Java Web Start

Leonardo Marcelino

Neste tutorial apresentaremos as vantagens e facilidades de usar a ferramenta de Deployment Java Web Start.

Introdução

Java Web Start é uma nova tecnologia para deployment de aplicações Java. Ele nos permite executar aplicações com um simples clique em uma página Web. O usuário pode baixar e executar aplicações sem passar por procedimentos complicados de instalação. Se aplicação não estiver presente na máquina do cliente, o Java Web Start irá automaticamente baixar todos os arquivos necessários para execução da aplicação. Se existir alguma versão da aplicação na máquina do cliente, está aplicação estará sempre pronta para ser executada a qualquer hora que o usuário necessitar, através de um ícone em seu desktop ou através de um link em um Web Browser. Independente do modo que aplicação for invocada, uma versão mais recente da aplicação sempre estará presente para o usuário.

Java Web Start utiliza a tecnologia Java Network Lauching Protocol & API (JNLP). A tecnologia JNLP define, além de outras coisas, um formato padrão de arquivo (arquivo JNLP) que descreve como a aplicação será executada.

Java Web Start suporta:

- Versionamento e atualização incremental;
- Operações Off-line;
- Integração com Desktop;
- Sandboxing (Ambiente protegido com restrições de acesso a disco ou recurso de rede);
- Code-signing;
- Instalação automática de JRE e pacotes adicionais da aplicação.

Mais detalhes

Mais informações sobre Java Web Start, consulte o site da Sun Microsystem.
http://java.sun.com/products/javawebstart/

Como Funciona o Java Web Start?

O Java Web Start utiliza um arquivo *JNLP* que descreve como a aplicação será executada. Através de uma página Web com um link para este arquivo JNLP, o usuário com um simples clique ativa o Java Web Start.

O Java Web Start irá então verificar se todos os recursos necessários para execução da aplicação estão disponíveis na máquina do cliente, caso seja a primeira vez que aplicação esteja sendo executada, o Java Web Start detectou que nenhuma versão da aplicação ou até mesmo o Java Runtime Environment (JRE) estão presentes na máquina do cliente, neste caso ele irá primeiro fazer download do JRE e depois o download da aplicação Java. Caso já exista uma versão da aplicação instalada no cliente e o Java Web Start identifique que houve alguma alteração da aplicação no servidor, imediatamente inicia o processo de atualização da versão mais recente da aplicação. Após todo processo de verificação e atualização a aplicação é inicializada e o usuário já pode utiliza-la.

Criando uma aplicação para utilizar o Java Web Start

Vamos criar uma simples aplicação Hello World e disponibilizá-la em um servidor para poder mostrar na prática o uso do Java Web Start.

```
package br.com.guj.tutorial.jws;

import java.awt.BorderLayout;
import java.awt.Dimension;
import java.awt.Font;
```

```
import java.awt.Toolkit;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;

import javax.swing.ImageIcon;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;

public class HelloJWS extends JFrame {

    private JLabel lblGreeting;

    private ImageIcon imgLogo;

    private JButton btnClose;

    public HelloJWS() {
        super("Hello JWS");

        initComponents();

        this.addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent event) {
                exit();
            }
        });

        this.setSize(new Dimension(300, 200));

        centralize();

        this.setVisible(true);
    }

    private void initComponents() {

        this.lblGreeting = new JLabel("Hello Java Web Start", JLabel.CENTER);
        this.lblGreeting.setFont(new Font("Arial", Font.BOLD, 18));

        this.imgLogo = new ImageIcon(this.getClass().getClassLoader().getResource("guj.gif"));

        this.btnClose = new JButton("Fechar");
        this.btnClose.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                exit();
            }
        });

        JPanel pnlGretting = new JPanel();
        pnlGretting.add(this.lblGreeting);

        JPanel pnlButton = new JPanel();
        pnlButton.add(this.btnClose);

        this.getContentPane().setLayout(new BorderLayout());
        this.getContentPane().add(BorderLayout.NORTH, pnlGretting);
        this.getContentPane().add(BorderLayout.CENTER, new JLabel(this.imgLogo));
        this.getContentPane().add(BorderLayout.SOUTH, pnlButton);
    }

    private void centralize() {
        Dimension screenSize = Toolkit.getDefaultToolkit().getScreenSize();
        Dimension windowSize = this.getSize();

        int x = (int)((screenSize.getWidth()  - windowSize.getWidth()) / 2);
        int y = (int)((screenSize.getHeight() - windowSize.getHeight()) / 2);

        this.setLocation(x, y);
    }

    private void exit() {
        System.exit(0);
    }

    public static void main(String args[]) {
```

```
        new HelloJWS();
    }
}
```

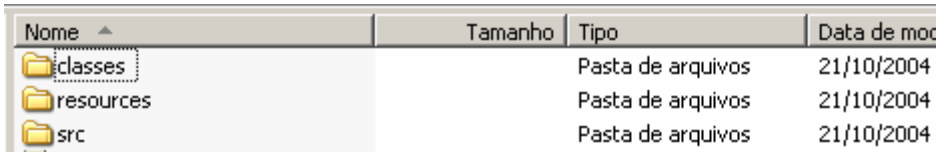
O código acima nada mais faz do que exibir uma janela com uma mensagem de saudação, o logo do GUJ e um botão fechar.

Tela da aplicação HelloJWS

Empacotando nossa aplicação

Após a compilação da nossa aplicação, iremos agora empacotá-la em um arquivo *JAR*. Esse arquivo irá conter todos a classes, imagens e outros arquivos utilizados por nossa aplicação.

Supondo que nossa estrutura de diretório seja:



Para gerar o arquivo *JAR* da nossa aplicação iremos utilizar seguinte comando:

```
jar -cvf TutorialJWS.jar -C classes . -C resorces .
```

```
added manifest
adding: br/(in = 0) (out= 0)(stored 0%)
adding: br/com/(in = 0) (out= 0)(stored 0%)
adding: br/com/guj/(in = 0) (out= 0)(stored 0%)
adding: br/com/guj/tutorial/(in = 0) (out= 0)(stored 0%)
adding: br/com/guj/tutorial/jws/(in = 0) (out= 0)(stored 0%)
adding: br/com/guj/tutorial/jws/HelloJWS$2.class(in = 724) (out= 407)(deflated 4
3%)
adding: br/com/guj/tutorial/jws/HelloJWS.class(in = 3546) (out= 1915)(deflated 4
5%)
adding: br/com/guj/tutorial/jws/HelloJWS$1.class(in = 701) (out= 387)(deflated 4
4%)
adding: guj.gif(in = 2358) (out= 2363)(deflated 0%)
adding: Thumbs.db(in = 6144) (out= 3212)(deflated 47%)
```

Mais detalhes

Para saber mais sobre como gerar arquivos JAR de uma olhada no tutorial no nosso amigo Samuel Mota. http://www.guj.com.br/java.artigo.42.1.guj

Preparando ambiente no servidor Web

Criaremos agora nosso ambiente no servidor Web que irá disponibilizar nossa aplicação. Em nosso exemplo utilizaremos como servidor Web, o Jakarta Tomcat versão 5.0.19, mas vale lembrar que o Java Web Start funciona em qualquer servidor Web.

Para o Java Web Start funcionar devemos:

- no arquivo *web.xml* que se encontra dentro da pasta *conf* do nosso servidor, verificar a existência do *MIME-TYPE, JNLP*, caso o servidor não possua esse *MIME-TYPE*, devemos adicionar as seguintes linhas:

```
<mime-mapping>
    <extension>jnlp</extension>
    <mime-type>application/x-java-jnlp-file</mime-type>
</mime-mapping>
```

- no arquivo *server.xml* que também se encontra dentro da pasta *conf* do nosso servidor, devemos adicionar um novo contexto para nossa aplicação. Inclua entre a tag *Host*, as seguintes linhas:

```
<Context path="/tutorialjws" docBase="tutorialjws" debug="0" reloadable="true"/>
```

- devemos criar também dentro da pasta *webapps* do nosso servidor a pasta da nossa aplicação, que neste caso irá se chamar *tutorialjws*. Após criar a pasta, podemos copiar o arquivo *JAR* gerando anteriormente para dentro deste pasta.

Criando o protocolo JNLP

O Java Network Lauching Protocol (JNLP) descreve como nossa aplicação será executada. Ele nada mais é do que um arquivo XML com a extensão *.jnlp*. Apesar de ser um arquivo XML é imprescindível que a extensão seja *.jnlp*.

Abaixo o arquivo JNLP da nossa aplicação:

```
<?xml version="1.0" encoding="utf-8"?>
<jnlp
  spec="1.0+"
  codebase="http://localhost:8080/tutorialjws"
  href="TutorialJWS.jnlp">

    <information>
        <title>Tutorial Java Web Start</title>
        <vendor>GUJ – Grupo de Usuarios Java</vendor>
        <homepage href="http://localhost:8080/tutorialjws/index.html"/>
        <description>Tutorial Java Web Start</description>
        <description kind="short">Tutorial JWS</description>
        <icon href="images/logo.jpg"/>
        <offline-allowed/>
    </information>

    <resources>
        <j2se version="1.3+" href="http://java.sun.com/products/autodl/j2se"/>

        <jar href="TutorialJWS.jar"/>

        <property name="myProperty" value="Isso é um exemplo de propriedade"/>
    </resources>

    <application-desc main-class="br.com.guj.tutorial.jws.HelloJWS"/>
</jnlp>
```

O arquivo JNLP acima possui as tags necessárias para o funcionamento do Java Web Start, portanto iremos ver mais detalhadamente todas as *tags* que um arquivo JNLP pode conter.

O Elemento JNLP

- **atributo spec:** Por padrão o valor é “1.0+”, podendo ser versões 1.0 ou superiores do *release* que está trabalhando.

- **atributo codbase:** URL do servidor Web onde a aplicação deverá ser iniciada.

- **atributo href:** URL que aponta a localização do próprio arquivo JNLP. O Java Web Start necessita desse atributo para identificar as características de execução de um aplicativo.

O Elemento Information

- **elemento title:** Nome da aplicação.

- **elemento vendor:** Nome da empresa que desenvolveu a aplicação.

- **elemento home page:** Contém um simples atributo, *href,* que aponta para URL da página que inicia a aplicação. Esse elemento é usado pelo *Application Manager* para permitir ao usuário ir até a pagina da aplicação para obter mais informações.

- **elemento description:** Uma pequena descrição da aplicação. O elemento *description* é opcional. O atributo *kind* é utilizado para definir como a descrição deve ser usada. Ele pode ter um dos seguintes valores: *one-line, short e tooltip.*

- **elemento icon:** Contém o atributo *href*, onde informamos uma URL que aponta para uma imagem correspondente ao ícone da aplicação. O atributo opcional *kind="splash",* pode ser usado para informar que uma imagem será usada na *splash screen* na inicialização da aplicação.

- **elemento offline-allowed:** Esse elemento é opcional e indica se a aplicação pode ser executada offline.

Se esse elemento não for informado, a aplicação só poderá ser executada online, fazendo com que o Java Web Start sempre verifique a versão da aplicação para saber se deve ou não atualizá-la e depois executá-la. Agora se esse elemento for informado, o Java Web Start irá tentar verificar se existe uma atualização nova disponível, entretanto se já existir uma versão da aplicação na máquina do cliente, esse verificação irá terminar em poucos segundos e a aplicação existente no cliente será executada.

Elemento Security

Toda aplicação é por padrão executada em um ambiente restrito, similar ao Applet Sandbox. O elemento *security* pode ser utilizado para solicitar acesso irrestrito.

Se o elemento *all-permissions* for especificado, a aplicação terá acesso total a máquina do cliente e a rede, porém se uma aplicação solicitar acesso total, todos os *JARS* deverão ser assinados. Para mais informações sobre assinatura de arquivos *JAR* acessa a página:
http://java.sun.com/developer/Books/javaprogramming/JAR/sign/signing.html

Elemento Resource

O elemento *resource* server para especificar todos os recursos, como classes java, bibliotecas nativas e propriedades do sistema, que fazem parte da aplicação. A definição de um recurso pode ser restrita para um especifico sistema operacional, arquitetura ou uso local dos atributos. *os*, *arch* e *locale.*

O elemento *resource* possui 6 sub-elementos que são: *jar*, *nativelib*, *j2se*, *property*, *package* e *extension*. Não iremos falar sobre *package* e *extension*.

O elemento *jar* especifica um arquivo JAR utilizado pela aplicação. Exemplo:

```
<jar href="myjar.jar"/>
```

O elemento *nativelib* especifica um arquivo JAR que contém bibliotecas nativas. Exemplo:

```
<nativelib href="lib/windows/corelib.jar"/>
```

O elemento *j2se* especifica qual versão da JRE a aplicação suporta, bem como os parâmetros padrões para JVM. O atributo *href* pode ser setado com a URL para o Java Web Start poder fazer o download da JRE caso o cliente não possua instalado, por padrão devemos sempre setar com a URL do site da Sun. Exemplo:

```
<j2se version="1.3" initial-heap-size="64m" href="http://java.sun.com/products/autodl/j2se"/>
```

O elemento *property* define uma propriedade do sistema que poderá ser acessado pelo método *getProperty* da classe *System* (*System.getProperty*). Ele elemento requer dois atributos: *name* e *value*. Exemplo:

```
<property name=”greeting” value=”Olá Mundo”/>
```

Elemento application-desc

Esse elemento informa que o arquivo JNLP está executando uma aplicação e não uma Applet. Ele possui um atributo opcional, *main-class*, que é usado para especificar o nome da classe principal da aplicação, isto é, a classe que possui o método *public static void main(String[] args[])*. O atributo *main-class* pode ser omitido caso o primeiro JAR definido contenha o arquivo *Manifest* informando a classe principal.

Argumentos podem ser especificados. Por exemplo:

```
<application-desc main-class="Main">
    <argument>arg1</argument>
    <argument>arg2</argument>
</application-desc>
```

Elemento applet-desc

Java Web Start também possui suporte para execução de Applets Java. Exemplo:

```
<applet-desc
      documentBase="http://..."
      name="TimePilot"
      main-class="TimePilot.TimePilotApp"
      width="527"
      height="428">
    <param name="key1" value="value1"/>
    <param name="key2" value="value2"/>
</applet-desc>
```

Criando a página Web para acessar nossa aplicação

Bom para podermos disponibilizar nossa aplicação, falta criarmos uma página Web que será responsável por invocar nossa aplicação através de um link para o arquivo *JNLP*. Nossa página Web, *index.html*, ficará no diretório raiz da nosso aplicação, ou seja dentro da pasta *tutorialjws.*

Um detalhe muito importante que não podemos deixar passar é que precisamos verificar se existe na máquina do cliente o Java Web Start instalado, pois se este não estiver disponível, o cliente com certeza irá dizer que o sistema não funciona. Para evitar essa situação vamos adicionar um código *Javascript* que faz essa verificação, caso o browser não encontre o Java Web Start instalado, vamos fazer donwload do Java Web Start, e logo após invocamos a página *index.html*.

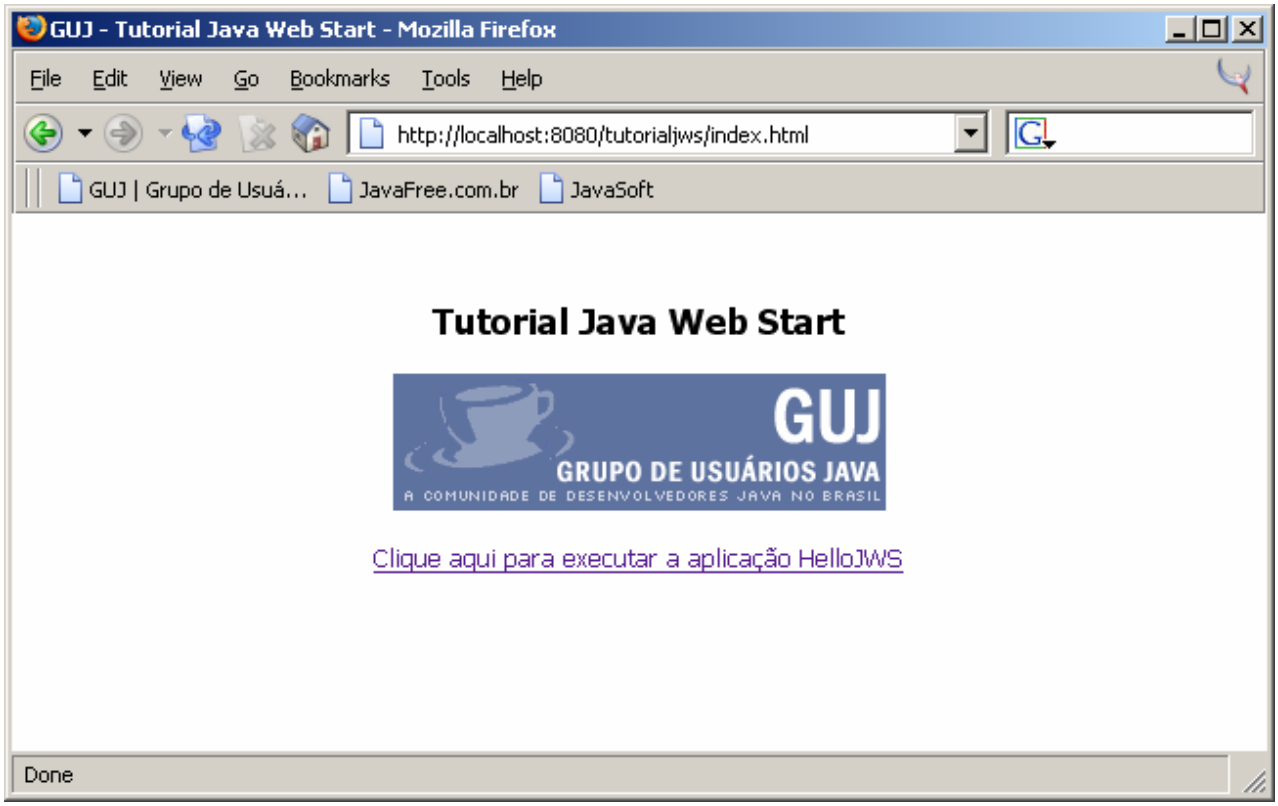
Abaixo a listagem completa da nossa página Web, *index.html*:

```
<html>

<head>
    <title>GUJ - Tutorial Java Web Start</title>
</head>

<script language="Javascript">
   var javawsInstalled = 0;
   isIE = "false";

   if (navigator.mimeTypes && navigator.mimeTypes.length) {
      x = navigator.mimeTypes['application/x-java-jnlp-file'];
      if (x) javawsInstalled = 1;
    } else {
      isIE = "true";
    }

    function insertLink(url, name) {
       if (javawsInstalled) {
          document.write("<a href=" + url + ">"  + name + "</a>");
       } else {
          document.write("Voc&ccedil; precisa intalar o Java Web Start");
       }
    }
```

```
</script>

<body>

    <br>

    <p align="center"><font face="tahoma" size="4"><b>Tutorial Java Web Start</b></h1>

    <p align="center"><img src="images/guj.gif" border="0"/></p>

    <p align="center"><font face="tahoma" size="2">

    <script>
        insertLink("TutorialJWS.jnlp", "Clique aqui para executar a aplicação HelloJWS");
    </script>
</body>

</html>
```

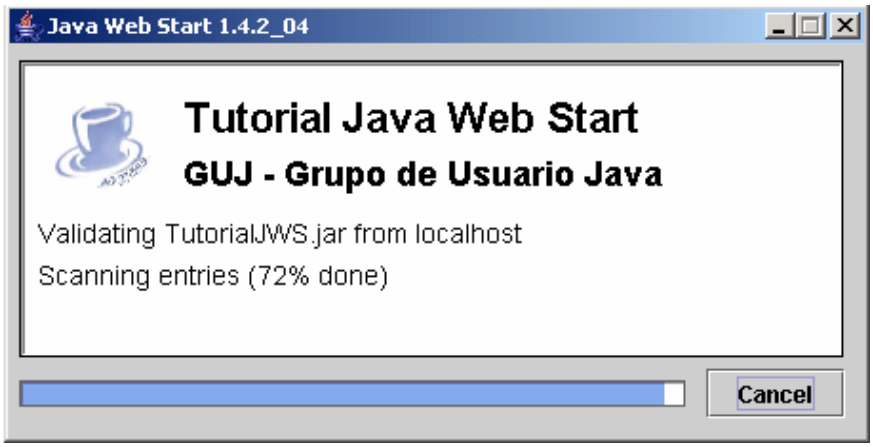
Colocando tudo para funcionar

Agora com tudo pronto vamos iniciar nosso servidor Web e acessar o endereço: http://localhost:8080/tutorialjws/index.html.



Página responsável por iniciar nossa aplicação.

O usuário através de um simples clique no link, irá iniciar a aplicação de forma simples e transparente.



Java Web Start em Ação.

Aplicação rodando na máquina do cliente.

Conclusão

Como podemos ver, o Java Web Start facilita muito a instalação, atualização e execução de aplicações Java. O exemplo que vimos aqui foi apenas uma simples demonstração do que o Java Web Start pode fazer, pois o esse tecnologia é muito mais ampla e completa.

Referências

http://java.sun.com/products/javawebstart/1.2/docs/developersguide.html
http://java.sun.com/developer/technicalArticles/Programming/jnlp/
Chinalia, Marcelo – Tutorial Java Web Start

Leonardo Marcelino (*leonardo.marcelino@gmail.com*) é programador certificado Java e a um ano e meio vem trabalhando com a tecnologia Java desenvolvendo aplicações Web e Deskop.