Contents

1. Getting started with applets
2. Developing an applet
3. Deploying an applet
4. Parameterizing an applet
5. Using Swing in an applet

Demo project:
DemoApplets

2

1. Getting Started with Applets

- Overview of applets
- How to write an applet
- Applet UIs

Overview of Applets

- What is an applet?
  - A special kind of Java program embedded in a Web page
  - Runs client-side within a browser
  - Managed and executed by a Java plug-in (built into the browser)
- Purpose of applets:
  - Perform intelligent client-side processing via Java code
  - E.g. UI animations
  - E.g. complex algorithmic calculations

How to Write an Applet

- An applet is a Java class
  - Must inherit from `java.applet.Applet` (AWT applets)
  - Or inherit from `javax.swing.JApplet` (Swing applets)

- You can implement the following lifecycle methods:
  - `void init()`
    - Useful for one-time initialization (if needed) when applet is loaded
    - Put your initialization code here, rather than in a constructor
  - `void start()`
    - Called when applet is displayed – you should override this method
    - E.g. create a new thread to do time-consuming tasks, play animations, etc.
  - `void stop()`
    - Called when user navigates away from page – you should override this method
    - E.g. stop background threads
  - `void destroy()`
    - Useful for releasing resources (if needed)

5

Applet UIs

- Applets can contain UI components
  - Text boxes, buttons, etc.
- The `Applet` class inherits from `java.awt.Panel`
  - You can add AWT UI components directly to the applet
  - E.g. `TextField`, `Button`
- The `JApplet` class inherits from `Applet`
  - Swing applets have a single root pane (just like a `JFrame`)
  - You can add Swing UI components to the root pane
  - E.g. `JTextField`, `JButton`

6

2. Developing an Applet

- A "hello world" applet
- Running an applet in Eclipse
- Packaging an applet as a JAR

A Hello World Applet

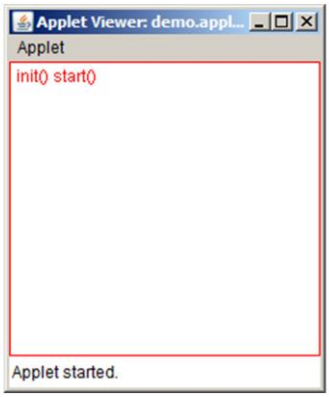
- Here's a simple applet, to display applet lifecycle info

```
public class HelloWorldApplet extends Applet {

  private StringBuffer buffer;

  public void init() {
    buffer = new StringBuffer();
    addItem("init() ");
  }

  public void start() {
    addItem("start() ");
  }

  public void stop() {
    addItem("stop() ");
  }

  public void destroy() {
    addItem("destroy() ");
  }

  …
}
```

```
private void addItem(String message) {
  System.out.println(message);
  buffer.append(message);
  repaint();
}

public void paint(Graphics g) {
  g.setColor(Color.RED);
  g.drawRect(5, 5, getWidth() - 10, getHeight() - 10);
  g.drawString(buffer.toString(), 10, 20);
}
```

Running an Applet in Eclipse

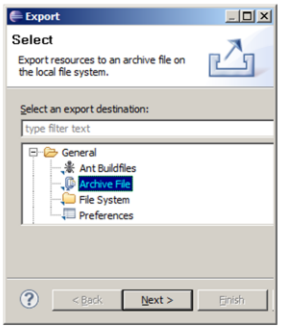
- You can run an applet within Eclipse
  - Run As | Java Applet
  - Launches `appletviewer.exe` (JDK tool for hosting an applet)



9

Packaging an Applet as a JAR

- When you are ready to deploy your applet to the production Web server…
  - … it's common practice to package the applet as a JAR file
- To do this within Eclipse:
  - Right-click the Java file(s)
  - Select Export from pop-up menu
  - Select General | Archive File
  - Specify JAR filename / location



- To do this from the command prompt:

```
cd bin  (i.e. the build folder for your Java classes)
jar cvf MyApplet.jar .
```

10

3. Deploying an Applet

- Using a simple <applet> tag
- Using the Deployment Toolkit script
- The Java Network Launch Protocol

Using a Simple <applet> Tag

- Prior to Java SE 6 Update 10:
  - You deploy an applet by using an `<applet>` tag in an HTML page
  - The browser's Java Plug-In automatically runs the latest version of the JRE installed on the client machine
- Example:

```
<html>
…
<body>

  <h1>Here's my HelloWorld applet</h1>
  <applet archive='MyApplet.jar'
          code='demo.applets.HelloWorldApplet.class'
          width='300'
          height='300'>
  </applet>
  p>Do you like it?</p>

</body>
</html>
```

HelloWorldHostPage1.html

Using <applet>:

- Don't have to use a JAR
- Can specify codebase

12

Using the Deployment Toolkit Script

- For Java SE 6 Update 10 and above:
  - You should use the Deployment Toolkit JavaScript script
  - The script exposes a `deployJava` object, which has functions for downloading RIAs (i.e. applets and Java Web Start applications)

- Example:

```
<html>
<script src="http://www.java.com/js/deployJava.js"></script>
…
<body>

  <h1>Here's my HelloWorld applet</h1>
  <script>
    var attributes = { archive:'MyApplet.jar',
                       code:'demo.applets.HelloWorldApplet.class',
                       width:300, height:300 };
    deployJava.runApplet(attributes, null, '1.6');
  </script>
  p>Do you like it?</p>

</body>
</html>
```

HelloWorldHostPage2.html

runApplet() arguments:

- attributes
- parameters
- JRE min version (browser will download if necessary)

13

The Java Network Launch Protocol

- The `deployJava` object provides access to the Java Network Launch Protocol (JNLP)
  - Gives you very powerful control over how RIAs are deployed
- For full details on JNLP, see:
  - `http://download.oracle.com/javase/6/docs/technotes/guides/jweb/deployment_advice.html`
- We also show an example of JNLP in Appendix H

14

4. Parameterizing an Applet

- Overview
- Defining parameters
- Accessing parameters

Overview

- You can define parameters for an applet
  - Define the parameters in the HTML page, as part of the `<applet>` tag or in the `deployJava` JavaScript object
  - Access the parameters in the applet code
- Benefits:
  - Avoid hard-coding details in the applet code
  - Easier to modify in the HTML page

16

Defining Parameters

- If you are using an `<applet>` tag:

```
<applet archive='MyApplet.jar'
        code='demo.applets.ParameterizedHelloWorldApplet.class'
        width='300'
        height='300'>

  <param name='greeting1' value='Bonjour' />
  <param name='greeting2' value='mon ami'  />

</applet>                                          ParameterizedHelloWorldHostPage1.html
```

- If you are using the `deployJava` object:

```
<script>
  var attributes = { archive:'MyApplet.jar',
                     code:'demo.applets.ParameterizedHelloWorldApplet.class',
                     width:300, height:300 };
  var parameters = { greeting1:'Hello', greeting2:'mate' }
  deployJava.runApplet(attributes, parameters, '1.6');
</script>                                          ParameterizedHelloWorldHostPage2.html
```

Accessing Parameters

- To access parameters in an applet, call `getParameter()`

```
public class ParameterizedHelloWorldApplet extends Applet {

  private StringBuffer buffer;

  public void init() {
    buffer = new StringBuffer();

    String greeting1 = this.getParameter("greeting1");
    String greeting2 = this.getParameter("greeting2");

    addItem("init() says " + greeting1 + ", " + greeting2);
  }
  …

}
```

5. Using Swing in an Applet

- Overview
- Example Swing applet
- Running the applet

Overview

- You can use Swing components in an applet
  - Inherit from `JApplet` rather than `Applet`
- `JApplet` has a content pane (just like `JFrame`)
  - You can assign a standard pane (e.g. `JPanel`) or a custom pane
  - You can add components to the pane, as normal
- Note:
  - You should perform UI tasks in the event-dispatching thread
  - Ensures thread safety

20

Example Swing Applet (1 of 2)

- Here's the outline for a simple Swing applet

```
public class SwingApplet extends JApplet implements ActionListener {

  public void init() {
    try {
      SwingUtilities.invokeAndWait(new Runnable() {
                                     public void run() {
                                       createGUI();
                                     }
                                   });
    } catch (Exception e) {
      System.err.println("createGUI() exception occurred");
    }
  }

  // See createGUI() method on next slide.
  …
}
```

Example Swing Applet (2 of 2)

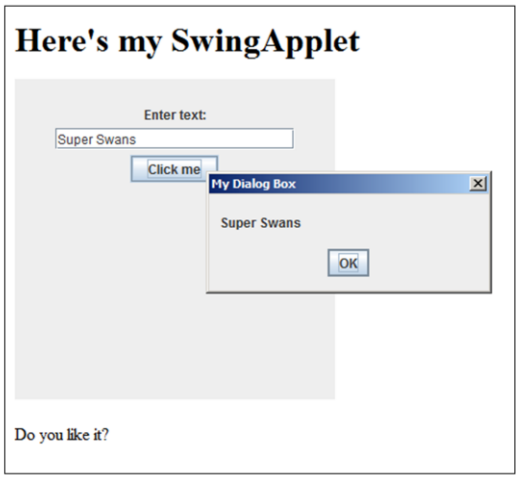
- Here's the code to create the GUI, and to handle events

```
public class SwingApplet extends JApplet implements ActionListener {
  …
  private JLabel label         = new JLabel("Enter text:");
  private JTextField textField = new JTextField(20);
  private JButton button       = new JButton("Click me");

  private void createGUI() {
    JPanel pane = new JPanel();
    pane.setBorder(BorderFactory.createEmptyBorder(20, 20, 20, 20));

    pane.add(label);
    pane.add(textField);
    pane.add(button);
    button.addActionListener(this);

    this.setContentPane(pane);
  }

  public void actionPerformed(ActionEvent evt) {
    JOptionPane.showMessageDialog(this,
                      textField.getText(),
                      "My Dialog Box",
                      JOptionPane.PLAIN_MESSAGE);
  }
}
```

22

Running the Applet

- This is how the applet appears when you run it
  - Also see the `SwingAppletHostPage.html` host page



23

Any Questions?

24