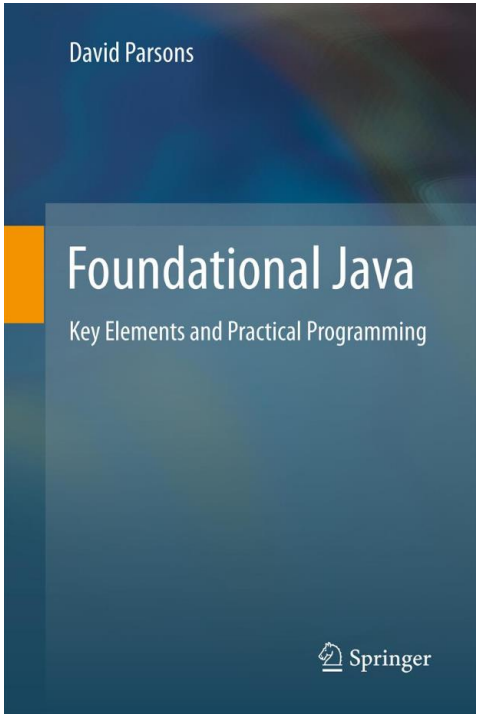


Chapter 20
Java Web Start and Applets

Foundational Java

Key Elements and Practical Programming

Web Start and Applets

- Java first came to prominence because of Java applets, running in web browsers
  - Still widely used, and can be a useful component of rich internet applications
- Also associated with the web, but less directly, is Java Web Start
  - Allows desk top applications to be deployed over the web
- Applets can now use the same deployment mechanism as Java Web Start, making it very easy to switch between the two modes of deployment

Web Browsers, URLs and HTTP

- Web browser software retrieves information from the World Wide Web (WWW) using Uniform Resource Locators (URLs)
- URL
  - The Internet address of a resource on a particular server
  - Comprises:
    - Protocol
    - Server address
    - Name of the resource (including any path information).
- HTTP
  - The protocol prefix for web pages is ‘http://’ (hypertext transfer protocol)

Complete URLs

- The server address
  - Usually begins ‘www’ (World Wide Web), followed by the name of the site and its ‘domain’
  - Separated by periods, e.g.

```
http://www.introjava.com
```

  - If you are running a test server on your local machine, the domain name becomes ‘localhost’.
- Path and resource name
  - The URL can include the location (directory) and name of the particular file

```
http://www.introjava.com/documentation/index.html
```

  - The file name will generally end in ‘html’ (or sometimes just ‘htm’)
  - If no filename is specified, ‘index.html’ is often the server’s default file name that it will send back to the browser

HTML (HyperText Markup Language)

- Web browsers display screens of information written using HTML
- The browser formats the content using tags embedded into the text of the file
- All HTML files begin with an <html> tag and end with </html>

```
<!DOCTYPE html>
<html>
  <head>
    <title> A simple HTML page </title>
  </head>
  <body>
    <h1>This is a heading</h1>
    <p>Here is the text of my HTML page</p>
  </body>
</html>
```

Java Web Start

- Provides a way to deploy standard applications via the web
- An alternative to applets or web applications (from a web perspective)
- Eases deployment of desktop applications
- Can automatically download and install a JRE
- Programmers can specify which JRE version a given program needs in order to execute
- No Internet connection is required to execute the downloaded programs

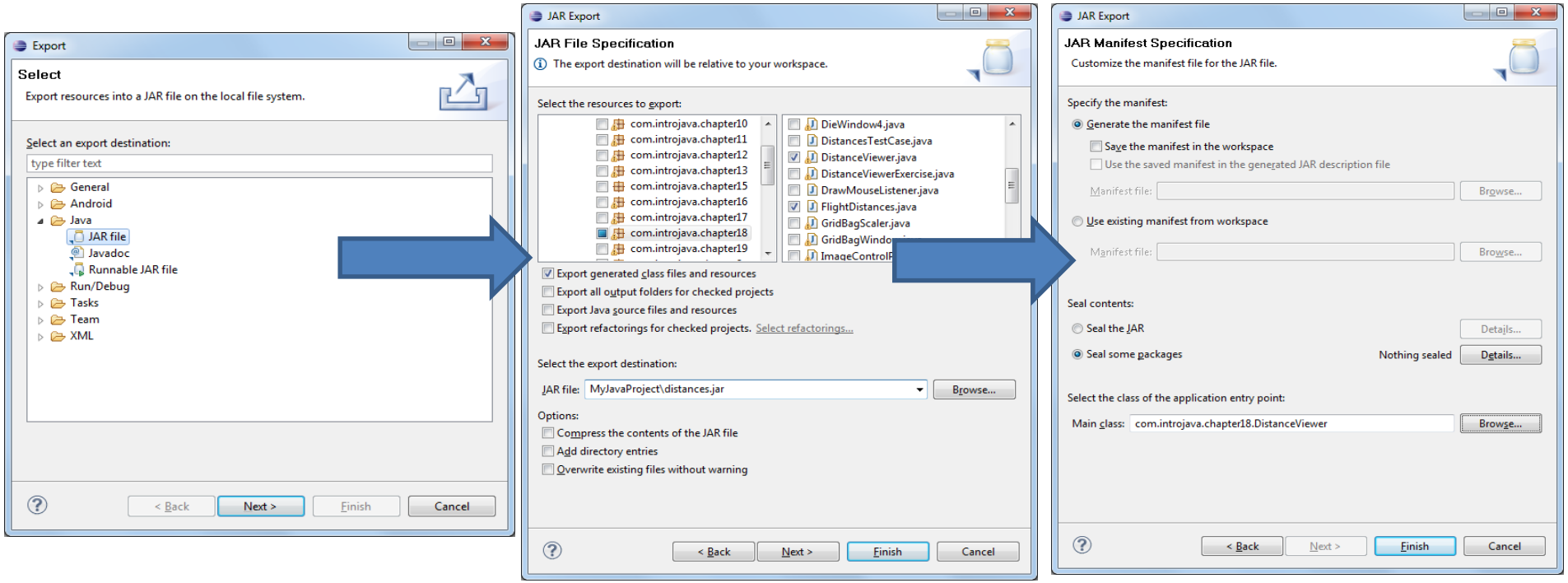
Core Components

- A Java application
  - Archived in a JAR file
  - With an identifiable ‘main’ class
- A Java Network Launch Protocol (JNLP) file
  - An XML file that describes how the application should be launched
  - A generic rich internet application (RIA) launch mechanism – can also be used for applets
- A Web (HTML) page
  - To host the application launcher

Creating a JAR in Eclipse

- A JAR file can be created using the Eclipse export facility
- Select ‘export’ from the ‘file’ menu
- In the ‘Export’ dialog, within the ‘Java’ folder, you can select ‘JAR File’ as the type of export
- In the ‘JAR Export’ dialog, choose all of the files that need to be in the archive
- Select the export folder and the name of the JAR file
- Select one of the classes in the JAR as the class used as the application entry point

Creating a JAR in Eclipse



JNLP File

- Note that the 'codebase' and 'href' attributes are empty in this example – they are not required when testing local files

```
<?xml version="1.0" encoding="UTF-8"?>
<jnlp spec="1.0+" codebase="" href="">
  <information>
    <title>Flight Distances Viewer</title>
    <vendor>Introductory Java</vendor>
  </information>
  <resources>
    <j2se version="1.6+" href="http://java.sun.com/products/autodl/j2se"/>
    <jar href="distances.jar" main="true" />
  </resources>
  <application-desc name="Flight Distances Viewer"
    main-class="com.introjava.chapter18.DistanceViewer">
  </application-desc>
</jnlp>
```

Web Page

- The web page can contain a simple hyperlink to the JNLP file

```
<a href="jnlp_file">Launch Application</a>
```

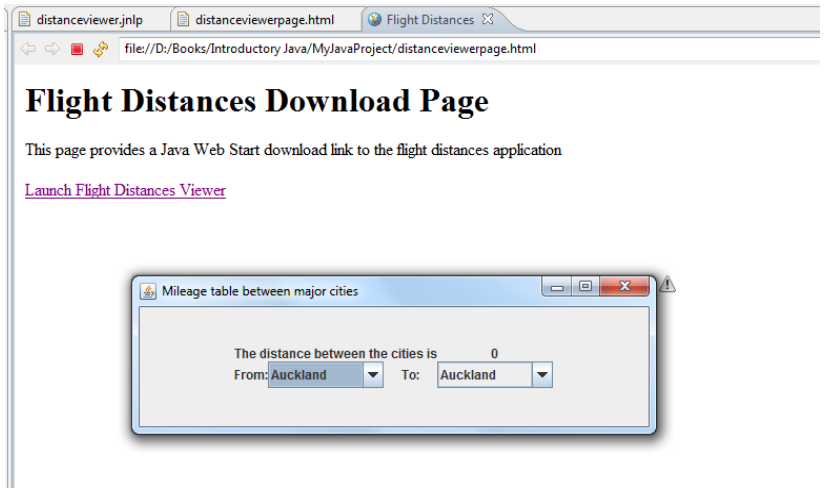
- This link launches the DistanceViewer

```
<a href="distanceviewer.jnlp">Launch Flight Distances Viewer</a>
```

Launching the Application

- When the hyperlink is clicked the Java icon appears before the application is launched

- All being well, the application should launch in the same way that it does when running directly as a Java application



JavaScript Launch Button

- Another way of launching a Java Web Start application is to use a JavaScript button
  - Use the 'deployJava.createWebStartLaunchButton' method

```
<script src="http://www.java.com/js/deployJava.js"></script>
<script>
// using JavaScript to get location of JNLP file relative to HTML page
  var dir = location.href.substring(0, location.href.lastIndexOf('/')+1);
  var url = dir + "distanceviewer.jnlp";
  deployJava.createWebStartLaunchButton(url, '1.6.0');
</script>
```

Deploying to a Server

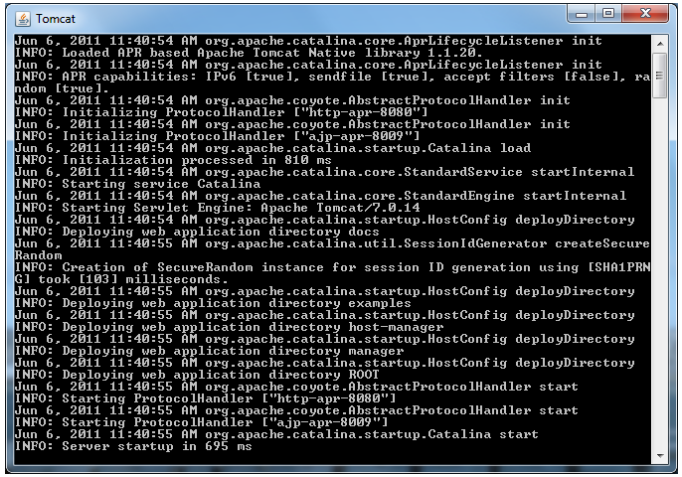
- Web Start applications need to be deployed to a server in order to be downloaded over the internet
- Tomcat is an open source Java application server that supports the Web application components of the Java Enterprise Edition specification
- Tomcat can be downloaded as zipped archive

Starting Tomcat

- To start Tomcat, navigate to the 'bin' folder of the Tomcat installation directory, e.g.

```
C:\apache-tomcat-7\bin
```

- In the 'bin' folder there will be a file called 'startup' that can be used to start the server
- As Tomcat starts up, you should see a command window appear

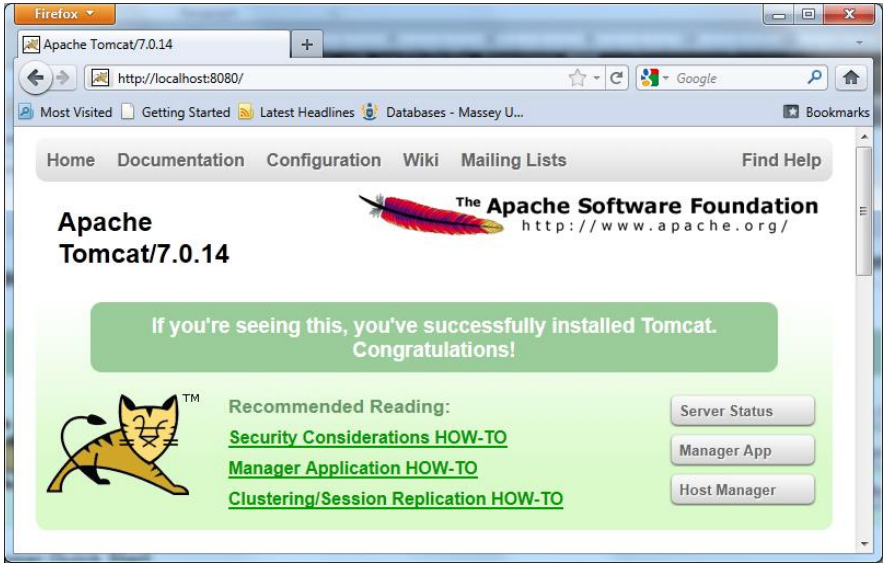


The ‘localhost’ URL and port number

- When we run a test server on the local machine, we use the loopback address
  - IP address 127.0.0.1, which is also known as ‘localhost’
- The HTTP server that is included with Tomcat runs by default on port 8080

http://localhost:8080

- To check that Tomcat is running correctly, open a Web browser and direct it to this URL



Web Application Structure and Deployment

- Java Enterprise Edition (Java EE) specifies some folders and files that are used to deploy Java web applications
- The static contact (e.g. HTML files) can be put into the root of the folder structure
  - Along with other deployable resources, such as JNLP files and JAR files
- A Java web application may include a deployment descriptor
  - An XML file used to configure how the server deploys the application
  - Called ‘web.xml’, put into a special folder called ‘WEB-INF’
- Java EE application components may be packaged in JAR files
  - a JAR file for a web application archive is given a ‘.war’ extension

XML Deployment Descriptors

- The ‘web.xml’ deployment descriptor must be both well-formed and valid XML so that its elements can be processed by the application server.
- The following example uses an XML Schema for validation

```
<?xml version="1.0" ?>
<web-app xmlns="http://java.sun.com/xml/ns/javaee"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
    http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
    version="2.5">
     ...
</web-app>
```

- The root element is called ‘web-app’
  - Inside this element are many optional nested elements

web.xml Entries

- The web.xml needs to contain the MIME mapping for the JNLP file so that the server handles this type of file correctly

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

- Another useful element to add is the 'welcome-file-list'
  - Configures the default page that will be served when a client connects to the URI of the Web application

Complete web.xml

```
<?xml version="1.0" ?>
<web-app xmlns="http://java.sun.com/xml/ns/j2ee"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee/web-app_2_5.xsd"
 version="2.5">

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

  <welcome-file-list>
    <welcome-file>distanceviewerpage.html</welcome-file>
  </welcome-file-list>

</web-app>
```

mime mapping for JNLP files

default welcome page

Setting the ‘codebase’ and ‘href’

- If a JNLP file is to be used for deployment from a web server, then it needs to include values for the ‘codebase’ and ‘href’ attributes
- codebase
  - The URI of the web application
- href
  - The name of the JNLP file

```
<?xml version="1.0" encoding="UTF-8"?>
<jnlp spec="1.0+" codebase=http://localhost:8080/introjava
   href="distanceviewer.jnlp">
```

Deploying the WAR file with Ant

- Much of the build file used in this example uses tasks that should be familiar from Chapter 14
- First, a series of properties are set for various folders and file names, including the deployment folder for Tomcat web applications

```
<property name="deployfolder" value="c:\WebStart" />
<property name="source" value="${deployfolder}\javasource" />
<property name="build" value="${deployfolder}\javabuild" />
<property name="webroot" value="${deployfolder}\webroot" />
<property name="webapp"
  value="${deployfolder}\introjava.war" />
<property name="tomcat-deploy" value="C:\apache-tomcat-7.0.14\webapps" />
```

Properties

- deployfolder
  - The root folder for the various deployment files, which will also be used as the build destination folder for the web archive
- source
  - The folder used to initially copy the Java source code to before it is compiled and packaged into a JAR file. It is also used as the temporary folder for the web.xml file before it is add to the WAR file
- build
  - The output folder for the compiled Java code
- webroot
  - This folder will contain files that need to be in the root folder of the web application. This will include the HTML file, the JNLP file and the JAR file
- webapp
  - This specifies the file name of the web application to be built (‘introjava.war’)
- tomcat-deploy
  - The deployment folder for the Tomcat web server

Ant Targets

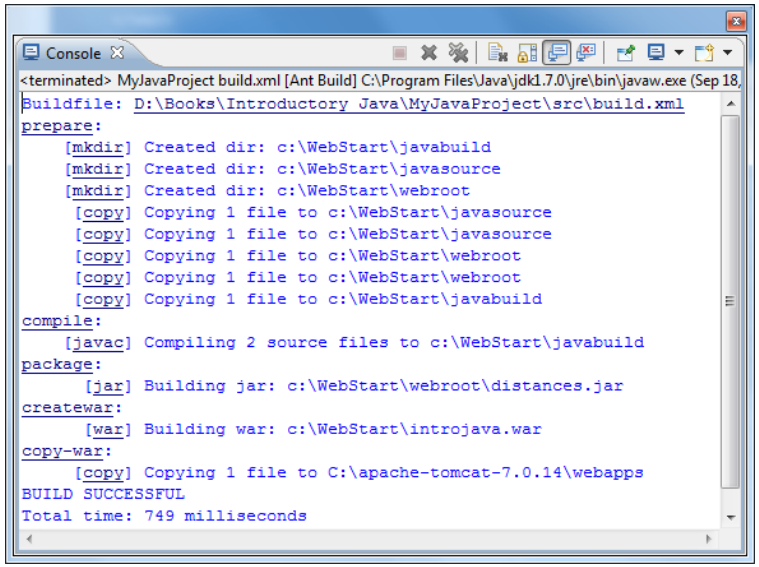
- The ‘prepare’ target creates the required folders and copies files into them from the Eclipse source folders
- The ‘compile’ and ‘package’ targets compile the Java code and create a JAR file
- The ‘createwar’ target uses the ‘war’ task to create a web archive

```
<!-- build the war file -->
<target name="createwar" depends="package">
  <war destfile="${webapp}" webxml="${source}/web.xml" >
    <fileset dir="${webroot}"/>
  </war>
</target>
```

- ‘copy-war’, copies the war file to the web server’s deployment folder

Running the Build File

- The WAR file is built, and then it is deployed to the server
- When the war is copied to Tomcat, the Web application is rebuilt and redeployed



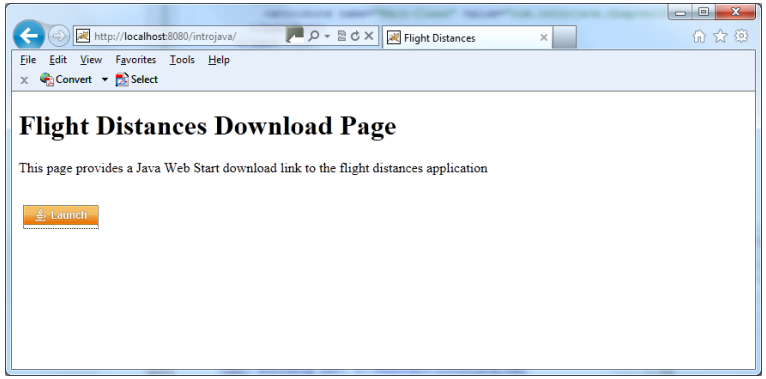
The Deployed Application

- The application will be deployed using the name of the WAR file as the name of the web application

```
http://localhost:8080/introjava
```

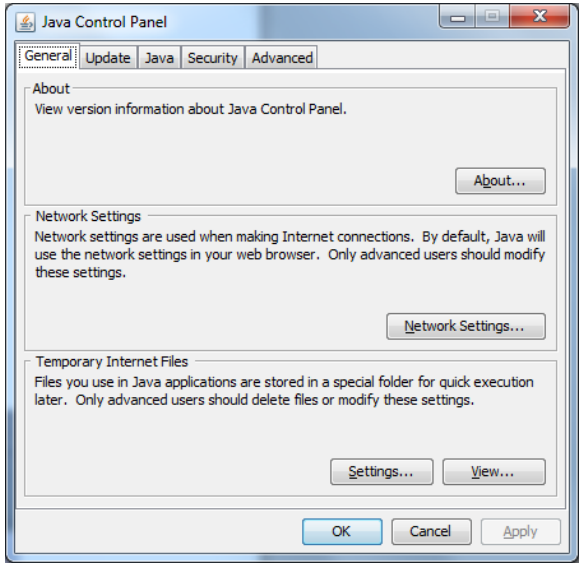
- Because ‘distanceviewer.html’ is the default welcome file, this page should appear without needing to be specifically requested
- We could alternatively invoke it directly

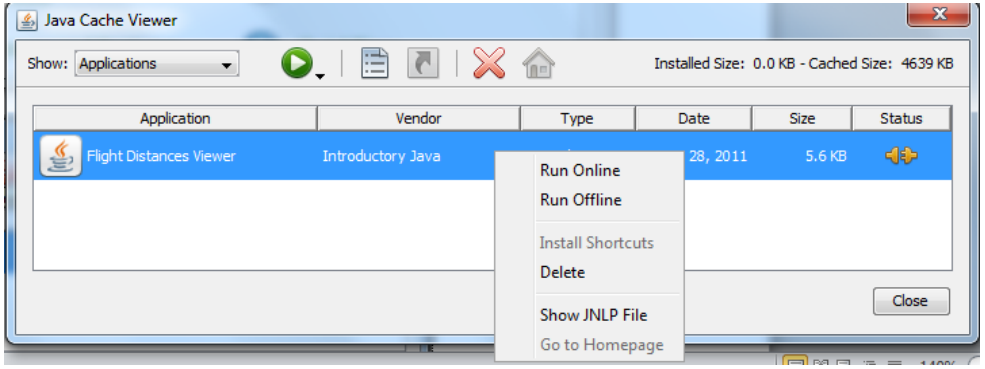
```
http://localhost:8080/webhomecover/distanceviewer.html
```



Running Applications from the Cache

- Open the Java control panel
- Press the ‘view’ button under the ‘Temporary Internet Files’ section on the ‘General’ tab
- This will open a dialog containing a list of all the Java applications in the cache
- Right click on an application to run it
- If the JNLP file includes the ‘offline-allowed’ element, a cached application can be run offline,
- It may also be possible to create a desktop shortcut to the cached application





Security

- Applications launched with Java Web Start are run in a *sandbox*
- Protects users against malicious code
- Unsigned JAR files launched by Java Web Start remain in this sandbox, meaning they cannot access local files or the network
- Alternatively, JARs can be signed and verified

Exercise 20.1

- Deploy the ‘DrawShapes’ application using Java Web Start

Applets

- An applet is a Java program that is specifically designed to run within a web browser
- Not downloaded as a locally run application, but runs directly in the browser
- The main problem with applets has historically been how to manage seamless deployment across different browsers
  - 'applet' tag deprecated tag in HTML 5
  - Other tags that could be used, 'object' and 'embed', had a range of issues that made applet deployment difficult
- A more recent approach is to use JNLP to launch applets

How Applets Differ from Applications

- Swing Applets must inherit from the JApplet class
- The ‘init’ method
  - Rather than using the constructor, we use the init method for all initialization processes
- No ‘main’ method
  - Loading a web page containing an applet will start the applet running, calling the ‘init’ method
  - There is no ‘main’ method in an applet
- The default layout manager
  - The default layout manager for a JApplet is FlowLayout

Converting to an Applet

- Converting the distance viewer application used in the Java Web Start example into an applet
- The Distances class, since it does not involve any user interaction, can remain unchanged
- The DistanceViewer class remains much as before but does need some modification
- DistanceViewerApplet inherits from JApplet

```
public class DistanceViewerApplet extends JApplet
```

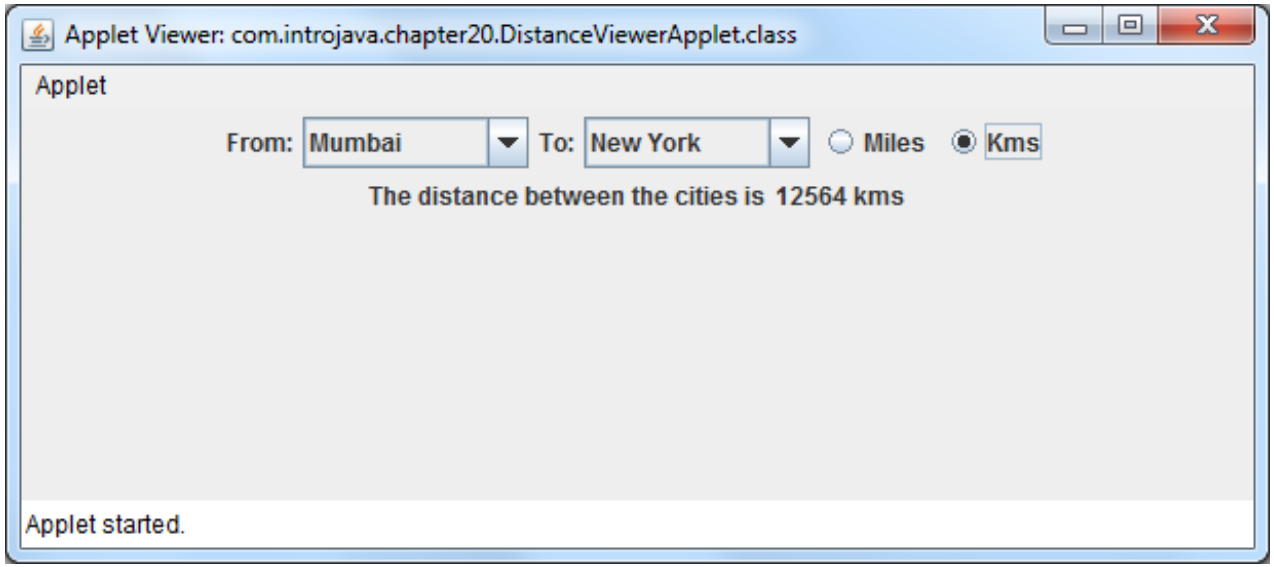
The ‘init’ Method

- Instead of using the constructor, all initliazation is done in the ‘init’ method.
- The ‘invokeandWait’ code used here is for handling the threads correctly

```
public void init() {
// Execute a job on the event-dispatching thread, creating this applet's GUI
  try {
    SwingUtilities.invokeAndWait(new Runnable() {
      public void run() {
        createGUI();
      }
    });
  } catch (Exception e) { …
```

Running an Applet in the Applet Viewer

- To test an applet in Eclipse without having to deploy it in a web page, you can run an applet directly in the Applet Viewer
  - Select 'Run As' -> 'Java Applet'



Applet JNLP File

- The ‘applet-desc’ element incudes the applet name, main class, height and width

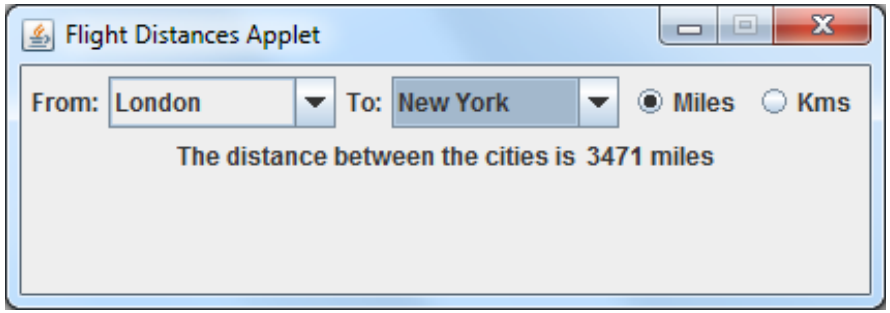
```
<jnlp spec="1.0+" codebase=http://localhost:8080/introapplet href="distanceapplet.jnlp">
  <information>
    <title>Flight Distances Applet</title>
    <vendor>Introductory Java</vendor>
  </information>
  <resources>
    <j2se version="1.6+" href="http://java.sun.com/products/autodl/j2se" />
    <jar href="distances.jar" main="true" />
  </resources>
  <applet-desc
    name="Distance Viewer Applet"
    main-class="com.introjava.chapter20.DistanceViewerApplet"
    width="400" height="100">
  </applet-desc>
</jnlp>
```

Launching the Applet

- The HTML page will refer to the applet JNLP file

```
<script>
   var dir = location.href.substring(0, location.href.lastIndexOf('/')+1);
   var url = dir + "distanceapplet.jnlp";
   deployJava.createWebStartLaunchButton(url, '1.6.0');
</script>
```

- The functionality is the same as the Java Web Start application, but it is hosted by its web page



Exercise 20.2

- Write an ‘Ohms law’ applet using Swing components
- Ohms law calculates the resistance of a circuit by dividing the voltage by the current
- Your applet should allow the user to enter the voltage and the current into text fields, displaying the resistance when a button is pressed
- Deploy your applet using JNLP

Summary

- Distributing applications over the web using Java Web Start
- How applets differ from applications
  - Though both can be deployed using JNLP
- Three key differences between applications and applets
  - Applets run in web browsers or in the applet viewer
  - Applets are subclasses of JApplet, not JFrame
  - Applets have no main method, but are initialized using the ‘init’ method