JAV - TD 10
Applets & ant

Les applets java

Le packaging d'applications

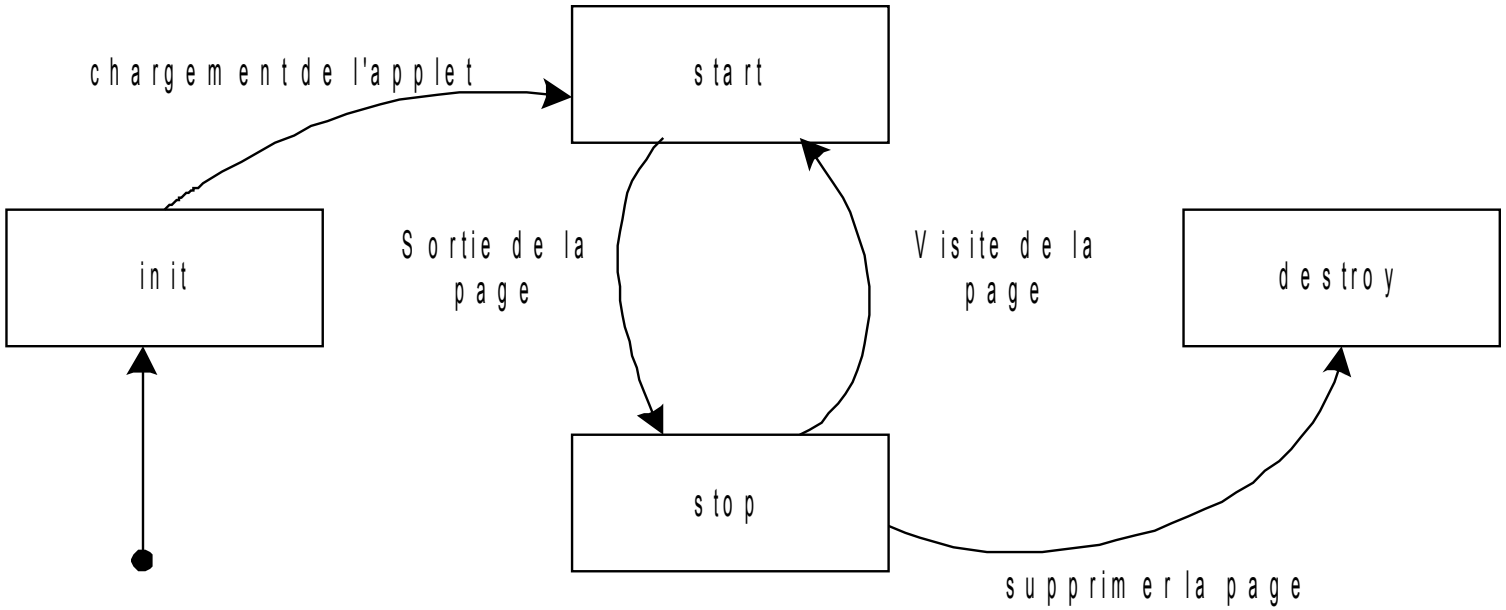
Les applets – Objectifs

- Une applet est une classe compilée héritant de `java.applet.Applet`.
- Une applet est à la fois un composant graphique (hérite de `Panel`) et un thread d'exécution

Les applets – Objectifs

- Elle est diffusée par un serveur web dans une page HTML
- Elle est téléchargée puis exécutée par le browser.
- Elle est soumise au *Security Manager* du browser
  - pas d'accès en lecture ni en écriture sur le disque du browser
  - connexion réseau uniquement sur le serveur d'origine
  - pas de chargement de librairie native
  - pas de lancement de processus, ...

Les applets – Vie et mort



Les applets – Syntaxe

```
public class BonjourApplet extends Applet {
   String msg = "bonjour";
   public void init() {...}
   public void start() {...}
   public void paint(java.awt.Graphics g)
   {g.drawString(msg, 20, 20);}
   public void stop() {...}
   public void destroy() {...}
}
```

```
<HTML>
  <BODY>
    <APPLET code="BonjourApplet.class"
       codebase="http://citi.insa-lyon.fr/~flemouel/applets/"
       width=300 height=200>
    <PARAM name="message" value="Bonjour">
   </applet>
  </BODY>
</HTML>
```

Les applets – Démarrage, arrêt

- `init` : méthode appelée au chargement de l'applet (avant le `start`)
- `start` : méthode appelée pour démarrer l'applet
- `isActive` : l'applet est vivante ?
- `stop` : méthode appelée pour arrêter l'applet
- `destroy` : méthode appelée pour détruire l'applet (libération des ressources, etc.) (`stop` doit toujours être appelée avant)

Les applets – Exemple

```
public class BonjourApplet extends Applet {
   StringBuffer buffer;
   public void init() {
          buffer = new StringBuffer();
          this.addItem("initialisation..."); }
   public void start() { this.addItem("démarrage..."); }
   public void stop() { this.addItem("arrêt..."); }
   public void destroy() { this.addItem("déchargement..."); }
   private void addItem(String newWord) {
          buffer.append(newWord); repaint(); }
   public void paint(Graphics g) {
          g.drawRect(0, 0, size().width - 1, size().height -
   1);
          g.drawString(buffer.toString(), 5, 15); } //
   Message
}
```

Exercice

- Écrire l'applet précédente affichant « Bonjour ! » à l'aide de la fonction `drawString` de la classe `Graphics`. Visualisez-là dans une page Web à l'aide d'un navigateur
- Modifier l'applet pour expérimenter d'autres méthodes de la classe `Graphics`, comme `drawLine`, `drawRect`, `drawImage`, `fillRect`, etc.
- A l'aide des interfaces `MouseListener` et `MouseMotionListener`, écrivez une applet `Tableau` qui permet de tracer une ligne sur simple pression du bouton de la souris

Le packaging d'applications

Le retour du makefile...

Objectifs du Makefile

- **Gérer une chaîne complète de production de logiciels (et plus si affinité)**
  - Ex :
    - Compilation
    - Génération de squelettes
    - Intégration de code
    - Tests
    - Packaging
    - Mailing
    - Déploiement...
  - ==> Le pb : chaque tâche ci-dessus repose sur des outils spécifiques. Il faut des outils pour mettre bout à bout l'exécution de ces outils
- **Pour réaliser l'ensemble de ces tâches, il existe deux solutions :**
  - Les langages de scripts
  - Les outils de gestion de projet comme Makefile ou Ant

Objectifs du Makefile

- Les objectifs du makefile sont de gérer proprement un ensemble de tâches. Ex en C :
  - compilation, linkage, exécution

```
run: test
    ./test
test: test.o
    gcc -o test.o test.c
test.o: test.c
    gcc -c test.c
```

- Les tâches déjà réalisées ne sont pas ré-exécutées, seuls les fichiers à regénérer le sont.
  - ant est une adaptation du make file au monde java, et aux projets plus « complexes »

Ant

- Repose sur un fichier de description de tâches : build.xml
- La commande est `ant`
- Il existe toute une collection de tâches standardisées
  - java, javac, mail, cc, jar, tar, move, copy, documentation...
- Il est possible de définir de nouvelles tâches
- http://ant.apache.org/manual/

Ant exemple

Build.xml

```
<project name="tp1" default="compile" basedir=".">
 <target name="clean">
  <delete verbose="false"
        includeEmptyDirs="true">
   <fileset dir="classes"/>
  </delete>
 </target>
 <target name="compile" depends="init">
  <javac srcdir="src"
   destdir="classes"
    debug="on">
 </target>
</project>
```

- Les commandes possibles pour cet exemple
  - ant
  - ant clean
  - ant compile
- Les répertoires concernés sont src et classes

Exercice

- Réaliser votre projet Division sous ant.
- Votre outil doit vous permettre de :
  - Compiler et lancer votre projet
  - Créer un jar exécutable de votre projet

- A partir de maintenant tous vos projets de développement sont packagés dans des scripts ant