



*Solutions based on*  
*Software development*  
*Project Management*  
*ICT Security*

# Java 6

Examples of new features

zondag 2 juli 2006

## ➔ Walter van der Heiden

- ➔ Programming in Java since 2003
- ➔ SCJP 1.4, SCJA, SCWCD
- ➔ Working for clients with C#/.NET and Java/SE, ME and EE

## ➔ Patrick Versteeg

- ➔ ...

- **Compiler API**
- **JDBC 4.0**
- **AWT & Swing**
- **Streaming API for XML (StAX)**
- **Scripting for the Java platform**
- **Java API for XML-Based Webservices 2.0 (JAX-WS)**

## ➤ Interfaces that describes the functions provided by a Java Language Compiler

- Allow invocation of a Java compiler from a Java program using standardized interfaces
- Interfaces enabling the compiler to report diagnostics in a structured way.
- Interfaces enabling clients of the compiler to override how source objects are found. Source objects is a common term for Java source files and Java class files.

## ➤ **JavaCompilerTool**

➤ `JavaCompilerTool compiler =  
ToolProvider.getSystemJavaCompilerTool();`

## ➤ **DiagnosticListener, Diagnostic**

➤ `DiagnosticCollector<JavaFileObject> diagnostics = new  
DiagnosticCollector<JavaFileObject>();`

## ➤ **JavaFileManager**

➤ `StandardJavaFileManager fm =  
compiler.getStandardFileManager(diagnostics);`

## ➤ Example

- `Iterable<? extends JavaFileObject> compilationUnits = fileManager.getJavaFileObjectsFromFiles(Arrays.asList(srcFiles));`
- `compiler.getTask(null, fileManager, diagnostics, null, null, compilationUnits).run();`
- `Class clazz = ClassLoader.getSystemClassLoader().loadClass(srcName); clazz.newInstance();`

## ➤ Example

```
➤ for (Diagnostic diagnostic : diagnostics.getDiagnostics()) {  
➤ System.out.format("Error on line %d%s%n",  
    diagnostic.getLineNumber(), diagnostic);  
➤ }
```

## ➤ Automatic loading of `java.sql.Driver` class:

- `Class.forName("org.apache.derby.jdbc.EmbeddedDriver");`
- `Connection connection = DriverManager.getConnection("jdbc:derby:TestDB");`

## ➤ But...

- `Context context = new InitialContext();`
- `DataSource datasource = (DataSource) context.lookup("jdbc/TestDB");`
- `Connection connection = datasource.getConnection("uid", "pwd");`

## ➔ SQLXML

- ➔ This interface provides methods for accessing the XML value as a String, Reader/Writer, or as a Stream.
- ➔ Access through a Source or set as a Result, for use with XML Parser API's.

## ➤ Example

- `String sql = "insert into content (xml) values (?)";`
- `PreparedStatement prepStmt = conn.prepareStatement(sql);`
- `SQLXML xml = ...`
- `prepStmt.setSQLXML(1, xml);`
- `prepStmt.executeUpdate ();`

## ➤ Example

- `SQLXML sqlxml = resultSet.getSQLXML(content);`
- `InputStream binaryStream = sqlxml.getBinaryStream();`
- `XMLInputFactory factory = XMLInputFactory.newInstance();`
- `XMLStreamReader streamReader =  
factory.createXMLStreamReader(binaryStream);`

## ➔ JDBC Annotations

- ➔ Helps reducing the amount of code that must be written when using the JDBC API's.
- ➔ Will be used in combination with **Query** interfaces and **DataSet** objects
- ➔ Simplify the access and processing of data that is returned as a single SQL result set.
- ➔ **AutoGeneratedKeys, ResultColumn, Select, Update**

## ➤ Example

```
➤ public class Person {  
    public String name;  
}  
  
➤ interface PersonQueries extends BaseQuery {  
    @Select("select name from persons")  
    DataSet<Person> getAllPersons();  
}  
  
➤ PersonQueries personQueries =  
    connection.createQueryObject(PersonQueries.class);  
  
➤ DataSet persons = personQueries.getAllPerson();  
  
➤ for (Person person : persons) {  
    System.out.format("Name: %s%n", person.name);  
}
```

## ➤ Splash screen

- Splash screen displays before the JVM starts. Can be (non-) animated GIF or (non) transparent (GIF, PNG).
- Splash screen is closed as first window is displayed by Swing.
- Usage: **java -splash:LogoAnim.png -cp . YourLegacyApp**
- Or add: **SplashScreen-Image: filename.gif** to manifest.mf of JAR.

## ➤ SplashScreen class provides the API for controlling the splash screen

➤ `getSplashScreen()` return SplashScreen object

➤ `setImageURL(URL imageURL)` replaces current image

➤ Example:

```
URL anotherImage = new URL("file:Logo.gif");
```

```
SplashScreen splashScreen = SplashScreen.getSplashScreen();
```

```
splashScreen.setImageURL(anotherImage);
```

## ➤ **SystemTray** class represents the system tray for a desktop

- SystemTray may contain one or more TrayIcons
- **getSystemTray()** gets the SystemTray instance that represents the desktop's tray area
- **add(TrayIcon trayIcon)** adds a TrayIcon to the SystemTray
- Example:

```
SystemTray systemTray = SystemTray.getSystemTray();
TrayIcon trayIcon = new TrayIcon(read(new
File("red.png")), "Hi, I am red!");
systemTray.add(trayIcon);
```

## ➤ TableRowSorter

- `JTable table = new JTable(model);`
- `RowSorter<TableModel> rowSorter = new TableRowSorter<TableModel>(model);`
- `table.setRowSorter(sorter);`

## ➔ **SwingWorker**<T, V>

- ➔ An abstract class to perform lengthy GUI-interacting tasks in a dedicated thread.
- ➔ **execute()** schedules SwingWorker for execution on a *worker* thread.
- ➔ **doInBackground()** contains long running task.
- ➔ **done()** executed on the *EDT* after the **doInBackground** is finished.
- ➔ **publish(V... chunks)** and **process(V... chunks)** sends/receives data chunks to/from.
- ➔ Type Parameters:
  - ➔ T - result type returned by SwingWorker
  - ➔ V – type SwingWorker's publish/process methods



# Java SE 6 a.k.a. Mustang

Patrick Versteeg

zondag 2 juli 2006

## Agenda

- Streaming API for XML (StAX)
- Scripting for the Java platform
- Java API for XML-Based Webservices 2.0 (JAX-WS)

## Java SE 5 XML api's:

### ➤ Document Object Model API

- DOM structure
- memory intensive

### ➤ Simple API for XML (SAX)

- push parsing
- observer pattern
- Read only

## Java SE 6 XML api's:

- Document Object Model API
- Simple API for XML (SAX)
- Streaming API for XML (Stax)
  - pull parsing
  - Iterator pattern
  - Read and write API

## Two API's

- **low-level, cursor-based API**  
**(XMLStreamWriter/Reader)**
  
- **higher-level, event-based API**  
**(XMLEventWriter/Reader)**

```
FileWriter fileWriter = new FileWriter(fileName);  
XMLOutputFactory xmlOutputFactory = XMLOutputFactory.newInstance();  
XMLStreamWriter xmlStreamWriter = xmlOutputFactory.createXMLStreamWriter(fileWriter);  
xmlStreamWriter.writeStartDocument();  
xmlStreamWriter.writeComment("J-Spring sample document");  
xmlStreamWriter.writeStartElement("Root");  
xmlStreamWriter.writeStartElement("tree");  
xmlStreamWriter.writeAttribute("type", "lets make it an oak");  
xmlStreamWriter.writeStartElement("evergreen");  
xmlStreamWriter.writeCharacters("No");  
xmlStreamWriter.writeEndElement();  
xmlStreamWriter.writeEndElement();
```

```
FileReader fileReader = new FileReader(fileName);
XMLInputFactory xmlInputFactory = XMLInputFactory.newInstance();
XMLStreamReader xmlStreamReader = xmlInputFactory.createXMLStreamReader(fileReader);
while (xmlStreamReader.hasNext()) {
    if (xmlStreamReader.isStartElement() || xmlStreamReader.isEndElement()) {
        System.out.print(xmlStreamReader.getText());
    } else if (xmlStreamReader.isCharacters() && !xmlStreamReader.isWhiteSpace()) {
        System.out.print(xmlStreamReader.getText());
    }
    xmlStreamReader.next();
}
xmlStreamReader.close();
```

## Scripting for the Java platform

This is a standard framework to enable scripting language programs to be executed from and have access to the Java platform

## Main building blocks

- ➔ ScriptEngineManager
- ➔ ScriptEngineFactory
- ➔ ScriptEngine
- ➔ ScriptContext, Bindings

```
public static void exec() throws ScriptException {  
  
    ScriptEngineManager factory = new ScriptEngineManager();  
  
    ScriptEngine engine = factory.getEngineByName("JavaScript");  
  
    engine.eval("print('hello world')");  
  
}
```

## Retrieve results :

- `eval()` method returns by default the value of the last executed expression.
- Bindings object : A map which allows information exchange between an Java application and a script.

- ➔ **Compilable, execute without recompilation**
- ➔ **Invocable, invoke functions/methods**

## Java Resources:

➔ Use `importPackage` function

➔ Call methods on Java objects

String script =

```
    "importPackage(java.util);"
```

```
+ "var date = new Date();"
```

```
+ "print('The current time in milliseconds is:' +  
    date.getTime());";
```

```
engine.eval(script);
```



# Java API for XML-Based Webservices (JAX-WS) 2.0

## A WebService is a regular class with annotations

```
@WebService(serviceName="FaxService")  
  
public class FaxMachine {  
  
    @WebMethod  
  
    public void fax(String text) {  
  
    }  
  
}
```

## Compile the endpoint

```
javac -d bin com/hinttech/mustang/jaxws/ScriptStore.java
```

## Generate the endpoint:

```
wsgen -cp bin -d bin -wsdl  
com.hinttech.mustang.jaxws.ScriptStore
```

## Generate the client:

```
wsimport
```

```
-p com.hinttech.mustang.client
```

```
-d bin
```

```
-wsdllocation
```

```
http://127.0.0.1:8080/scriptstore?WSDL
```

```
bin/ScriptStoreService.wsdl
```



**Run the endpoint:**

```
Java -cp bin  
com.hinttech.mustang.jaxws.ScriptStore
```

➤ Questions?

➤ Presentation can be found @ <http://dotnet.hinttech.com/blogs/walterh/>

➤ Next year... Java 7?

Walter van der Heiden  
Patrick Versteeg

+31-(0)6-000 00 000

walter@Hinttech.com

Patrick.versteeg@Hinttech.com