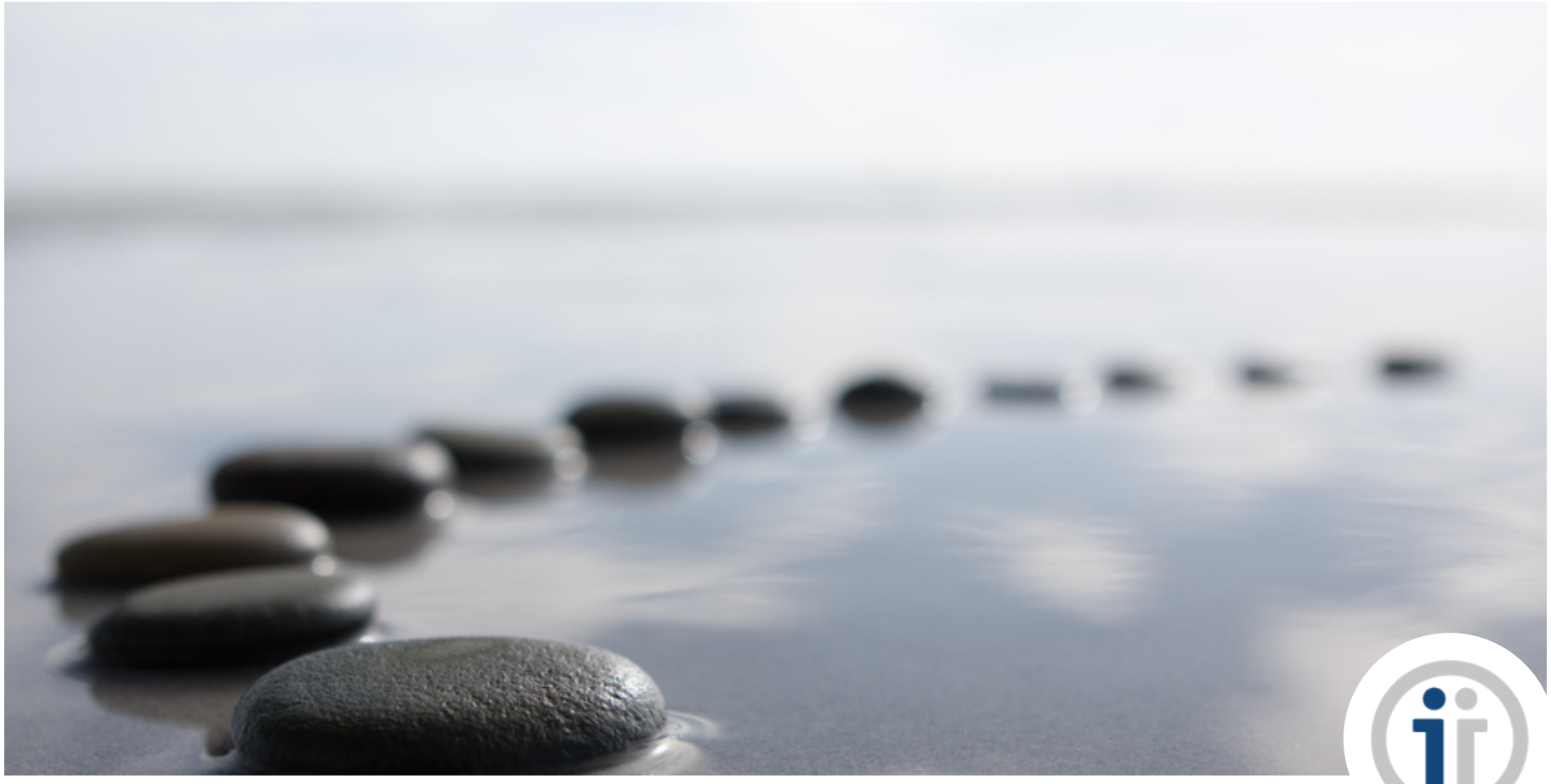
itemis



Interpreting Xtend - How to do it and why

Marco Eilers, July 3, 2015

Overview

1 What is Xtend?

2 The How

3 The Why

Xtend as a language

Xtend as a language

- Java-like structure
- Compiles to Java 5 or 8 source code

Xtend as a language

- Java-like structure
- Compiles to Java 5 or 8 source code
- Type inference
- Cleaner syntax
- Extension methods, lambdas, polymorphic dispatch, rich strings, ...

Xtend as a language

- Java-like structure
- Compiles to Java 5 or 8 source code
- Type inference
- Cleaner syntax
- Extension methods, lambdas, polymorphic dispatch, rich strings, ...
- Awesome

Xtend as a model

Xtend as a model

- Xtend is an Xtext language
- An Xtend class is an EMF model

Xtend as a model

- Xtend is an Xtext language
- An Xtend class is an EMF model
- Xtend uses Xbase for its expressions

Xtend as a model

- Xtend is an Xtext language
- An Xtend class is an EMF model
- Xtend uses Xbase for its expressions
- $\rightarrow$ Xtend wraps Xbase expressions in methods and classes

The How: Basics

The How: Basics

- Use Xtext tooling

The How: Basics

- Use Xtext tooling
- Extend the Xbase interpreter

The How: Basics

- Use Xtext tooling
- Extend the Xbase interpreter
- Add object scope, fields
- Add context switches for method calls
- Add new expressions
- Add polymorphic dispatch and other mechanics
- Add enums, interfaces etc.

The How: Basics

- Use Xtext tooling
- Extend the Xbase interpreter
- Add object scope, fields
- Add context switches for method calls
- Add new expressions
- Add polymorphic dispatch and other mechanics
- Add enums, interfaces etc.
- Done?

The problem

The problem

Virtually all Xtend programs freely mix Xtend code with Java code.*

The problem

Virtually all Xtend programs freely mix Xtend code with Java code.*

*(There are some restrictions)

Java-Xtend-interaction

Java-Xtend-interaction

Method calls Xtend to Java:

Java-Xtend-interaction

Method calls Xtend to Java:

- Decide whether to call Xtend or Java
  - Possibly dispatch in the Xtend case

Java-Xtend-interaction

Method calls Xtend to Java:

- Decide whether to call Xtend or Java
  - Possibly dispatch in the Xtend case
- Polymorphic decision based on runtime type

Java-Xtend-interaction

Method calls Xtend to Java:

- Decide whether to call Xtend or Java
  - Possibly dispatch in the Xtend case
- Polymorphic decision based on runtime type
- Important: Java method needs to be accessible

Java-Xtend-interaction

Method calls Xtend to Java:

- Decide whether to call Xtend or Java
  - Possibly dispatch in the Xtend case
- Polymorphic decision based on runtime type
- Important: Java method needs to be accessible

Method calls from Java to Xtend:

Java-Xtend-interaction

Method calls Xtend to Java:

- Decide whether to call Xtend or Java
  - Possibly dispatch in the Xtend case
- Polymorphic decision based on runtime type
- Important: Java method needs to be accessible

Method calls from Java to Xtend:

- Java method calls may stay in the Java realm

Java-Xtend-interaction

Method calls Xtend to Java:

- Decide whether to call Xtend or Java
  - Possibly dispatch in the Xtend case
- Polymorphic decision based on runtime type
- Important: Java method needs to be accessible

Method calls from Java to Xtend:

- Java method calls may stay in the Java realm
- Or not. In that case, interpreter has to step in somehow.

Java-Xtend-interaction

Method calls Xtend to Java:

- Decide whether to call Xtend or Java
  - Possibly dispatch in the Xtend case
- Polymorphic decision based on runtime type
- Important: Java method needs to be accessible

Method calls from Java to Xtend:

- Java method calls may stay in the Java realm
- Or not. In that case, interpreter has to step in somehow.
- Again, decision may depend on method overrides in Xtend class and runtime type

Java-Xtend-interaction

Method calls Xtend to Java:

- Decide whether to call Xtend or Java
  - Possibly dispatch in the Xtend case
- Polymorphic decision based on runtime type
- Important: Java method needs to be accessible

Method calls from Java to Xtend:

- Java method calls may stay in the Java realm
- Or not. In that case, interpreter has to step in somehow.
- Again, decision may depend on method overrides in Xtend class and runtime type

Reflection?

Solution

Attempt number zero

Create a class "XtendObject"!

Solution

Attempt number zero

Create a class "XtendObject"!

Does not work.

Solution

Attempt number one

Solution

Attempt number one

- Use proxies!

Solution

Attempt number one

- Use proxies!
- Java proxies are too weak, use Javassist!

Solution

Attempt number one

- Use proxies!
- Java proxies are too weak, use Javassist!

- Messy, behaviour may depend on caller.
- No reflection
- Constructors are problematic

Solution

Attempt number two

Solution

Attempt number two

- Create classes using Javassist!

Solution

Attempt number two

- Create classes using Javassist!
- Create correct fields, methods etc.

Solution

Attempt number two

- Create classes using Javassist!
- Create correct fields, methods etc.
- Methods delegate to the interpreter

Solution

Attempt number two

- Create classes using Javassist!
- Create correct fields, methods etc.
- Methods delegate to the interpreter
- JVM handles a lot of messy issues

Solution

Attempt number two

- Create classes using Javassist!
- Create correct fields, methods etc.
- Methods delegate to the interpreter
- JVM handles a lot of messy issues

- Messy if compiled classes exist, ClassLoader magic needed
- Otherwise very natural solution that solves a lot of problems.

Limitations

Limitations

It works, but...

Limitations

It works, but...

- No Active Annotations
- RichString evaluation is somewhat messy
- Prototype status

Why an interpreter?

Why an interpreter?

1. Dynamically create and execute code at runtime

Why an interpreter?

1. Dynamically create and execute code at runtime
2. No Java compiler needed

Why an interpreter?

1. Dynamically create and execute code at runtime
2. No Java compiler needed
3. Execution can be manipulated and/or tracked

Why an interpreter?

1. Dynamically create and execute code at runtime
2. No Java compiler needed
3. Execution can be manipulated and/or tracked
4. For Xtend: Existing tooling is very helpful

Why an interpreter?

1. Dynamically create and execute code at runtime
2. No Java compiler needed
3. Execution can be manipulated and/or tracked
4. For Xtend: Existing tooling is very helpful
5. I like writing interpreters

Applications

Some ideas:

Applications

Some ideas:

- Generate Xtend code using Xtend and execute it right away

Applications

Some ideas:

- Generate Xtend code using Xtend and execute it right away
- Dynamically create data structures and behaviour

Applications

Some ideas:

- Generate Xtend code using Xtend and execute it right away
- Dynamically create data structures and behaviour
- Command line REPL

Applications

Some ideas:

- Generate Xtend code using Xtend and execute it right away
- Dynamically create data structures and behaviour
- Command line REPL
- Tracing program execution

Applications

Some ideas:

- Generate Xtend code using Xtend and execute it right away
- Dynamically create data structures and behaviour
- Command line REPL
- Tracing program execution
  - e.g. model transformations

And now...

Demo!

Thank you for your attention

Use it, give us feedback, improve it, extend it!

`https://github.com/kbirken/xtendency`