



Deploying Java Applications

Originals of Slides and Source Code for Examples:
<http://courses.coreservlets.com/Course-Materials/java5.html>

Customized Java EE Training: <http://courses.coreservlets.com/>
Servlets, JSP, JSF 1.x & JSF 2.0, Struts Classic & Struts 2, Ajax, GWT, Spring, Hibernate/JPA, Java 5 & 6.
Developed and taught by well-known author and developer. At public venues or onsite at *your* location.



For live Java training, please see training courses at <http://courses.coreservlets.com/>. Servlets, JSP, Struts Classic, Struts 2, JSF 1.x, JSF 2.0, Ajax (with jQuery, Dojo, Prototype, Ext, etc.), GWT, Java 5, Java 6, Spring, Hibernate/JPA, and customized combinations of topics.



Taught by the author of *Core Servlets and JSP*, *More Servlets and JSP*, and this tutorial. Available at public venues, or customized versions can be held on-site at your organization. Contact hall@coreservlets.com for details.

Agenda

- **Major Java Deployment Options**

- Individual .class files
- JAR files
- OS wrapper around class files or JAR files
- Applets
- Java Web Start
- Server-based Alternatives

4

Individual Class Files: Approach

- **Send all required .class files**

- Packaged classes go in subdirectories matching package name

- **Identify a class with a main method**

```
public class SomeClass {  
    public static void main(String[] args) {  
        startTheWholeshebang();  
    }  
}
```

- **Invoke the class**

- Packageless class
 - > `java SomeClass command-line-args`
- Packaged class
 - > `java somePackage.SomeClass command-line-args`

5

Individual Class Files: Pros/Cons

- **Advantages**

- Very simple to set up
- Programmers can modify individual classes easily

- **Disadvantages**

- Requires lots of separate files
 - Painful to install
- Messy for non-Java-programmers
- Requires the right Java version
- No option for updates of class files
- No option to assist in installing Java

6

Individual Class Files: Example

```
public class Launcher {  
    public static void main(String[] args) {  
        String[] names =  
            {"Ebay", "Amazon.com", "Chase Bank", "PayPal"};  
        int index = (int)(Math.random() * names.length);  
        new Phisher(names[index]);  
    }  
}
```

7

Individual Class Files: Example

```
public class Phisher extends JFrame {
    public Phisher(String company) {
        WindowUtilities.setNativeLookAndFeel();
        addWindowListener(new ExitListener());
        Container content = getContentPane();
        String title = company + " Security Verification";
        setTitle(title);
        String imageURL =
            "http://images.encarta.msn.com/xrefmedia/" +
            "sharemed/targets/images/pho/000a5/000a5038.jpg";
        String labelText =
            "<html><CENTER><H1>" + title + "</H1>" +
            "<IMG SRC=\"" + imageURL + "\"><BR>" +
            "<H2>Your " + company + " account may have been " +
            "compromised.<BR>To avoid cancellation, please " +
            "reenter your account information.<BR>Sorry for " +
            "the inconvenience, but security is our " +
            "priority.</H2>";
        JLabel label = new JLabel(labelText); ...
    }
}
```

8

Individual Class Files: Example

```
DOS>dir/B
ExitListener.class
ExitListener.java
JFrameExample.java
LabeledTextField.class
LabeledTextField.java
Launcher.class
Launcher.java
Phisher.class
Phisher.java
WindowUtilities.class
WindowUtilities.java
```

```
DOS> java Launcher
```



9

JAR Files: Approach

- **Create a text file that designates main class**
 - `Main-Class: classname`
 - Must have carriage return at the end of the line
- **Create a JAR file with extra entry in manifest**
 - `jar -cmf yourtextfile.txt yourjarfile.jar *.class`
 - Must have m and f in same order as text/JAR filenames
- **Execute class from JAR file**
 - `java -jar yourjarfile.jar`

10

JAR Files: Pros/Cons

- **Advantages**
 - Only one file to send to user
 - No subdirectories or other files
- **Disadvantages**
 - Not easy for programmer to modify
 - Requires the right Java version
 - No option for updates of class files
 - No option to assist in installing Java

11

JAR Files: Example

- **ManifestEntry.txt**

Main-Class: Launcher

← Blank line here!

- DOS> jar -cmf ManifestEntry.txt Phisher.jar *.class

- DOS> java -jar Phisher.jar



12

OS Wrapper

- **Wrap the call to Java inside a .bat file (Windows) or shell script (Unix/Linux)**

- User can execute it in normal way (e.g., double click it)
- Use javaw instead of java to avoid a popup console

- **Advantages**

- User does not need to open DOS window or Unix shell
- Double clicking icon is more natural to most users

- **Disadvantages**

- File must be in same directory as .class files or JAR file
 - Or contain full path to class/JAR files
- Requires the right Java version
- No option for updates of class files
- No option to assist in installing Java

- **Launcher.bat**

javaw -jar Launcher.jar



13

Applets: Approach

- **Create a Web page that refers to an applet**

```
<APPLET CLASS="MyApplet.class" ...>  
  Warning for users without Java  
</APPLET>
```

 - There is also Java plugin alternative with extra options
 - Applets covered in earlier lecture
- **User loads URL in browser**
 - `http://host/path/filewithapplet.html`
- **Applets can be embedded within browser or launch separate popup windows**

14

Applets: Pros/Cons

- **Advantages**
 - User can bookmark location
 - User gets updates automatically
- **Disadvantages**
 - Security restrictions
 - Applets cannot read/write local files, execute local programs, open arbitrary network connections, etc.
 - Digitally signed applets partially mitigate this
 - User must have right version of Java plugin
 - Accessed through browser
 - Does not seem like a "regular" program

15

Applets: Example

- **Yahoo Games (<http://games.yahoo.com/>)**



16

Java WebStart: General Approach

- **Create XML file referring to JAR file and main class name**
 - XML file is called *mylauncher.jnlp*
- **Access the XML file in a browser**
 - <http://host/path/mylauncher.jnlp>
- **Code is cached locally**
 - Check for new version is automatic
 - Can also be run offline
 - Can create desktop icon automatically

17

Java WebStart: Pros/Cons

- **Advantages**

- Ensures user has latest application updates
 - Downloads automatically
- Ensures user has proper Java version
 - Downloads with minimal user intervention
- JAR files cached locally
 - Faster download, offline execution
- Can create desktop shortcut as launcher

- **Disadvantages**

- Similar security restrictions to applets
 - Again, digital signatures can mitigate this
- User must have *some* WebStart version installed
 - Fails for users that do not have Java installed at all
 - Autodetection of WebStart requires JavaScript code

18

Java WebStart: Example (Launcher)

```
public class Launcher2 {
    public static void main(String[] args) {
        String[] names =
            {"Ebay", "Amazon.com", "Chase Bank", "PayPal"};
        int index = (int)(Math.random() * names.length);
        new Phisher2(names[index]);
    }
}
```

19

Java WebStart: Example (GUI Code)

```
public class Phisher2 extends JFrame {
    public Phisher2(String company) {
        WindowUtilities.setNativeLookAndFeel();
        addWindowListener(new ExitListener());
        Container content = getContentPane();
        String title = company + " Security Verification";
        setTitle(title);
        String labelText =
            "<html><CENTER><H1>" + title + "</H1>" +
            "<H2>Your " + company + " account may have been " +
            "compromised.<BR>To avoid cancellation, please " +
            "reenter your account information.<BR>Sorry for " +
            "the inconvenience, but security is our " +
            "priority.</H2>";
        JLabel label = new JLabel(labelText);
        ClassLoader cl = getClass().getClassLoader();
        Icon bankVaultIcon =
            new ImageIcon(cl.getResource("images/bankvault.jpg"));
        label.setIcon(bankVaultIcon);
    }
}
```

20

Java WebStart: Example (JAR File)

- JAR file must contain images and class files
- No extra manifest entry needed
- DOS> jar -cf Phisher2.jar images *.class

21

Java WebStart: Example (JNLP File)

```
<jnlp spec="1.0"
      codebase="http://www.coreservlets.com/webstartdemo"
      href="Phisher2.jnlp">
  <information>
    <title>Phisher2</title>
    <vendor>freesecuritywarnings.com</vendor>
    <offline-allowed/>
  </information>
  <resources>
    <j2se version="1.5+"
          href="http://java.sun.com/products/autodl/j2se"/>
    <jar href="Phisher2.jar"/>
  </resources>
  <application-desc main-class="Launcher2"/>
</jnlp>
```

22

Java WebStart: Example (Execution)

- After entering URL
 - <http://www.coreservlets.com/webstartdemo/Phisher2.jnlp>



- Later WebStart options (from Start Menu)



23

Java WebStart: Other JNLP File Capabilities

- **Bypassing security**
 - `<security><all-permissions/></security>`
 - Asks user for permission to run in unrestricted mode
 - Requires your JAR file(s) to be digitally signed
- **Auto-Shortcut**
 - `<shortcut online="false">`
 - `<desktop/>`
 - `<menu submenu="My Corporation Apps"/>`
 - `</shortcut>`
- **java-vm-args (attribute of j2se element)**
 - Lets you pass args to java
- **More JNLP syntax info**
 - <http://java.sun.com/j2se/1.5.0/docs/guide/javaws/developersguide/syntax.html>

24

Java WebStart: More Info

- **Top-Level Documentation Page**
 - <http://java.sun.com/j2se/1.5.0/docs/guide/javaws/>
- **FAQ**
 - <http://java.sun.com/j2se/1.5.0/docs/guide/javaws/developersguide/faq.html>
- **Developer's Guide**
 - <http://java.sun.com/j2se/1.5.0/docs/guide/javaws/developersguide/contents.html>
 - Includes downloadable JavaScript and VBScript code for detecting users that do not have *any* version of WebStart, and for letting them download it in such a case
 - Shows how to digitally sign JAR files so that application can request unrestricted privileges

25

Server-Based Alternatives

- **Regular Sockets**
 - Request data from a normal server.
Parse response explicitly.
- **Object Streams**
 - Request Java objects from a server running same major version of Java. No parsing required.
- **RMI**
 - Convenient wrapper to automate exchange of Java objects via object streams.
- **Servlets and Web Services**
 - Use HTTP to get HTML (regular Web applications), XML (Web services), or Java object streams (HTTP tunneling).

26

Summary

- **Individual class files**
 - Pros: simple to set up, easy Java programmer to edit
 - Cons: no auto-updates of code or Java version, nonintuitive to non-programmer
- **JAR files**
 - Pros: single file
 - Cons: same as class files
- **OS wrapper**
 - Pros: more intuitive (clickable)
 - Cons: same as class files
- **Applets**
 - Pros: Web access, auto-updates of code, security
 - Cons: Web only, no auto-update of Java version, security
- **Java WebStart**
 - Pros: Web/desktop access, auto-updates of code and Java, security
 - Cons: more complicated, security

27



Questions?

Customized Java EE Training: <http://courses.coreservlets.com/>

Servlets, JSP, JSF 1.x & JSF 2.0, Struts Classic & Struts 2, Ajax, GWT, Spring, Hibernate/JPA, Java 5 & 6.

Developed and taught by well-known author and developer. At public venues or onsite at *your* location.