Java Avanzado

Java Applets

Copyright

¿Qué es un Applet Java?

- Es un tipo de aplicación Java que se inserta dentro de páginas HTML. Cuando estas se descargan, los Applets se ejecutan en el navegador.
- Los Applets Java se cargan de la siguiente forma:
  - Se escribe una URL en el navegador.
  - El navegador carga la página HTML.
  - El navegador carga las clases del Applet Java.
  - Se ejecuta el Applet Java.

Restricciones de seguridad

- Los Applets Java, a diferencia de las aplicaciones Java no se ejecutan siempre con el conocimiento del usuario. Podemos estar navegando y cargar un Applet Java sin saberlo.
- Por ello, sufren una serie de restricciones de seguridad no existentes en las aplicaciones Java convencionales.
- Los navegadores implementan un entorno de ejecución seguro habitualmente conocido con el nombre de SandBox.
- Existen técnicas de firma digital para poder evitar estas restricciones. (http://java.sun.com/docs/books/tutorial/security/index.html)

Restricciones de seguridad

- Algunas restricciones son:
  - Realizar llamadas llamadas a programas externos.
  - Realizar operaciones de entrada/salida.
  - Realizar llamadas a métodos nativos (JNI).
  - Abrir conexiones con servidores remotos distintos al origen del propio Applet Java.
- De esta manera, un Applet Java jamás podrá dañar o robar información del sistema.

Las clases Applet y JApplet

- La clase Applet es un contenedor visual del estilo de los Frame y Panel o los JFrame y JPanel.
- Existen dos implementaciones: AWT y Swing
- AWT:
  - Se encuentra en el paquete java.applet.*
  - Hereda de java.awt.Panel
  - Se le aplican LayoutManagers y se le añaden componentes como ya hiciéramos con los contenedores estándar AWT. Ocurre lo mismo con la gestión de eventos.

Las clases Applet y JApplet

- Swing:
  - Se encuentra en el paquete javax.swing.*
  - Hereda de java.applet.Applet
  - Se le aplican LayoutManagers y se le añaden componentes como ya hiciéramos con los contenedores estándar Swing. Ocurre lo mismo con la gestión de eventos.

Ciclo de vida

- A diferencia de las aplicaciones Java convencionales, los Applets Java no tienen un método: *public static void main(String[] args)* para ejecutarse.
- Por el contrario tienen estos cinco métodos:
  - **public void** init();
  - **public void** start();
  - **public void** stop();
  - **public void** destroy();
  - **public void** paint(Graphics g);

Ciclo de vida

- **public void** init();
  - Es el primer método que se ejecuta una vez que se ha llamado al constructor del applet.
  - Solo se ejecuta una vez en la vida del applet.
  - Se suele usar para la inicialización del applet.
- **public void** start();
  - Se ejecuta una vez haya terminado el método init().
  - A diferencia del método init(), start() se ejecuta cada vez que haya que arrancar el applet (recarga, maximizar,...).
  - Se suele usar para arrancar procesos (ej: Threads).

Ciclo de vida

- **public void** stop();
  - Se ejecuta cada vez que haya que parar el applet (reload, minimizar, cargar otra página,.....).
  - Se suele usar para parar procesos (ej: Threads).
- **public void** destroy();
  - Se ejecuta al cerrar el Navegador o cuando este decide eliminarle (ej: purgado de caché).
  - Solo se ejecuta una vez en la vida del applet.
  - Se suele usar para liberar cualquier recurso utilizado.

Ciclo de vida

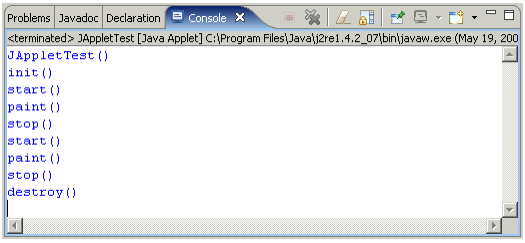
- **public void** paint(Graphics g);
  - Recibe una instancia de la clase java.awt.Graphics
  - Esta instancia se puede utilizar para escribir o pintar en el applet.
  - Se ejecuta cada vez que se necesita refrescar el Applet (ej: recarga, maximizar,....).

Ejemplo del Ciclo de vida

- Carga del Applet:
  - Se ejecuta el constructor por defecto (sin parámetros) del applet.
  - Se ejecuta el método init().
  - Se ejecuta el método start().
- Salida y vuelta a entrar en la página HTML:
  - Se ejecuta el método stop() al salir.
  - Se ejecuta el método start() al volver.
- Cierre del navegador:
  - Se ejecuta el método stop().
  - Se ejecuta el método destroy().

Ejemplo del Ciclo de Vida

```
import java.awt.Graphics;
import javax.swing.JApplet;

public class JAppletTest extends JApplet
{
   public JAppletTest() { System.out.println("JAppletTest()"); }
   public void init() { System.out.println("init()"); }
   public void start() { System.out.println("start()"); }
   public void paint(Graphics g) { System.out.println("paint()"); }
   public void stop() { System.out.println("stop()"); }
   public void destroy() { System.out.println("destroy()"); }
}
```



HTML

- HTML – HiperText Markup Language
- Es un lenguaje simple utilizado para crear documentos Web de extensión *.html o *.htm
- Se limita a describir la estructura y el contenido de un documento.
- Actualmente se encuentra en su versión 4.01
- Es un lenguaje basado en etiquetas. Cada etiqueta se escribe entre los caracteres: < >.
- Las etiquetas pueden tener atributos.

Estructura básica

- Las etiquetas suelen ir formando bloques: <> y </>.
- Esqueleto básico:

```
<HTML>
  <HEAD>
    <TITLE>Mi primer HTML</TITLE>
  </HEAD>
  <BODY>

  </BODY>
</HTML>
```

La etiqueta <applet>

- Sintaxis:

```
<applet [name=?] [codebase=?] code=? [archive=?] width=? height=?
        [align=?] [vspace=?] [hspace=?] [alt=?]>
   [<param name=? value=?]>
   .....
</applet>
```

- Descripción:
  - name: especifica un nombre al applet.
  - codebase: especifica la URL por defecto del applet en caso de que sea distinta a la del HTML.
  - code: especifica el nombre de la clase principal del applet, incluyendo el paquete y la extensión (edu.upco.einf.MiApplet.class)

La etiqueta <applet>

- Descripción (cont.):
  - archive: especifica los archivos JAR o ZIP que contengan clases necesarias y que deban ser precargados por el navegador.
  - width: especifica el ancho del applet en pixels.
  - height: especifica el alto del applet en pixels.
  - align: especifica el alineamiento del applet respecto del texto. Algunos valores válidos son: left, right, top, middle....
  - vspace: especifica el espacio a dejar por encima y por debajo del applet en pixels.
  - hspace: especifica el espacio a dejar por derecha e izquierda del applet en pixels.

La etiqueta <applet>

- Descripción (cont.):
  - alt: especifica el texto a mostrar en caso de que el navegador no sea capaz de mostrar el applet.
  - <param>: es la manera de pasar parámetros al applet desde el exterior. Esta etiqueta tiene dos atributos: *name* para el nombre del parámetro y *value* para su valor.
- Los applets cuentan con dos métodos para acceder a los parámetros (no usar en el constructor):
  - **public** String getParameter(String name);
  - **public** String[][] getParameterInfo();

Ejemplo

```
<HTML>
 <HEAD>
  <TITLE>Mi primer HTML</TITLE>
 </HEAD>
 <BODY>

  <APPLET name="Test1" codebase="." code="edu.upco.einf.MiApplet.class"
          archive="miApplet.jar" width="200" height="300" align="middle"
          vspace="5" hspace="5" alt="No tiene soporte de Applets Java">
   <PARAM name="param1" value="value1">
   <PARAM name="parma2" value="value2">
  </APPLET>

  <APPLET code= "edu.upco.einf.OtroApplet.class" width="200" height="300">
  </APPLET>

 </BODY>
</HTML>
```

Probando los Applets

- El SDK (o JDK) contiene una herramienta para probar los Applets Java sin necesidad de un navegador.
- Esta herramienta se llama appletviewer.exe
- Carga un fichero HTML como argumento:

  appletviewer.exe HelloWorld.html

- Ese fichero HTML requiere como mínimo:

```
<HTML>
  <APPLET code= "HelloWorld.class" width="200" height="300">
  </APPLET>
</HTML>
```

Ejemplo

```
import java.applet.Applet;
import java.awt.Graphics;

public class HelloWorld extends Applet
{
  public void paint(Graphics g)
  {
    g.drawString("Hello World!",10,10);
  }
}
```

```
<HTML>
  <APPLET code="HelloWorld.class"
          width="100" height="50">
  </APPLET>
</HTML>
```

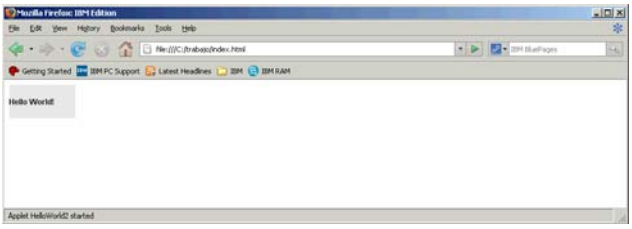
Probando los Applets

- No obstante podemos probar directamente en un navegador abriendo en local la página HTML:
  - file:///c:/test/index.html
- O desplegando la página HTML y el Applet Java a un Servidor Web:
  - http://www.upco.es/test/index.html

Ejemplo

```
import javax.swing.JApplet;
import javax.swing.JLabel;

public class HelloWorld2 extends JApplet
{
  public void init()
  {
    JLabel l = new JLabel("Hello World!");
    this.getContentPane().add(l);
  }
}
```

```
<HTML>
  <APPLET code="HelloWorld2.class"
          width="100" height="50">
  </APPLET>
</HTML>
```



Otras posibilidades del API

- public AudioClip getAudioClip(URL url, String clip):

```
import java.applet.Applet;
import java.applet.AudioClip;

public class AudioClipTest extends Applet
{
  AudioClip clip = null;

  public void init()
  {
    clip = this.getAudioClip(this.getDocumentBase(),“audio/test.au");
  }

  public void start() { clip.play(); }

  public void stop() { clip.stop(); }
}
```

Otras posibilidades del API

- public Image getImage(URL url, String img):

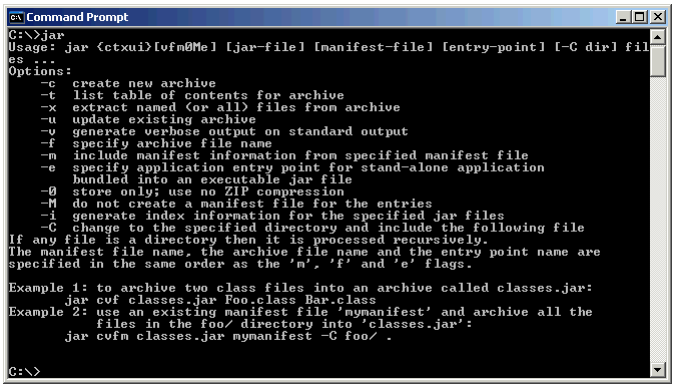
```
import java.applet.Applet;
import java.awt.Graphics;
import java.awt.Image;

public class ImageTest extends Applet
{
  Image img = null;

  public void init()
  {
    img = this.getImage(this.getDocumentBase(),"images/test.gif");
  }
  public void paint(Graphics g)
  {
    g.drawImage(img, 10,20, this);
  }
}
```

El fichero JAR

- Es un formato de fichero.
- Empaqueta comprimidos en un único fichero clases Java y otro tipo de recursos (imágenes, sonidos,....).
- El SDK (o JDK) contiene una herramienta para crear, listar o expandir archivos JAR.
- Sintaxis:

jar.exe {ctxu} [vfm0M] [fichero-jar] [fichero-manifest] [-C dir] ficheros....

El fichero JAR



Seguridad

- Ya hemos comentado, que un Applet se ejecuta en un entorno restringido, de forma que no tiene acceso a los recursos de la máquina donde se ejecuta.
- Esto es debido a que la ejecución de los Applets se realiza sin el conocimiento y aprobación de los usuarios.
- Pero existen los certificados y firmas digitales para evitar estas restricciones.

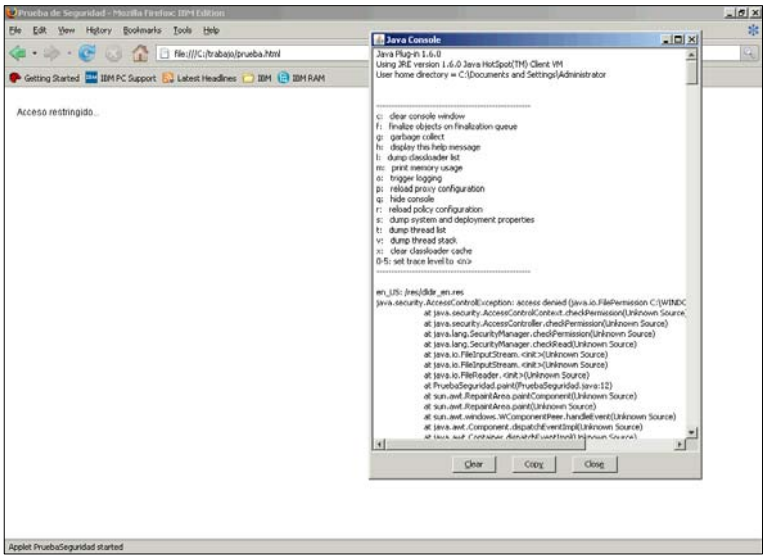
Ejemplo

```
import java.applet.Applet;
import java.awt.Graphics;
import java.io.*;

public class PruebaSeguridad extends Applet
{
  public void paint(Graphics g)
  {
    BufferedReader br = null;
    try
    {
      br = new BufferedReader(new FileReader("C:\\WINDOWS\\system32\\drivers\\etc\\hosts"));
      StringBuffer buffer = new StringBuffer();
      String temp = null;
      while((temp = br.readLine()) != null)
        buffer.append(temp);
      g.drawString("Lectura realizada...",10,30);
      br.close();
    }
    catch(IOException ex)
    {
      g.drawString("Problemas de I/O...",10,30);
    }
    catch(SecurityException ex)
    {
      ex.printStackTrace();
      g.drawString("Acceso restringido...",10,30);
    }
  }
}
```

```
<HTML>
  <HEAD>
    <TITLE>Prueba de Seguridad</TITLE>
  </HEAD>
  <BODY>
    <APPLET code="PruebaSeguridad.class"
                archive="prueba.jar" width="300" height="50">
    </APPLET>
  </BODY>
</HTML>
```

Nota: Se trata de un Applet que intenta conocer otras máquinas de nuestra red accediendo al fichero c:\windows\system32\drivers\etc\hosts

Ejemplo



Firma digital

- La idea básica de la firma digital es:
  - Generamos un par de claves: pública y privada.
  - Firmamos el código con la clave privada.
    Nota: firmar significa aplicar un algoritmo hash sobre el código y el resultado, cifrarlo con la clave privada, obteniendo así la firma digital.
  - Enviamos el código firmado junto con la clave pública.
  - En la recepción, se aplica el mismo algoritmo hash sobre el código y se compara el resultado con la firma digital (después de haberla descifrado, esta vez, con la clave pública).
  - Normalmente, la clave pública se envía en un certificado digital para validar la identidad del firmante.

Firma digital

- Para firmar digitalmente código Java, hay que seguir los siguientes pasos:
  - Crear ficheros JAR con todo el código compilado.
  - Generar un par de claves (pública y privada) así como un certificado digital con la clave pública.
  - Firmar digitalmente los ficheros JAR con la clave privada.
- En la JDK tenemos todo lo necesario para llevar a cabo estos pasos.

Generar un par de claves

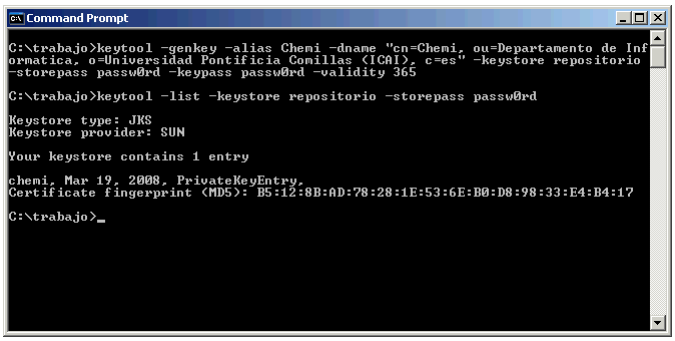
- El JDK incluye una herramienta para este fin:
  - keytool.exe
- Usaremos los siguientes parámetros:
  - -genkey: comando para generar las claves.
  - -dname: con el *distinguished name* del propietario de las claves.
  - -alias: con el identificador a usar para referirse a las claves.
  - -keystore: con el repositorio (es un fichero) donde se va a guardar toda la información generada.
  - -storepass: con la contraseña de acceso al repositorio.

Generar un par de claves

- -keypass: con la contraseña para utilizar las claves.
- -validity: con el número de días de validez del certificado.
- Esta herramienta admite otros comandos como por ejemplo:
  - -list: lista el contenido de un repositorio.
  - -delete: borra una entrada de un repositorio.
  - -export: exporta un certificado.
  - -import: importa un certificado.
  - -help: muestra la ayuda (listando todos los comandos).

Generar un par de claves

- Ejemplo:



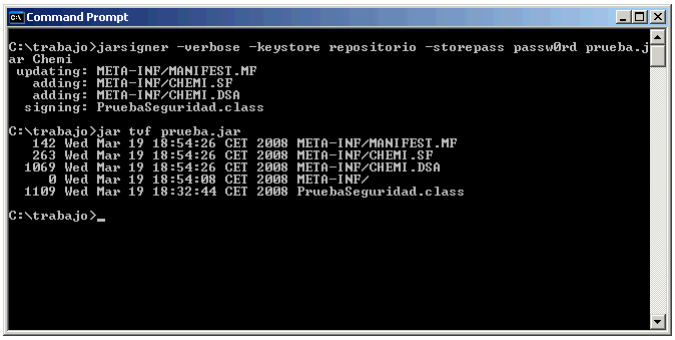
- Como resultado, se habrá creado un fichero llamado repositorio con toda la información: un par de claves pública y privada, y un certificado digital.

Firmar digitalmente un JAR

- El JDK incluye una herramienta para este fin:
  - jarsigner.exe
- Usaremos los siguientes parámetros:
  - -verbose: para que muestre información de lo que va haciendo por pantalla.
  - -keystore: con el repositorio donde está la clave privada.
  - -storepass: con la contraseña de acceso al repositorio.
  - %fichero%: fichero a firmar.
  - %alias%: identificador de la clave a utilizar.

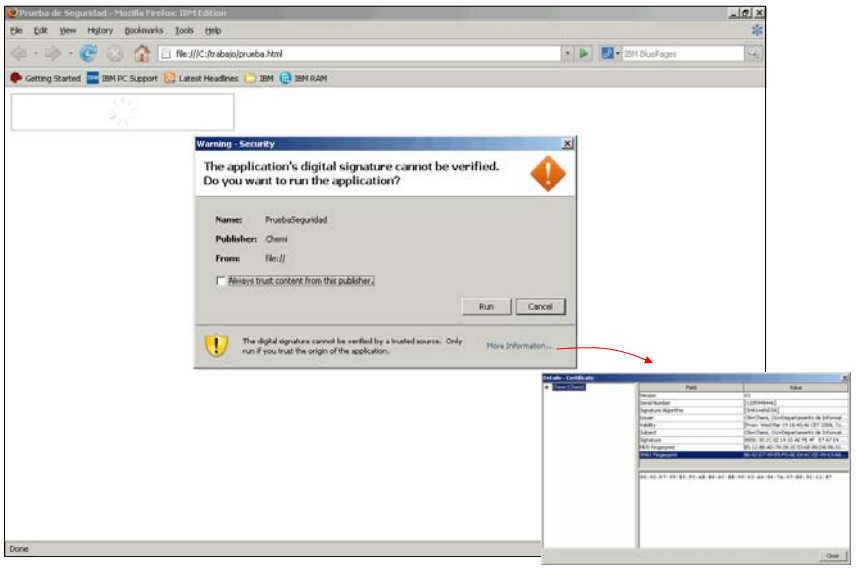
Firmar digitalmente un JAR

- Ejemplo:



- La herramienta ha añadido el fichero de la firma (*.SF) y el fichero de bloques de firmas (*.DSA).

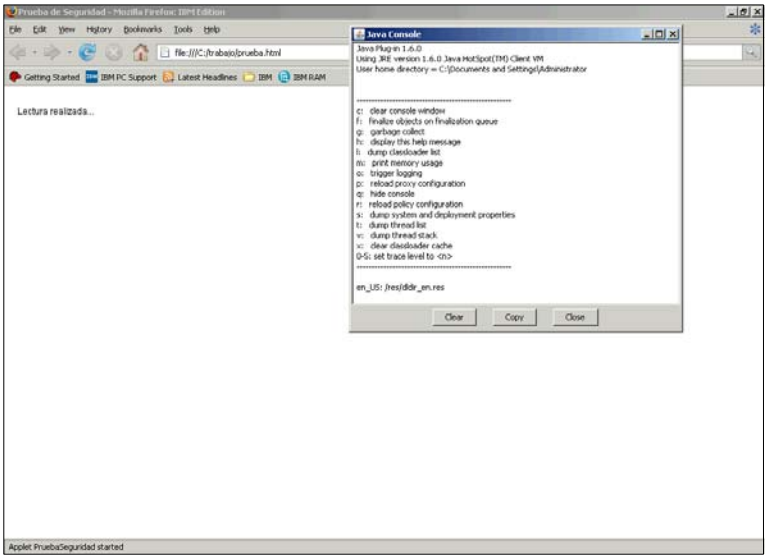
Firmar digitalmente un JAR



Firmar digitalmente un JAR

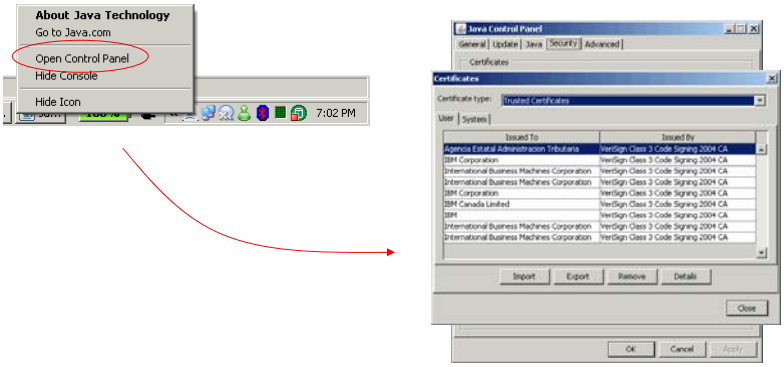
- Esta vez el navegador detecta que el Applet que intenta acceder a los recursos del sistema está firmado.
- Nos muestra información sobre el firmante:
  - El nombre del firmante.
  - La Autoridad Certificadora que verifica la identidad del firmante.
  - Si se confía en dicha Autoridad Certificadora o no.
  - Si el certificado es válido y no ha caducado.
- Y nos pregunta si confiamos en el firmante y le damos permiso de acceso.

Firmar digitalmente un JAR



Certificados digitales

- ¿Podemos evitar que nos pregunte y le de permiso de acceso por defecto?
- Si, pero para ello, tenemos que añadir el certificado del firmante a la lista de certificados en los que confiamos.



Certificados digitales

- Un certificado digital está formado por:
  - La clave pública del firmante.
  - El *distinguised name* del firmante: nombre, unidad organizativa, organización, ciudad, provincia y país.
  - La firma digital del emisor del certificado (denominado Autoridad Certificadora).
  - El *distinguised name* del emisor del certificado.
- Una Autoridad Certificadora, es una entidad que certifica la identidad de un firmante.
  - Algunos ejemplos: Equifax, FNMT, VeriSign, Thawte, etc...

Certificados digitales

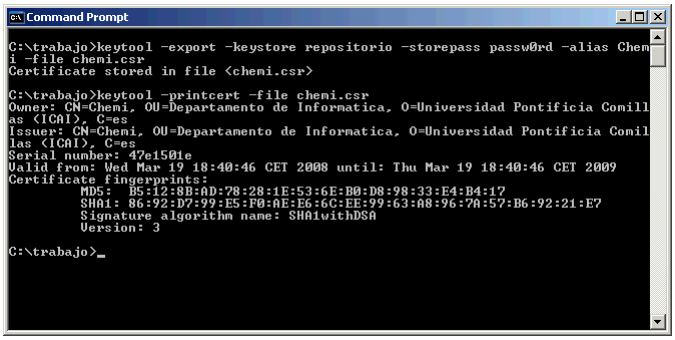
- Conseguir un certificado digital cuesta dinero.
- Por ello, normalmente para realizar pruebas creamos nuestros propios certificados:
  - Certificados auto-firmados, es decir, el propio firmante certifica que es quien dice ser.
- En España, la Fabrica Nacional de Moneda y Timbre (FNMT) emite gratuitamente certificados a todos los españoles con DNI.
  - http://www.cert.fnmt.es/index.php?cha=cit&sec=obtain_cert
- Nota: la firma digital tiene validez legal, por lo que su uso debe ser cuidadoso y responsable.

Certificados digitales

- Para exportar nuestro certificado, debemos utilizar de nuevo la herramienta:
  - keytool.exe
- Con los siguientes parámetros:
  - -export: comando para exportar el certificado.
  - -keystore: con el nombre del repositorio donde está el certificado.
  - -storepass: con la contraseña de acceso al repositorio.
  - -alias: con el identificador de las claves/certificado.
  - -file: con el nombre del fichero donde guardar el certificado exportado.

Certificados digitales

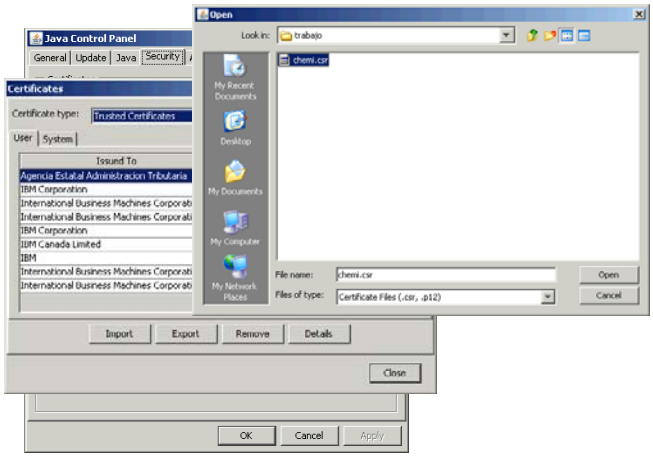
- Ejemplo:



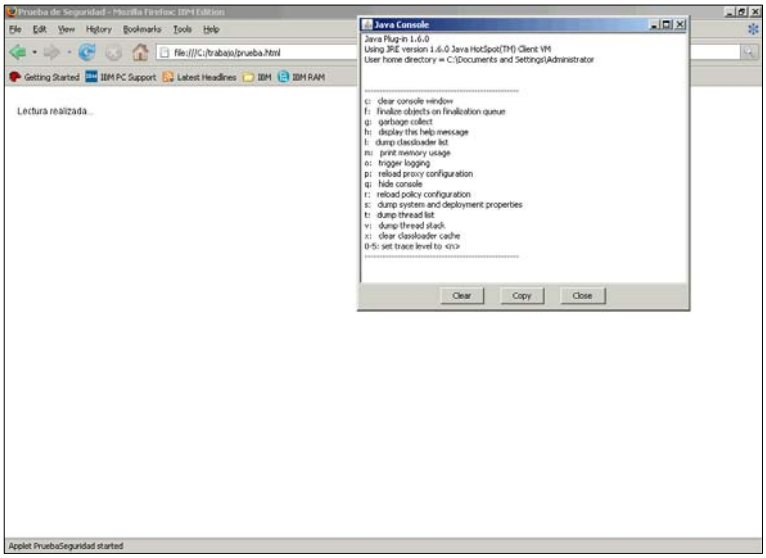
- Con el comando –printcert podemos ver los detalles del certificado.

Certificados digitales

- Por último, utilizando la consola del Java Plug-in importamos el certificado:

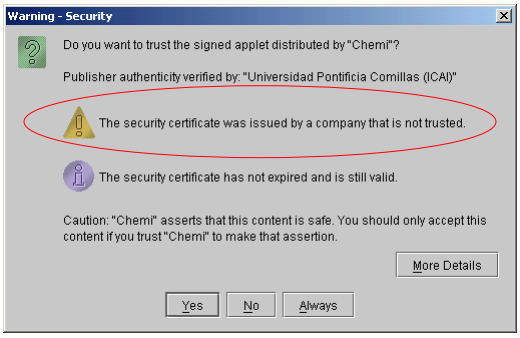


Certificados digitales



Certificados digitales

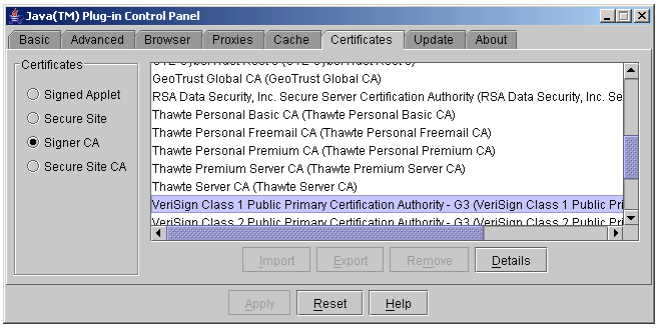
- ¿Cómo hacemos para que el Java Plug-in confíe en nosotros como emisores del certificado?



- El problema es que se trataba de un certificado auto-firmado.
- Esto ya no es necesario a partir de Java SE 5.0

Certificados digitales

- Para solventarlo podemos:
- Obtener el certificado de una Autoridad Certificadora conocida por el Java Plug-in y usarlo en vez del nuestro.
- Añadir nuestro certificado auto-firmado a la lista de Autoridades Certificadoras en las que confía.

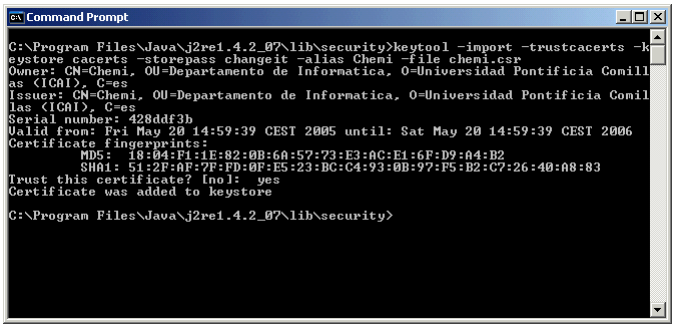


Certificados digitales

- Todos los certificados de las Autoridades Certificadoras en las que confía el Java Plug-in se encuentran en el repositorio (keystore):
- C:\Program Files\Java\jre1.6.0\lib\security\cacerts
- Una vez mas, usaremos la herramienta keytool para importarlo:
- -import: comando para importar el certificado.
- -trustcacerts: indicando que se trata el certificado de una Autoridad Certificadora.
- -keystore y –storepass: indicando el respositorio.
- -alias: indicando el identificador para referirse al certificado.
- -file: indicando el fichero con el certificado.

Certificados digitales

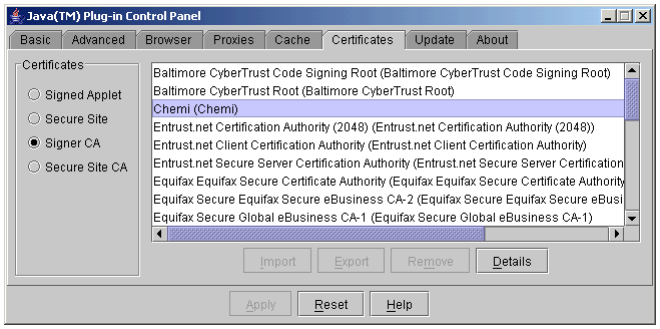
- Ejemplo:



- La contraseña del repositorio “cacerts” es: changeit.
- Como la inserción de una nueva Autoridad Certificadora es crítica, el proceso pide confirmación.

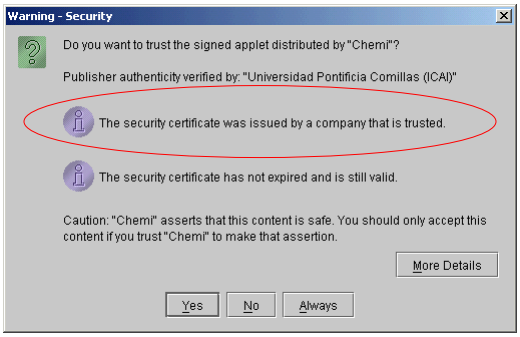
Certificados digitales

- Lista de Autoridades Certificadoras después de la importación:



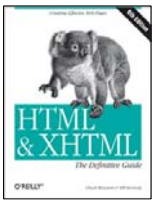
Certificados digitales

- El Java Plug-in ya confía en nosotros, como emisores del certificado:

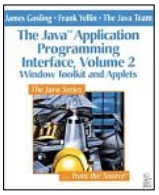


Bibliografía

- HTML & XHTML (6th edition).
  Chuck Musciano y Bill Kennedy.
  O'Reilly.



- Window Toolkit and Applets, Volume 2
  James Gosling y Frank Yellin.
  Addison-Wesley.



- Developing professional Java Applets
  K.C.Hopson, Stephen E. Ingram y Patrick Chan.
  Sams Publishing.

- The Java tutorial (on-line)
  http://java.sun.com/docs/books/tutorial/deployment/applet/