javaday

Roma 2 Dicembre 2006

Tiger, Mustang e Dolphin: le nuove caratteristiche di JAVA 5, 6 ... 7(?)

Matteo Baccan matteo@baccan.it JUG Novara

Tiger: le nuove caratteristiche di Java 5

- Relatore: **Matteo Baccan** matteo@baccan.it
- Data: Roma 2/12/2006 10:00
- Area: Programming & development
- Target: Tecnici
- Difficoltà: Media

50 minuti per svelare la tigre, il cavallo e sapere che esiste il delfino.
La nuova versione di Java implementa caratteristiche che da tempo erano attese nello standard del linguaggio: Metadata, Generic Types, AutoBoxing e Unboxing dei tipi primitivi, Static import, gestione dinamica dei Loop e delle Enumeration.
Per Java 6: gestione di SystemTray e motori di scripting.
Vediamo di cosa si tratta e di come poter utilizzare queste nuove feature all'interno dei nostri programmi

Obiettivi

Dopo aver completato questo seminario sarai in grado di:

- Descrivere le nuove caratteristiche di Java 5
- Sfruttare i nuovi costrutti che sono stati introdotti con Tiger, attendendo Mustang (Java 6) e Dolphin (Java 7)
- Creare codice più compatto ed efficiente
- Migliorare lo stile della tua programmazione

Storia

- 1995 Java 1.0
  Prima release
- 1997 Java 1.1
  Gestione degli eventi, ereditarietà, miglioramento della JVM
- 1999 Java 1.2
  Swing, list, set, hashmap
- 2000 Java 1.3
  Performance
- 2001 Java 1.4
  Nuovo IO, XML, ASSERT
- 2004 Java 5
  La svolta!!
- 2006 Java 6 RC a breve sui vostri schermi: http://java.sun.com/
- 2008 java 7 snapshot disponibili su https://jdk7.dev.java.net/

Tiger

- Metadata (Annotations)
- Generics
- AutoBoxing e Auto-Unboxing
- Static import
- Foreach
- Enumeration
- Varargs
- Input/Output formattati (estesa in Java 6)
- Network (estesa in Java 6)
- Diagnostica e monitoring (estesa in Java 6)
- Property: setXXX() getXXX() (Java 7)

Riferimenti

Q: Cos'è JCP? Java Community Process (http://www.jcp.org)

A: Fin dalla sua introduzione, nel 1998, è un processo nato per sviluppare e revisionare le specifiche tecniche di Java e le sue implementazioni. JCP contribuisce all'evoluzione della piattaforma Java cooperando con la comunità internazionale di sviluppatori.

- Java 5 Tiger
  - JSR-014 Generics
  - JSR-175 Metadata
  - JSR-201 Foreach, Boxing/Unboxing, enumeration etc
- Java 6 Mustang
  - JSR-223 javax.script API : Java 6 : esecuzione di PHP, JavaScript, Java, Python, Ruby, Tcl etc etc (https://scripting.dev.java.net/)
  - 202: Class File Update, 199: Compiler API, 269: Annotation Processors, 221: JDBCTM 4.0 software, 223: Scripting, 105: XML Digital Signature, 173: Streaming API for XML, 222: JAXB 2.0, 250: Common Annotations, 181: WS Metadata, 224: JAX-WS 2.0

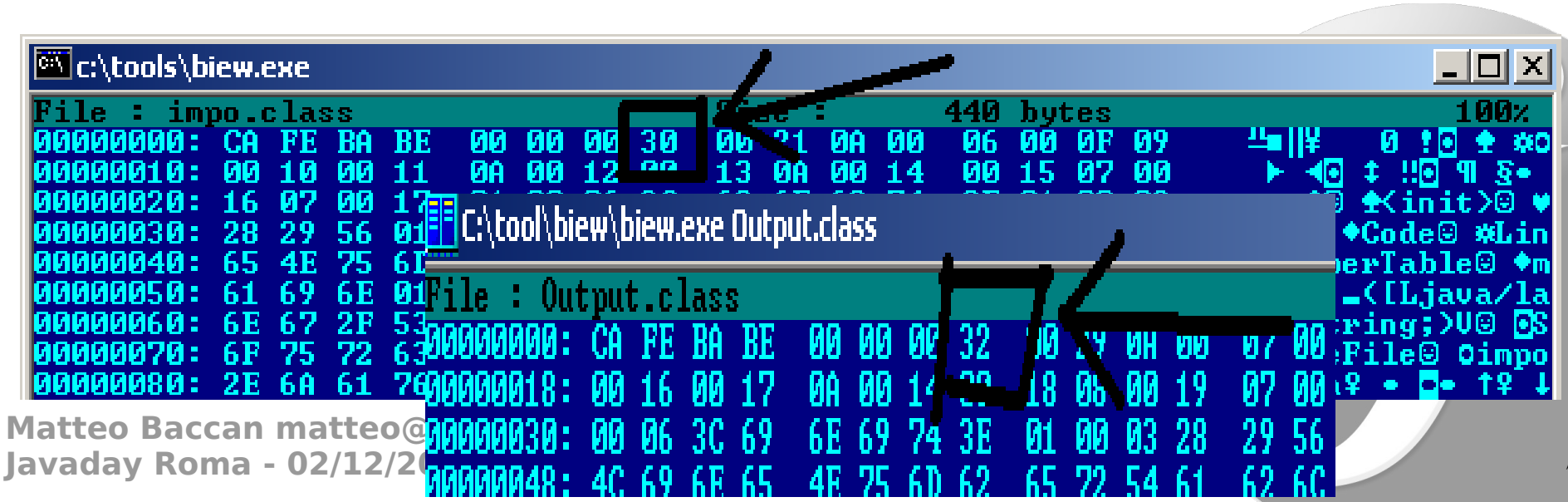
JAVAC

Nuove opzioni di compilazione di Javac:

-source 1.5: Abilita l'uso delle nuove caratteristiche di Java 5 all'interno dei sorgenti (-target 1.5 è implicito)

-target 1.5: Abilita javac a usare ne nuove feature di Java 5 all'interno di librerie e virtual machine

-Xlint: Abilita javac a produrre dei warning su costrutti legali ma logicamente sospetti come, ad esempio, la dichiarazione di una classe Serializable senza la definizione di un serialVersionUID.



Metadata/Annotations

Un sistema per poter arricchire il codice con nuove informazioni

Vantaggi

- Permette a tool che agiscono su codice Java di non ricorrere a file di proprietà esterni
- Permette di semplificare la manipolazione del codice
- Arricchisce i programmi di nuove informazioni
- Non influisce sulle performance

Sintassi

@annotation([param=valore])

Esempio 01

Metadata/Annotations

Enum RetentionPolicy:

Indica la validità dei metadata:

CLASS:

Le annotazioni sono registrate nel file .class dal compilatore ma non sono mantenute dalla VM a runtime

RUNTIME:

Le annotazioni sono registrate nel file .class dal compilatore e mantenute dalla VM a runtime, cosi possono essere lette tramite reflection

SOURCE:

Le annotazioni sono scartate dal compilatore

Generics

- Permette la scrittura di classi generiche (C++ Template)
- Elimina il problema del runtime-casting
- Sposta gli errori nella fase di compilazione

Javac non è il primo compilatore java a supportare i Generics. Il primo è stato:
Pizza. http://pizzacompiler.sourceforge.net/

Esempio 02

Generics – 1.4 senza

```
public static void main(String[] args) {
   List myIntList = new LinkedList(); // Object container
   myIntList.add(new Integer(2));
   myIntList.add("String");
   Integer iValue1 = (Integer)myIntList.get(0);
   Integer iValue2 = (Integer)myIntList.get(1);
   System.out.println( "*"+iValue1.intValue()+"*" );
   System.out.println( "*"+iValue2.intValue()+"*" );
}
```

```
Exception in thread "main" java.lang.ClassCastException: java.lang.String
        at gen14.main(gen14.java:10)
```

Generics – 5 con

```
public static void main(String[] args) {
   List<Integer> myIntList = new LinkedList<Integer>(); //container
   myIntList.add(new Integer(2));
   myIntList.add("String");
   Integer iValue1 = myIntList.get(0);
   Integer iValue2 = myIntList.get(1);
   System.out.println( "*"+iValue1.intValue()+"*" );
   System.out.println( "*"+iValue2.intValue()+"*" );
}
```

```
gen1.java:8: cannot find symbol
symbol  : method add(java.lang.String)
location: interface java.util.List<java.lang.Integer>
    myIntList.add("String");
```

Generics – una classe

```
public class gen2<GEN> {
  GEN uno;
  GEN due;
  public static void main(String[] args) {
     gen2<Integer> myInt2 = new gen2<Integer>();
     myInt2.setUno( new Integer(10) );
     myInt2.setDue( new Integer(7) );
     System.out.println( myInt2.getUno()+myInt2.getDue() );
  }
  public void setUno( GEN arg ) {     uno = arg;  }
  public void setDue( GEN arg ) {     due = arg;  }
  public GEN getUno() {     return uno;  }
  public GEN getDue() {     return due;  }
}
```

Auto boxing/unboxing

Capacità di convertire tipi primitivi in oggetti e viceversa

Presente anche in altri linguaggi, ad esempio: C#

http://msdn.microsoft.com/library/default.asp?url=/library/en-us/csref/html/vclrfBoxingUnboxingPG.asp

Esempio 03

Auto boxing/unboxing - prima

```
private static final Integer ONE = new Integer(1);
public static void main(String[] args) {
    // Frequenza di stringhe
    Map m = new TreeMap();
    for (int i=0; i<args.length; i++) {
        Integer freq = (Integer) m.get(args[i]);
        m.put(args[i], freq==null ? ONE :
            new Integer(freq.intValue() + 1));
    }
    System.out.println(m);
}
```

Auto boxing/unboxing - dopo

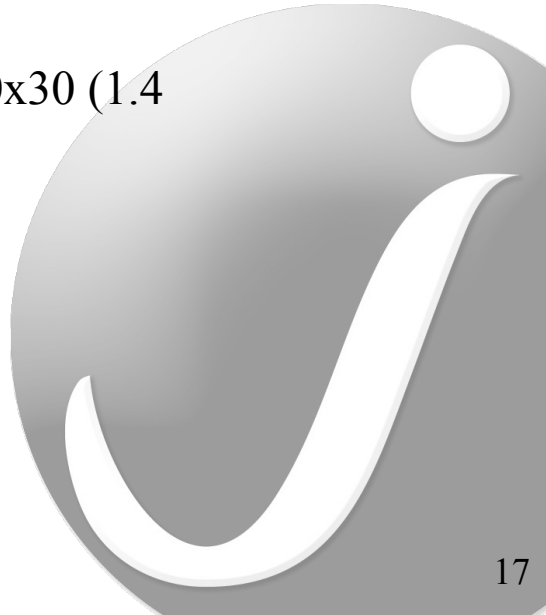
```
public static void main(String[] args) {
    // Frequenza di stringhe
    Map<String,Integer> m = new TreeMap<String,Integer>();
    for (String word : args) {
        Integer freq = m.get(word);
        m.put(word, freq==null ? 1 : freq+1 );
    }
    System.out.println(m);
}
```

Static import

- Permette di importare dei metodi static, risolti a livello di compilazione
- Si possono importare singoli metodi, l'intera classe o l'intero package
- Semplice da usare

NB: All'uso dell'import static cambia la versione del .class: da 0x30 (1.4 compatibile) a 0x31 (1.4 non compatibile)

Esempio 04



Static import

```
import static java.lang.Math.max; // Singolo metodo
import static java.lang.Math.*; // Tutti gli static

public class impo2 {
    public static void main(String[] args) {
       int uno = 10;
       int due = 20;
       System.out.println( max( uno, due ) );
    }
}
```

Foreach

Nuovo modo di scrivere i cicli, il **for** viene potenziato

- Chiarezza: Molto più chiaro da scrivere e leggere, meno codice
- Errori: Minor uso di variabili temporanee, non c'è il rischio di fare cicli annidati incrociati
- Compilatore: Pensa a tutto lui

EnhancedForStatement:

```
for ( FormalParameter : Expression )
    Statement
```

Esempio 05

Foreach – l'evoluzione

```
// Old Style
Vector c = new Vector();
for( Enumeration e = c.elements(); e.hasMoreElements();  )
   String x = (String) e.nextElement();
```

```
// New Style
Vector<String> c = new Vector<String>();
for( String s : c )
   String x = s;
```

Enumerations

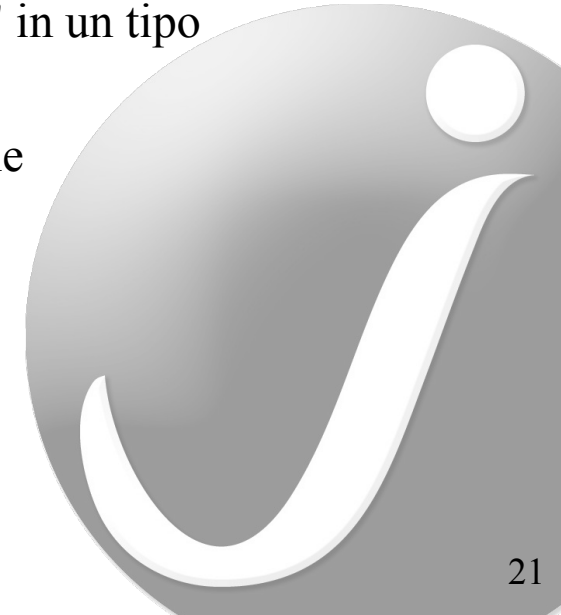
Prima

```
public static final int matteo   = 0;
public static final int andrea    = 1;
public static final int francesco = 2;
```

Problemi

- Non typesafe. Sono solo numeri non li posso "raggruppare" in un tipo
- Le costanti sono "compilate" nel codice
- Se cambio l'ordine o tolgo una costante ho un errore runtime

Esempio 06



Enumerations

Soluzione

Introduzione delle enum

public **enum** nomi { matteo, andrea, francesco };

Gestione del tipo

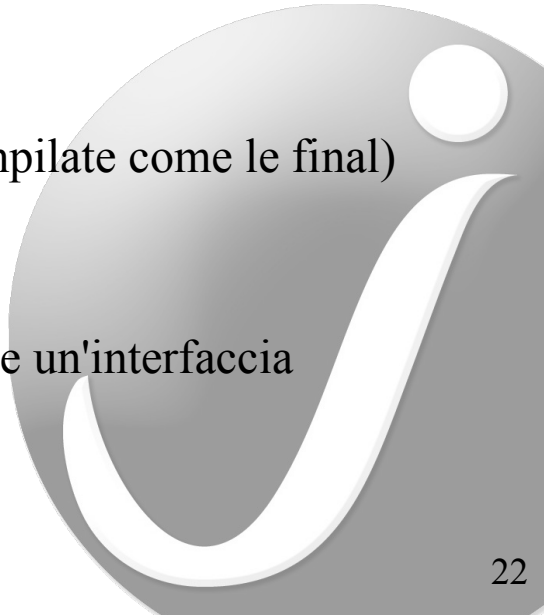
Non compilazione nel codice

Gestione degli errori

Da non confondere con le enum di C (non sono classi, sono compilate come le final)

Vantaggi

- Sono classi e sono quindi estendibili e possono implementare un'interfaccia



Enumerations

```
public enum nomieta { matteo(35), andrea(30), francesco(30);
   private final int eta;
   nomieta( int v ){
      eta = v;
   }
   public int eta(){
      return eta;
   }
};
```

Uno sguardo al decompilato ...

Varargs

Possibilità di definire un numero di parametri variabile

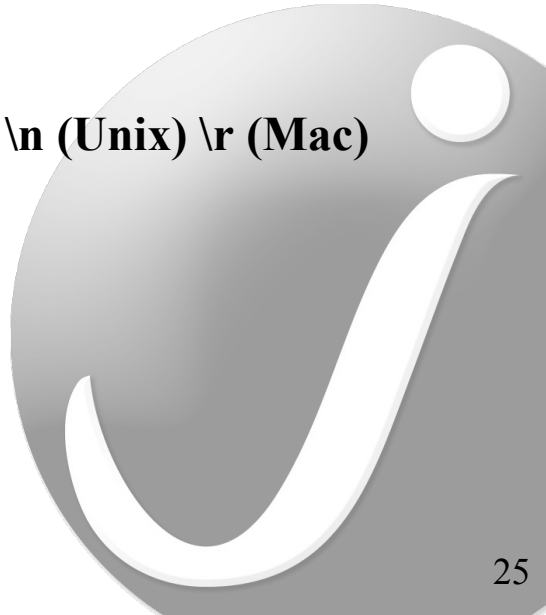
```
public class varTest {
    public static void main(String[] args) {
      varTest x = new varTest();
      x.testArg( "ciao", "io", "sono", "Matteo", 10, 20 );
    }
    void testArg(Object ... args) {
      for( Object s : args ) {
        System.out.println( s );
      }
    }
}
```

Esempio 07

Input/Output formattati

```
// Grazie a varargs abbiamo un output formattato
System.out.printf("[%s] [%5d] [%5x]\n", "Matteo", 33, 2459);
// Input formattato
Scanner s= new Scanner(System.in);
System.out.println( "Ciao come ti chiami? " );
String str = s.next();
System.out.println( "Quanti anni hai? " );
int val    = s.nextInt();
s.close();
// Java 6 introduce %n = in base alla piattaforma genera un \n (Unix) \r (Mac)
     (\r\n) Windows
System.out.printf("[%s] [%5d] [%5x]%n", "Matteo", 33, 2459);
```

Esempio 08



Network

Introduzione dei timeout a livello di classe: **URLConnection.java**

```
public void setConnectTimeout(int timeout) {
    if (timeout < 0)  throw new IllegalArgumentException("...");
    connectTimeout = timeout;
 }
public void setReadTimeout(int timeout) {
    if (timeout < 0)  throw new IllegalArgumentException("...");
    readTimeout = timeout;
}
```

Fino a JDK 1.4: settaggio di sistema comune a tutte le istanze

-D**sun.net.client.defaultConnectTimeout**=...

-D**sun.net.client.defaultReadTimeout**=...

Network

- Introduzione del “PING” nella classe **InetAddress**

  public boolean **isReachable**(int timeout) throws IOException {

  ...

  }
- Miglioramento del supporto dei Cookie (CookieHandler) (Java 6 CookiePolicy, CookieStore)
- Miglioramento della gestione dei Proxy server: si possono gestire le casistiche di non connessione col server
- Fino a JDK 1.4: settaggio di sistema comune a tutte le istanze

  -D**http.proxyHost**=...

  -D**http.proxyPort**=...

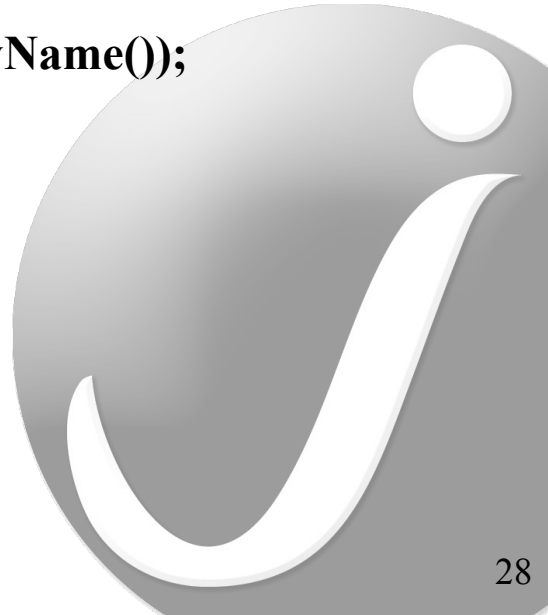
Java 6: NetworkInterface

NetworkInterface.getNetworkInterfaces() : accesso alle informazioni relative alle interfacce di rete attive

```
Enumeration<NetworkInterface> nets =
  NetworkInterface.getNetworkInterfaces();

for (NetworkInterface netint : Collections.list(nets)) {
  console.printf("Display name: %s%n",    netint.getDisplayName());
  console.printf("Name: %s%n", netint.getName());
....
```

Esempio 08a



Java 6: TrayIcon

```
// AWT introduce l'accesso alla SystemTray

if (SystemTray.isSupported()) {
    SystemTray tray = SystemTray.getSystemTray();
....

Image image = Toolkit.getDefaultToolkit().getImage("tray.gif");
PopupMenu popup = new PopupMenu();
...
TrayIcon trayIcon = new TrayIcon(image, "Tray Demo", popup);
```

Esempio 11

Java 6: JSR 223

Integrazione con gli interpreti di script (PHP, Ruby, Javascript, ...)

```
 ScriptEngineManager manager = new ScriptEngineManager();
 List<ScriptEngineFactory> factories = manager.getEngineFactories();
 for (ScriptEngineFactory factory: factories) {
  ScriptEngine engine = factory.getScriptEngine();
}
ScriptEngine engine = manager.getEngineByName("javascript");
try {
  Double hour = (Double)engine.eval( "var date = new Date();" +
      "date.getHours();");
  ...
} catch (ScriptException e) {
  System.err.println(e);
}
```

Esempio 12

Java 6: Varie

```
// isEmpty()
String dummy = "";
dummy.isEmpty();

public boolean isEmpty() {
    return count == 0;
}

// Input di stringhe senza output
Console console = System.console();
char password[] = console.readPassword("Enter password: ");
```

Non abbiamo parlato di

Sicuramente da approfondire

Concurrency Utilities

Sono una serie di costrutti di alto livello che includono: executors, che sono un framework per la creazione di thread, code thread safe, Timers, locks e altre primitive di sincronizzazione

JDBC 4.0

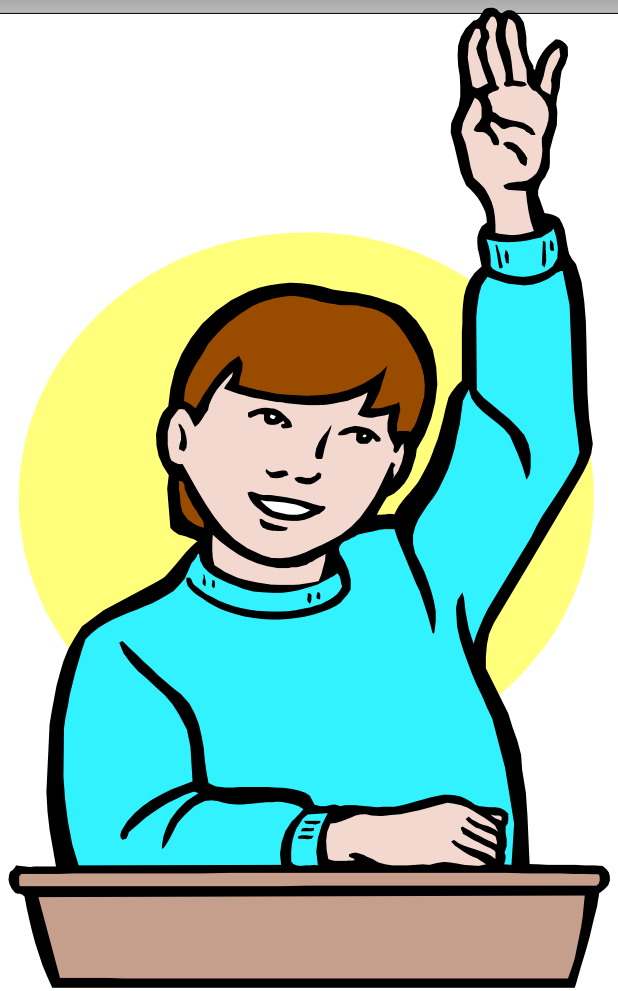
Java Compiler API

Desktop API

Riferimenti

- Java 5 download
  http://java.sun.com/javase/downloads/index.jsp
- Breve storia delle versioni di Java
  http://www.java.com/en/javahistory/
- Java Community Process
  http://www.jcp.org
- Mokabyte – j2se 1.5 beta1
  http://www.mokabyte.it/2004/03/jdk1_5.htm
- Esempi di codice
  http://zamples.com/JspExplorer/samples/samplesJDK1_5.html
- Dove poter scaricare questo materiale
  http://www.baccan.it

Q&A



Grazie

Matteo Baccan
Enterprise Architect
Email: matteo@baccan.it