JSUGA Tech Tips

Christoph Pickl, 2008-06-16

IT SOLUTIONS

Compiler API

IT SOLUTIONS

Compiler API

- "*Compiling with the Java Compiler API*"
  - http://java.sun.com/mailers/techtips/corejava/2007/tt0307.html
- Standardized with Java6 (JSR-199)
  - before `com.sun.tools.javac.Main`
    - not standard, public programming interface
- Also compiles dependent classes
- Two possibilities: simple and advanced
  - both will be covered in next slides

Compiler API - Simple - Roadmap

- Create sourcecode (programatically created)
- Retrieve `JavaCompiler`
  - via `ToolProvider`
- Invoke `run` method
- Check return code
  - 0 == successfull
- Get class via reflection
  - create instance, invoke methods, get/set fields, ...

IT SOLUTIONS

Compiler API - Simple - The Source

- First of all: Create desired sourcecode
  - Could be created dynamically
- Save it in right location
  - Be aware of Eclipse' standard `src` folder

```
package at.sitsolutions.techtips.s4666;

public class Hello {
  public static void sayHello() {
    System.out.println("Hello TechTips!");
  }
}
```

IT SOLUTIONS

Compiler API - Simple - Compile it

```
package at.sitsolutions.techtips.s4666;

import javax.tools.JavaCompiler;
import javax.tools.ToolProvider;

public class CompilerApi {
  public static void main(String[] args) {
    String path = “src/at/sitsolutions/techtips”
                + “/s4666/Hello.java”;
    JavaCompiler compiler =
        ToolProvider.getSystemJavaCompiler();
    if(compiler.run(null, null, null, path) == 0) {
      // do something with fresh compiled class
    } else {
      System.err.println(“Ups, something went wrong!”);
    }
  }
}
```

IT SOLUTIONS

Compiler API - Simple - Use it

- After compiling sourcecode...
  - ... access class reflectively
  - ... invoke static method

```
// assume we have compiled our sourcecode successfully

String className = “at.sitsolutions.techtips.s4666.Hello”;

try {
  Class<?> c = Class.forName(className);
  Method m = c.getDeclaredMethod(“sayHello”, new Class[] {});
  m.invoke(null, new Object[] {}); // prints “Hello TechTips”
} catch(Exception e) {
  e.printStackTrace();
}
```

IT SOLUTIONS

Compiler API - Simple - Core Method

- The `JavaCompiler.run` method in detail:
  - (actually inherited from `javax.tools.Tool`)

```
int run(
 InputSream in,        ... use null for System.in
 OutputStream out,     ... use null for System.out
 OutputStream err,     ... use null for System.err
 String... args        ... files to compile
);
```

IT SOLUTIONS

Compiler API - Advanced

- Benefit of advanced version:
  - Access to error messages
  - More options (usefull if developing an IDE)
- Additionally involved classes:
  - `DiagnosticCollector, JavaFileObject, StandardJavaFileManager, CompilationTask`

```
package at.sitsolutions.techtips.s4666;

public class Hello2 {
  public static void sayHello() {
    System.out.printnl(“Hello TechTips!”);
  }
}
```

IT SOLUTIONS

Compiler API - Advanced - Compile it

```
JavaCompiler compiler = ToolProvider.getSystemJavaCompiler();
DiagnosticCollector<JavaFileObject> diagnostics =
    new DiagnosticCollector<JavaFileObject>();

StandardJavaFileManager manager =
    compiler.getStandardFileManager(diagnostics,
      Locale.ENGLISH, new ISO_8859_11());

String file="src/at/sitsolutions/techtips/s4666/Hello2.java";
Iterable<? extends JavaFileObject> compilationUnits =
    manager.getJavaFileObjects(new String[] {file});

CompilationTask task = compiler.getTask(null, manager,
                  diagnostics, null, null, compilationUnits);
if(task.call() == false) {
  // put error handling in here
}
manager.close();
```

IT SOLUTIONS

Compiler API - Advanced - Error Handling

```
// error handling

for (Diagnostic d : diagnostics.getDiagnostics())
  System.out.printf(
    "Code: %s%nKind: %s%n" +
    "Position: %s (%s/%s)%n" +
    "Source: %s%nMessage: %s%n", d.getCode(), d.getKind(),
    d.getPosition(), d.getStartPosition(), d.getEndPosition(),
    d.getSource(), d.getMessage(null));
}

/*
Code: compiler.err.cant.resolve.location
Kind: ERROR
Position: 123 (113/131)
Source: src\at\sitsolutions\techtips\s4666\Hello2.java
Message: src\at\sitsolutions\techtips\s4666\Hello2.java:5: ↵
  cannot find symbol
*/
```

IT SOLUTIONS

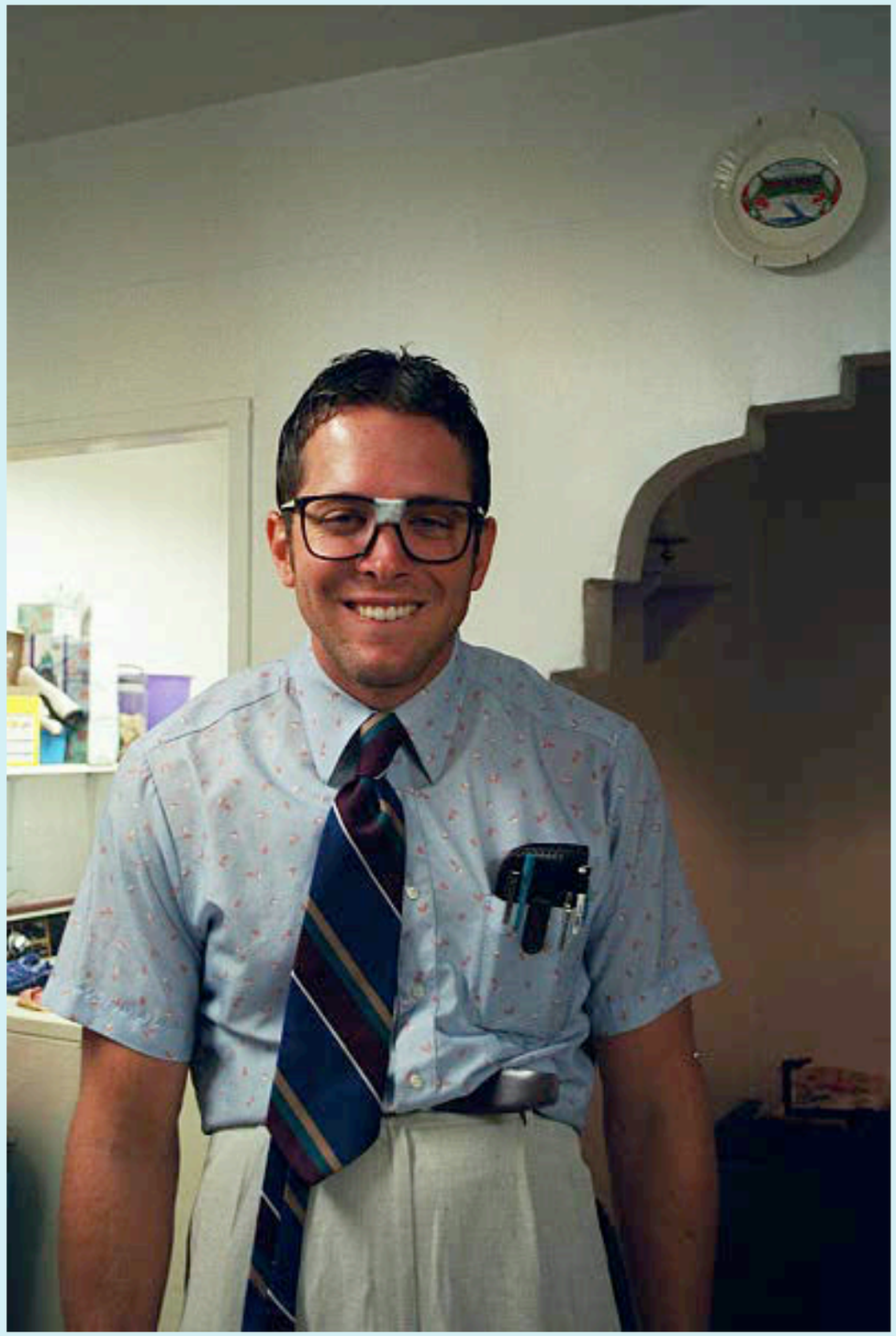
Compiler API - Advanced - Core Method

- `JavaCompiler.getTask` method in detail:

```
CompilationTask getTask(
  Writer out,                       ... use null for System.err
  JavaFileManager mgr,  ... use null for compiler's standard mgr
  DiagnosticListener lst,  ... somehow mandatory
  Iterable<String> options,  ... options to compiler
  Iterable<String> classes,  ... used for annotation processing
  Iterable<? extends JavaFileObject>
     compilationUnits        ... files to compile
);
```

Compiler API - What for?

- Be more dynamic
  - Create classes on-the-fly
  - Let application gain “experience”
- Build your own IDE
  - Provide meaningfull error messages
  - Highlight specific error in sourcecode
- It’s fun



IT SOLUTIONS

Using printf

IT SOLUTIONS

Using printf

- "*Using printf with Customized Formattable Classes*"
  - http://blogs.sun.com/CoreJavaTechTips/entry/using_printf_with_customized_formattable
- Format output available since Java5
  - well known from C language (`%5.2f%n`)
- `Formattable` interface
  - `formatTo(Formatter, int, int, int):void`
  - use locales, flags, width, precision
- Flags available:
  - ALTERNATE (#), LEFT_JUSTIFY (-), UPPERCASE (^)
- Usage just as known from C language

IT SOLUTIONS

Using printf - The Formattable Object

```
public class Alfa implements Formattable {
  private final String stamp12 =
    "ABCDEF123456"; // alternate
  private final String stamp24 =
    "ABCDEF123456GHIJKL123456"; // default

  public void formatTo(
   Formatter formatter, int flags, int width, int precision) {
    StringBuilder sb = new StringBuilder();

    // 1. check flags (alternate)
    // 2. set precision (cut off length)
    // 3. check locale
    // 4. setup output justification

    formatter.format(sb.toString());
  }
}
```

IT SOLUTIONS

Using printf - formatTo Implementation 1/2

```
// 1. check flags (alternate)
boolean alternate = (flags & FormattableFlags.ALTERNATE)
                          == FormattableFlags.ALTERNATE;
alternate |= (precision >= 0 && precision <= 12);
String stamp = (alternate ? stamp12 : stamp24);

// 2. set precision (cut off length)
if (precision == -1 || stamp.length() <= precision)
  sb.append(stamp);
else
  sb.append(stamp.substring(0, precision - 1)).append('*');

// 3. check locale
if (formatter.locale().equals(Locale.CHINESE))
  sb.reverse();

// 4. setup output justification
...
```

IT SOLUTIONS

Using printf - formatTo Implementation 2/2

```
// 4. setup output justification

int n = sb.length();
if (n < width) {
  boolean left = (flags & FormattableFlags.LEFT_JUSTIFY)
                       == FormattableFlags.LEFT_JUSTIFY;
  for (int i = 0; i < (width - n); i++) {
    if (left) {
      sb.append(‘ ’);
    } else {
      sb.insert(0, ‘ ‘);
    }
  }
}
```

IT SOLUTIONS

Using printf - Examples

```
final Alfa alfa = new Alfa();

System.out.printf(“>%s<%n”, alfa);
// >ABCDEF123456GHIJKL123456<
System.out.printf(“>%#s<%n”, alfa);
// >ABCDEF123456<
System.out.printf(“>%.5s<%n”, alfa);
// >ABCD*<
System.out.printf(“>%.8s<%n”, alfa);
// >ABCDEF1*<
System.out.printf(“>%-25s<%n”, alfa);
// >ABCDEF123456GHIJKL123456 <
System.out.printf(“>%15.10s<%n”, alfa);
// >     ABCDEF123*<
System.out.printf(Locale.CHINESE, “>%15.10s<%n”, alfa);
// >     *321FEDCBA<
```

IT SOLUTIONS

Using printf - What for?

- Much more powerfull than simple `toString`
- Unified use of `printf` (even on own objects)
- Usefull in CLI apps
- Again: It’s fun

IT SOLUTIONS

JSUGA Tech Tips

That's all folks

IT SOLUTIONS