La réflexion

contenu de la section

Les objectifs et les principes

les objectifs et les principes

- permettre à une application de :
  - découvrir à l'exécution son implantation et son environnement
  - découvrir au runtime les annotations (java5)

  - s'autoconfigurer (i.e. de modifier son implantation et son environnement)
  - utiliser des méta-protocoles génériques

- le package java.lang, java.lang.reflect offre des descripteurs pour tous les éléments du langage :
  - la classe Class
  - les classes Field, Constructor, Method, Modifier, Array, …

La description des classes

le descripteur de classe

- son type est java.lang.Class<E>
- singleton créé par le ClassLoader au chargement de la classe
- il permet de :
  - découvrir les informations de type, les modificateurs, les champs, les constructeurs, les méthodes
  - découvrir les informations de sécurité (ProtectionDomain, signers, ...), les annotations (Java5)
  - créer une instance, caster une instance

```
public class Class<E> {
   public static Class forName(String name) throws ClassNotFoundException;          //crée un objet Class de la classe name
   public E newInstance() throws InstanciationException, IllegalAccessException;    //crée une instance de la classe
   public boolean isInterface( );
   public boolean isArray( );

   Class[ ] getInterfaces( );
   Class getSuperClass( );
   Field[ ] getFields( ) throws SecurityException ;                  //retourne un tableau des champs déclarés et hérités
   Field getField( String name) throws SecurityException, NoSuchFieldException;
   Field[ ] getDeclaredFields( ) throws SecurityException;           //retourne un tableau des champs déclarés de la classe
   Field getDeclaredField( String name)  throws SecurityException, NoSuchFieldException;

   Method[ ] getMethods( ) throws SecurityException;                 //retourne un tableau des méthodes de la classe
   Method getMethod(String nm, Class... paramTypes) throws SecurityException, NoSuchMethodException;
   Method[ ] getDeclaredMethods( ) throws SecurityException;     //retourne un tableau des méthodes déclarées de la classe
   Constructor<E> getConstructor(Class... argTypes)
   Constructor[ ] getConstructors();
   ……
}
```

java et la réflexivité (2)

quelques autres classes réflexives

- classes de java.lang.reflect

- class **Constructor<E>**
  String getName();
  Class<?>[ ] getParameterTypes();
  Class<?>[ ] getExceptionTypes();
  Object **newInstance**(Object... args) throws ...
  void **setAccessible**(boolean b);
  .....

- class **Method**
  String getName();
  Class<?>[ ] getParameterTypes();
  Class<?>[ ] getExceptionTypes();
  Class<?> getReturnType();
  Modifier getModifier();
  void **setAccessible**(boolean b);
  Object **invoke**(Object support, Object... args) throws ...

- class **Field**
  String getName();
  Class<?> getType();
  Modifier getModifier();
  void **setAccessible**(boolean b);
  int **getInt**(Object obj), double **getDouble**(Object obj), ...,
  void setInt(Object obj ,int val), void setDouble(Object obj ,double val), ...

java et la réflexivité (3)

un exemple de programme utilisant la réflexivité …

- une forme simplifiée de Junit

```
package divers;
public class MyJUNIT {
    public static void fail(String msg) throws MyJUNITException { throw new MyJUNITException(msg); }

    @SuppressWarnings("unchecked")
    public static void main(String[] args) {
        if (args.length !=1) return;
        try {
            Class clz = Class.forName(args[0]);
            Method[] meths = clz.getDeclaredMethods();
            for (int i = 0; i < meths.length; i++) {
                if (!meths[i].getName().startsWith("test")) continue;
                if (meths[i].getReturnType() != void.class) continue;
                if (meths[i].getParameterTypes().length != 0) continue;
                Object obj = clz.newInstance();
                try {
                    meths[i].invoke(obj);
                    System.out.println("methode " + meths[i].getName() + " OK ");
                } catch(InvocationTargetException e) {
                    Throwable e1 = e.getCause();
                    if (e1 instanceof MyJUNITException) System.out.println("methode " + meths[i].getName() + " KO : " + e1.getMessage());
                    else e.printStackTrace();
                }
            }
        } catch (Exception e) { e.printStackTrace(); }
    }
}
```

```
public class MyJUNITException extends Exception {
    public MyJUNITException(String msg) { super(msg); }
}
```