

# Java Web Start

par [johann Sorel \(jsorel.developpez.com\)](http://jsorel.developpez.com)

Date de publication : 26/01/2007

Dernière mise à jour : 28/01/2007

On a tous un jour ou l'autre eu la fâcheuse expérience de devoir réinstaller un logiciel sur plusieurs postes, à cause d'une erreur de manipulation, d'un reformatage ou d'une simple mise à jour.  
Il existe plusieurs façons de palier à ce problème, en voici une.  
Déployez ces logiciels avec Java Web Start.

## POURQUOI JWS ?

- 1 - Le nécessaire
  - 1.1 - Le Serveur
  - 1.2 - Le logiciel
- 2 - Ajouter au serveur
- 3 - Allons plus loin
- 4 - Voyons le résultat
- 5 - L'intégralité des balises
  - 5.1 - information
  - 5.2 - applet-desc
  - 5.3 - application-desc
  - 5.4 - component-desc
  - 5.5 - installer-desc
  - 5.6 - ressources
  - 5.7 - security
- 6 - Signer une archive jar
- 7 - En savoir plus
  - 7.1 - Annexe 1
  - 7.2 - Annexe 2
- 8 - Remerciement

## POURQUOI JWS ?

### Les avantages :

- Déploiement à distance des logiciels
- Mise à jour automatique
- Exécution sur le poste client
- Pas d'exécutable / désinstallation et de raccourcis à gérer

### Le principe :

- On installe le logiciel à partir d'un serveur web (apache ou équivalent)
- La mise à jour se fait à chaque connexion s'il y a besoin
- Un raccourci est créé sur le bureau et dans le menu démarrer

## 1 - Le nécessaire

Il vous faut un serveur web, le plus simple qu'il existe suffit.

Apache dans sa version minimale ou IIS sont plus que suffisants.

S'il en existe un en place, il servira à cette tâche.

### 1.1 - Le Serveur

Commençons par mettre en place notre serveur.

#### **Vous utilisez APACHE :**

Pour activer la reconnaissance des fichiers .jnlp, il vous faudra éditer le fichier mime.types .

Installation classique sous Windows : C:\Program Files\Apache Group\Apache2\conf\mime.types

Ajouter dans le fichier la ligne : application/x-java-jnlp\_file jnlp

#### **Vous utilisez IIS :**

Il faut ajouter un type de fichier.



Rien de plus côté serveur.

### 1.2 - Le logiciel

Attachons nous maintenant au logiciel.

Dans ce tutorial, je vais utiliser une JFrame avec un petit mot.

Ce sera un logiciel exécutable uniquement à partir d'internet, sans création de raccourcis, sans accès aux ressources systèmes... bref le strict minimum.



```
package prog;

import javax.swing.JFrame;
import javax.swing.JLabel;

public class HelloYou extends JFrame{

    private static final long serialVersionUID = 1L;

    public static void main( String[] args ){
        new HelloYou();
    }

    public HelloYou(){
        super("Hello You");
        getContentPane().add(new JLabel("Hello YOU!"));
        setDefaultCloseOperation(EXIT_ON_CLOSE);
        setLocationRelativeTo(null);
        pack();
        setVisible(true);
    }
}
```

Compilez et faites-en un jar.

### Particularité :

Tout n'est pas merveilleux, voici un défaut de JWS , toutes les données dont vous avez besoin doivent être contenues dans des archives JAR. Heureusement ça ne change pas grand chose dans la majorité des logiciels, c'est même plus propre ( c'est devenu une habitude chez moi ).

### En pratique (cas d'une image) :

```
ImageIcon jpg = new ImageIcon( "modele" + File.separator + "terrain" + File.separator + "land.jpg"
);
devient
ImageIcon jpg = new ImageIcon( ImageIcon.class.getResource( "modele/terrain/land.jpg" ) );
```

## 2 - Ajouter au serveur

Pour rendre notre application accessible, il va nous falloir créer un fichier .JNLP, ce fichier est au format xml, donc compréhensible. Vous trouverez plus loin un exemple ainsi que la totalité des balises jnlp.

### Créer un fichier logiciel.jnlp :

```
<?xml version="1.0" encoding="utf-8"?>
<jnlp spec="1.0+" codebase="http://localhost/prog/" href="logiciel.jnlp">

  <information>
    <title>Mon Logiciel par JNLP</title>
    <vendor>Johann Sorel</vendor>

  </information>

  <resources>
    <j2se version="1.5+" href="http://java.sun.com/products/autodl/j2se"/>
    <jar href="http://localhost/prog/prog.jar"/>
  </resources>

  <application-desc main-class="prog.HelloYou" />

</jnlp>
```

### Créer la page web :

```
<html>
<head>
  <title>Mon Logiciel</title>
</head>

<body>
  Mon Programme a déployer :
  <br>
  <a href="logiciel.jnlp">PROG</a>
</body>

</html>
```

Ajoutez la page web, le fichier jnlp et prog.jar dans le même dossier (ajustez les chemins d'accès selon votre serveur).

**Et voilà, rien de plus.**

Bien sûr, ce n'est que la version minimaliste.

### 3 - Allons plus loin

Java Web Start peut paraître très simpliste, détrompez-vous, il y a possibilité de le rendre très performant et très avantageux ! Aussi bien pour l'entreprise sur un intranet que pour un développeur seul qui veut partager ses logiciels sur internet.

Pour mieux comprendre toutes les possibilités de Java Web Start, voici un fichier jnlp classique.

```
<?xml version="1.0" encoding="utf-8"?>
<jnlp spec="1.0+" codebase="http://localhost/jnlp/" href="alter.jnlp">
```

```
  <information>
    <title>Alter-SIG</title>
    <vendor>Johann Sorel</vendor>
    <homepage href="http://localhost/jnlp/" />

    <description>Alter-SIG</description>
    <description kind="short">Logiciel 3D SIG Java, technologie de
      déploiement JWS/JNLP</description>
    <description kind="tooltip">Alter-SIG</description>

    <icon href="http://localhost/jnlp/icon.gif" width="64"
      height="64" />
    <offline-allowed />
  </information>
```

```
<security>
  <all-permissions />
</security>
```

```
<resources>
  <j2se version="1.5+"
    href="http://java.sun.com/products/autodl/j2se"
    initial-heap-size="128m"
    max-heap-size="1024m" />

  <jar href="altersig.jar" main="true" download="eager" />
  <jar href="data.jar" />
  <jar href="lib/jsl.jar" />
  <jar href="lib/looks-2.0.4.jar" />

  <extension name="unAutresource" version="1" href="/3DEngine.jnlp" />
</resources>

<resources os="Windows">
  <nativlib href="lib/winnativ.dll.jar" />
</resources>

  <resources os="Linux">
    <nativlib href="lib/linuxnativ.so.jar" />
```

```
</resources>
```

```
<application-desc main-class="asig.load.Boot">  
  <argument>-jconsole disable</argument>  
  <argument>show</argument>  
</application-desc>
```

```
</jnlp>
```

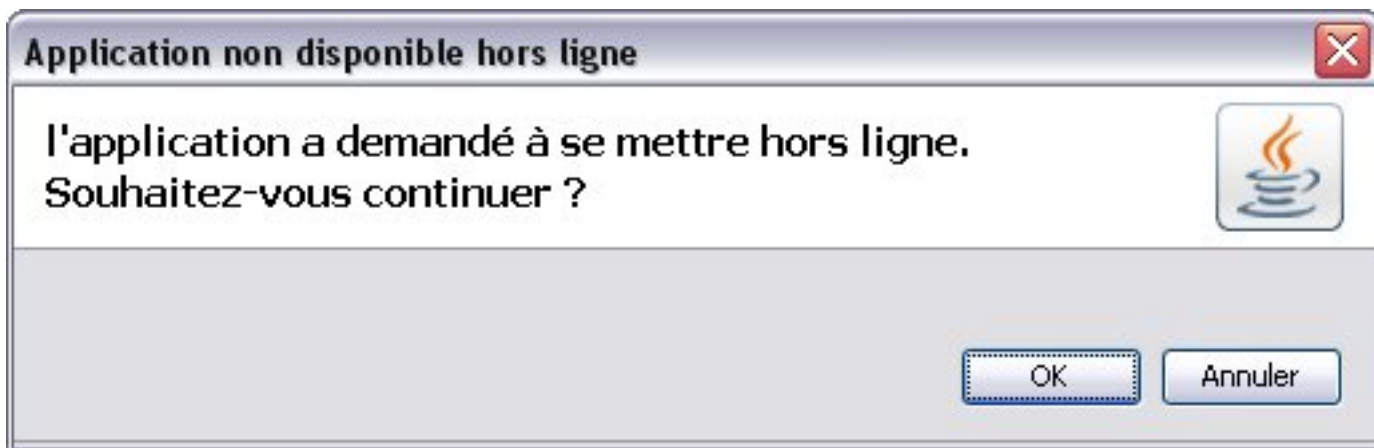
Voici donc un fichier jnlp d'un logiciel de taille « normal ». Mais il y a plus de balises encore.

## 4 - Voyons le résultat

Si tout est bien fait on obtient les écrans suivants lors de l'exécution (résultat avec JRE 1.6.0, les écrans sont légèrement différents selon les JRE).



Celui-ci témoigne que votre serveur est bien configuré, et que votre fichier jnlp va être traité par la machine virtuelle java.



Si vous en êtes à vos premiers pas avec Java Web Start et que vous travaillez en local, il arrive que cet écran s'affiche.

Vient ensuite le moment le plus important, si votre fichier jnlp est bien structuré.



Cet écran demande la confirmation de l'installation à l'utilisateur.

Comme on le voit, signer un logiciel ne garanti pas qu'il est honnête, à la charge de l'utilisateur de savoir ce qu'il fait.

Mais voir cette écran est rassurant, il prouve que la machine virtuelle java veille à la sécurité.

**Erreur d'application**

**Impossible de lancer l'application.**

**Nom :** Mon Logiciel par JNLP  
**Éditeur :** Johann Sorel  
**De :** http://localhost

OK Détails

---

**Plus d'informations**

**Erreur :** Une application non signée demande un accès illimité au système  
Ressource non signée : http://localhost/prog/prog.jar

Fichier de lancement Exception

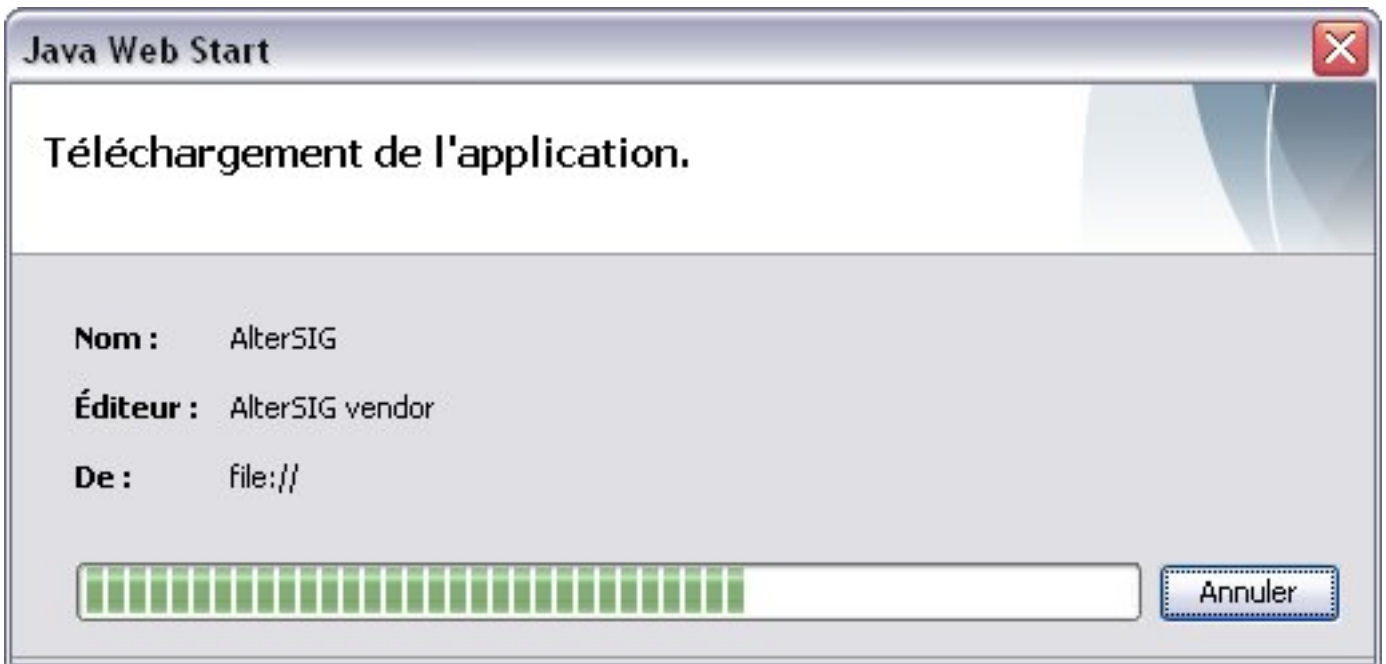
```
<?xml version="1.0" encoding="utf-8"?>
<jnlp spec="1.0+" codebase="http://localhost/prog/" href="logiciel.jnlp">
  <information>
    <title>Mon Logiciel par JNLP</title>
    <vendor>Johann Sorel</vendor>
    <offline-allowed />
  </information>
  <security>
    <all-permissions />
  </security>
  <resources>
    <j2se version="1.5+" href="http://java.sun.com/products/autodl/
i2se"/>
```

Fermer

A l'inverse voici le genre d'écran que produit un fichier jnlp mal configuré.

Ici on peut lire, en cliquant sur « détail » qu'il s'agit de notre archive « prog.jar » qui n'est pas signée.

N'hésitez pas à aller sur l'onglet « Exception », les exceptions en java sont compréhensibles et précises. (un point fort du Java)



Après avoir réglé les petits problèmes s'il y en avait, le chargement doit se faire normalement et le logiciel va démarrer.

## 5 - L'intégralité des balises

**1** OBLIGATOIRE (une seule balise possible)

**1..N** OBLIGATOIRE ( 1 ou N balises)

**0..1** OPTIONEL (0 ou une balise)

**0..N** OPTIONEL (0 ou N balises)

`<?xml version="1.0" encoding="utf-8"?>`

**1** version : version xml du fichier

**1** encoding : encodage utilisé

`<jnlp spec="1.0+" codebase="http://localhost/jnlp/" href="alter.jnlp">`

**1** spec : version du fichier jnlp

**1** codebase : chemin des fichiers

**1** href : nom complet du fichier jnlp

### 5.1 - information

`<information locale="fr_FR">`

**1** La balise information contient la description de l'application.

**0..1** Locale : précise le pays, la langue, la culture cf: annexe 1

**<title>AlterSIG</title>**

**0.1** titre de l'application

**<vendor>Johann Sorel</vendor>**

**0.1** nom du ou des développeurs/vendeurs

**<homepage href="http://localhost/jnlp/">**

**0.1** site de l'application

**1** href : url de l'application, adresse du site

**<description>Alter-SIG</description>**

**<description kind="short">Logiciel 3D SIG Java, technologie de déploiement JWS/JNLP</description>**

**<description kind="tooltip">Alter-SIG</description>**

**0.N** description de l'application

**0.1** kind : one-line / short / tooltip

**<icon href="http://localhost/jnlp/icon.gif" width="64" height="64"/>**

**<icon href="http://localhost/jnlp/icon32.jpg" width="32" height="32"/>**

**0.N** icône utilisé pour les raccourcis

**1** href : url de l'icone au format JPEG ou GIF

**0.1** version : version de l'image

**0.1** width : largeur de l'image

**0.1** height : hauteur de l'image

**0.1** kind : default / selected / disabled / rollover : type d'utilisation de l'image

**0.1** depth : profondeur de couleur, généralement 8,16,24,32

**0.1** size : poids de l'image en octets

**<shortcut online="true">**

**0.1** Ajouter un raccourci pour le logiciel

**0.1** online : visible uniquement si connexion au site mère présente

**<desktop />**

**0.1** Raccourci sur le bureau

**<menu submenu="nomlogiciel"/>**

**0.1** Raccourci dans le menu démarrer

**</shortcut>**

**<offline-allowed />**

**0.1** permet a l'application de se lancer sans avoir de connexion au site

**</information>**

## 5.2 - applet-desc

**<applet-desc main-class="maclasse.class" name="Prog" width=320 height=200>**

**0.1** Description de l'applet (s'il sagit d'un applet uniquement)

**1** main-class : nom de la classe principale

**1** name : nom de l'applet

**1** width : largeur de l'applet en pixel

**1** height : hauteur de l'applet en pixel

**0.1** documentbase : url pour les fichiers utilisés

**<param name="variableA" value="Monsieur Jean">**

**0.N** d'éventuels paramètres pour l'applet

**1** name : nom du paramètre

**1** value : valeur du paramètre

**</applet-desc>**

## 5.3 - application-desc

**<application-desc main-class="maclasse.class" >**

**0.1** Description de l'application (s'il s'agit d'une application uniquement)

**0.1** main-class : nom de la classe principale, on peut sauter ce paramètre si la classe principale est dans la première archive jar et qu'elle contient un fichier manifest avec la classe principale

**<argument>un argument</argument>**

**0.N** D'éventuels arguments pour l'application, ces arguments sont passés à la méthode main de l'application

**</application-desc>**

## 5.4 - component-desc

**<component-desc/>**

**0.1**

Spécifier une extension de composant

## 5.5 - installer-desc

**<installer-desc main-class="classedinstallation.class"/>**

**0.1**

Spécifiez une classe pour l'installation ou la désinstallation de l'application

**1**

main-class : nom de classe

## 5.6 - resources

**<resources os="Windows">**

**1.N**

La balise ressources contient les informations sur les données de l'application.

**0.1**

os : permet de spécifier pour quel système d'exploitation est destiné cette balise, on peut donc avoir des ressources différentes par système. cf: annexe 2

**0.1**

arch : permet de spécifier une architecture processeur. cf: annexe 2

**0.1**

locale: précise le pays, la langue, la culture. cf: annexe 1

**<j2se version="1.5+"  
href="http://java.sun.com/products/autodl/j2se"  
initial-heap-size="128m"**

**max-heap-size="1024m" />**



Spécifiez les caractéristiques nécessaires pour la machine virtuel java



version : 1.4+ / 1.2 1.5 : version de la JVM nécessaire. Un + pour spécifier la version minimum ou mettre un espace entre chaque version.



href : url où l'on peut télécharger la JVM



initial-heap-size : 128m / 64M / 1024K / 512k : spécifiez l'espace mémoire initial. M ou m pour mega-octets, K ou k pour kilo-octets.



max-heap-size : 128m / 64M / 1024K / 512k : spécifiez la taille mémoire maximum

**<jar href="altersig.jar" main="true" download="eager" />**

**<jar href="data.jar" download="lazy" />**



Spécifiez les jars nécessaires pour l'application



href : adresse de l'archive jar



version : version de l'archive



main : true / false : permet de spécifier si cette archive est la principale. Par défaut la première archive mentionnée est la principale.



download : eager / lazy : eager veut dire que l'archive sera téléchargée avant le lancement de l'application, lazy qu'elle le sera si nécessaire.



Size : taille en octets de l'archive



part : nom du groupe d'archive auquel appartient cette archive

**<nativlib href="lib/jogl\_awt.dll.jar" />**

```
<nativelib href="lib/jogl_cg.dll.jar" />
```

**D.N**

Spécifiez les librairies natives nécessaires pour l'application. Il est souvent nécessaire de faire deux (ou plus) balises ressources pour spécifier des librairies natives différentes par système d'exploitation. Il est nécessaire de charger dynamiquement ces librairies dans l'application ( `System.loadLibrary("jogl")` )

**1**

href : adresse des archives. Les librairies (.dll / .so) doivent être contenues dans des.jar

**0.1**

version : version de l'archive

**0.1**

download : eager / lazy : eager veut dire que l'archive sera téléchargée avant le lancement de l'application, lazy qu'elle le sera si nécessaire.

**0.1**

Size : taille en octets de l'archive

**0.1**

part : nom du groupe d'archive auquel appartient cette archive

```
<extension name="unAutresource" version="1" href="3Dengine.jnlp">
```

**D.N**

Spécifiez d'éventuel extension nécessaire à l'application.

**1**

href : adresse du fichier jnlp

**0.1**

version : version de l'extension

**0.1**

name : nom de l'extension

```
<ext-download ext-part="unepartie" download="lazy">
```

**D.N**

Permet de définir la manière dont sera téléchargé chaque partie de l'extensions

**1**

ext-part : nom de la partie d'extension

**0.1**

download: eager / lazy : eager veut dire que l'archive sera téléchargée avant le lancement de l'application, lazy qu'elle le sera si nécessaire.

**0.1**

Part : nom du groupe où elle appartient

**</extension>****<property name="user.name" value="Johann"/>****0.N**

Permet de donner une valeur à une propriété du système de type System.getProperty ou System.getProperties

**1**

name : nom de la propriété

**1**

value : valeur de la propriété

**<package name="oracle.\*" part="oracle" recursive="true"/>****0.N**

Permet de définir une relation entre un package (ou une classe) et un groupe

**1**

name : nom du paquet

**1**

part : nom du groupe concerné

**0.1**

recursive : true / false : permet de préciser si le sous-package sont aussi compris dans cette relation

**</resources>**

## 5.7 - security

**<security>****0.1**

La balise security: les sécurités à apporter pour l'application. S'il n'y a pas de précision l'applition s'exécute dans le « bac à sable », donc comme un applet avec juste assez de droit pour son fonctionnement. Pas d'accès au disque, de communication réseau ...

**<all-permissions />**

**0.1**

Définit l'application comme ayant tous les droits d'une application normale Java.

Il est nécessaire que TOUTES les archives jar soient signées (cf : signer un jar) et que l'utilisateur ait donné son accord au moment du lancement.

**<j2ee-application-client-permissions/>**

**0.1**

Indique que l'application doit avoir les droits d'une application de type J2EE

**</security>**

**</jnlp>**

## 6 - Signer une archive jar

La signature d'une archive est très importante, elle permet de savoir qui a fait l'archive de manière fiable. C'est une façon de prouver qu'il s'agit bien de votre travail. C'est une sorte de signature électronique. Elle va permettre à notre application d'avoir le droit d'accéder aux fichiers systèmes, à la communication sur le réseau ... etc ..., à l'identique d'un programme java standard.

Le principe de la signature marche sur un cryptage clé privée/clé public. La signature va ajouter quelques lignes dans les fichiers manifest ainsi que d'autres fichiers permettant de confirmer qu'il s'agit de vous et que l'archive n'a pas été modifiée après signature. Je n'en dirais pas plus sur ces sécurités, ce n'est pas le but de ce tutorial.

Il faut d'abord obtenir une clé privée associée à des clés publiques. Celles-ci vont être stockées dans une petite base de données appelée « Keystores ». Chaque clé est identifiée par un alias, généralement le nom du signataire. Un keystore peut contenir la signature de plusieurs personnes.

### **Pour créer un KeyStore :**

```
keytool -genkey -keystore FichierKeyStore -alias henry
```

Tapez le mot de passe du Keystore : (mdpkey)

Ressaisissez le nouveau mot de passe : (mdpkey)

Quels sont vos prénom et nom ?

[Unknown] : henry Dupont

Quel est le nom de votre unité organisationnelle ?

[Unknown] : etudiant

Quelle est le nom de votre organisation ?

[Unknown] : lupsig

Quel est le nom de votre ville de résidence ?

[Unknown] : paris

Quel est le nom de votre État ou province ?

[Unknown] : france

Quel est le code de pays à deux lettres pour cette unité ?

```
[Unknown] : fr

Est-ce CN=henry, OU=etudiant, O=lupsig, L=paris, ST=france, C=fr ?

[non] : oui

Spécifiez le mot de passe de la clé pour <henry>

(appuyez sur Entrée s'il s'agit du mot de passe du Keystore) : (mdpkenry)

Ressaisissez le nouveau mot de passe : (mdpkenry)
```

Un fichier « monKeyStore » a été créé.

On peut répéter cette opération pour ajouter des signatures dans le keystore.

**Pour signer notre archive jar, il faut exécuter la commande :**

```
 jarsigner -keystore FichierKeyStore monarchive.jar henry

Enter Passphrase for keystore: (mdpkey)

Enter key password for henry: (mdpkenry)

Warning:

The signer certificate will expire within six months.
```

C'est fait, votre archive est maintenant signée de votre main « virtuelle » et prête pour être autorisée à accéder aux données du système.

## 7 - En savoir plus

Page Java Web Start de SUN :

 [java.sun.com/products/javawebstart/index.jsp](http://java.sun.com/products/javawebstart/index.jsp)

Page des tags de sun (en anglais) :

 [java.sun.com/javase/6/docs/technotes/guides/javaws/developersguide/syntax.html](http://java.sun.com/javase/6/docs/technotes/guides/javaws/developersguide/syntax.html)

Un autre résumé complet sur les différents tags (en anglais) :

 [lopica.sourceforge.net/ref.html](http://lopica.sourceforge.net/ref.html)

Automatisation : Signer un jar avec Apache ANT :

 [ant.apache.org/manual/CoreTasks/signjar.html](http://ant.apache.org/manual/CoreTasks/signjar.html)

Comment déployer une application SWT via WebStart :

 [lfe.developpez.com/Java/SWT/WebStart/](http://lfe.developpez.com/Java/SWT/WebStart/)

Documentation avancée, compléments de johann pons :

 <http://developers.sun.com/learning/javaoneonline/>

Articles : TS-3212, TS-3133, TS-1319, TS-7904

### 7.1 - Annexe 1

La liste complète de toutes les langues :

 [www.unicode.org/unicode/onlinedat/languages.html](http://www.unicode.org/unicode/onlinedat/languages.html)

La liste complète de toutes les régions :

 [www.unicode.org/unicode/onlinedat/countries.html](http://www.unicode.org/unicode/onlinedat/countries.html)

### 7.2 - Annexe 2

**Abréviations des systèmes d'exploitations :**

 <http://www.vamphq.com/os.html>

## 8 - Remerciement

Correction : L.Lecocq

