Les Annotations en JAVA

Fabien LE BLEVEC
Informatique & Réseau - 3ème année
Exposé de Système

Ingénieurs 2000, Université de Marne la vallée

Plan

- Présentation
- Syntaxe
- Les Méta-annotations de Java 1.5
- Les annotations standards de Java 1.5
- APT (Annotation Processing Tool)
- Conclusion

Présentation

- Historique
  - En JAVA depuis la J2SE 5.0 Tiger (2005)
  - Déjà utilisé par les approches XDoclet/EJBGen…
  - Utilisé par les EJB3/Hibernate3.1 pour citer les plus connus...
- Définition
  - S'applique à :
    - Des méthodes
    - Des classes
    - Des attributs
  - Optionnel
  - Peux contenir des couples clé/valeur

Syntaxe

- Déclaration

```
public @interface Contrainte {
        int minLength;
        int maxLength;

}
```

- Utilisation sur une méthode

```
@Contrainte(minLength=3, maxlength=5)
public void setName(){
…
}
```

Les Méta-annotations (1/5)

- Présentation
  - **Ce sont des annotation qui s'appliquent aux annotations.**
  - Disponible dans java.lang.Annotation
  - Liste des méta-annotations :
    - @Documented
    - @Inhereted
    - @Retention
    - @Target
    - @SuppressWarning*

*** s'applique aussi à d'autres éléments**

Les Méta-Annotations (2/5)

- @SuppressWarning(« unchecked »)
  - Ne s'applique pas forcément à une annotation…
  - Indique au compilateur de ne pas lever de warning pour des raison de conversions non vérifiées
  - Utile pour les « **Unchecked Warning** »
    - casts ou conversions impliquant des *generics* non vérifiable par le VM.
    - 2 types :
      - **Unchecked conversion**

```
ArrayList<String> list=new ArrayList<String>();
ArrayList raw=list;
```

      - **Unsafe cast**

```
Object o=new ArrayList<String>();
ArrayList<String> list=(ArrayList<String>)o;
```

Les Méta-Annotations (3/5)

- @Retention
  - Politique d'usage de l'annotation
  - Définit la détention de l'annotation au niveau de la JVM.
  - Différentes politique :
    - **SOURCE** : Ignorer par le compilateur
    - **CLASS** : Enregistrer dans le fichier class mais non conserver par la JVM
    - **RUNTIME** : Enregistrer dans le fichier class et conserver par la JVM à l'éxécution, peut être lu par réflexion !

Les Méta-Annotations (4/5)

- @Target
  - Définie le domaine d'application de l'annotation
  - Les types d'éléments sont :
    - ANNOTATION_TYPE :
    - CONSTRUCTOR
    - FIELD(inclus les constantes des enums)
    - LOCAL_VARIABLE
    - METHOD
    - PACKAGE
    - PARAMETER
    - TYPE (class, interface, enum)

Les Méta-Annotations (5/5)

- @Documented
  - L'annotation doit être documenté par des tags javadoc.
  - Indique à javadoc qu'il doit prendre en compte cette annotation.
- @Inheret
  - L'annotation est automatiquement héritée.
  - Le fils d'un élément annoté aura aussi l'annotation.

Les Annotations standards

- Disponible dans java.lang
- @Deprecated
  - Redéfinit le tag Javadoc
  - Lance un warning quand le membre annoté est utilisé.
- @Overrides
  - Lance une erreur si la méthode annoté ne redéfinit pas celle de sa super-classe.

Annotations - Restrictions

- Pas d’héritage d’annotations(même si ce sont des interfaces)
- Méthodes d’annotation sans paramètres !
- Pas de gestion des exceptions

APT-Présentation

- Traitement avant compilation des sources Java
- API réflective basée sur les sources
- Procure une vue read-only des sources

- Equivalent de javac en plus puissant !

- Nécessite la librairie tools.jar dans lib/ du SDK1.5

APT-Fonctionnement

- Nécessite :
  - Une classe AnnotionProcessorFactory fournissant des AnnotationProcessor…
  - Une classe AnnotationProcessor fournissant le traitement des sources.
  - Une classe DeclarationVisitor fournissant le traitement des déclarations.
- Utilise le design-pattern Visitor pour parcourir les sources.

Avantages/Inconvénients

- Utilisation pendant l'éxécution
  - Avantages :
    - Simple
    - Facile à mettre en oeuvre
  - Inconvénients :
    - Le réflexion est un processus lent !
    - Gestion des exceptions laborieuses
- Utilisation avant la compilation
  - Avantages:
    - Très porteur !
    - Permet de générer à partir des sources, n'importe quel fichiers
    - Possibilité de faire de la méta-programmation(par aspect)
  - Inconvénients:
    - API encore mal documenté
    - APT non géré par les IDE

Conclusion

- Inclus dans Java SE 6 (Mustang)
  - Dans les JSR 199 and JSR 269
- De nombreuses API et framework utilisent déjà les annotations.
- Attention aux conflits d'API/framework utilisant les mêmes noms d'annotations !
- Bouleversement avec l'arrivée d'APT des APIs/Framework utilisant les annotations ???

Netographie

- http://www.eclipse.org/jdt/apt/
- http://java.sun.com/developer/technicalArticles/releases/j2se15/
- http://java.sun.com/developer/technicalArticles/J2SE/constraints/annotations.html
- http://adiguba.developpez.com/tutoriels/java/tiger/annotations/
- http://lroux.developpez.com/article/java/tiger/
- http://ricky81.developpez.com/tutoriel/java/api/reflection/
- http://java.sun.com/j2se/1.5.0/docs/guide/apt/mirror/
- http://java.sun.com/j2se/1.5.0/docs/guide/apt/
- http://www-106.ibm.com/developerworks/library/j-annotate1/
- http://www-106.ibm.com/developerworks/library/j-annotate2.html
- http://www.eclipse.org/jdt/apt/introToAPT.html
- http://java.sun.com/developer/technicalArticles/J2SE/Desktop/Mustang_build39.html