**JAVA Web Start**

**introduzione**

Lo sviluppo di applicazioni per la distribuzione con Java Web Start è generalmente lo stesso dello sviluppo di applicazioni autonome per Java (TM) Platform Standard Edition. Ad esempio, il punto di ingresso per l'applicazione è il vuoto statico pubblico standard principale (String [] argv) .

Tuttavia, al fine di supportare la distribuzione Web (download e avvio automatici di un'applicazione) e garantire che un'applicazione possa essere eseguita in un sandbox sicuro, ci sono alcune considerazioni aggiuntive:

* Un'applicazione deve essere consegnata come un insieme di file JAR.
* Tutte le risorse dell'applicazione, come file e immagini, devono essere archiviate in file JAR; e devono essere referenziati usando il meccanismo getResource in Java (TM) Platform Standard Edition (vedi sotto).
* Se un'applicazione è scritta per essere eseguita in un sandbox sicuro, deve seguire queste restrizioni:
  + Nessun accesso al disco locale.
  + Tutti i file JAR devono essere scaricati dallo stesso host.
  + Le connessioni di rete sono abilitate solo per l'host da cui vengono scaricati i file JAR.
  + Nessun gestore della sicurezza può essere installato.
  + Non è possibile utilizzare librerie native.
  + Accesso limitato alle proprietà del sistema. L'applicazione ha accesso in lettura / scrittura a tutte le proprietà di sistema definite nel file JNLP, nonché l'accesso in sola lettura allo stesso set di proprietà a cui un Applet ha accesso.

Alcune di queste restrizioni possono essere superate utilizzando l'API JNLP per accedere al file system e ad altre risorse di sistema.

* Un'applicazione può utilizzare la chiamata System.exit .
* Un'applicazione che necessita di accesso illimitato al sistema dovrà essere consegnata in un set di file JAR firmati. Tutte le voci in ciascun file JAR devono essere firmate.

**Recupero di risorse dai file JAR**

Java Web Start trasferisce solo i file JAR dal server Web al computer client. Determina dove archiviare i file JAR sul computer locale. Pertanto, un'applicazione non può utilizzare riferimenti relativi al disco a risorse come immagini e file di configurazione.

Tutte le risorse dell'applicazione devono essere recuperate dai file JAR specificati nella sezione risorse del file JNLP o recuperate esplicitamente utilizzando una richiesta HTTP al server Web. Si consiglia di archiviare risorse nei file JAR, poiché verranno memorizzati nella cache sul computer locale da Java Web Start.

L'esempio di codice seguente mostra come recuperare immagini da un file JAR:

// Ottieni il classloader corrente

ClassLoader cl = **this** .getClass (). GetClassLoader ();

// Crea icone

Icona saveIcon = **new** ImageIcon (cl.getResource ("images / save.gif"));

Icona cutIcon = **new** ImageIcon (cl.getResource ("images / cut.gif"));

...

L'esempio presuppone che le seguenti voci siano presenti in uno dei file JAR per l'applicazione:

images / save.gif

images / cut.gif

**Accesso al client mediante l'API JNLP**

L'API JNLP può essere utilizzata per accedere al file system del client e ad altre risorse. Vedere i seguenti argomenti per ulteriori informazioni sull'uso dell'API JNLPI per accedere al client:

* [Esempi di API JNLP](https://docs.oracle.com/javase/8/docs/technotes/guides/javaws/developersguide/examples.html)
* [Accesso al client mediante l'API JNLP](http://docs.oracle.com/javase/tutorial/deployment/doingMoreWithRIA/usingJNLPAPI.html) (Argomento della [traccia](http://docs.oracle.com/javase/tutorial/deployment/index.html) di [distribuzione](http://docs.oracle.com/javase/tutorial/deployment/index.html) dei tutorial Java. Sebbene l'esempio mostrato sia un'applet, le informazioni si applicano anche alle applicazioni Java Web Start.)

**Sicurezza e firma del codice**

Java Web Start risolve i problemi di sicurezza:

* Protezione degli utenti da codice dannoso (intenzionale e non intenzionale) che può influire sui file locali;
* Proteggere le aziende dal codice che potrebbe tentare di accedere o distruggere i dati sulle reti.

Le applicazioni avviate con Java Web Start sono, per impostazione predefinita, eseguite in un ambiente limitato in cui hanno accesso limitato alle risorse informatiche locali, come i dispositivi di archiviazione e la rete locale. In questo ambiente sandbox, Java Web Start può garantire che un'applicazione scaricata e potenzialmente non affidabile non possa compromettere la sicurezza dei file locali o della rete.

Un'ulteriore funzione di sicurezza supportata da Java Web Start è la firma digitale del codice. Se un'applicazione invocata viene recapitata in uno o più file JAR firmati, Java Web Start verificherà che i contenuti del file JAR non siano stati modificati da quando sono stati firmati. Se la verifica di una firma digitale non riesce, Java Web Start non eseguirà l'applicazione, poiché potrebbe essere stata compromessa da una terza parte.

Il supporto per la firma del codice è importante sia per gli utenti che per i fornitori di servizi applicativi. Questo servizio consente agli utenti di verificare che un'applicazione provenga da una fonte attendibile. Poiché il fornitore di servizi applicativi firma il codice, entrambi possono essere certi che nessun'altra parte può rappresentare l'applicazione sul Web. Un'applicazione firmata considerata attendibile dall'utente può anche richiedere privilegi di sistema aggiuntivi, come l'accesso a un disco locale.

Java Web Start presenta una finestra di dialogo che mostra l'origine dell'applicazione, in base al certificato del firmatario, prima dell'avvio dell'applicazione. Ciò consente all'utente di prendere una decisione informata sull'opportunità o meno di concedere ulteriori privilegi al codice scaricato.

Includendo le seguenti impostazioni nel file JNLP, un'applicazione può richiedere l'accesso completo a un sistema client se tutti i suoi file JAR sono firmati:

< **sicurezza** >

< **tutte le autorizzazioni** />

</ **sicurezza** >

L'implementazione della firma del codice in Java Web Start si basa sull'API di sicurezza nella piattaforma Java (TM) Platform Standard Edition. Java 2 SE JRE 1.4.2 supporta la firma del codice con gli algoritmi SHA1withDSA e MD5withRSA.

Gli sviluppatori firmano il codice per l'utilizzo con Java Web Start allo stesso modo delle applet Java, utilizzando lo strumento jarsigner standard dalla piattaforma Java (TM) Standard Edition. La documentazione dello strumento [jarsigner](https://docs.oracle.com/javase/8/docs/technotes/tools/index.html" \l "security) fornisce esempi su come firmare il codice e creare certificati di prova e discute di altri problemi relativi alla firma.

**Firma dei file JAR con un certificato di prova**

Ecco i passaggi necessari per firmare un file JAR con un certificato di prova:

Si noti che un certificato di prova autofirmato deve essere utilizzato solo per i test interni, poiché non garantisce l'identità dell'utente e pertanto non può essere considerato attendibile. Un certificato degno di fiducia può essere ottenuto da un'autorità di certificazione, come [VeriSign](http://www.verisign.com/) o [Thawte](http://www.thawte.com/) , e deve essere utilizzato quando l'applicazione viene messa in produzione.

**Come codificare i file JNLP**

A partire da Java Web Start versione 1.2, i file JNLP possono essere codificati in qualsiasi codifica dei caratteri supportata da Java (TM) Platform Standard Edition. (Consultare la documentazione [Java (TM) Platform Standard Edition](https://docs.oracle.com/javase/8/docs/technotes/guides/intl/encoding.doc.html) per un elenco di codifiche supportate.)

Per codificare un file JNLP, specificare una codifica nel prologo XML di quel file. Ad esempio, la seguente riga indica che il file JNLP verrà codificato in UTF-16.

<? xml version = "1.0" encoding = "utf-16"?>

Il prologo XML stesso deve essere codificato UTF-8.

**Download dinamico di certificati HTTPS**

A partire dalla 1.4.2, Java Web Start importa dinamicamente i certificati più o meno allo stesso modo dei browser. Per far funzionare tutto ciò, Java Web Start ora imposta il proprio gestore https, utilizzando le java.protocol.handler.pkgsproprietà di sistema, per inizializzare i valori predefiniti per SSLSocketFactorye HostnameVerifier. Imposta i valori predefiniti con HttpsURLConnection.setDefaultSSLSocketFactorye HttpsURLConnection.setDefaultHostnameVerifier.

Se l'applicazione utilizza questi due metodi, assicurarsi che vengano chiamati dopo l'inizializzazione del gestore https di Java Web Start, altrimenti il ​​gestore personalizzato verrà sostituito dal gestore predefinito di Java Web Start. Puoi assicurarti che tu sia personalizzato SSLSocketFactorye che HostnameVerifiter venga utilizzato effettuando una delle seguenti operazioni:

1. Installazione del proprio gestore https, che sostituirà completamente il gestore https di Java Web Start;
2. Chiamata HttpsURLConnection.setDefaultSSLSocketFactoryo HttpsURLConnection.setDefaultHostnameVerifiersolo dopo la creazione del primo oggetto url https, che eseguirà prima il codice di inizializzazione del gestore https di Java Web Start.

Per informazioni sulla creazione di un servlet di download, consultare il capitolo successivo, [JnlpDownloadServlet Guide](https://docs.oracle.com/javase/8/docs/technotes/guides/javaws/developersguide/downloadservletguide.html) .

**MappaSonoraAvo**

**Come utilizzarla con Java Web Start**

**Azioni da compiere per utilizzare Java Web Start**

Sul codice java per accedere ai file:

**nel file Aule.java**

String csvFile = "dati/DatiAule.csv";

String line = "";

String cvsSplitBy = ",";

ClassLoader classloader = Thread.currentThread().getContextClassLoader();

InputStream inputStream = classloader.getResourceAsStream(csvFile);

InputStreamReader streamReader = new InputStreamReader(inputStream);

try (BufferedReader br = new BufferedReader(streamReader)) {

line = br.readLine(); // skip della prima riga con le intestazioni

….

**nel file PianoPanel.java**

ClassLoader classloader = Thread.currentThread().getContextClassLoader();

InputStream is = classloader.getResourceAsStream(strFile);

try

{

BufferedImage img;

img=ImageIO.read(is);

.**Nel file AulaPanel.java (PaintComponent)**

/\*

\*\*\* disegna l'immagine di background con la planimetria dell'aula scelta

\*/

String strFilePlan = "plan/" + this.myAula.getNome()+ ".jpg";

System.out.println("leggo il file "+ strFilePlan);

ClassLoader classloader = Thread.currentThread().getContextClassLoader();

InputStream filePlan = classloader.getResourceAsStream(strFilePlan);

/\*

\*\*\* disegna l'immagine di background con la immagine dell'aula scelta

\*/

String strFileImg = "img/" + this.myAula.getNome()+ ".jpg";

System.out.println("leggo il file "+ strFileImg);

InputStream fileImg = classloader.getResourceAsStream(strFileImg);

/\*

\*\*\* disegna l'immagine di background con la immagine dell'aula scelta

\*/

String strFileDati = "dati/" + this.myAula.getNome()+ ".jpg";

System.out.println("leggo il file "+ strFileDati);

InputStream fileDati = classloader.getResourceAsStream(strFileDati);

try

{

imgPlan=ImageIO.read(filePlan);

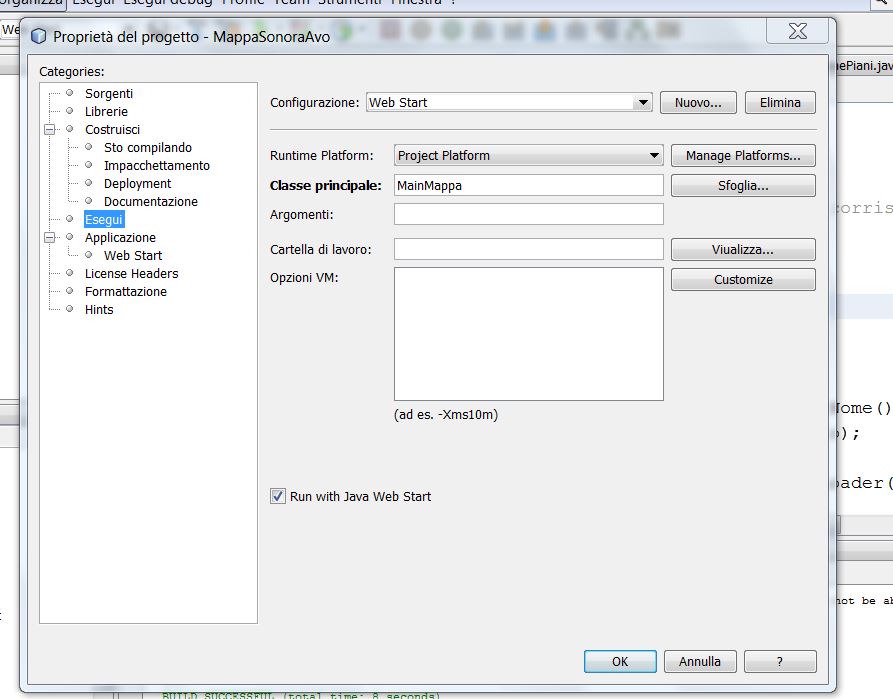
imgImg=ImageIO.read(fileImg);

imgDati=ImageIO.read(fileDati);

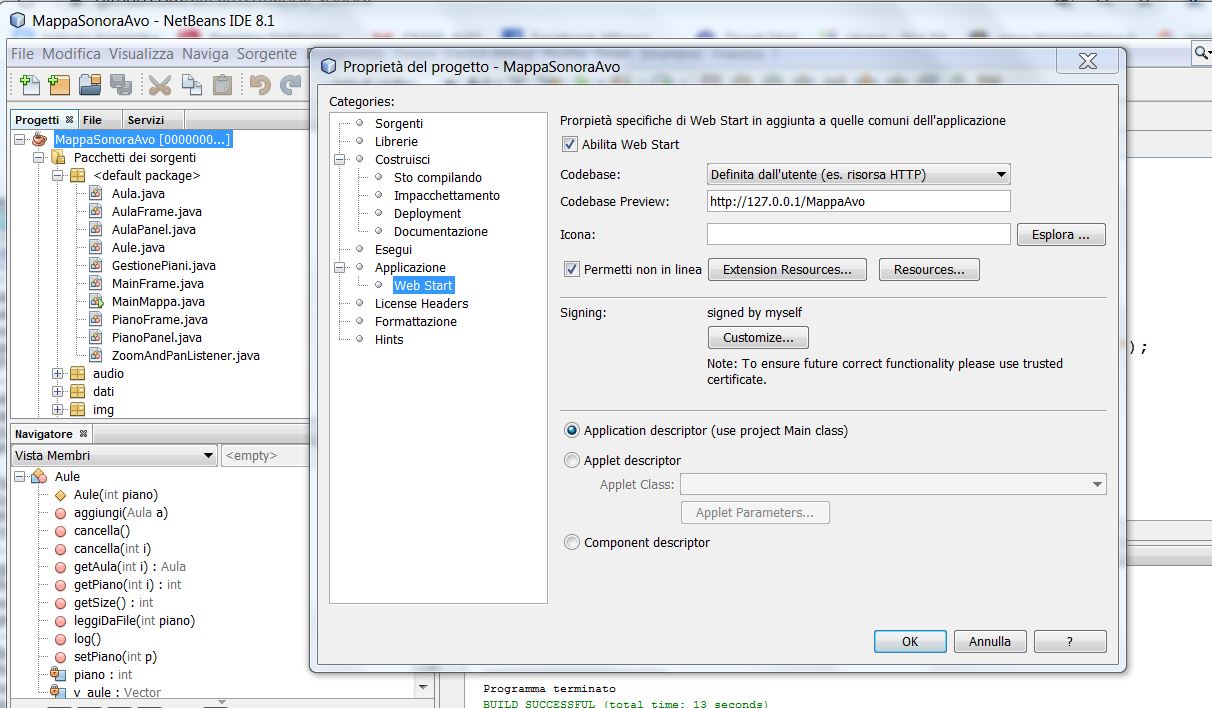
……

1.Impostazione su NetBeans

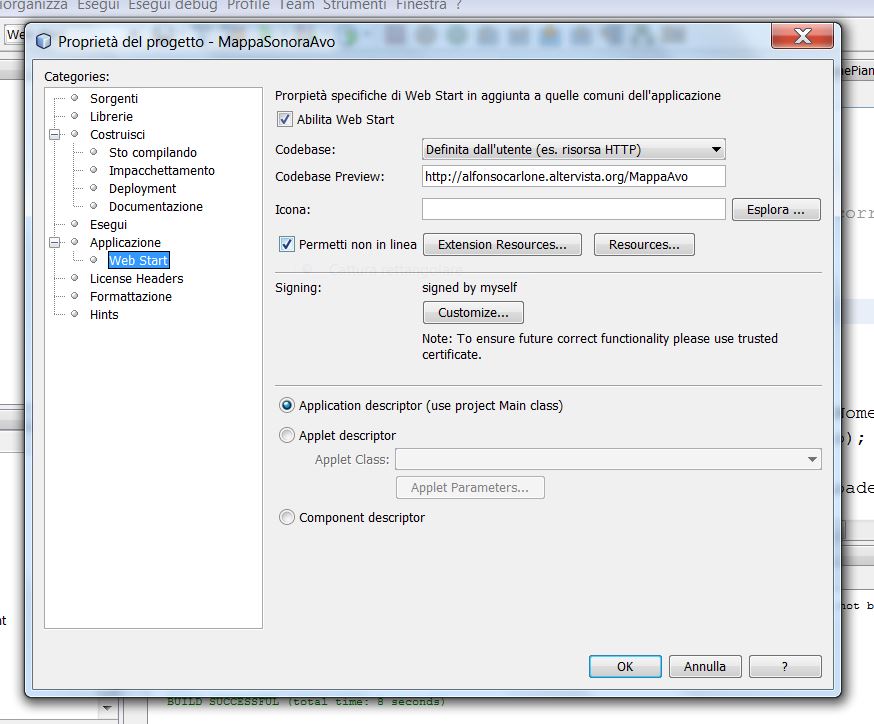
-Proprietà del progetto



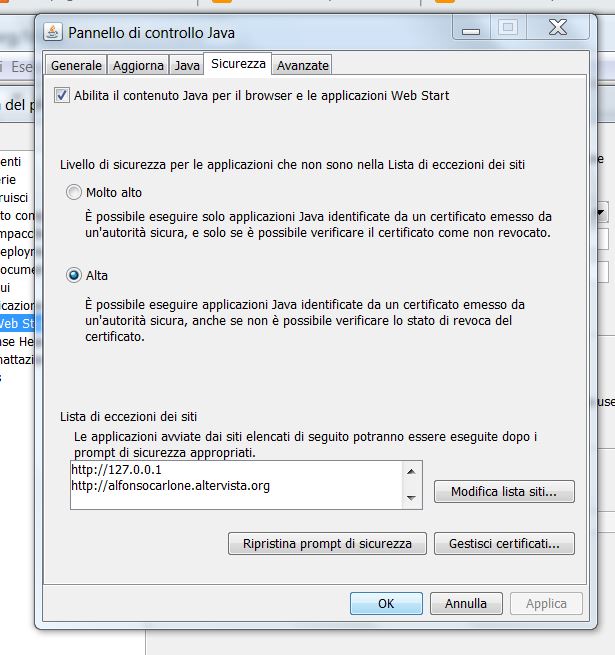
* Se in localhost



* Se su server remoto



Sistema: Sicurezza java

i

2.Creazione del certificato

1. Assicurati di avere un SDK 1.4.2 keytoole jarsignernel tuo percorso. Questi strumenti si trovano nella directory bin dell'SDK.

2. Creare una nuova chiave in una nuova keystorecome segue:

**keytool -genkey -keystore myKeystore -alias myself**

Ti verranno richieste informazioni sulla nuova chiave, come password, nome, ecc. Questo creerà il myKeystorefile sul disco.

3. Quindi creare un certificato di prova autofirmato come segue:

**keytool -selfcert -alias myself -keystore myKeystore**

Questo richiederà la password. La generazione del certificato potrebbe richiedere alcuni minuti.

4. Verificare che tutto sia a posto. Per elencare il contenuto del keystore, utilizzare questo comando:

**keytool -list -keystore myKeystore**

Dovrebbe elencare qualcosa come:

Tipo di archivio chiavi: jks

Fornitore di keystore: SUN

Il tuo keystore contiene 1 voce:

me stesso, martedì 23 gennaio 19:29:32 PST 2001, keyEntry,

Impronta digitale del certificato (MD5):

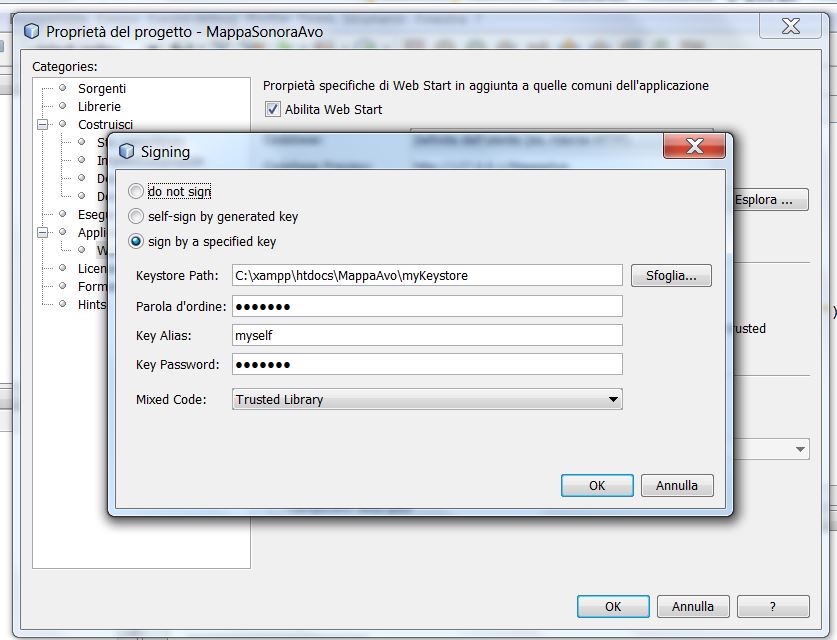
C2: E9: BF: F9: D3: DF: 4C: 8F: 3C: 5F: 22: 9E: AF: 0B: 42: 9D

5. Infine, firmare il file JAR con il certificato di prova come segue:

**jarsigner -keystore myKeystore test.jar myself**

Ripeti questo passaggio con tutti i tuoi file JAR.

**3. Inserire il certificato su netBeans**



**4.Avviare xampp**

Start apache

**5.Copiare i file presenti sotto la cartella dist**

sotto c:\xampp\htdocs\MappaAvo

**6.Da browser:**

http://127.0.0.1/MappaAvo