# Java SE 6 Update 10: Deployment

Sang Shin
http://www.javapassion.com
"Learn with Passion!"

# Topics

- Issue with JRE before Java SE 6 Update 10
- Java kernel
- Quick start
- New Plug-in architecture
- Java Deployment Kit
- Applet launching through JNLP
- Applet/JavaScript interaction

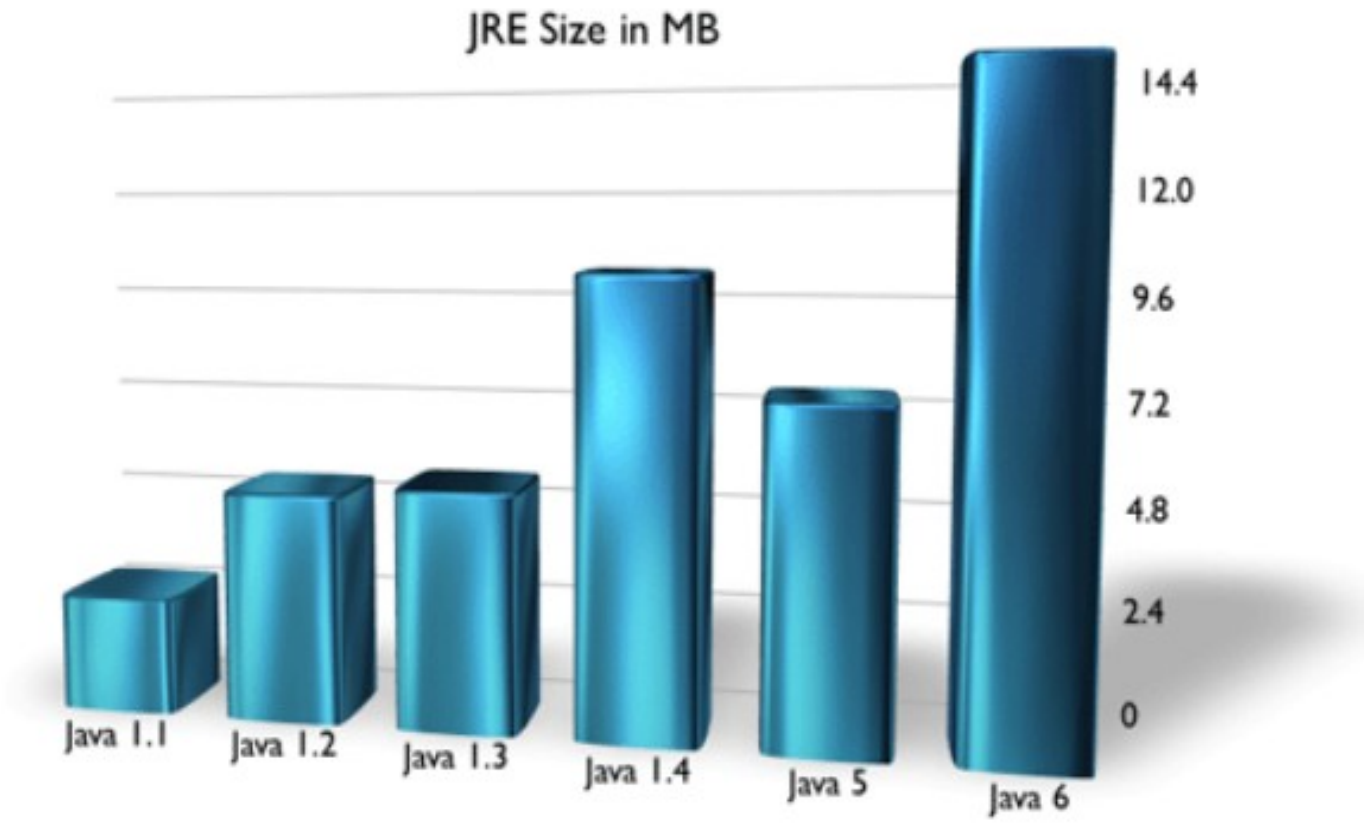
# Issues of JRE before Java SE 6 Update 10

# Issues with JRE before Java SE 6 u10

- JRE complaints
  - Too big, too long, too much resources, too ugly
- JRE to browser integration
  - Too brittle, breaking either one will undo the other
  - Not customizable per applet
  - Cannot patch in place

# Java Kernel:
# Reducing JRE Size

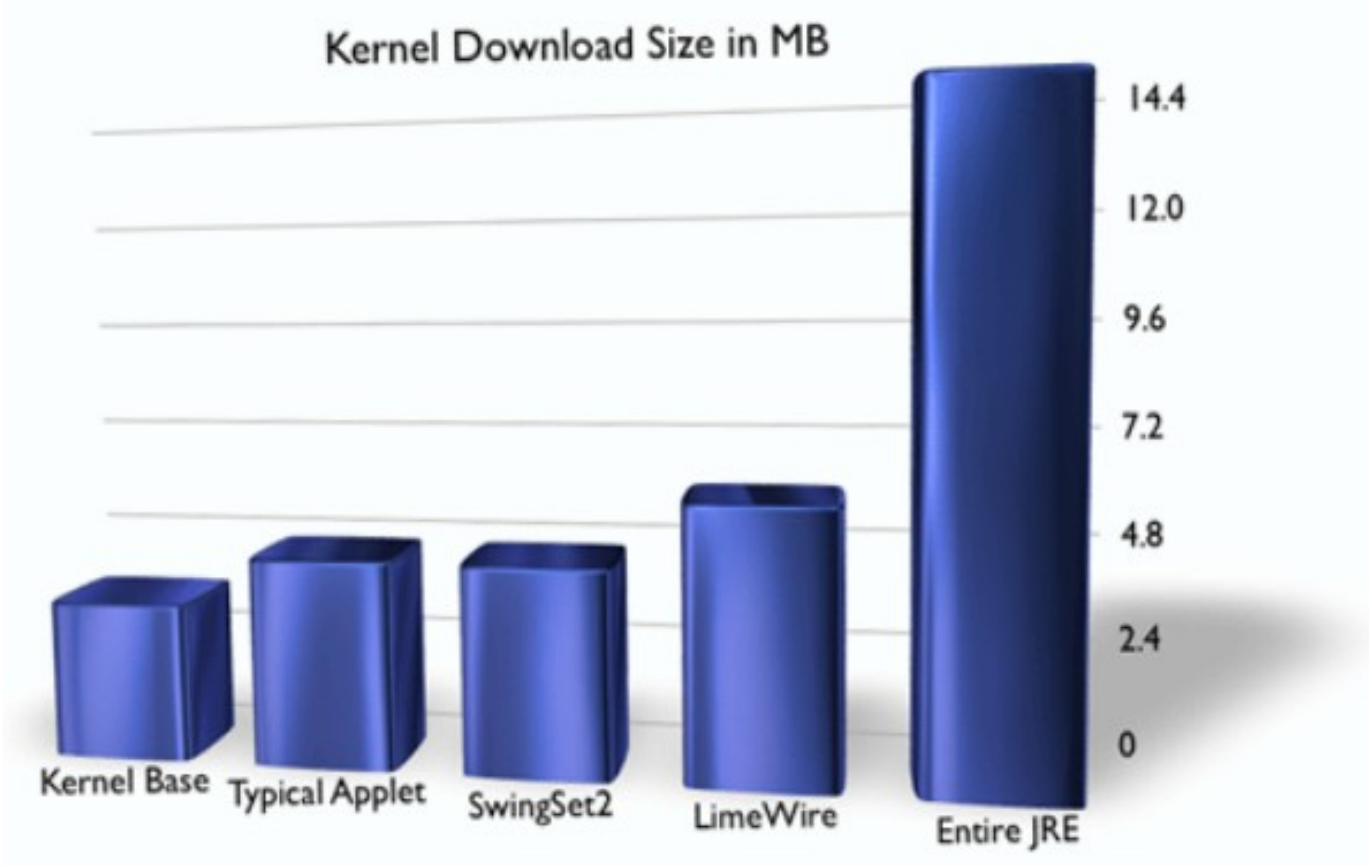
# JRE is Too Big!



# Solution – Java Kernel Installer

- JRE provides APIs for many function needs
  - Swing/AWT, SQL, RMI, XML, concurrency, etc...
- Solve the problem by packaging Java APIs to many sets
- When JRE needed to be installed
  - Java Kernel is downloaded first
  - Additional packages required for the application are downloaded
  - Application is launched
  - Other packages being download in the back end

# Java Kernel Download Size



# Quick Start: Reducing The Startup Time for Cold-Start

# Startup Too Slow

- Pre-Java SE 6 up10 startup state
  - "Warm" startup is acceptable in most cases (Warm start is the time that it takes for a Java application to start when you have recently run Java application on your system. Cold start, on the other hand, refers to the time that it takes to launch the first Java application after a fresh reboot.)
  - Utilizes class data sharing between VMs or previous VM execution
  - "Cold" startup takes longer with lots of disk access
- Java SE 6 up10 Quick starter background process reads JRE files, causing them to be cached
  - Sets up warm startup environment for both cold and warm startup

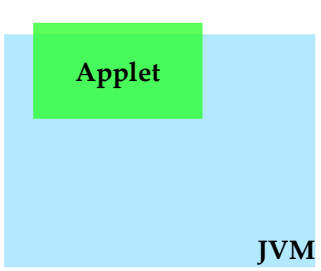
# New Plug-in Architecture

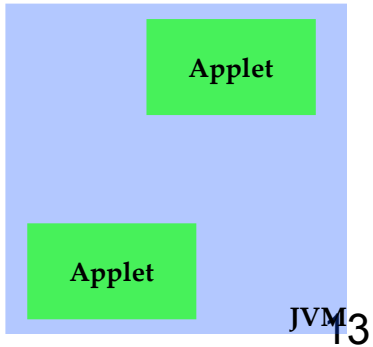
# Previous Java Plugin Architecture (Before Java SE 6 update 10)

- Applet in browser process space – a malfunctioning applet can crash the browser
- All applets share the same JVM instance owned by the browser
- Not configurable on a per Applet basis

# New Plugin Architecture Java SE 6 Update 10

- Live outside the browser
  - > Cannot crash the browser
- Applets can run in separate JVM
  - > Choose the version they want
  - > Configurable per applet basis
  - > Live beyond the browser
- Unify Applet and JavaWeb Start model
- Works on FF3 and IE7 and up only

# Java Plug-in and the Browser

- A Java applet runs in the context of a browser. The Java Plug-in software in the browser controls the launch and execution of Java applets.
- The Java Plug-in software creates a worker thread for every Java applet. It launches an applet in an instance of the Java Runtime Environment (JRE) software.

# Java Plug-in and Applet JRE

- Normally, all applets run in the same instance of the JRE. The Java Plug-in software starts a new instance of the JRE in the following cases:
  - When an applet requests to be executed in a specific version of the JRE.
  - When an applet specifies its own JRE startup parameters, for example, the heap size. A new applet uses an existing JRE if its requirements are a subset of an existing JRE, otherwise, a new JRE instance is started
- An applet will run in an existing JRE if the following conditions are met:
  - The JRE version required by the applet matches an existing JRE.
  - The JRE's startup parameters satisfy the applet's requirements.

# Parameters to Applet, Object, Embed tags

- Animated GIFs displayed before Applet is loaded
  - image: String
  - boxborder: boolean
  - centerimage:boolean
- java_arguments: specifies JVM arguments

```
<APPLET archive="my_applet.jar" code="MyApplet"
                width="300" height="300">
    <PARAM name="java_arguments"
                value="-Xmx256m -Dsun.java2d.noddraw=true">
</APPLET
```

- separate_jvm:boolean
- java_version: specify the minimum Java version

# Java Deployment Kit

# Java Deployment Kit

- A Javascript library that detects JRE, download missing software, create applet tags, etc
  - Accurate detection of installed JREs
  - Seamless JRE installation (JRE can be installed without user intervention)
  - Complete applet launching (JRE detection and, if necessary, upgrading) in a single line of code
  - Complete Web Start program launching in a single line of code
  - Enables easier Applet deployment
  - Applets can be deployed even on a browser which does not have any JRE installed - JRE will be installed
- http://java.com/js/deployJava.js (for Java applets)
  - Creates *deployJava* Javascript object

# Hosting HTML File of Java Applet

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
   <head>
      <title>Test page for launching the application via JNLP</title>
   </head>
   <body>
      <h3>Test page for launching the application via JNLP</h3>
      <a href="launch.jnlp">Launch the application</a>
      <!-- Or use the following script element to launch with the Deployment Toolkit -
      <!-- Open the deployJava.js script to view its documentation -->
      <!--
      <script src="http://java.com/js/deployJava.js"></script>
      <script>
         var url="http://[fill in your URL]/launch.jnlp"
         deployJava.createWebStartLaunchButton(url, "1.6")
      </script>
      -->
   </body>
</html>
```

# JavaScript Functions in *deployJava.js*

- *getJREs()*
  - returns a list of JRE versions
- *versionCheck(version)*
  - checks if a particular version is installed
- *runApplet(attr, params, version)*
  - ensures the appropriate version is installed and writes out an applet tag
- Many more

# Deploy Applets: Using Applet Tag Attributes

```
<script src="http://java.com/js/deployJava.js"/>
<script>
        function launch() {
            deployJava.runApplet({
                codebase:'http://www.mycompany.com/java/',
                archive:'myapplet.jar', code:'Main.class',
                width:400, height:300},
            null, "1.4");
        }
</script>
...
<button onclick="launch()">Launch Applet</button>
```

# Trigger JRE installation

```
<script>
   var attributes = {......};
   var parameters = {......};
   version = '1.6';

   deployJava.setInstallerType('kernel');
   deployJava.setAdditionalPackages(
         'javax.swing, javax.xml, ...');

   deployJava.runApplet(attributes, parameters,
        version);
</script>
```

# Draggable Java Applet

- Enable with new applet parameter
  - Default: Alt + Left-click

```
<APPLET archive="my_applet.jar" code="MyApplet">
    <PARAM name="draggable" value="true">
</APPLET>
```

- A set of interactive methods
  - Watch the method signature
  - `boolean isAppletDragStart(MouseEvent e);`
  - `void appletDragStarted();`
  - `void appletDragFinished();`
  - `void setAppletCloseListener(ActionListener l);`
  - `void appletRestored();`

# Demo:

applet_AppletWithParameters
1513_javase6u10_deployment.zip

# Applet Launching through JNLP

# Applets Can be Launched through JNLP

- Java Plug-In provides support for launching applets directly from JNLP files.
  - Previously, only Java Web Start utilized JNLP files, for the purpose of launching Java applications. Now Java applets can be described using the same meta-descriptor

# Applets Can be Launched through JNLP

- To launch an applet from a JNLP file, use the jnlp_href parameter in the <applet> tag

```
<applet width="300" height="300" code="com.mycompany.MyApplet">
  <param name="jnlp_href" value="my_applet.jnlp">
</applet>
```

- Or pass it as a parameter to deployJava.runApplet(..)

```
<script>
  var attributes = { code:'AppletTakesParams.class',
     archive:'applet_AppletWithParameters.jar',  width:800, height:50} ;
  var parameters = {jnlp_href: 'my_applet.jnlp',
                    paramOutsideJNLPFile: 'fooOutsideJNLP'} ;
  deployJava.runApplet(attributes, parameters, '1.6');
</script>
```

# JNLP File

```
<?xml version="1.0" encoding="UTF-8"?>
<jnlp spec="1.0+" codebase="" href="">
   <information>
      <title>Applet Takes Params</title>
      <vendor>Dynamic Team</vendor>
   </information>
   <resources>
      <!-- Application Resources -->
      <j2se version="1.6+"
           href="http://java.sun.com/products/autodl/j2se"/>
      <jar href="applet_AppletWithParameters.jar" main="true" />
   </resources>
   <applet-desc
       name="Applet Takes Params"
       main-class="AppletTakesParams"
       width="800"
       height="50">
          <param name="paramStr" value="someString"/>
          <param name="paramInt" value="22"/>
    </applet-desc>
    <update check="background"/>
</jnlp>
```

# Defining & Using Applet Parameters

- Parameters are to Java applets what command-line arguments are to applications. They enable the user to customize the applet's operation.
  - By defining parameters, you can increase your applet's flexibility, making your applet work in multiple situations without recoding and recompiling it.
- You can specify an applet's input parameters in the applet's Java Network Launch Protocol (JNLP) file or in the <parameter> element of the <applet> tag.
  - It is usually better to specify the parameters in the applet's JNLP file so that the parameters can be supplied consistently even if the applet is deployed on multiple web pages.
  - If the applet's parameters will vary by web page, then you should specify the parameters in the <parameter> element of the <applet> tag.

# Demo:

applet_AppletWithParameters
1513_javase6u10_deployment.zip

# Applet and JavaScript Interaction

# Java / Javascript Support

- Completely rewritten Java / Javascript integration
- Formerly Mozilla specific functionality now working in IE
  - Static method access
  - Construct new Java objects from Javascript
- Opportunity to respecify and reintroduce “LiveConnect”
  - Potentially could change the nature of AJAX applications
- Applet → JavaScript
  - Needs to enable MAYSCRIPT option in <applet> tag

# Invoke JavaScript from Applet

- `netscape.javascript.JSObject` provides easy access to HTML DOM
- Get a JSObject instance

```
public static JSObject getWindow(Applet a)
```

- A set of methods to navigate DOM

```
public Object call(String methodName, Object args[])
public Object eval(String s)
public Object getMember(String name)
public void setMember(String name, Object value)
```

# LiveConnect Example

```
import netscape.javascript.*;

class MyApplet extends Applet {
    public void init() {
        JSObject win = JSObject.getWindow(this);
        JSObject doc =
                (JSObject)win.getMember("document");
        JSObject loc =
                (JSObject)doc.getMember("location");

        String s = (String) loc.getMember("href");
        win.call("f", null);
    }
}
```

# DOM Support

- Access and traverse DOM
  - Alternative to LiveConnect
- Retrieve and process DOM with DOMService
- Need to implement DOMAction
  - Synchronous – invokeAndWait()
  - Asynchronous – invokeLater()

```
public MyApplet extends JApplet implement DOMAction {
    ...

    DOMService.getService(this).
```

# DOM Support Example

```
public Object run(DOMAccessor accessor) {
    HTMLDocument doc =
(HTMLDocument)accessor.getDocument(this);
    HTMLElement body = doc.getBody();
    //Get all <input> tag
    NodeList list = body.getElementsByTag("input");
    for (int i = 0; i < list.getLength(); i++) {
        NamedNodeMap map = list.item(i).getAttributes();
        //Look for specific ids
        String id = map.getNamedItem("id").getNodeValue();
        if ("name".equals(id)) {
            map.getNamedItem("value")
                    .setNodeValue("some text value");
        } ...
    }
```

# JavaScript to Java/Applets

- JavaScript can access applet methods

```
function setValue(val) {
    var myApplet = document.getElemenById("myApplet")
    myApplet.setText(val)
}
<applet id="myApplet" ...>
```

- JavaScript can instantiate any Java class with the applet.Packages namespace

```
function setValue() {
    var myApplet = document.getElemenById("myApplet")
    var swing = myApplet.Packages.javax.swing
    swing.JOptionPane.showMessageDialog(...)
}
```

# Demo:

applet_InvokingAppletMethodsFromJavaScript,
applet_TraversingDOM

1513_javase6u10_deployment.zip

# Thank you!

Check JavaPassion.com Codecamps!
http://www.javapassion.com/codecamps
"Learn with Passion!"