Nye features i Java 1.6/6.0

Denne artikel beskriver nye features i Java 1.6/6.0, som lige er kommet i endelig udgave.

Den forudsætter et vist kendskab til Java.

Skrevet den **14. Feb 2010** af **arne_v** I kategorien **Programmering / Java** | ★★★☆☆

Historie:
V1.0 - 26/03/2006 - original
V1.1 - 27/03/2006 - tilføj desktop eksempel
V1.2 - 20/05/2006 - tilføj lidt flere småændringer
V1.3 - 04/07/2006 - ret prompt og tilret for beta 2
V1.4 - 11/01/2007 - noter at den er ude i endelig udgave
V1.5 - 21/03/2008 - tilret og udvid JC16.java
V1.6 - 26/12/2008 - små ændringer og tilføj links
V1.7 - 14/02/2010 - smårettelser

Kodenavn Mustang

Java 1.6/6.0 med kodenavnet Mustang udkom i december 2006.

Den afløser Java 1.5/5.0 med kodenavnet Tiger (se evt. artikel http://www.eksperten.dk/guide/167 "Nye features i Java 1.5/5.0").

Den er stadig gal med versionsnummeret.

For at citere fra docs:

"Both version numbers (1.6.0 and 6) are used to identify this release of the Java Platform. Version 6 is the product version, while 1.6.0 is the developer version. The number 6 is used to reflect the evolving level of maturity, stability, scalability and security of Java SE."

"Version 6 is used in the platform and product names as given in this table:"

"Java SE keeps the version number 1.6.0 (or 1.6) in some places that are visible only to developers, or where the version number is parsed by programs. As mentioned, 1.6.0 refers to exactly the same platform and products numbered 6. Version numbers 1.6.0 and 1.6 are used at:"

Jeg er gammeldags og vil fortsætte med at kalde den 1.6 !

Indledning

Jeg vil opremse nogle af de nye features i 1.6 og illustrere med små kode eksempler.

Det er dog ikke sikkert at jeg har opdaget alle de

interessante nyheder.

Console

Der er tilføjet en ny klasse for at give visse ekstra muligheder for console IO.

Mest relevant er nok læsning af passwords uden echo.

Eksempel:

C16.java

```
import java.io.Console;

public class C16 {
    public static void main(String[] args) {
        Console c = System.console();
        String un = c.readLine("Enter username: ");
        String pw = new String(c.readPassword("Enter password: "));
        System.out.println(un + " " + pw);
    }
}
```

Nyttigt men ikke revolutionerende.

Disk info

Der er tilføjet nogle muligheder til File klassen for at få information om disk space.

Eksempel:

DI16.java

```
import java.io.File;

public class DI16 {
    public static void main(String[] args) {
        File f = new File("C:\\");
        long free = f.getFreeSpace();
        long tot = f.getTotalSpace();
        System.out.println(free + " " + tot);
    }
}
```

Nyttigt og længe efterspurgt.

Netkort info

Man har siden 1.4 kunnet hente alle IP adresser på systemet. Nu kan man også hente ethernet adresserne.

Gammel kode:

NI15.java

```
import java.net.NetworkInterface;
import java.net.InetAddress;
import java.util.Enumeration;

public class NI15 {
    public static void main(String[] args) throws Exception {
        Enumeration e = NetworkInterface.getNetworkInterfaces();
        while(e.hasMoreElements()) {
            NetworkInterface ni = (NetworkInterface)e.nextElement();
            System.out.println("Net interface: " + ni.getName());
            Enumeration e2 = ni.getInetAddresses();
            while (e2.hasMoreElements()){
                InetAddress ip = (InetAddress)e2.nextElement();
                System.out.println("IP address: " + ip.getHostAddress());
            }
        }
    }
}
```

Ny kode:

NI16.java

```
import java.net.NetworkInterface;
import java.net.InetAddress;
import java.util.Enumeration;

public class NI16 {
    public static void main(String[] args) throws Exception {
        Enumeration e = NetworkInterface.getNetworkInterfaces();
        while(e.hasMoreElements()) {
            NetworkInterface ni = (NetworkInterface)e.nextElement();
            System.out.println("Net interface: " + ni.getName());
            Enumeration e2 = ni.getInetAddresses();
            while (e2.hasMoreElements()){
                InetAddress ip = (InetAddress)e2.nextElement();
                System.out.println("IP address: " + ip.getHostAddress());
            }
            byte[] mac = ni.getHardwareAddress();
            if(mac != null) {
```

```
System.out.printf("%02X:%02X:%02X:%02X:%02X:%02X\n",mac[0],mac[1],mac[2],mac[3
],mac[4],mac[5]);
            }
        }
    }
}
```

Nyttigt i nogle tilfælde.

Callable compiler

Det har længe været ønsket at kunne kalde compileren fra kode.

Hidtil har man måttet være nødt til at starte den eksterne javac kommando, men nu kan man kalde compileren direkte.

Eksempel på brug af javac:

JC15.java

```
import java.io.PrintWriter;
import java.io.FileWriter;

public class JC15 {
    public static void main(String[] args) throws Exception {
        PrintWriter pw = new PrintWriter(new FileWriter("Temp1.java"));
        pw.println("public class Temp1 {");
        pw.println("    public void test() {");
        pw.println("        System.out.println(\"Temp1 OK\");");
        pw.println("    }");
        pw.println("}");
        pw.close();
        Runtime.getRuntime().exec(new String[] { "javac", "Temp1.java" });
        Class c = Class.forName("Temp1");
        Object o = c.newInstance();
        c.getMethod("test", new Class[] { }).invoke(o, new Object[] { });
    }
}
```

Eksempel på brug af callable compiler:

JC16.java

```
import java.io.ByteArrayInputStream;
import java.io.ByteArrayOutputStream;
import java.io.FileWriter;
import java.io.IOException;
import java.io.InputStream;
```

```
import java.io.OutputStream;
import java.io.PrintStream;
import java.io.PrintWriter;
import java.net.URI;
import java.util.Arrays;
import java.util.HashMap;
import java.util.Map;

import javax.tools.Diagnostic;
import javax.tools.DiagnosticCollector;
import javax.tools.FileObject;
import javax.tools.ForwardingJavaFileManager;
import javax.tools.JavaCompiler;
import javax.tools.JavaFileObject;
import javax.tools.SimpleJavaFileObject;
import javax.tools.StandardJavaFileManager;
import javax.tools.ToolProvider;
import javax.tools.JavaCompiler.CompilationTask;

public class JC16 {
    public static void main(String[] args) throws Exception {
        testFileFile();
        testMemoryFile();
        testFileMemory();
        testMemoryMemory();
    }
    public static void testFileFile() throws Exception {
        PrintWriter pw = new PrintWriter(new FileWriter("Temp2.java"));
        pw.println("public class Temp2 {");
        pw.println("    public void test() {");
        pw.println("        System.out.println(\"Temp2 OK\");");
        pw.println("    }");
        pw.println("}");
        pw.close();
        compileFileFile("Temp2.java", System.err);
        Class<?> c = Class.forName("Temp2");
        Object o = c.newInstance();
        c.getMethod("test", new Class[] { }).invoke(o, new Object[] { });
    }
    public static void compileFileFile(String fnm, PrintStream err) {
        JavaCompiler javac = ToolProvider.getSystemJavaCompiler();
        DiagnosticCollector<JavaFileObject> diacol = new
DiagnosticCollector<JavaFileObject>();
        StandardJavaFileManager sjfm = javac.getStandardFileManager(diacol,
null, null);
        CompilationTask compile = javac.getTask(null, sjfm, diacol,
Arrays.asList(new String[] { }), null,
                                                sjfm.getJavaFileObjects(new
String[] { fnm }));
        boolean status = compile.call();
        if(err != null) {
            err.println("Compile status: " + status);
            for(Diagnostic<? extends JavaFileObject> dia :
diacol.getDiagnostics()) {
                err.println(dia);
```

```
            }
        }
    }
    public static void testMemoryFile() throws Exception {
        String src = "public class Temp3 { public void test() {
System.out.println(\"Temp3 OK\"); } }";
        compileMemoryFile(src, "Temp3", System.err);
        Class<?> c = Class.forName("Temp3");
        Object o = c.newInstance();
        c.getMethod("test", new Class[] { }).invoke(o, new Object[] { });
    }
    public static void compileMemoryFile(String src, String name, PrintStream
err) {
        JavaCompiler javac = ToolProvider.getSystemJavaCompiler();
        DiagnosticCollector<JavaFileObject> diacol = new
DiagnosticCollector<JavaFileObject>();
        StandardJavaFileManager sjfm = javac.getStandardFileManager(diacol,
null, null);
        CompilationTask compile = javac.getTask(null, sjfm, diacol,
Arrays.asList(new String[] { }), null,
                                                Arrays.asList(new
JavaFileObject[] { new MemorySource(name, src) }));
        boolean status = compile.call();
        if(err != null) {
            err.println("Compile status: " + status);
            for(Diagnostic<? extends JavaFileObject> dia :
diacol.getDiagnostics()) {
                err.println(dia);
            }
        }
    }
    public static void testFileMemory() throws Exception {
        PrintWriter pw = new PrintWriter(new FileWriter("Temp4.java"));
        pw.println("public class Temp4 {");
        pw.println("    public void test() {");
        pw.println("        System.out.println(\"Temp4 OK\");");
        pw.println("    }");
        pw.println("}");
        pw.close();
        SpecialClassLoader xcl = new SpecialClassLoader();
        compileFileMemory("Temp4.java", xcl, System.err);
        Class<?> c = Class.forName("Temp4", true, xcl);
        Object o = c.newInstance();
        c.getMethod("test", new Class[] { }).invoke(o, new Object[] { });
    }
    public static void compileFileMemory(String fnm, SpecialClassLoader xcl,
PrintStream err) {
        JavaCompiler javac = ToolProvider.getSystemJavaCompiler();
        DiagnosticCollector<JavaFileObject> diacol = new
DiagnosticCollector<JavaFileObject>();
        StandardJavaFileManager sjfm = javac.getStandardFileManager(diacol,
null, null);
        SpecialJavaFileManager xfm = new SpecialJavaFileManager(sjfm, xcl);
        CompilationTask compile = javac.getTask(null, xfm, diacol,
Arrays.asList(new String[] { }), null,
```

```
                                                    sjfm.getJavaFileObjects(new
String[] { fnm }));
        boolean status = compile.call();
        if(err != null) {
            err.println("Compile status: " + status);
            for(Diagnostic<? extends JavaFileObject> dia :
diacol.getDiagnostics()) {
                err.println(dia);
            }
        }
    }
    public static void testMemoryMemory() throws Exception {
        String src = "public class Temp5 { public void test() {
System.out.println(\"Temp5 OK\"); } }";
        SpecialClassLoader xcl = new SpecialClassLoader();
        compileMemoryMemory(src, "Temp5", xcl, System.err);
        Class<?> c = Class.forName("Temp5", true, xcl);
        Object o = c.newInstance();
        c.getMethod("test", new Class[] { }).invoke(o, new Object[] { });
    }
    public static void compileMemoryMemory(String src, String name,
SpecialClassLoader xcl, PrintStream err) {
        JavaCompiler javac = ToolProvider.getSystemJavaCompiler();
        DiagnosticCollector<JavaFileObject> diacol = new
DiagnosticCollector<JavaFileObject>();
        StandardJavaFileManager sjfm = javac.getStandardFileManager(diacol,
null, null);
        SpecialJavaFileManager xfm = new SpecialJavaFileManager(sjfm, xcl);
        CompilationTask compile = javac.getTask(null, xfm, diacol,
Arrays.asList(new String[] { }), null,
                                                Arrays.asList(new
JavaFileObject[] { new MemorySource(name, src) }));
        boolean status = compile.call();
        if(err != null) {
            err.println("Compile status: " + status);
            for(Diagnostic<? extends JavaFileObject> dia :
diacol.getDiagnostics()) {
                err.println(dia);
            }
        }
    }
}

class MemorySource extends SimpleJavaFileObject {
    private String src;
    public MemorySource(String name, String src) {
        super(URI.create("string:///" + name + ".java"), Kind.SOURCE);
        this.src = src;
    }
    public CharSequence getCharContent(boolean ignoreEncodingErrors) {
        return src;
    }
    public OutputStream openOutputStream() {
        throw new IllegalStateException();
    }
```

```
    public InputStream openInputStream() {
        return new ByteArrayInputStream(src.getBytes());
    }
}

class MemoryByteCode extends SimpleJavaFileObject {
    private ByteArrayOutputStream baos;
    public MemoryByteCode(String name) {
        super(URI.create("byte:///" + name + ".class"), Kind.CLASS);
    }
    public CharSequence getCharContent(boolean ignoreEncodingErrors) {
        throw new IllegalStateException();
    }
    public OutputStream openOutputStream() {
        baos = new ByteArrayOutputStream();
        return baos;
    }
    public InputStream openInputStream() {
        throw new IllegalStateException();
    }
    public byte[] getBytes() {
        return baos.toByteArray();
    }
}

class SpecialJavaFileManager extends
ForwardingJavaFileManager<StandardJavaFileManager> {
    private SpecialClassLoader xcl;
    public SpecialJavaFileManager(StandardJavaFileManager sjfm,
SpecialClassLoader xcl) {
        super(sjfm);
        this.xcl = xcl;
    }
    public JavaFileObject getJavaFileForOutput(Location location, String name,
JavaFileObject.Kind kind, FileObject sibling) throws IOException {
        MemoryByteCode mbc = new MemoryByteCode(name);
        xcl.addClass(name, mbc);
        return mbc;
    }
}

class SpecialClassLoader extends ClassLoader {
    private Map<String,MemoryByteCode> m;
    public SpecialClassLoader() {
        m = new HashMap<String, MemoryByteCode>();
    }
    protected Class<?> findClass(String name) {
        MemoryByteCode mbc = m.get(name);
        return defineClass(name, mbc.getBytes(), 0, mbc.getBytes().length);
    }
    public void addClass(String name, MemoryByteCode mbc) {
        m.put(name, mbc);
    }
}
```

NB: Dette eksempel virker ikke med beta 1 og beta 2 - kun med final. Og eksemplet som virkede med beta 1 og beta 2 virker ikke med final. Man har lavet en del om i dette mellem beta 1 og beta 2 samt mellem beta 2 og final. Og ikke til det bedre efter min mening.

Særdeles nyttigt og formentligt vil dette erstatte mange af de byte code generatorer som eksisterer idag.

Script evaluation

Der er blevet indbygget support for at embedded script engines i Java programmer, så man f.eks. kan få udført en stump JavaScript inde fra sit Java program.

Eksempel:

S16.java

```
import javax.script.ScriptEngineManager;
import javax.script.ScriptEngine;

public class S16 {
    public static void main(String[] args) throws Exception {
        ScriptEngineManager mgr = new ScriptEngineManager();
        ScriptEngine js = mgr.getEngineByName("js");
        System.out.println(js.eval("(1+2)*(3+4)"));
        js.put("a", 1);
        js.put("b", 2);
        js.put("c", 3);
        System.out.println(js.eval("a+b+c"));
        System.out.println(js.eval("var x = 1; var y = 4; var z = 9; x+y+z"));
    }
}
```

Java shipper med JavaScript engine, men det er muligt at tilføje andre script engines.

Jeg ved ikke om det er nyttigt.

JDBC 4.0

Der er mange forbedringer til JDBC i forbindelse med version 4.0, som kommer med Java 1.6.

Nogle af de mest interessante er:
* man behøver ikke længere bruge Class.forName til at loade driveren med
* SQLException har fået sub klasser, så man kan teste mere præcist på hvilken fejl man har
* der er en ny SQLXML data type som opfører sig lidt ligesom CLOB/BLOB
* Connection har fået create metoder til at lave CLOB/BLOB/SQLXML objekter

Alt sammen yderst interessant. Problemet er at der stort set ikke er nogen JDBC 4.0 compliant drivere og at det derfor vil tage nogle år inden man kan udnytte disse features.

Og derfor vil jeg heller ikke vise kode eksempler.

SQL annotations

Denne feature hører egentligt med i JDBC 4.0, men bør kunne bruges uafhængigt af driveren fordi logikken ligger i kode som kommer med selve Java.

Featuren tillader at man laver data gateways ved hjælp af kun et interface og annotations med SQL sætninger.

Traditionel kode (uden data gateway):

A15.java

```
import java.sql.DriverManager;
import java.sql.Connection;
import java.sql.Statement;
import java.sql.ResultSet;

public class A15 {
    public static void main(String[] args) throws Exception {
        Class.forName("com.mysql.jdbc.Driver");
        Connection con =
DriverManager.getConnection("jdbc:mysql://localhost/Test", "", "");
        Statement stmt = con.createStatement();
        ResultSet rs = stmt.executeQuery("SELECT * FROM t1");
        while(rs.next()) {
            int f1 = rs.getInt(1);
            String f2 = rs.getString(2);
            System.out.println(f1 + " " + f2);
        }
        rs.close();
        stmt.executeUpdate("INSERT INTO t1 VALUES (6, 'FFF')");
        stmt.executeUpdate("UPDATE t1 SET f2 = 'FFFFFF' WHERE f1 = " + 6);
        ResultSet rs2 = stmt.executeQuery("SELECT * FROM t1 WHERE f1 > " + 3);
        while(rs2.next()) {
            int f1 = rs2.getInt(1);
            String f2 = rs2.getString(2);
            System.out.println(f1 + " " + f2);
        }
        rs2.close();
        stmt.executeUpdate("DELETE FROM t1 WHERE f1 = " + 6);
        stmt.close();
        con.close();
    }
}
```

Data gateway via annotations med SQL:

A16.java

```
import java.sql.DriverManager;
import java.sql.Connection;
import java.sql.QueryObjectFactory;
import java.sql.DataSet;

public class A16 {
    public static void main(String[] args) throws Exception {
        Class.forName("com.mysql.jdbc.Driver");
        Connection con =
DriverManager.getConnection("jdbc:mysql://localhost/Test", "", "");
        T1Gate t1 = QueryObjectFactory.createDefaultQueryObject(T1Gate.class,
con);
        DataSet<T1> ds = t1.getAll();
        for(T1 row : ds) {
            System.out.println(row.f1 + " " + row.f2);
        }
        t1.insertNew("6","FFF");
        t1.updateOne(6,"FFFFFF");
        DataSet<T1> ds2 = t1.getSome(3);
        for(T1 row : ds2) {
            System.out.println(row.f1 + " " + row.f2);
        }
        t1.deleteOne(6);
        con.close();
    }
}
```

T1.java

```
public class T1 {
    public int f1;
    public String f2;
}
```

T1Gate.java

```
import java.sql.BaseQuery;
import java.sql.DataSet;
import java.sql.Select;
import java.sql.Update;

public interface T1Gate extends BaseQuery {
```

```
    @Select(sql="SELECT * FROM t1")
    DataSet<T1> getAll();
    @Update(sql="INSERT INTO t1 VALUES(?1, ?2)")
    int insertNew(String f1, String f2);
    @Update(sql="UPDATE t1 SET f2 = ?2 WHERE f1 = ?1")
    int updateOne(int f1, String f2);
    @Update(sql="DELETE FROM t1 WHERE f1 = ?1")
    int deleteOne(int f1);
    @Select(sql="SELECT * FROM t1 WHERE f1 > ?1")
    DataSet<T1> getSome(int minf1);
}
```

To bemærkninger:

1) QueryObjectFactory.createDefaultQueryObject(T1Gate.class, con) skal erstattes
af con.createQueryObject(T1Gate.class) når driveren bliver JDBC 4.0 compliant

2) Eksemplet virkede ikke beta 1 (tilsyneladende bliver tallene efter ? ikke fjernet
som de skal når der skal genereres PreparedStatement), men det gør det i beta 2

Nogle vil nok sige at det er en nyttig feature. Jeg er ikke glad for den.
Det er faktisk lykkes at putte implementations logic i et interface !

Array resize

Det er nu blivet nemmere at resize et array.

Gammel kode:

AR15.java

```
public class AR15 {
    public static void main(String[] args) {
        int[] ia = new int[2];
        ia[0] = 1;
        ia[1] = 2;
        int[] ia2 = new int[4];
        System.arraycopy(ia, 0, ia2, 0, 2);
        ia2[2] = 3;
        ia2[3] = 4;
        for(int i = 0; i < ia2.length; i++) {
            System.out.println(ia2[i]);
        }
    }
}
```

Ny kode:

AR16.java

```
import java.util.*;

public class AR16 {
    public static void main(String[] args) {
        int[] ia = new int[2];
        ia[0] = 1;
        ia[1] = 2;
        int[] ia2 = Arrays.copyOf(ia, 4);
        ia2[2] = 3;
        ia2[3] = 4;
        for(int i = 0; i < ia2.length; i++) {
            System.out.println(ia2[i]);
        }
    }
}
```

Meget praktisk.

IDN support

Java understøtter nu IDN.

IDN klassen i java.net har en toASCII og en toUnicode metode
til at konvertere domain navne med non-ASCII tegn i.

IDN16.java

```
import java.net.*;

public class IDN16 {
    public static void main(String[] args) {
    System.out.println(IDN.toASCII("www.rød-grød-med-fløde.dk"));
    }
}
```

Næppe specielt vigtigt.

Wildcard i classpath

En * i classpath betyder alle jar filer i det pågældende dir.

Windows eksempel:

java -classpath .;C:\foobar\lib* MainClass

vil tilføje alle jar filer i C:\foobar\lib til classpath.

Helt klart praktisk.

Andet

Der er tilføjet to nye XML API'er: StAX og JAXB. For mere info se artikel:
http://www.eksperten.dk/guide/1261 "Nye Java XML API'er"

Efter at UNISYS'es patent på LZW komprimering er udløbet, så er skrivning af GIF filer nu muligt uden eksterne libraries.

Java web service API'erne er blevet indlemmet i J2SE. Jeg vil ikke komme med kode eksempler, da det mere er en flytning af funktionalitet end ny funktionalitet.

Der er en del ændringer til AWT og Swing plus support for brug af desktop og systray.

Jeg vil kun komme med et eksempel på brug af desktop til start af default browser.

Før Java 1.6 måtte man lave en windows specific og browser specific løsning som:

B15.java

```
public class B15 {
    public static void main(String[] args) throws Exception {
        Runtime.getRuntime().exec(new String[] { "C:\\Program Files\\Mozilla
Firefox\\firefox.exe", "http://www.eksperten.dk/" });
    }
}
```

Med Java 1.6 kan man lave en generel løsning:

B16.java

```
import java.net.URI;
import java.awt.Desktop;

public class B16 {
    public static void main(String[] args) throws Exception {
        if(Desktop.isDesktopSupported()) {
            Desktop dsktop = Desktop.getDesktop();
            if(dsktop.isSupported(Desktop.Action.BROWSE)) {
                dsktop.browse(new URI("http://www.eksperten.dk/"));
            }
        }
    }
}
```

Kommentar af jensgram d. 27. Mar 2006 | 1

Velskrevet artikel, men det var da nogle kedelige features...

Kommentar af krukken d. 27. Jul 2006 | 2

God og hurtig oversigt. Som altid ved arve_v hvad han snakket om.

Kommentar af mysitesolution d. 14. Jun 2006 | 3

Holder hvad den lover og fin som dine andre artikler...Der er dog en lille fejl, du skriver "Enter username:" 2 gange? dette er vel ikke meningen... Næstne ligegyldigt men...

Kommentar af o-zone d. 22. Feb 2007 | 4

Velskrevet og lige til at gå til!

Kommentar af stich d. 19. May 2006 | 5

Synes eksemplerne kunne være mere illustrative, og ser fx ikke den store grund til at medtage NI15 - ej heller samme kode i NI16.
Der er en god del andre tilføjelser, ændringer osv. (viser et par hurtige Googlinger), men hvor vigtige de hver især er, kommer selvfølgelig an på ens interesseområder (et godt eksempel herpå er nok elliptisk kurve-kryptografi). Der er dog, som forventet, øjensynligt ikke tale om en nær så interessant udgivelse for alm. udviklere, som Java (1.)5 var.

Kommentar af imago-dei d. 28. Mar 2006 | 6

Artiklen er fin nok, men hvis denne artikel lægger vægt på højdepunkterne i den nye udgave af Java, er virkelig ikke meget at hente ved at opgradere.

Kommentar af 2c d. 12. Apr 2006 | 7

God artikel. Den holder hvad den lover, og de nye features demonstreres på en god måde.

Java 6 virker umiddelbart ikke så revolutionerende.

Kommentar af jlykkegaard d. 27. Mar 2006 | 8

udmærket :)

Kommentar af psycosoft-funware d. 02. Apr 2006 | 9

artiklen er godt skrevet, selvom jeg ikke fatter en brik af java ;-)

/FunteX! :-)