Reflexe

Aplikační programování v Javě (BI-APJ) - 8

Ing. Jiří Daněček
Katedra softwarového inženýrství
Fakulta informačních technologií ČVUT Praha

Evropský sociální fond
Praha & EU: Investujeme do vaší budoucnosti

Úvod

- Reflexe (také introspekce) je schopnost programu získat informace o vlastní struktuře.
- Programové konstrukce jsou v reflexi reprezetovány objekty tříd (balík *java.lang.reflection*):
  - *Method,*
  - *Constructor,*
  - *Field,*
  - *Annotation*,
  - atd.
- Reflexe umožňuje programovat v Javě speciální úlohy. Např. interpret skriptového jazyka, který má přístup k JRE.

Reflexní metody třídy Class

- Metody pro získání metod třídy:
  - veřejné včetně zděděných:

```
Method getMethod(String name,
                 Class<?>... parameterTypes)
    throws NoSuchMethodException,
           SecurityException
```

  - všechny veřejné metody včetně zděděných:

```
Method[] getMethods()
             throws SecurityException
```

  - všechny s výjimkou zděděných:

```
Method[] getDeclaredMethods()
             throws SecurityException
```

- Obdobné metody jsou k dispozici pro ostatní prvky tříd: položky (*Field*), konstruktory (*Constructor*), anotace (*Annotation*) atd.

Třídy *Field*, *Method*, *Constructor*

- Třídy reflexní objektů (*Field*, *Method*, *Constructor*) jsou následníky třídy *AccesibleObject*, která definuje metodu:

Annotation[] getAnnotations()

- Jednotlivé reflexní třídy pak mají metody umožňující:
  - *Field* - čtení a nastavování položek,
  - *Method* - volání metod:

```
Object invoke(Object obj, Object... args)
        throws IllegalAccessException,
               IllegalArgumentException,
               InvocationTargetException
```

  - *Constructor* - vytváření instancí:

```
T newInstance(Object... initargs)
      throws InstantiationException,
             IllegalAccessException,
             IllegalArgumentException,
             InvocationTargetException
```

Dynamická proxy třída

- **Dynamická proxy třída** (zkráceně proxy třída) je mechanismus, který umožňuje vytvářet třídy dynamicky tedy v runtimu:
  - proxy třídy implementují zadané rozhraní,
  - od proxy tříd lze instancionovat objekty.
- Pro vytvoření proxy třídy slouží statická metoda *Proxy.getProxyClass*. Parametrem metody je pole rozhraní, která mají být implemetována:

```
static Class<?> getProxyClass(
  ClassLoader loader,
  Class<?>... interfaces)
        throws IllegalArgumentException
```

- Instanci proxy třídy lze pak vytvořit standardně metodou *newInstance* z třídy *Constructor*.

Použití metody getProxyClass

- Příklad:

```
InvocationHandler handler
        = new MyInvocationHandler(...);
Class proxyClass = Proxy.getProxyClass(
          Foo.class.getClassLoader(),
          new Class[] { Foo.class });
Foo f = (Foo) proxyClass. getConstructor(
        new Class[] { InvocationHandler.class }).
        newInstance(new Object[] { handler });
```

- Rozhraní *InvocationHandler* obsahuje metodu, která definuje vlastní chování proxy třídy, pokud je volána nějaká metoda implementovaného rozhraní:

```
Object invoke(Object proxy, Method method,
              Object[] args) throws Throwable
```

Vlastnosti proxy třídy

- Proxy třída:
  - je podtřídou *java.lang.reflect.Proxy,*
  - je public, final,
  - má jeden konstruktor, jehož parametrem je rozhraní *InvocationHandler,*
  - implementuje rozhraní zadaná při jejím vytvoření, tj. reflexní metody *getInterfaces* a *getMethods* se chovají, jakoby byla proxy třída deklarována s klauzulí extends.

Vlastnosti instancí proxy třídy

- Pro instance proxy třídy implementující rozhraní *Foo* platí podmínka:

proxy instanceof Foo

a je tedy možno provádět přetypování:

(Foo) proxy

- Každá instance má při konstrukci přiřazen objekt handleru volání (*InvocationHandler*). Volání metody z rozhraní *Foo* bude delegováno na metodu *invoke* z invokačního handleru.
- Stejně je zpracováno volání metod *hashCode*, *equals*, a *toString* deklarovaných ve třídě *Object.*
- Ostatní metody nejsou proxy delegovány takže se chovají implicitním způsobem.

Metoda newProxyInstance

- Metoda umožňující přímé vytvoření instance proxy třídy:

```
static Object newProxyInstance(
    ClassLoader loader,
    Class<?>[] interfaces,
    InvocationHandler h)
            throws IllegalArgumentException
```

- Příklad:

```
InvocationHandler handler =
                new MyInvocationHandler(...);
Foo f = (Foo) Proxy.newProxyInstance(
                Foo.class.getClassLoader(),
                new Class[] { Foo.class },
                handler);
```