

Applets as front-ends to server-side programming

Objectives

- ▶ Introduce applets
 - ▶ Examples of Java graphical programming
 - ▶ How-to put an applet in a HTML page
 - ▶ The HTML Applet tag and alternatives
- ▶ Applet communication with the environment
 - ▶ Applet-Browser (AppletContext)
 - ▶ Applet-Applet
 - ▶ Applet-JavaScript and JavaScript-Applet
 - ▶ Applet-page using DOM
- ▶ Applet signing
- ▶ Applet-server communication
- ▶ Media in Applets and in Java

Example - java code

```
import java.applet.*;
import java.awt.*; // needed for Graphics
public class FirstApplet extends Applet {
    // we draw a Hello. No interaction
    public void paint(Graphics g) {
        g.drawString("Hello!", 25, 50);
    }
}
```

Applets

- ▶ Applets are based on a Java Virtual Machine running inside a browser as a *Plug-in*
- ▶ As graphical applications, applets can give more interactive interfaces than e.g. HTML forms
- ▶ Since they have all the Java functionality, applets can connect to a server and communicate with it just like any Java app.
 - ▶ However, there are security restrictions on applets downloaded from other sites than the applet's site
 - ▶ E.g. the local file system cannot be freely accessed
- ▶ Applets are written as subclasses of `java.applet.Applet`
 - ▶ They redefine some methods to achieve desired functionality
- ▶ Since applets are downloaded before execution, large code may make the user wait quite a lot
 - ▶ Caching in browser cache was the initial mechanism
 - ▶ Now the Java plugin has more sophisticated caching features

Example - HTML code

```
<html>
  <head>
    <title>
      My first applet
    </title>
  </head>
  <body>
    <applet code="FirstApplet.class"
            width="150"
            height="50" />
  </body>
</html>
```

Applet How-To

- ▶ Make a subclass of `java.applet.Applet` and compile
 - ▶ To use the latest GUI libraries, use `javax.swing.JApplet`
- ▶ Make a HTML file that refers to the applet via the `APPLET` tag and its `CODE` attribute
- ▶ Test with
 - ▶ `appletviewer` file.html or
 - ▶ (most often) load the HTML in a www-browser.
 - ▶ Normally, browsers have a "Java Console" where you can see exceptions, `System.out` output etc.
 - ▶ To reload the applet class after a change, reloading the page may not be enough!
 - ▶ Shift-reload may work. Ctrl-Shift-R, or Ctrl-Shift-F5
 - ▶ In the Java Plugin console, press x to clean class cache (press h for other commands)

A simple graphical applet

```
import java.applet.*;
import java.awt.event.*;
import java.awt.*;
public class SimpleGraphicalApplet extends Applet
implements java.awt.event.ActionListener {
    TextField input= new TextField();
    TextArea output= new TextArea(3, 20);
    /* constructor: arrange the two buttons nicely */
    public SimpleGraphicalApplet(){
        output.setEditable(false); //no input!
        setLayout(new java.awt.BorderLayout());
        add(input, "North");
        add(output, "Center");
        input.addActionListener(this);
    }
    public void actionPerformed(ActionEvent ae){
        output.setText(input.getText());
        input.setText("");
    }
}

<applet code="SimpleGraphicalApplet" width="200" height="300">
</applet>
```

Applet lifecycle

- ▶ As in the previous example, the applet constructor is a good place to define the graphical layout and interaction
 - ▶ Once the applet is constructed, it will react to user input
 - ▶ So nothing like a `main()` or `service()` method are needed unless you want to be able to start your application either as an applet *or* as a stand alone application
- ▶ `java.applet.Applet` also defines a number of methods to treat interaction with the browser
 - ▶ `init()`
 - ▶ Is called after the browser has downloaded the applet
 - ▶ `start()`
 - ▶ Called after `init()` and every time the user comes back to the applet page (if the applet hasn't been destroyed)
 - ▶ `stop()`
 - ▶ Called when the user leaves the applet page
 - ▶ `destroy()`
 - ▶ Called when the browser exits, or the applet terminates, etc.
 - ▶ It is entirely up to the browser when to call `destroy()`
 - ▶ `stop()` is always called before `destroy`

```
import java.awt.*;
import java.applet.*;
public class TheLifeOfAnApplet extends Applet {
    public void init() { trace("init"); }

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

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

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

    public void paint(Graphics g){ trace(g, "paint"); }

    private void trace(String s) {
        System.out.println(s);
        trace(getGraphics(), s); //retrieve the graphical context
    }
    private void trace(Graphics g, String s) {trace(g, s, 50, 20);}

    private void trace(Graphics g, String s, int x, int y) {
        g.drawString("***", x, y);
        g.drawString(s, x, y + 30);
        g.drawString("***", x, y + 60);
    }
}
```

The APPLET tag attributes

- ▶ **CODEBASE**
URL to the applet base if another than the current dir
- ▶ **ALT**
Text shown if the browser can't show applet
- ▶ **NAME**
An applet name, used for communicating from other applets in the same page
- ▶ **ALIGN, VSPACE, HSPACE, HEIGHT, WIDTH**
Placing in the page
- ▶ **ARCHIVE**
Comma-separated JAR files with applet code, resources, libraries needed, etc
- ▶ **OBJECT**
Refers to an already-instantiated applet saved in a file on the server

APPLET tag alternatives

- ▶ **APPLET** is deprecated in XHTML 1.0
It is still used and recognized by browsers
- ▶ **OBJECT** is used with Internet Explorer
It ensures that if Java is not installed in Explorer, the Java Plugin will be downloaded and installed at the first applet use
OBJECT is also understood by Mozilla in XHTML, but attributes are different
<https://eyeasme.com/Shayne/XHTML/appletObject.html>
- ▶ **EMBED** is used in Mozilla
The situation does not appear to be very "standard" at the moment
http://java.sun.com/j2se/1.5.0/docs/guide/plugin/developer_guide/using_tags.html
- ▶ In **JSP** you can look at the User-agent header to decide what kind of browser you serve to
JSP has a special action, `<jsp:plugin>`, which will generate correct code
<http://java.sun.com/products/jsp/syntax/2.0/syntaxref2023.html#1004158>

Applet parameters

HTML:

```
<applet code="SomeApplet.class" width="500" height="320">
  <param name="CourseName" value="Internet Programming" />
  <param name="CourseID" value="2D1335" />
  <param name="LectureNumber" value="7" />
</applet>
```

Java:

```
String course = getParameter("CourseName");
if (course == null) course = "A KTH course";
```

```
String lectno = getParameter("LectureNumber ");
int no = Integer.parseInt(lectno);
```

The applet context

- ▶ `java.applet.AppletContext getAppletContext()`
Represents basically the browser in which the applet runs. Capabilities:
 - ▶ get another applet from the same page in order to call its methods:
`getApplet(String name)`
 - ▶ enumerate all other applets in the page: `getApplets()`
- ▶ Read audio clips and images from the net and give them to the applet:
`getAudioClip(URL), getImage(URL)`
- ▶ Retrieve and show a URL in this browser frame or another:
`showDocument(URL, String frame)`
- ▶ Retrieve and save information to communicate with applets from this page or other pages ("applet persistence")
 - ▶ `void setStream(String key, InputStream stream)`
 - ▶ `InputStream getStream(String key)`

Applet-Applet and JavaScript communication

- ▶ Adding the info of all applets in the page to a `java.awt.TextArea` called 'text'

```
Enumeration e = getAppletContext().getApplets();
while(e.hasMoreElements()) {
    text.append("\n" + ((Applet) e.nextElement()).getAppletInfo());
}
```

- ▶ Calling a method of another applet defined as

```
<applet codebase="." code="examples.Applet3.class"
        name="Paint" width="400" height="300" />

Applet other = getAppletContext().getApplet("Paint");
if(other != null) {
    ((Applet3) other).setInfo(new Date().toString());
}
```

- ▶ The same from Javascript

```
var paintapplet = document.applet.Paint;
paintapplet.setInfo("Hello");
```

Accessing the HTML document through DOM

- ▶ Since Java 1.4 an applet can examine and modify the HTML document just like JavaScript can

- ▶ DOM = Document Object Model, <http://www.w3.org/DOM/>

- ▶ `org.w3c.dom`, `org.w3c.dom.html`

- ▶ Retrieving the document object:

```
com.sun.browser.dom.DOMService
http://java.sun.com/j2se/1.5.0/docs/guide/plugin/developer_guide/
    java_js.html#common_dom
```

- ▶ Manipulating the object

```
org.w3c.dom.html.HTMLDocument
```

Applet Security Restrictions

- ▶ In principle, all `java.*` packages are accessible to applets
- ▶ Also you can use ARCHIVE to add other code
- ▶ But that doesn't mean that applets have all the power of Java.

For example

- ▶ They can't open TCP connections (sockets) to any other host than the host they are downloaded from
- ▶ Can't read or write files (Though reading files via file URLs is possible)
- ▶ Can't start programs, or load native code
- ▶ Can't access certain System properties
- ▶ `java.awt.Window` objects made by applets look different, to warn the user
- ▶ If the applet was loaded through the `file://` protocol, security restrictions don't apply
- ▶ If the applet is signed, the browser (or java plug-in) should prompt the user that the applet requires permission for one of the above operations. How to sign applets:

```
http://java.sun.com/j2se/1.5.0/docs/guide/plugin/
    developer_guide/rsa_signing.html
```

Applet-servlet communication

- ▶ We can use a URL to connect to the HTTP server that the applet comes from
 - ▶ We can connect to any server on that host
- ▶ There we can e.g. invoke a servlet (or even a CGI for that matter), that can do something useful for our applet

```
URL page = getCodeBase();
String protocol = page.getProtocol();
String host = page.getHost();
int port = page.getPort();
String servlet = "/servlet/SomeServlet";
URL dataUrl = new URL(protocol, host, port, servlet);
```

POST using URLConnection

```

URL dataURL = new URL(protocol, host, port, servlet);
URLConnection conn = dataURL.openConnection();
conn.setUseCaches(false);
conn.setDoOutput(true);
String query = "firstName=" + URLEncoder.encode(firstName) +
    "&lastName=" + URLEncoder.encode(lastName) +
    "&emailAddress=" + URLEncoder.encode(emailAddress);
// we have to write the content length
conn.setRequestProperty("Content-Length",
    String.valueOf(query.length()));
conn.setRequestProperty("Content-Type",
    "application/x-www-form-urlencoded");
// finished headers, now write the POST content
conn.getOutputStream().write(query.getBytes());
// start reading
BufferedReader in =
    new BufferedReader(new InputStreamReader(conn.getInputStream()));

```

Audio/Media in Java

- ▶ Applets have direct support to play sound

```

AudioClip audioClip = getAudioClip(baseUrl, relativeURL);
audioClip.loop();

```

When you want it to stop playing (e.g. in the `stop()` method)

```

audioClip.stop();

```

- ▶ To use this functionality outside applets

```

AudioClip audioClip = Applet.newAudioClip(completeURL);

```

- ▶ Details:

<http://java.sun.com/docs/books/tutorial/sound/index.html>
<http://java.sun.com/javase/6/docs/technotes/guides/sound/>

- ▶ The latest in sound and video is the Java Media Framework

<http://java.sun.com/javase/technologies/desktop/media/jmf/>