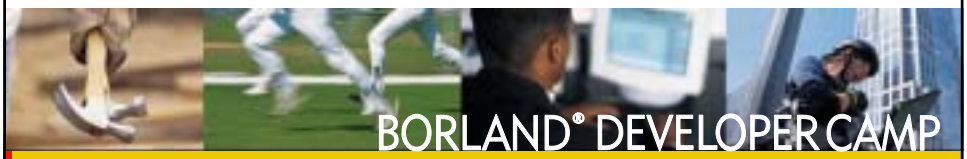


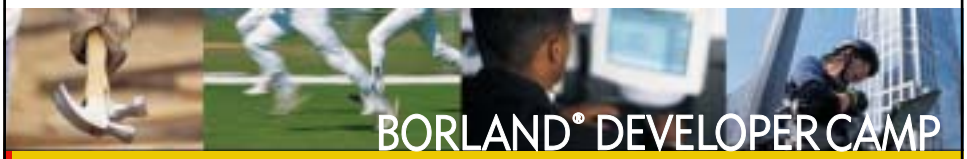
Javaで体験するスクリプト言語の威力

フリーランス
ITアーキテクト
鈴木雄介
ブログ：アークランプ (http://www.arclamp.jp)

Borland

アジェンダ

- Javaにおけるスクリプト言語
  - Scripting API (JSR223)
- スクリプト言語の威力
  - JavaScript (ECMAScript)
  - Groovy
- J2EEでスクリプティング
  - Project Phobos
  - Sarugau JS
- まとめ



Javaにおけるスクリプト言語

Scripting API (JSR223) :概要 1/3

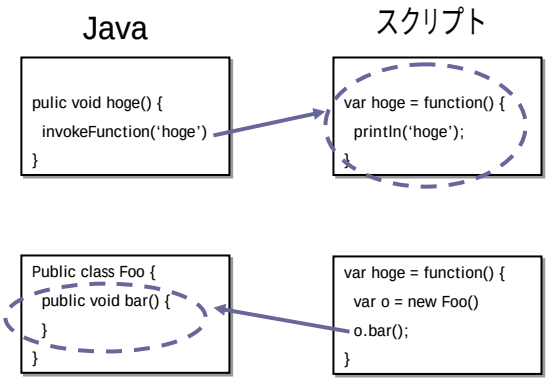
- JavaVM内でJava言語とスクリプト言語をつなぐための標準仕様
- JCP (Java Community Process) で策定
  - JSR 223: Scripting for the Java™ Platform
    - http://jcp.org/en/jsr/detail?id=223
  - 2006年11月6日に最終承認完了 (2003年から検討してました)
- JavaSE6 (Mustang) で採用
  - Scripting API (javax.script)
  - http://java.sun.com/javase/6/docs/technotes/guides/scripting/index.html
  - SUNのJavaSE6 β にJavaScript (ECMAScript) が同梱済み

Scripting API (JSR223) : 概要 2/3

- できること
  - Java言語から「スクリプト言語のファイル」を呼び出し
  - スクリプト言語側でJavaオブジェクトを操作
  - Java言語からインターフェースを通じてスクリプトを操作
  - などなど
- 言語には非依存
  - Apache Jakarta Bean Scripting Framework (BSF) と同じアイデア
    - http://jakarta.apache.org/bsf/
  - JDBCみたいなもの

Scripting API (JSR223) : 概要 3/3

- ようは、Java言語からスクリプト言語が呼べる。逆もOK。



Scripting API (JSR223) : API

- javax.script.ScriptEngine
- javax.script.ScriptEngineManager
- javax.script.Invocable

- javax.script.ScriptContext
- javax.script.Bindings

BORLAND® DEVELOPER CAMP
第3回 ボーランド デベロッパー キャンプ

Scripting API (JSR223) : サンプル

- サンプル
  - 1.直接、スクリプトを実行する
  - 2.スクリプトファイルを実行する
  - 3.スクリプトファイルの関数を呼び出す
  - 4.スクリプトファイルにパラメタを渡す
  - 5.スクリプトファイルからJavaオブジェクト生成する
  - 6.スクリプトの関数をインターフェースを通じて呼び出す

※すべてのコードは、講演後にhttp://www.arclamp.jpからダウンロード可能にします

Scripting API (JSR223) :サンプル1

- 1.直接、スクリプトを実行する
  - src/java
    - Hello1.java

BORLAND® DEVELOPER CAMP
第3回 ボーランド デベロッパー キャンプ

Scripting API (JSR223) :サンプル1

- Hello1.java

```
import javax.script.ScriptEngine;
import javax.script.ScriptEngineManager;
public class Hello1 {
  public static void main(String[] args) throws Exception {
    ScriptEngineManager manager = new ScriptEngineManager();
    ScriptEngine engine = manager
        .getEngineByName("JavaScript");
    //直接スクリプトを実行するパターン
    engine.eval("println(‘Hello World')");
  }
}
```

Scripting API (JSR223) : サンプル2

- 2.スクリプトファイルを実行処理する
  - src/java
    - Hello2.js
    - Hello2.java

Scripting API (JSR223) : サンプル2

- Hello2.js

```
//標準出力
println('Hello World');
```

Scripting API (JSR223) : サンプル2

- Hello2.java

```
…
public class Hello2 {
  public static void main(String[] args) throws Exception {
    engine = …
    engine.eval(new InputStreamReader(Hello2.class
        .getResourceAsStream("hello2.js"), "UTF-8"));
  }
}
```


Scripting API (JSR223) : サンプル3

- 3.スクリプトファイルの関数を呼び出す
  - src/java
    - Hello3.js
    - Hello3.java

Scripting API (JSR223) : サンプル3

- Hello3.js

```
println('init');

var hello = function() {
  println('Hello World');
}

var helloMessage = function(p) {
  println('Hello ' + p);
}
```


Scripting API (JSR223) : サンプル3

- Hello3.java

```
import javax.script.Invocable;
…
public class Hello3 {
  public static void main(String[] args) throws Exception {
   ScriptEngine engine = …
   engine.eval(new InputStreamReader(Hello2.class
        .getResourceAsStream("hello3.js"), "UTF-8"));
    Invocable invocable = (Invocable)engine;
    invocable.invoke("hello", null);
    invocable.invoke("helloMessage",
        new Object[]{"Message"});
  }
}
```

Scripting API (JSR223) : サンプル4

- 4.スクリプトファイルにパラメタを渡す
  - src/java
    - Hello4.js
    - Hello4.java

BORLAND® DEVELOPER CAMP
第3回 ボーランド デベロッパー キャンプ

Scripting API (JSR223) : サンプル4

- Hello4.js

```
println(helloMessage + d.time);
```

Scripting API (JSR223) : サンプル4

- Hello4.java

```
…
public class Hello4 {
  public static void main(String[] args) throws Exception {
    ScriptEngine engine = …
    engine.put("helloMessage", "time=");
    engine.put("d", new Date());
    engine.eval(new InputStreamReader(Hello2.class
        .getResourceAsStream("hello4.js"), "UTF-8"));
  }
}
```


Scripting API (JSR223) : サンプル5

- 5.スクリプトファイルからJavaオブジェクト生成する
  - src/java
    - Hello5.java
    - Hello5.js

Scripting API (JSR223) : サンプル5

- Hello5.java

```
…
public class Hello5 {
  public static void main(String[] args) throws Exception {
    ScriptEngine engine = …
    engine.eval(new InputStreamReader(Hello2.class
        .getResourceAsStream("hello5.js"), "UTF-8"));
  }
  public String getValue() { return "1"; }
  public void helloMessage(String p) {
    System.out.println("hello " + p);
  }
}
```


BORLAND® DEVELOPER CAMP
第3回 ボーランド デベロッパー キャンプ

Scripting API (JSR223) : サンプル5

- Hello5.js

```
var s = new ();
var d = new Packages.java.util.Date();
var o = new Packages.Hello5();
println(d.time);
o.helloMessage('Message' + o.value);
```

Scripting API (JSR223) : サンプル6

- 6.スクリプトの関数をインターフェースを通じて呼び出す
  - src/java
    - MyInterface.java
    - Hello6.js
    - Hello6.java

BORLAND® DEVELOPER CAMP
第3回 ボーランド デベロッパー キャンプ

Scripting API (JSR223) : サンプル6

- MyInterface.java

```
public interface MyInterface {
  void exec(String p);
}
```

Scripting API (JSR223) : サンプル6

- Hello6.js

```
var exec = function(p) {
  println('Hello ' + p);
}
```


Scripting API (JSR223) : サンプル6

- Hello6.java

```
…
public class Hello6 {
  public static void main(String[] args) throws Exception {
    ScriptEngine engine = …
    engine.eval(new InputStreamReader(Hello2.class
        .getResourceAsStream("hello6.js"), "UTF-8"));
    Invocable inv = (Invocable) engine;
    MyInterface i = inv.getInterface(MyInterface.class);
    i.exec("Message");
  }
}
```

Scripting API (JSR223) : jrunscript 1/2

- jrunscript
  - JavaSE6に付属しているスクリプト実行シェル
  - 言語は切替可能
  - JAVA_HOME¥binにあります

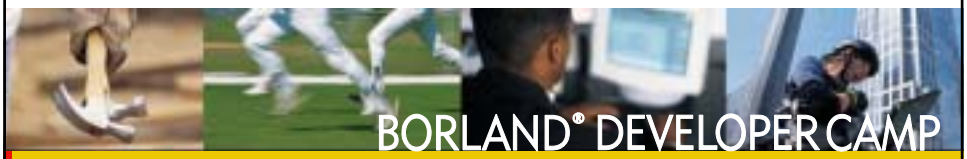
Scripting API (JSR223) : jrunscript 2/2

- コマンドラインから実行

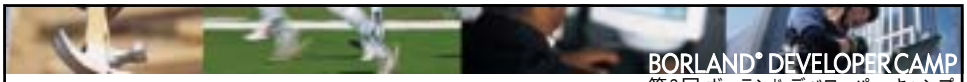
```
C:>jrunscript
js> println('hello world');
hello world
js>exit();
```

- ファイルを読み込み実行

```
c:>jrunscript -f src/java/Hello2.js
Hello World
```



スクリプト言語の威力



スクリプトの威力

- スクリプトの有利な点
  - 型決定が動的、変数スコープが柔軟
  - シンプルで直感的な記述
  - 頭を柔らかくして”感じる”べし
- スクリプトの不利な点
  - コンパイラによる型チェックはできない
    - Javaの有利な点

JavaScript(ECMAScript)

- ネットスケープ2.0より提供。当時はLiveScript
- Javaとは確かに似ているけど、なんの互換性もない
- 1997年よりECMA（ヨーロッパ電子計算機工業会）にてECMAScriptとして標準化
  - http://www.ecma-international.org/publications/standards/Ecma-262.htm
- 現在は**A**synchronous **Ja**vaScript + **X**MLで有名
- 参照：http://ja.wikipedia.org/wiki/JavaScript

Groovy

- Javaで記述されたスクリプト言語
  - James Strachan, Bob McWhirterによってCodehausにて開発
    - http://groovy.codehaus.org/
  - Ruby, Python, and Smalltalk™にインスパイアされている
- JSR241：The Groovy Programming Language として標準化作業中
  - http://jcp.org/en/jsr/detail?id=241
- 使える
  - テンプレートエンジン、コード生成、XML操作など、充実したライブラリが存在

スクリプトの威力：サンプル

- 1.MapとList
- 2.関数を引数や返り値に
- 3.Closureとレキシカルスコープ

スクリプトの威力：サンプル1

- 1.MapとList
  - src/javascript/maplist.js
  - src/groovy/maplist.groovy
  - src/java/MapList.java
- 処理
  - a,b,cのリストを作る
  - キーa値A、キーb値B、キーc値Cのマップ
  - マップとリストの複雑な組み合わせ

スクリプトの威力：サンプル1

- MapList.js

```
var l = ['a','b','c'];
println(l[1]);
var m = {a:'A', b:'B', c:'C'};
println(m.b);
var lm = ['a', {b:'B', c:[1,2,3]} ];
println(lm[1].c[2]);
```


BORLAND® DEVELOPER CAMP
第3回 ボーランド デベロッパー キャンプ

スクリプトの威力：サンプル1

- MapList.groovy

```
def l = ['a','b','c']
println l[1]
def m = ['a':'A', 'b':'B', 'c':'C']
println m.b
def lm = ['a', [b:'B', c:[1,2,3]] ]
println lm[1].c[2]
```

スクリプトの威力：サンプル1

- MapList.java

```
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

public class MapList {
  public static void main(String[] args) throws Exception {
    List<String> l = new ArrayList<String>();
    l.add("a");
    l.add("b");
    l.add("b");
    System.out.println(l.get(1));

    HashMap<String, String> m = new HashMap<String, String>();
    m.put("a", "A");
    m.put("b", "B");
    m.put("c", "C");
    System.out.println(m.get("b"));
```

右上へ続く

```
    List<Object> lm = new ArrayList<Object>();
    lm.add("a");
    Map<String, Object> b = new HashMap<String, Object>();
    b.put("b", "B");
    List<Integer> c = new ArrayList<Integer>();
    c.add(1);
    c.add(2);
    c.add(3);
    b.put("c", c);
    lm.add(b);
    System.out.println(((List) ((Map) lm.get(1)).get("c"))
        .get(2));
  }
}
```


スクリプトの威力：サンプル2

- 2.関数を引数や返り値に
  - src/javascript/functions.js
  - src/groovy/function.groovy
- 処理
  - 関数を引数に、返り値に
    - ()をつけると関数が呼び出せる
    - ()をつけない場合は関数そのもの
    - Groovyの場合は、いろいろ省略できる
  - 高階関数であれば、引数の関数を呼び出せる

スクリプトの威力：サンプル2

- functions.js

```
1  println('a'); //関数を呼び出し
2  println('[' + println + ']'); //関数そのものを表示
3
4  var runFunction = function(f) {
5    f('Hello'); //引数の関数を実行
6  }
7  runFunction( println ); //関数そのものを引数に
8
9  var returnFunction = function() {
10   return println; //関数そのものを返り値に
11 }
12 var f = returnFunction();
13 f('Hello2');
14 returnFunction()('Hello3'); //返り値の関数をそのまま呼び出す
```


BORLAND® DEVELOPER CAMP
第3回 ボーランド デベロッパー キャンプ

スクリプトの威力：サンプル3

- functions.groovy

```
1  println('a')
2  println 'a' //()は空白1文字に省略可能
3  println {return println} //{}を関数として解釈
4
5  def sample1(p) { println(p)}
6  sample1('Hello1');
7  def sample2 = { p -> println(p) } //->で引数をしめす
8  sample2('Hello2')
9
10 def runFunction(f) { f('Hello3') }
11 runFunction( sample2 )
12
13 def returnFunction = { return sample2 }
14 returnFunction()('Hello4')
```

スクリプトの威力：サンプル3

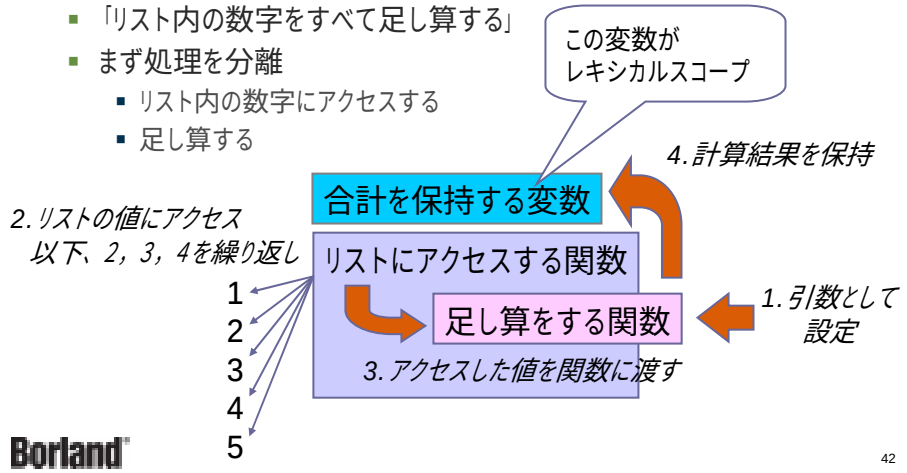
- 3.Closure
  - src/javascript/closure.js
  - src/groovy/closure.groovy
  - src/java/Closure.java

- クロージャー
  - 内側の関数が、実行位置ブロックの外側にある変数にアクセスできる（レキシカル・スコープ）
  - Javaでは実装できない（でもJavaSE7で導入されるかも）
    - http://blogs.sun.com/ahe/resource/closures.pdf

BORLAND® DEVELOPER CAMP
第3回 ボーランド デベロッパー キャンプ

スクリプトの威力：サンプル3

- 処理
  - 「リスト内の数字をすべて足し算する」
  - まず処理を分離
    - リスト内の数字にアクセスする
    - 足し算する



スクリプトの威力：サンプル3

- Closure by JavaScript

```
var forEach = function(l, f) {
  for ( i=0;i<l.length;i++ ) {
    f(l[i]);
  }
}
var list = [1,2,3,4,5];
var total=0;
forEach(list, function(v){total+=v});

println(total);
```


スクリプトの威力：サンプル3

- Closure by Groovy

```
def list = [1,2,3,4,5]
def total = 0
list.each {i -> total += i}
println total
```

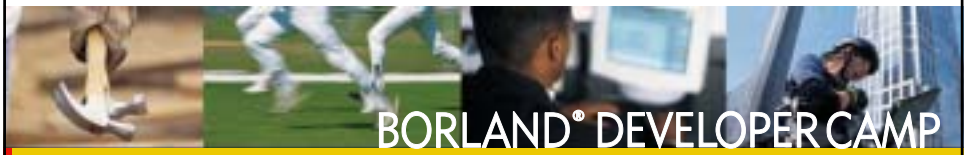
- Groovyではライブラリとして Closure対応機能を用意
  - http://groovy.codehaus.org/Closures

スクリプトの威力：サンプル3

- Closure by Java

```
public class Closure {
  public static void main(String[] args) {
    int[] list = new int[] { 1, 2, 3, 4, 5 };
    int total = 0;
    forEach(list, new Function() {
      public void exec(int i) { total += i; } // ここでコンパイルエラー！
    });
    System.out.println(total);
  }

  public static void forEach(int[] l, Function f) {
    for (int i : l) { f.exec(i); }
  }
  public interface Function { void exec(int i); }
}
```

J2EEでスクリプティング

Project Phobos 1/4

- Scripting APIを利用したWeb層アプリケーション
- Project GlassFishの一部としてSUNメンバーが実装
  - https://phobos.dev.java.net/
- MVCモデルを採用（Strutsのような）
  - モデルは、スクリプトのMapを利用
  - ビューは、JSPシンタックス（<%%>）でJavaScriptを実行するejs形式をサポート
  - コントローラーは、URLに対応するスクリプトファイルの関数を実行

BORLAND® DEVELOPER CAMP
第3回 ボーランド デベロッパー キャンプ

Project Phobos 2/4

- サンプルを実行
  - AJAXList
    - WEBサイトのドキュメント -> samples – 2007/07/24
    - WARをダウンロードして、Servletコンテナにデプロイ でOK
  - 構造
    - WEB-INF/applicationがアプリケーションコード
      - controller：list.js
      - script：index.js（/でアクセスされたときに実行される）
      - view：list.ejs
      - startup.js
    - WEB-INF/その他はフレームワークのコード

Project Phobos 3/4

- 処理概要
  - ルートのURLにアクセス
    - index.jsが実行
    - “/list/show“にリダイレクト
  - /list/showにアクセス
    - controller/list.jsの関数showを実行
    - view/list.ejsをレンダリング
  - “Add to List”を押す
    - クライアントの関数submitData()が実行
    - “/list/compute”に値とコマンドとしてaddを送信
  - /list/computeにアクセス
    - controller/list.jsの関数computeを実行
    - リストの一覧をしめすXMLを返却

Project Phobos 4/4

- まだ、使うのは危険
  - 実験的要素が散見される
  - そもそもフレームワーク全体をスクリプトで記述する意味があるのか不明（Java言語でいいのでは…）
  - 細かいユーティリティが全然なくて、ものすごく開発しにくい
- 未来への期待
  - JSR223対応なのでマルチ言語。JRuby対応の後が見える
  - 考え方はJSF。ManagedBeanもスクリプトでいいかも？
  - 確かにStrutsよりもコード量は少ない（半分ぐらいになるはず）
    - クラス宣言、インポート宣言ない、設定ファイルない、ActionFormない
    - 設定ファイルの削減は暗黙ルール。暗黙ルールは強力（Seasarのように）

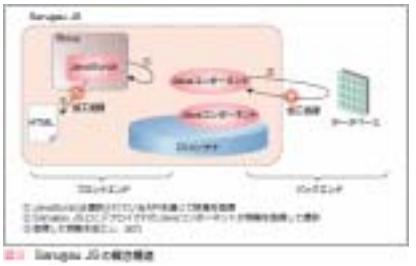
Sarugau JS 1/3

- “D”HTMLテンプレートエンジン
  - 読み：さるごー じぇいえす
  - オープンソースで開発（CCの帰属2.1日本ライセンス）
  - http://www.sarugau.org/js/
- Ajaxで使われているDHTMLを、そのままサーバサイドで
  - そもそものコンセプトはデザイナとエンジニアの分業の実現
    - DHTMLならHTMLコーダーでも使えるのでは
    - JavaScriptのグローバル変数を呼ぶ感じで、Javaオブジェクト使えればいいのでは
  - Ajax用ライブラリも利用可能
    - ex. prototype.js, Mochikit

Sarugau JS 2/3

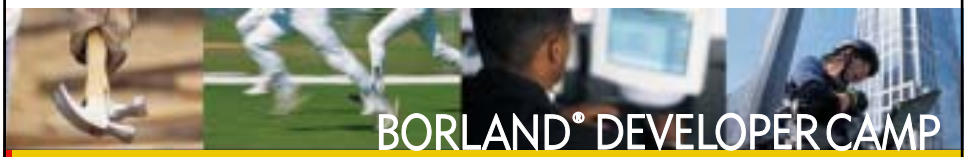
- 構造概要
  - DHTML処理 on サーバー
    - リクエストされたURLに対応したHTMLをDOM化
    - 対応したスクリプトで処理
    - 結果のDOMを書き出す
  - ブラウザの動きを再現
    - document変数など
    - MapやListへのアクセスをJavaScriptっぽく
  - JavaScript内からDIコンテナに簡単アクセス
    - services.hogehoge

DBマガジン　2006年6月号より



Sarugau JS 3/3

- 効果
  - ビジネスロジック・エンジニアとUIエンジニアの分業が実現。サービスレイヤーをDIコンテナで実現
  - JavaScriptやDOMは使いにくいが、Ajaxライブラリが強い
  - テストはそんなにたいへんではない
    - というか、テストがすごくたいへんになる複雑なコードをスクリプト言語で書いてはいけない
- 課題
  - Javaとスクリプト言語（とXML）の使い分け箇所
  - サービス層のAPI設計（どこまでUI側に自由にさせるの？）



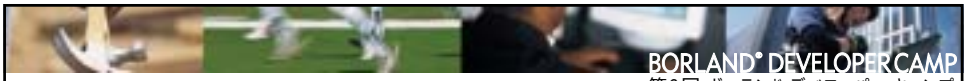
まとめ

まとめ 1/2

- JavaVMにおけるマルチ言語化の流れ
  - Javaはビジネスロジックの記述に最適
    - 静的型宣言。コンパイラ、IDE...
    - たくさんのエンジニアが同時に実装しても安心
  - スクリプト言語はシンプルで直感的
    - 動的型付け。美しい構文
      - IDEは時間の問題
    - ともかく作って、ばりばり変える
  - スクリプト言語がJavaに置き換わるわけではない
    - どちらも使える

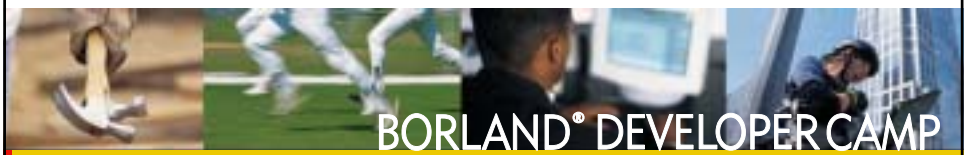
まとめ 2/2

- どちらも使っていくべき？
  - YES
- アプリケーション部位（レイヤー、コンポーネントなど）に応じて使い分ける
  - 開発者の分業が大前提。分業すると道具の専門性が強くなる
  - すでにJSPがある
  - スクリプト言語のよさはJSTLで体験済み
  - XMLは構造に向くが条件分岐ができない
    - 設定ファイルの代替としてスクリプト言語が使えるかも



スクリプトの威力を体験できた？

Borland®



BORLAND® DEVELOPER CAMP

ありがとうございました

Borland®