Java First-Tier: Aplicações

Construção de Applets

Grupo de Linguagens de Programação

Departamento de Informática
PUC-Rio

Applets

- Programas Java que executam em *browsers "java enabled"*
  - JVM que interpreta os *bytecodes*
- Imagem "viva" em uma página HTML
  - reage a comandos
  - muda sua aparência
  } manipulação "local"
  - interação (envio de dados) com o servidor

2

Applets: Internet × Intranet

- Uso de *applets* na Internet tem limitações
  - incompatibilidade de versões de Java nos *browsers*
  - *download* em linhas de baixa capacidade
- O ambiente em Intranets é diferente
  - ambiente é controlado
  - redes com taxas de transmissão bem mais elevadas

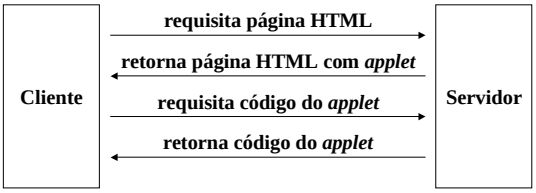
3

Applets

- Execução de código Java em um *browser*
  - definido em uma página do servidor
  - carregado e executado no cliente
- "Tag" HTML especifica
  - onde obter o código do *applet* (arquivos .class)
  - como posicionar o *applet* na página
- Passagem de parâmetros pela página HTML

4

"Protocolo"



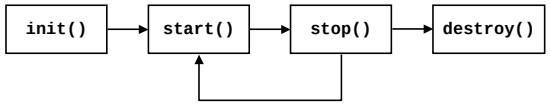
5

Execução de *Applets*

- Criação de área no *browser* para o *applet*
- Criação do objeto *applet* (um *panel* especial)
- Inicialização e execução do *applet*
  - execução do método **init**
  - execução do método **start**
- Posterior execução dos métodos **stop** e **destroy**

6

Ciclo de Vida



7

Ciclo de Vida: **init**

- Chamado pelo *browser* quando o *applet* é carregado
- Responsável pela inicialização do *applet*
  - processamento de parâmetros
  - criação e adição de componentes de interface
- O "ambiente" do *applet* pode não estar completo no momento de sua construção
  - está completo quando **`init`** é chamado

8

Ciclo de Vida: **start**

- Chamado pelo *browser* para informar que o *applet* deve iniciar sua execução
  - após a chamada ao método **`init`**
  - toda vez que o usuário retorna à página que contém o *applet*
- Só é necessário se alguma atividade deve ser suspensa quando o usuário deixa a página, e reativada na volta
  - *threads*, animações, audio

9

Ciclo de Vida: **stop**

- Chamado pelo *browser* para informar que o *applet* deve interromper sua execução
  - quando a página que contém o *applet* é substituída
  - imediatamente antes da "destruição" do *applet*
- As atividades interrompidas serão retomadas com a execução do método **`start`**.

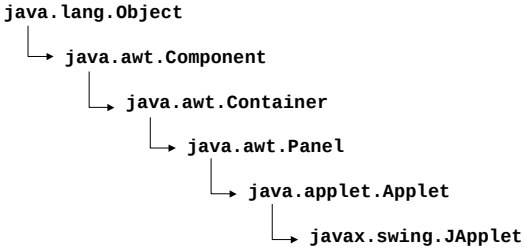
10

Ciclo de Vida: **destroy**

- Chamado pelo *browser* para que o *applet* libere recursos que tenha alocado
  - por exemplo, um *applet* com *threads* usaria o método **`init`** para criá-las e o método **`destroy`** para destruí-las.
- O método **`stop`** será sempre chamado antes

11

Hierarquia de Classes



12

Pacote **java.applet**

- **java.applet.Applet**
  - classe que modela um *applet*
- **java.applet.AppletContext**
  - interface que modela um visualizador (*browser*)
- **java.applet.AudioClip**
  - interface que modela um *clip* de áudio

13

Classe **Applet**

- Sub-classe de **Panel**, do pacote **awt**
- É, essencialmente, um painel inserido em uma página HTML, em um navegador
- Embora restrito em algumas funcionalidades (por uma questão de segurança), pode interagir com o usuário como qualquer aplicação gráfica **awt**

14

Métodos de **Applet**

```
void init()
void start()
void stop()
void destroy()
String getParameter(String name)
```

15

Exemplo de **Applet**

```
import java.awt.*;
import java.applet.*;

public class MeuApplet extends Applet {
  public void init() {
    String msg = getParameter("Mensagem");
    Label l = new Label(msg);
    l.setAlignment(Label.CENTER);
    setLayout(new BorderLayout());
    add(l, BorderLayout.CENTER);
  }
}
```

16

Exemplo de Página

```
<html>
 <head>
  <title>Teste do MeuApplet</title>
 </head>
 <body>
  <applet code=MeuApplet width=300 height=200>
   <param name="Mensagem" value="Olá Internet!">
  </applet>
 </body>
</html>
```

17

Outros Métodos de **Applet**

```
URL getDocumentBase()
URL getCodeBase()
String getAppletInfo()
String[][] getParameterInfo()
```

18

Fornecendo Informações

```
public String getAppletInfo() {
  return "Exibe uma mensagem na tela";
}

public String[][] getParameterInfo() {
  return new String[][] {
    {"Mensagem", "String", "Mensagem a exibir"}
  };
}
```

nome tipo descrição

19

Interface **AppletContext**

- Modela o navegador, isto é, o programa que hospeda o *applet*
- Disponibiliza métodos de conveniência para obter dados do servidor de onde veio o *applet*
  - imagens, arquivos de áudio
- Também disponibiliza métodos de controle do navegador

20

Obtendo o **AppletContext**

- A classe **Applet** disponibiliza um método específico para este fim:

```
AppletContext getAppletContext()
```

21

Métodos de **AppletContext**

```
Applet getApplet(String name)
Enumeration getApplets()
AudioClip getAudioClip(URL url)
Image getImage(URL url)
void showDocument(URL url)
void showDocument(URL url, String target)
void showStatus(String status)
```

22

Métodos de Conveniência

- Algumas funcionalidades providas pelo contexto podem ser acessadas diretamente através do *applet*. Para isso, a classe **Applet** disponibiliza métodos como:

```
void showStatus(String msg)
AudioClip getAudioClip(URL url)
Image getImage(URL url, String name)
```

23

Interface **AudioClip**

- Modela um *clip* de áudio
- Disponibiliza métodos para executar e interromper a execução do *clip*

24

Métodos de **AudioClip**

```
void play()
void loop()
void stop()
```

25

Mais Métodos de Conveniência

- Novamente por conveniência, é possível executar um *clip* de áudio diretamente a partir do *applet*. Para isso, a classe **Applet** disponibiliza:

```
void play(URL url)
void play(URL url, String name)
```

26

Exemplo de **AudioClip**

```
import java.awt.*;
import java.awt.event.*;
import java.applet.*;

public class MeuApplet extends Applet {
  AudioClip clip;
  public void init() {
    Button play = new Button("Play");
    Button stop = new Button("Stop");
    add(play);
    add(stop);
    clip = getAudioClip(getDocumentBase(), "spacemusic.au");
    play.addActionListener(new ActionListener() {
      public void actionPerformed(ActionEvent ev) {
        clip.loop(); }});
    stop.addActionListener(new ActionListener() {
      public void actionPerformed(ActionEvent ev) {
        clip.stop(); }});
  }
  public void stop() {
    clip.stop();
  }
}
```

27

Tag HTML para *Applets*

```
<APPLET
  CODE=nome da classe
  WIDTH=largura em pixels
  HEIGHT=altura em pixels
  [ARCHIVE=lista de arquivos jar]
  [CODEBASE=diretório]
  [ALT=texto alternativo]
  [NAME=nome do applet]
  [ALIGN=alinhamento]
  [VSPACE=espaço vertical em pixels]
  [HSPACE=espaço horizontal em pixels]
>
[<PARAM NAME=nome VALUE=valor>]
[<PARAM NAME=nome VALUE=valor>]
...
[texto alternativo]
</APPLET>
```

} **obrigatórios !**

28

Obtenção do código do *applet*

- atributo **CODE**
  - nome da classe que implementa o *applet*
  - se a classe pertence a um pacote, este deve ser especificado
- atributo **CODEBASE**
  - localização (**URL**) do código do *applet*
  - se omitido, é assumida a mesma localização do documento HTML

29

Obtenção do código do *applet*

- atributo **ARCHIVE**
  - lista de arquivos (.jar) que contém classes e outros recursos que serão “pré-carregados”

30

Exemplos

```
<applet code=MeuApplet
        width=300 height=200>
</applet>

<applet code=meupacote.MeuApplet
        width=300 height=200>
</applet>

<applet code=MeuApplet codebase=diretorioDosApplets
        width=300 height=200>
</applet>
```

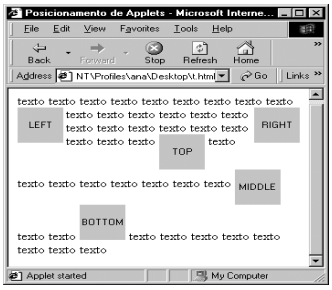
31

Posicionamento do *applet*

- Semelhante ao de imagens (*tag* **IMG**)
- Tamanho: atributos **WIDTH** e **HEIGHT**
- Espaçamento: atributos **VSPACE** e **HSPACE**
- Alinhamento: atributo **ALIGN**
  - **LEFT, RIGHT**
  - **TOP, TEXTTOP, MIDDLE, ABSMIDDLE, BASELINE, BOTTOM, ABSBOTTOM**

32

Exemplos de alinhamento



33

Comunicação entre *applets*

- *Applets* em uma mesma página podem se comunicar
- O atributo **NAME** especifica um nome para um *applet*
- O método **getApplet** (de **AppletContext**) retorna uma referência para um *applet* a partir de seu nome

34

Acessando outro *Applet*

- No arquivo HTML:

```
<applet code=MeuApplet width=300 height=200 name="Applet1">
</applet>
```

- Na classe que define um applet:

```
AppletContext ctx = getAppletContext();
MeuApplet applet1 = (MeuApplet)(ctx.getApplet("Applet1"));
applet1.recebeDados(…);
```

35

Arquivos **.jar**

- ***Java Archive*** (**jar**) é um formato de arquivo que permite juntar diversos arquivos (classes, imagens, sons) em um único arquivo compactado (como um .zip)
- Esse arquivo pode ser transmitido pela rede, do servidor para o cliente, otimizando o processo de troca das informações necessárias para o *applet*

36

Arquivos **.jar**

- São também comumente utilizados para empacotar/distribuir bibliotecas
  - as classes que compõem a biblioteca são armazenadas em um arquivo **.jar**
  - o arquivo **.jar** é adicionado ao **CLASSPATH**
- Um arquivo **.jar** também pode ser usado para distribuir uma aplicação
  - entrada no "manifest file" especifica a classe principal (*main)* aplicação

37

Programa **jar**

- O programa **jar** cria arquivos no formato **.jar** para serem usados em *applets* ou aplicações
- Um exemplo da sintaxe básica é:

```
jar cf MeuApplet.jar *.class *.au
```

38

Utilizando o Arquivo **.jar**

- No *tag* <**applet**>, podemos especificar o arquivo **.jar** e a JVM do navegador se encarrega de "extrair" a(s) classe(s) desse arquivo:

```
<html>
 <head>
  <title>Teste de Arquivos .jar</title>
 </head>
 <body>
  <applet code=MeuApplet archive=MeuApplet.jar
          width=300 height=200></applet>
 </body>
</html>
```

39

Obtendo recursos

- A JVM se encarrega de obter as classes do arquivo **.jar**, mas o mesmo não acontece com outros "recursos" (arquivos que não são de classes: imagens, textos etc.)
- Esses recursos devem ser "obtidos" de forma especial para que possam ser extraídos do arquivo **.jar**
  - o carregador ("class loader") sabe de onde a classe foi extraída e pode extrair o recurso da mesma localização

40

Ajustando o exemplo de **AudioClip**

- É necessário trocar uma linha:

```
clip = getAudioClip(getDocumentBase(), "spacemusic.au");
```



```
getAudioClip(MeuApplet.class.getResource("spacemusic.au"));
```

URL

41

Segurança

- Como o *applet* é um programa externo que irá executar na máquina do cliente, é **muito** importante que haja um controle sobre o que ele pode ou não fazer
  - o usuário não tem como impedir sua execução!
- Esse controle pode ser menos rigoroso se o *applet* for "assinado"

42

Modelo Caixa de Areia

- Tipicamente, *applets* executam em um modelo chamado "caixa de areia" *(sandbox*), no qual uma série de restrições são impostas
  - uma exceção do tipo **SecurityException** é lançada se o *applet* tenta violar uma dessas restrições

43

Restrições de Segurança

- No modelo "caixa de areia" o *applet*:
  - não pode executar código nativo
  - não pode acessar arquivos locais
  - não pode executar nenhum programa
  - não pode ler propriedades do sistema
    - ✓ somente versão do Java e do SO, e separadores
  - tem suas janelas (*pop up*) identificadas
  - só pode se comunicar com o computador de onde ele veio

44

Applets versus Java 2

- Apenas *applets* compatíveis com o Java 1.1 podem ser diretamente visualizados nos navegadores Netscape e Internet Explorer
- *Applets* desenvolvidos utilizando recursos do Java 2 precisam de um *plug-in* instalado para que possam ser visualizados
- A *Sun Microsystems* disponibiliza *plug-ins* para Netscape e para Internet Explorer

45

Mudanças no HTML

- Para utilizar o *plug-in*, é preciso que a página HTML diga isso, o que é feito através de uma mudança no *tag* HTML
- A *Sun* disponibiliza uma ferramenta que converte automaticamente arquivos HTML para que o *plug-in* seja utilizado
  - o *tag* que habilita uso do *plug-in* é bem mais complexo!

46

Obtenção do *Plug-in*

- O *tag* HTML gerado pela ferramenta da *Sun*, além de indicar ao navegador que o *plug-in* deve ser utilizado, traz a informação de onde este *plug-in* pode ser obtido caso ele ainda não esteja instalado
- O *plug-in* só precisa ser instalado uma única vez, permanecendo no navegador

47

Applets versus Swing

- A biblioteca Swing disponibiliza uma classe **JApplet** que pode ser utilizada (no lugar da classe **Applet**) para criar *applets* utilizando os elementos de interface do Swing
  - assim como um **JFrame**, um **JApplet** é um "top level container"
- Um *applet* swing pode ter uma barra de menu, o que não é possível no *applet* comum

48

Exemplo de **JApplet**

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

public class MeuApplet extends JApplet {
  public void init() {
    String msg = getParameter("Mensagem");
    JLabel l = new JLabel(msg);
    l.setHorizontalAlignment(JLabel.CENTER);
    getContentPane().add(l, BorderLayout.CENTER);
  }
}
```

49

Desenhando em um **Applet**

- A classe **Applet** herda de **Component** o método **paint**
  - esse método é chamado quando o conteúdo de um componente deve ser "pintado"
- Para "desenhar" em um *applet* deve-se redefinir o método **paint**

```
public void paint(Graphics g)
```

- A classe **Graphics** modela o *graphics context* que permite que aplicações desenhem sobre componentes

50

Métodos de **Graphics**

```
void setColor(Color c)
void drawRect(int x, int y, int width, int height)
void fillRect(int x, int y, int width, int height)
void drawOval(int x, int y, int width, int height)
void fillOval(int x, int y, int width, int height)
void drawLine(int x1, int y1, int x2, int y2)
void drawString(String str, int x, int y)
void setFont(Font font)
```

51