Java EE Spring, prêt à l'emploi

Annotations

Guillaume Dufrêne – Lionel Seinturier – Julien Wittouck

Université de Lille

Annotation

- Élément de syntaxe du langage Java
- Apparu avec Java 5 (2004)


```
public class Fenetre extends JFrame {

  private JPanel container = new JPanel();

  public Fenetre() {
    this.setTitle("Application Test");
  }
```

Annotation

- Élément de syntaxe du langage Java
- Apparu avec Java 5 (2004)

- But : pouvoir ajouter des informations (métadonnées) au code d'un programme

- Pas directement exécutable
- Concept existant dans d'autres langages (ex. C#)

Utilisation

Deux façons

1. utilisation d'annotations existantes
2. définition et utilisation d'annotations personnalisées

Utilisation

Syntaxe

```
@nom_de_l_annotation
élément de code Java
```

Exemple

```
@Override
public String toString() { return nom+" "+prenom; }
```

Utilisation

- Code général
  - @Override, @NotNull, etc.
- Framework Spring
  - Contrôleurs
  - JPA (entité, clé, relations, etc.)
  - etc.
- Framework de tests JUnit
- Compilateur, éditeur de code

Utilisation

Certaines annotations ont des paramètres

```
@Table(name = "PARTICIPANTS")
public class Personne { ... }

@Deprecated(since="9", forRemoval=true)
public interface ClassDoc { ... }
```

Utilisation

Deux façons

1. utilisation d'annotations existantes
2. définition et utilisation d'annotations personnalisées

Annotations personnalisées

Définir l'annotation `CreePar`

Fournir des informations sur l'auteur d'une classe

Paramètres : auteur, version, date, commentaires

```
@Retention(RetentionPolicy.RUNTIME)
@Target({ElementType.TYPE})
public @interface CreePar {
        String auteur();
        int version();
        String date() default "1970-01-01";
        String[] commentaires() default {};
}
```

Annotations personnalisées

Utiliser l'annotation `CreePar`

```
@CreePar(auteur="Lionel", version=7)
public class Carre { ... }

@CreePar(
    auteur="Guillaume", version=9, date="2019-08-21"
    commentaires={"définir PI","ajouter diamètre"})
public class Cercle { ... }
```

Annotations personnalisées

Types utilisables pour les paramètres des annotations

- Types primitifs
  - int, short, long, byte, char, double, float, boolean
- String
- Le type Class
- Un type énuméré défini
- Un type annotation
- Un tableau des types précédents

Annotations personnalisées

Éléments de programme pouvant être annotés

```
@Target({ElementType.   })
```

- un type : classe, interface, énumération
- méthode, constructeur
- variable d'instance (field), variable locale
- package, module
- paramètre

Annotations personnalisées

Visibilité des annotations

```
@Retention(RetentionPolicy.     )
```

- SOURCE visible dans le code source
- CLASS SOURCE + visible dans le code compilé
- RUNTIME SOURCE + CLASS + visible à l'exécution

Raccourcis d'écriture

Un seul élément dans un tableau

- omission possible des accolades `{ }`

```
@CreePar(auteur="Lionel",version=7,commentaires="définir PI")
```

Annotation avec un seul paramètre `value`

- omission possible du nom du paramètre `value=`

```
public @interface SuppressWarnings {
        String[] value();
}

@SuppressWarnings("unchecked")
public class Triangle { ... }
```

En résumé

- Un mécanisme pour ajouter des informations sur le code Java
  - on parle de métadonnées

```
@Annotation(param=valeur, ...)
```

- Utilisation d'annotations prédéfinies en Java
- Définition d'annotations personnalisées