destinada apenas ao usuário com o nome **PATH** e informar o diretório do **JDK**.

Após confirmar, pode ser necessário reiniciar seu computador. Para verificar se a instalação funcionou, abra o prompt de comando (acesse o menu Iniciar, digite 'cmd' e pressione ENTER). Então, digite javac e pressione ENTER. Se não aparecer nenhuma mensagem de erro, apenas uma lista de comandos, a instalação e a configuração estão corretas.

Vídeo 02 - Arquivos Jar

Criando um Arquivo Jar no Prompt de Comando

Para criar um arquivo **.jar**, primeiramente você deve acessar a janela do prompt de comando através do caminho (Windows 7):

Iniciar > Todos os programas > Acessórios > Prompt de Comando

E fazer o seguinte:

- Acesse o diretório onde se encontra os seus arquivos **.class**.
- Em seguida, digite o seguinte comando no console:

>jar cvf meuPrograma.jar Arquivo1.class Arquivo2.class...

O arquivo **meuPrograma** é o nome que você quer dar ao seu pacote. Escolha um nome compatível com a sua aplicação, de forma que você possa se lembrar do que se trata no futuro (de preferência o próprio nome do seu projeto). Substitua **Arquivo1** e **Arquivo2** pelos nomes de seus arquivos com extensão **class**.

O que esse comando faz é criar um arquivo chamado **meuPrograma.jar** contendo os arquivos **Arquivo1.class** e **Arquivos2.class**. Os parâmetros passados - **cvf** - significam o seguinte:

-c = crie um novo arquivo;
-v = gere um output sobre o que está acontecendo;
-f = especifique o nome do arquivo.

Pode-se também usar um comando mais "requintado" para inserir um arquivo **Manisfest**, determinado por você, em seu **.jar**. Isso porque o comando informado anteriormente cria um **Manifest.mf** automaticamente sem configurações mais

avançadas. Um arquivo **manifest** determina o funcionamento do seu **.jar**. Ele é apenas um documento de texto que pode indicar a classe principal da sua aplicação ou outros arquivos necessários para que a sua aplicação funcione corretamente. Um exemplo de arquivo **Manifest** poderia ser da seguinte forma:

Manifest-Version: 1.0
Main-Class: pacote1.pacote11.ClassePrincipal
Created-By: Ant 1.4.1
Class-Path: lib/outrojar.jar

A última linha do **Manifest** deve estar em branco. O arquivo **manifest** é geralmente criado com o nome de **MANIFEST.MF** e, por default, fica num diretório chamado **META-INF** dentro do **.jar**. O comando para gerar um **.jar** com um **manifest** especificado por você é:

>jar cvfm meuPrograma.jar mymanifest classes...

Onde:

- **classes...** – são as suas classes que devem constar do pacote **.jar** .

Vídeo 03 - Arquivos Jar: Criando

Descompactando um Arquivo Jar

Os arquivos **.jar** usam um padrão de compressão baseado no formato de compressão zip, de modo que qualquer aplicativo que suporte esse formato poderá abrir e manipular um **.jar**. Entretanto, você poderá realizar essas operações usando linhas de comando também. Por exemplo, para extrair os arquivos de um **.jar** basta digitar no prompt de comando:

>jar xvf arquivo.jar

Se você quiser extrair um determinado arquivo de um pacote **jar**, use o seguinte comando:

>jar xvf arquivoAExtrair arquivo.jar

Onde:

- **arquivoAExtrair** – é o arquivo que você pretende extrair.
- **arquivo.jar** – é o pacote no qual se encontra o arquivo desejado.

Executando um Jar

Ao contrário do que possa se pensar de imediato, arquivos **.jar** são executados através do comando **java** e não do comando **jar**. Se o **manifest** estiver corretamente definido para executar o arquivo, ou seja, se ele contém uma indicação de qual é a classe principal da aplicação, basta digitar o seguinte comando para executar o **.jar**:

>java –jar arquivo.jar

Conheça outros Parâmetros do Comando Jar

Caso você queira se aprofundar mais na criação de arquivos dessa natureza, aqui vão mais alguns recursos:

- **c** - Cria um novo arquivo.
- **t** - Lista o conteúdo de um arquivo **jar**.
- **x file** – Extrai todos os arquivos ou apenas os especificados. Se o argumento **file** for omitido, todos os arquivos serão extraídos, caso contrário os arquivo(s) especificado(s) serão extraído(s).
- **v** – Ativa a saída para o comando que está sendo executado.
- **f** – Argumento que indica qual arquivo **jar** está sendo manipulado. Num comando de criação, indica o nome do arquivo que está sendo criado e, no caso de listar ou extrair, indica qual arquivo será listado ou extraído.
- **m** – Indica o arquivo **manifest** que será inserido no **.jar**.

- **O** – Indica que se deve apenas armazenar os arquivos sem compressão.
- **M** – Para não criar um **manifest** default.
- **u** – Atualiza um arquivo **jar**. Ex.: **jar uf arquivo.jar Arquivo.class**.

Vídeo 04 - Arquivos Jar: Outros Comandos

Usando Arquivos Jar de Terceiros

Há duas maneiras de se usar um **JAR** distribuído por alguém. Isso depende para que o **.jar** serve. Alguns podem ser aplicações prontas e para usá-los será necessário apenas executar o **.jar**, como explicado anteriormente. Caso o **.jar** seja um framework, ou um driver JDBC ou qualquer outro tipo de **lib** (biblioteca) que você possa usar para desenvolver uma aplicação, a maneira correta de usar seria colocar o **.jar** no seu **classpath**. Para isso, basta adicionar o caminho completo do arquivo **jar** ao seu **classpath**. Entretanto, se você estiver usando uma IDE, algumas delas (aliás, a maioria) ignoram o seu **classpath** de maneira que você terá que adicionar o arquivo **.jar** manualmente ao conjunto de **libs** reconhecidos pela sua IDE, pois para cada IDE há uma maneira diferente de se fazer isso.

Você pode também ir por um caminho mais simples e apenas colocar o **.jar** que deseja usar no seguinte diretório:

JAVA_HOME\jre\lib\ext

Assim o próprio **JDK** irá carregá-lo sempre que for necessário e todas as aplicações irão reconhecê-lo.

Atividade 01

1. De acordo com a afirmação a seguir, marque a alternativa correta.

   Arquivos com a extensão .jar só agrupam arquivos com as extensões .java e .class.

   a. Verdadeiro

   b. Falso

2. Para se criar um arquivo .jar não é necessária a instalação do JDK, já que se trata de um aplicativo independente do Java. Essa afirmação é verdadeira ou falsa? Justifique sua resposta.

3. Que arquivo do pacote .jar indica a classe principal e os arquivos necessários para o funcionamento correto de uma aplicação?

   a. Javac.exe

   b. Java.exe

   c. Manifest.mf

   d. Javaw.exe

4. Qual dos comandos a seguir é o correto para se criar um arquivo .jar?

   a. jar cvf meuPrograma.jar Produtos.java Clientes.java

   b. jar cvf meuPrograma.jar Produtos.class;Clientes.class

   c. jar cvf meuPrograma.jar Produtos.class Clientes.class

Como Criar Arquivos Jar no NetBeans

Agora que você já sabe como criar arquivos **.jar** no prompt de comando, veremos, nesta seção, como criar arquivos **.jar** utilizando o NetBeans. Para isso:

1. Execute o NetBeans e abra o projeto que você pretende empacotar.

2. No menu principal, selecione **Executar > Limpar e construir projeto principal**, ou utilize o atalho **Shift + F11**.
3. Feito isso, será criado um diretório chamado **dist** contendo um arquivo **.jar** com todos os arquivos do seu projeto e um arquivo **txt** com as informações necessárias para você executá-lo.

Para executar seu projeto a partir do prompt de comando:

1. Abra a janela do prompt de comando.
2. Dirija-se ao diretório **dist** e execute o seguinte comando:

java -jar "SeuProjeto.jar"

Para executar seu projeto a partir do Windows Explorer:

Dirija-se ao diretório **dist** e dê um duplo clique no arquivo **jar**.

Se você quiser saber quais os arquivos que estão empacotados em um arquivo **jar** qualquer, basta descompactá-lo da mesma forma que você descompacta um arquivo **zip** ou **rar**.

Atividade 02

---

1. Os arquivos com a extensão .jar uma vez compactados só poderão ser executados a partir do prompt de comando:
    a. Verdadeiro
    b. Falso
2. Arquivos com a extensão .jar não podem mais ser descompactados. Essa afirmação é verdadeira ou falsa? Justifique sua resposta.
3. Um projeto empacotado no NetBeans só poderá ser executado no NetBeans. Essa afirmação é verdadeira ou falsa? Justifique sua resposta.

Resumo

Nesta aula, vimos duas formas de como empacotar os arquivos de um projeto em apenas um arquivo **.jar**. A partir do prompt de comando e através no NetBeans. Você viu que esse tipo de arquivo é muito utilizado para distribuir as suas aplicações de uma forma prática. Você aprendeu também que esse tipo de arquivo pode ser descompactado da mesma forma que descompactamos qualquer arquivo **zip** ou **rar**. Além disso, você ainda pode executar a sua aplicação diretamente tanto no prompt de comando quanto no Windows Explorer, sem ter que utilizar o NetBeans. Esperamos que você tenha aprendido muito com este curso e desejamos muito sucesso na sua carreira.

Autoavaliação

1. Qual o comando que devemos utilizar para extrairmos apenas um arquivo de um pacote **jar** a partir do prompt de comando?
2. Que comando devemos utilizar para executarmos um arquivo **jar** a partir do prompt de comando?

Referências

Links:

Disponível em: <http://pt.wikipedia.org/wiki/Java_Archive>. Acesso em: 31 jul. 2012.

Disponível em: <http://javafree.uol.com.br/wiki/jar>. Acesso em: 31 jul. 2012.

Disponível em: <http://www.plugmasters.com.br/sys/materias/793/1/Criando-arquivos-.jar>. Acesso em: 31 jul. 2012.

Livros:

Java – Use a Cabeça – Autores: Kathy Sierra & Bert Bates, Editora: Alta Books, 2ª Edição, Rio de Janeiro, 470 págs.