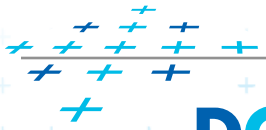
Webové aplikace 2

Anotace v Javě

Martin Klíma



Anotace

- jsou to vlastně aditivní procesní instrukce
- píše je programátor do zdrojového kódu
- využívá je nějaký externí nástroj

```
@Resource(name = "customerDB")
   public void setDataSource(DataSource myDB) {
      this.ds = myDB;
   }
@EJB
   public ShoppingCart myShoppingCart;

 @Local
   public interface RepeaterSessionBeanLocal {
   }

 @Copyright("2002 Yoyodyne Propulsion Systems")
   public class OscillationOverthruster {
      ...
   }
```


Anotace jsou definovány pomocí konstruktu

```
public @interface RequestForEnhancement {
    int    id();
    String synopsis();
    String engineer() default "[unassigned]";
    String date()  default "[unimplemented]";
}
```

Použití

```
public class EnhancementTest {

    @RequestForEnhancement(id = 2868724,
    synopsis = "Enable time-travel",
    engineer = "Mr. Peabody",
    date = "4/1/3007")
    public static void travelThroughTime(Date destination) {
        // tady neco udelej
    }
}
```

Komplexní příklad

převzato z http://java.sun.com/j2se/1.5.0/docs/guide/language/annotations.html

```
@Retention(RetentionPolicy.RUNTIME)
@Target(ElementType.METHOD)
public @interface Test { }
```

Anotace anotace = metadata

Anotovaný program

```
public class Foo {
   @Test public static void m1() { }
   public static void m2() { }
   @Test public static void m3() {
      throw new RuntimeException("Boom");
   }
   public static void m4() { }
   @Test public static void m5() { }
   public static void m6() { }
   @Test public static void m7() {
      throw new RuntimeException("Crash");
   }
   public static void m8() { }
}
```

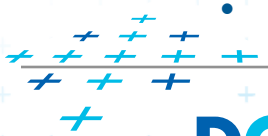
Využití anotace v kontrolním programu

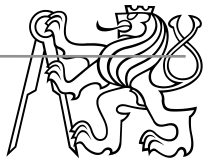
```
public class RunTests {
  public static void main(String[] args) throws Exception
{
    int passed = 0, failed = 0;
    for (Method m : Class.forName(args[0]).getMethods()) {
      if (m.isAnnotationPresent(Test.class)) {
        try {
          m.invoke(null);   passed++;
        } catch (Throwable ex) {
          System.out.printf("Test %s failed: %s %n", m,
ex.getCause());   failed++;
        }
      }
    }
    System.out.printf("Passed: %d, Failed %d%n",
passed, failed);
  }
}
```

Využití reflexe

Drobnosti kolem anotací

- Lze definovat default hodnoty
- Některé anotace už v jazyce Java existují
  - @Retention
    - SOURCE (jen ve zdrojovém kódu), CLASS (v binární třídě), RUNTIME (za běhu)
  - @Target – výčet z ElementType
    - TYPE
    - FIELD
    - METHOD
    - PARAMETER
    - CONSTRUCTOR
    - LOCAL_VARIABLE
    - ANNOTATION_TYPE
    - PACKAGE
  - @Inherited
    - potomci anotované třídy jsou také anotováni





Anotace finále

- Anotace bez hodnoty

```
@Retention(RetentionPolicy.RUNTIME)
@Target(ElementType.METHOD)
public @interface Test { }
```

- Anotace s jedinou hodnotou

```
@Retention(RetentionPolicy.RUNTIME)
@Target(ElementType.TYPE)
@Inherited
/**
 * Trida bude vracet chybovy stav uvedeny ve {@code value}.
 */
public @interface ErrorPage {
   int value();
}
```

- S více hodnotami a default

```
public @interface RequestForEnhancement {
   int    id();
   String synopsis();
   String engineer() default "[unassigned]";
   String date()  default "[unimplemented]";
}
```