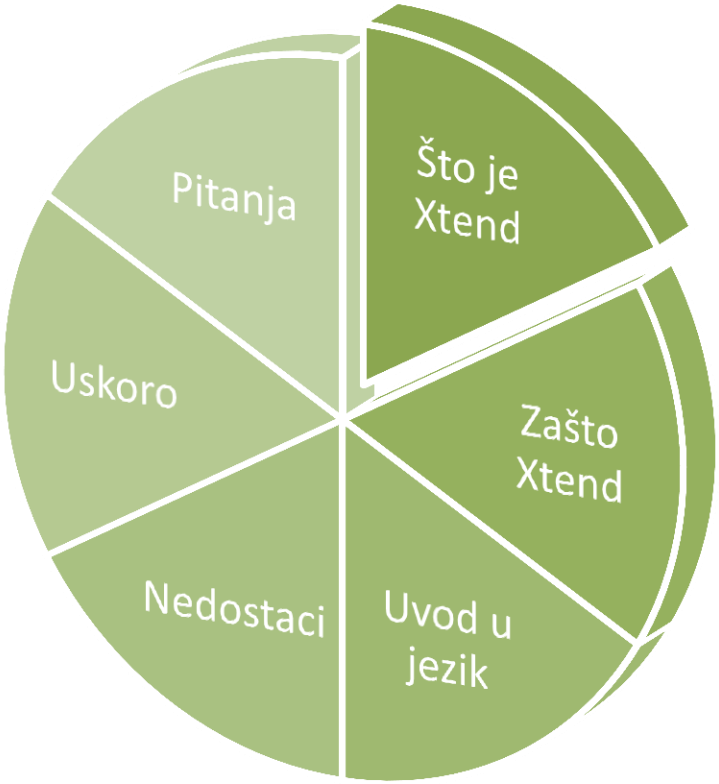
Xtend - prihvatljiva Java

Ivica Lončar

iloncar@croz.net

CROZ d.o.o.


Što je Xtend
Zašto Xtend
Uvod u jezik
Nedostaci
Uskoro
Pitanja

Što je Xtend

- Programski jezik
- Na JVM-u
- Xtend = Java + extra features by Xtend

  + pomoćne runtime biblioteke

- Kompajlira se u source kod
  - Ne u bytecode kao Groovy ili Scala
  - Slično CoffeeScript (generira Javascript)

Zašto Xtend

- JVM
- Kompatibilnost s Javom
- Malo boilerplate koda
- Type inference
- Lambde
- Operator overloading

Java 8?

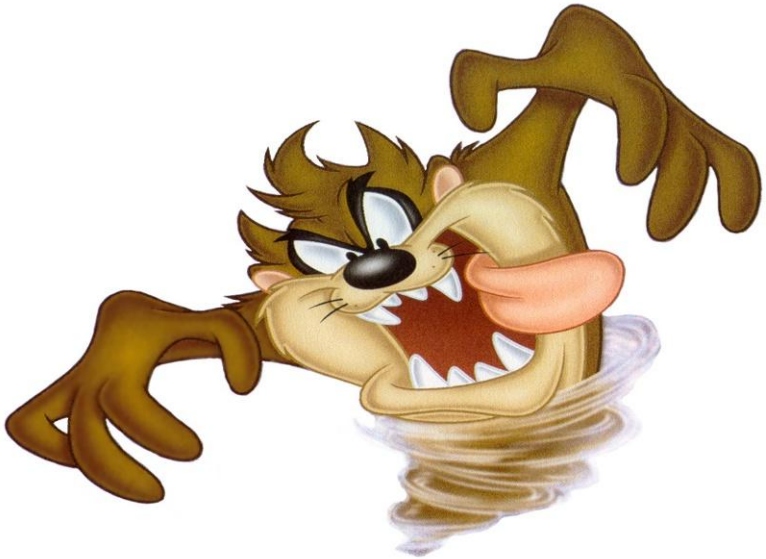
- Iz iskustva:
  - Vrijeme potrebno da nova inačica jezika uđe u produkciju u RH je oko 2 godine
  - Većina projekata NISU green field
- Java 8 – službeno u ljeto sljedeće godine
- Xtend – dostupan danas

- DA, kada Java 8 bude dostupna ☺

Tooling support

- Eclipse project
- Odličan tooling
  - U usporedbi s ostalim alternativnim jezicima
- Debugger
  - Paralelno debugiranje Xtend i Java koda
- Maven
  - Xtend kompajler kao maven plugin

Uvod u Xtend



Klase

- Default: public klase
- Konstruktor: new()

```
 9  class DataChannel {
10      OutputStream os
11
12      new(OutputStream os) {
13          this.os = os
14      }
15
16      def send(File file) {
17          var FileInputStream fis = new FileInputStream(file)
18          fis.copy(os)
19      }
```

Fields

- val - Final, mora imati pridruženu vrijednost
- var - Non final

```
19      def private ResponseCode acceptNonErors(ControlChannel c) {
20          val resp = c.readResponse
21          switch(resp.code) {
```

Generated Code

/ftpclient/src/main/xtend-gen/net/ilx/ftpclient/ftp/FtpProtocol.java

```
22  */
23  private ResponseCode acceptNonErors(final ControlChannel c) {
24    ResponseCode _xblockexpression = null;
25    {
26      final FtpResponse resp = c.readResponse();
```

Metode

- Mogu biti public (default), protected ili private
- Povratne vrijednosti se ne moraju definirati (inferred return types)

```
def void send(String command, String value) {
    writer.write(command + " " + value)
```

Generated Code

/ftpclient/src/main/xtend-gen/net/ilx/ftpclient/ftp/ControlChannel.java

```
53  public void send(final String command, final String value) {
54    try {
55      String plus = (command + " ");
```

Metode

- Override

```
34
35    override toString() {
36        '''
37        user     = «username»
```

- Static

```
1  class Demo {
2
3      def static createString() {
```

- Varargs

```
8      def printAll(String... strings) {
9          strings.forEach[ s | println(s) ]
10     }
11 }
```

- Abstract (moraju imati povratni tip)

Metode

- Apstraktne
  - Moraju imati definiran povratni tip

```
14 abstract class MyAbstractClass {
15   def String abstractMethod()
16 }
17
18
```

- Generičke

```
19   def <T> second(List<T> elements) {
20     elements.get(1)
21   }
```

```
 7 public <T extends Object> T second(final List<T> elements) {
 8   T _get = elements.get(1);
 9   return _get;
10 }
```

Anotacije

- Kao u Javi
- Procesuirane annotacije
  - @Data
  - @Property

```
11      @Property
12      @Parameter(names="-u", description="
13      String username = "user"
```

```
13  @Parameter(names = "-u", description =
14  private String _username = "user";
15
16  public String getUsername() {
17    return this._username;
18  }
19
20  public void setUsername(final String use
21    this._username = username;
22  }
```

Expressions

- Ne postoji statement, sve je expression
  - Čak i blok koda
- U prijevodu, sve može vratiti vrijednost, npr. if izraz ili try-catch blok

```
39      def expressYourself() {
40          val data = try {
41              fileContentsToString('data.txt')
42          } catch (IOException e) {
43              'dummy data'
44          }
45          data
46      }
```

Type casts, type literal

- As

```
val majorType = responseLine.subSequence(0,1)
code = switch (majorType) {
    case "1":  new WaitForReply(responseLine)
    case "2":  new Success(responseLine)
    case "3":  new Intermidiate(responseLine)
    case "4":  new Error(responseLine)
    case "5":  new FatalError(responseLine)
    default: new Unknown()
} as ResponseCode
```

- Typeof

```
private static val Logger logger = LoggerFactory::getLogger(typeof(ControlChannel))
```

Operator overloading

- Predefinirana lista
- Npr.

e1 => e2	e1.operator_doubleArrow(e2)

Pristup propertijima

- Umjesto gettera i settera jednostavno se referencira naziv property-ja

```
socket = new Socket(addr, options.port)
```

```
int _port = options.getPort();
Socket _socket = new Socket(addr, _port);
```

Static access

```
var stacktrace = Thread::currentThread.stackTrace
```

Null safe feature call

```
69⊖     def getLog() {
70          logWriter?.flush
```

```
119     if (this.logWriter!=null) this.logWriter.flush();
120     _xblockexpression = (this.log);
```

If expression

```
def if_expression(String foo) {
    val bar = if (foo != "") "foobar" else "bar"
    bar
}
```

```
public String if_expression(final String foo) {
  String _xblockexpression = null;
  {
    String _xifexpression = null;
    boolean _notEquals = (!Objects.equal(foo, ""));
    if (_notEquals) {
      _xifexpression = "foobar";
    } else {
      _xifexpression = "bar";
    }
    final String bar = _xifexpression;
    _xblockexpression = (bar);
  }
  return _xblockexpression;
}
```

Switch expression

```
def static switch_expression(Object obj) {
    val res = switch obj {
        Number case obj.intValue > 100_000: "BIIG n
        Number: "NUMBER"
        case null: "NULL"
        default: "UNKNOWN"
    }
    res
}

def static void main(String[] args) {
    println(switch_expression(null))
    println(switch_expression(100_001BI))
    println(switch_expression(100))
}
```

```
<terminated> Prin
NULL
BIIG number
NUMBER
```

```
public static String switch_expression(final Object obj) {
  String _xblockexpression = null;
  {
    String _switchResult = null;
    boolean _matched = false;
    if (!_matched) {
      if (obj instanceof Number) {
        final Number _number = (Number)obj;
        int _intValue = _number.intValue();
        boolean _greaterThan = (_intValue > 100000);
        if (_greaterThan) {
          _matched=true;
          _switchResult = "BIIG number";
        }
      }
    }
    if (!_matched) {
      if (obj instanceof Number) {
        final Number _number = (Number)obj;
        _matched=true;
        _switchResult = "NUMBER";
      }
    }
    if (!_matched) {
      if (Objects.equal(obj,null)) {
        _matched=true;
        _switchResult = "NULL";
      }
    }
```

Loop-ing

- For loop
  - Izraz!

```
var List<File> files = new ArrayList
for (f: clientOptions.files) {
    var file = new File(f)
    files += file
}
```

  - Return type je void!

Loop-ing

- While:

```
while (predicate) expression
```

- Do while:

```
do expression while (predicate)
```

Exceptions

- Checked exceptions se pretvaraju u runtime!
  - Sneaky throws iz Lomboka!

```
        }
        catch (ParameterException ex) {
            println(ex.message)
            println("Try with -h for usage description.")
        }


        return
```

```
} catch (final Throwable _t) {
  if (_t instanceof ParameterException) {
    final ParameterException ex = (ParameterException)_t;
    String _message = ex.getMessage();
    InputOutput.<String>println(_message);
    InputOutput.<String>println("Try with -h for usage description.");
  } else {
    throw Exceptions.sneakyThrow(_t);
  }
}
return;
```

Exceptions

- Try-catch je izraz:

```
val name = try {
    person.name
  } catch (NullPointerException e) {
    "no name"
  }
```

Template expressions

```
override toString() {
    '''
    user     = «username»
    password = «password»
    server   = «server»
    files    = «FOR file: files BEFORE '<' SEPARATOR ',' AFTER '>'»«file»«ENDFOR»
    '''
}
```

- Poravnavanje!
- Mini templating jezik!

Dispatch metode

- polimorfizam

```
class Printer {
    def dispatch printType(Number x) {
        "it's some number"
    }

    def dispatch printType(Integer x) {
      "it's an int"
    }

    def dispatch printType(Void x) {
      "it's null"
    }
}
```

```
@SuppressWarnings("all")
public class Printer {
 protected String _printType(final Number x) {
  return "it\'s some number";
 }

 protected String _printType(final Integer x) {
  return "it\'s an int";
 }

 protected String _printType(final Void x) {
  return "it\'s null";
 }

 public String printType(final Number x) {
  if (x instanceof Integer) {
   return _printType((Integer)x);
  } else if (x != null) {
   return _printType(x);
  } else if (x == null) {
   return _printType((Void)null);
  } else {
   throw new IllegalArgumentException("Unhandled parameter types: " +
    Arrays.<Object>asList(x).toString());
  }
 }
}
```

Lambda expressions

```
def private format(StackTraceElement[] stackTrace) {
    stackTrace.join("\n", "\t\n", "\n", [se | "\t" + se.toString])
}
```

```
82 private String format(final StackTraceElement[] stackTrace) {
83   final Function1<StackTraceElement,String> _function = new Function1<StackTraceElement,String>() {
84     public String apply(final StackTraceElement se) {
85       String _string = se.toString();
86       String _plus = ("\t" + _string);
87       return _plus;
88     }
89   };
90   String _join = IterableExtensions.<StackTraceElement>join(((Iterable<StackTraceElement>)Conversions.doWrapArray(stackTrace)), "\n", "\t\n", "\n", _function);
91   return _join;
```

Lambda expressions

```
        val foo = new Person("foo", "bar")
        val (Person)=>String p = [ name ]
        println (p.apply(foo))
    }
}

@Data
class Person {
    String name
    String lastName;
}
```

- Ispisati ce ”foo”

Implicitne varijable

- this
  - Kao u Javi
- it
  - Može biti ‘’shadowed‘’:

```
        val foo = new Person("foo", "bar")
        val (Person)=>String p = [
            val it = new Person("da", "bar")
            name
        ]
        println (p.apply(foo))
    }
}

@Data
class Person {
    String name
    String lastName;
}
```

- it.nesto ima veći prioritet od this.nesto!

Extension methods

- Local extensions
- Extension imports
- Extension fields

Local extensions

```
class MyClass {
  def doSomething(String obj) {
    println("extension called")
  }

  def extensionCall(String obj) {
    obj.doSomething()  // calls this.doSomething(obj)
  }

  def static main(String[] args) {
    val mc = new MyClass
    mc.extensionCall("foobar")
    return
  }

}
```

Extension imports

```
import static org.eclipse.xtext.xbase.lib.StringExtensions.*

class SimpleExtension {

    def static void main(String[] args) {
        println("pero".toFirstUpper)
    }
}
```

- U StringExtensions:

```
 */
@Pure
public static String toFirstUpper(String s) {
    if (s == null || s.length() == 0)
        return s;
    if (Character.isUpperCase(s.charAt(0)))
        return s;
    if (s.length() == 1)
        return s.toUpperCase();
    return s.substring(0, 1).toUpperCase() + s.substring(1);
}
```

Extension fields

```
package samples;
interface EntityPersistence {
    public void save(Entity e);

    public void update(Entity e);

    public void delete(Entity e);
}
```

```
@Data
class Entity {
    String name;
    String lastName;
}
```

```
class Factory {
    def static EntityPersistence get(Class<EntityPersistence> ep) {
        return new EntityPersistenceImpl
    }
}
```

```
public class FieldExtensions {
    extension EntityPersistence ep = Factory::get(typeof(EntityPersistence))

    def m() {
        val foo = new Entity("foo", "bar")
        foo.save
        foo.delete
    }

    def static void main(String[] args) {
        new FieldExtensions => [
            m
        ]
    }
}
```

Extension fields

```
@Component
public class FieldExtensions {
    @Inject
    extension EntityPersistence ep

    def m() {
        val foo = new Entity("foo", "bar")
        foo.save
        foo.delete
    }
}
```

```
class SpringFieldExtensions {

    def static main(String [] args) {
        val ctx = new AnnotationConfigApplicationContext(typeof(EntityPersistenceImpl), typeof(FieldExtensions));
        val bean = ctx.getBean(typeof(FieldExtensions));
        bean.m
    }
}
```

Primjer #1

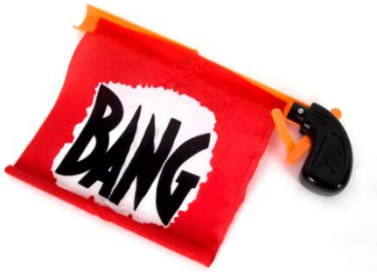
```
def buildDom() {
    new Html => [
      head [
        it.title [$("HTML with Xtend")]
      ]
      body [
        h1 [$("HTML with Xtend")]
        p [$("this format can be used as an alternative to templat

        // an element with attributes and text content
        a("http://www.xtend-lang.org") [$("Xtend")]

        // mixed content
        p [
          $("This is some ")
          b[$("mixed")]
          $(" text. For more see the ")
          a("http://www.xtend-lang.org")[$("Xtend")]
          $(" project")
        ]
        p [$("More text.")]
      ]
    ]
}
```

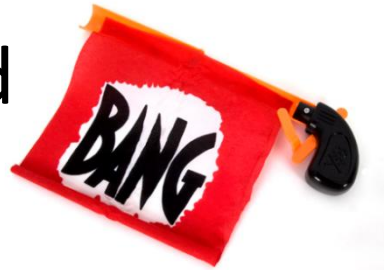
Primjer #2

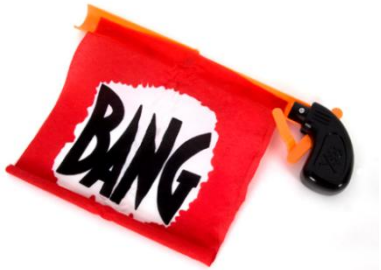
```
@Test def distances() {
    assertEquals(15.km, 13.km + 2_000.m)
    assertEquals(30.km, (13.km + 2_000.m) * 2)
}

@Test def time() {
    assertEquals(1.h, 65.sec + 59.min - 5_000.msec)
}

@Test def speed() {
    assertEquals(42.km/h, (40_000.m + 2.km) / 60.min)
}
```

Nedostaci

- Ne zna za anonimne klase -> go-to Java
- Runtime exceptioni su ponekad
- Refactoring - d'n'd
- Code formatting
- Synchronized keyword
- 'c' je string a ne character!
- Extension je ključna riječ ☹







Xtend 2.4 - uskoro

- 2.4.0
  - Prosinac 2012
  - Active annotations
    - Apt, annotations on steroids
    - Ideje: scala macros, spring roo, grails…
  - Orion editor
  - Gwt lib
    - ☺


Pitanja???
IWORM

Linkovi

- http://www.eclipse.org/xtend/
- http://blog.efftinge.de/2012/10/current-development-and-future-plans.html
- https://oracleus.activeevents.com/connect/sessionDetail.ww?SESSION_ID=6630