Write cool scalable enterprise application tests with Xtend & embedded DSLs

Boris Brodski

EclipseCon Europe 2014
Outline

1. Introduction
   - Conventional tests
   - Three types of tests

2. New Xtend-based technologies
   - Xtend & test contrib
   - XFactory
   - JMockit-Xtend

3. Demo
   - Live-Demo
Outline

1. **Introduction**
   - Conventional tests
     - Three types of tests

2. **New Xtend-based technologies**
   - Xtend & test contrib
   - XFactory
   - JMockit-Xtend

3. **Demo**
   - Live-Demo
Common problems of conventional tests

- Writing tests is boring
Introduction

New Xtend-based technologies

Demo

Conventional tests

Three types of tests

Common problems of conventional tests

Writing tests is boring

You need to rethink the entire use case

Testing afterwards is trickier than writing implementation

Low value of the tests

Watching slow tests running is demotivating

Causes tests to be ran infrequently

Result: lots of broken tests

Fixing broken tests is annoying

Good knowledge of the entire system is required

Boris Brodski

Write cool scalable enterprise application tests with Xtend & embedded DSLs

EclipseCon Europe 2014 4/26
Common problems of conventional tests

- Writing tests is boring
  - You need to rethink the entire use case
  - Testing afterwards is trickier than writing implementation
Common problems of conventional tests

- Writing tests is boring
  - You need to rethink the entire use case
  - Testing afterwards is trickier than writing implemention

- Watching slow tests running is demotivating
Introduction

New Xtend-based technologies

Demo

Conventional tests

Three types of tests

Common problems of conventional tests

Writing tests is boring

You need to rethink the entire use case

Testing afterwards is trickier than writing implementation

Low value of the tests

Watching slow tests running is demotivating

Causes tests to be run infrequently

Result: lots of broken tests

Fixing broken tests is annoying

Boris Brodski

Write cool scalable enterprise application tests with Xtend & embedded DSLs

EclipseCon Europe 2014 4/26
Common problems of conventional tests

- Writing tests is boring
  - You need to rethink the entire use case
  - Testing afterwards is trickier than writing implementation

- Watching slow tests running is demotivating
  - Causes tests to be ran infrequently
  - Result: lots of broken tests
Common problems of conventional tests

- Writing tests is boring
  - You need to rethink the entire use case
  - Testing afterwards is trickier than writing implementation
  - Low value of the tests

- Watching slow tests running is demotivating
  - Causes tests to be ran infrequently
  - Result: lots of broken tests
Common problems of conventional tests

- Writing tests is boring
  - You need to rethink the entire use case
  - Testing afterwards is trickier than writing implementation
  - Low value of the tests

- Watching slow tests running is demotivating
  - Causes tests to be ran infrequently
  - Result: lots of broken tests

- Fixing broken tests is annoying
Introduction

New Xtend-based technologies

Demo

Conventional tests

Three types of tests

Common problems of conventional tests

Writing tests is boring

You need to rethink the entire use case

Testing afterwards is trickier than writing implementation

Low value of the tests

Watching slow tests running is demotivating

Causes tests to be ran infrequently

Result: lots of broken tests

Fixing broken tests is annoying

Boris Brodski

Write cool scalable enterprise application tests with Xtend & embedded DSLs

EclipseCon Europe 2014 4/26
Common problems of conventional tests

- Writing tests is boring
  - You need to rethink the entire use case
  - Testing afterwards is trickier than writing implementation
  - Low value of the tests

- Watching slow tests running is demotivating
  - Causes tests to be run infrequently
  - Result: lots of broken tests

- Fixing broken tests is annoying
  - Good knowledge of the entire system is required
Goal: Make writing tests fun

- Problem: Tests are too complex
  - Test first or better TDD
Goal: Make writing tests fun

- Problem: Tests are too complex
  - Test first or better TDD
  - Modern languages reduce boilerplate code (Xtend, Jnario)
  - Use advance testing techniques: factories, mocking, ...

- Problem: Tests are too slow
  - Integration tests
  - Unit tests

- Problem: Broken tests in the repository
  - Outsource test execution (also for uncommitted code)
  - Gerrit+Jenkins run tests automatically before commit

- Problem: Writing tests is boring
  - Add gamification (e.g. code coverage challenges)
  - Make writing tests cool!
Goal: Make writing tests fun

- Problem: Tests are too complex
  - Test first or better TDD
  - Modern languages reduce boilerplate code (Xtend, Jnario)
  - Use advance testing techniques: factories, mocking, ...

- Problem: Tests are too slow
  - integration tests
  + unit tests
Goal: Make writing tests fun

- **Problem: Tests are too complex**
  - Test first or better TDD
  - Modern languages reduce boilerplate code (Xtend, Jnario)
  - Use advance testing techniques: factories, mocking, ...

- **Problem: Tests are too slow**
  - integration tests
  - unit tests
  - **Outsource test execution (also for uncommitted code)**
Goal: Make writing tests fun

- **Problem:** Tests are too complex
  - Test first or better TDD
  - Modern languages reduce boilerplate code (Xtend, Jnario)
  - Use advance testing techniques: factories, mocking, ...

- **Problem:** Tests are too slow
  - integration tests
  - unit tests
  - Outsource test execution (also for uncommitted code)

- **Problem:** Broken tests in the repository
  - Gerrit + Jenkins run tests automatically before commit
Goal: Make writing tests fun

• Problem: Tests are too complex
  • Test first or better TDD
  • Modern languages reduce boilerplate code (Xtend, Jnario)
  • Use advance testing techniques: factories, mocking, ...

• Problem: Tests are too slow
  – integration tests
  + unit tests
  • Outsource test execution (also for uncommitted code)

• Problem: Broken tests in the repository
  • Gerrit+Jenkins run tests automatically before commit

• Problem: Writing tests is boring
  • Add gamification (e.g. code coverage challenges)
Goal: Make writing tests fun

- Problem: Tests are too complex
  - Test first or better TDD
  - Modern languages reduce boilerplate code
  - Use advance testing techniques: factories, mocking, ...

- Problem: Tests are too slow
  - Integration tests
  - Unit tests
  - Outsource test execution (also for uncommitted code)

- Problem: Broken tests in the repository
  - Gerrit+Jenkins run tests automatically before commit

- Problem: Writing tests is boring
  - Add gamification (e.g. code coverage challenges)
Outline

1. **Introduction**
   - Conventional tests
   - Three types of tests

2. **New Xtend-based technologies**
   - Xtend & test contrib
   - XFactory
   - JMockit-Xtend

3. **Demo**
   - Live-Demo
Integration test

- Test entire system

![Diagram showing components and interactions for an integration test]

- App-Server JVM
- Dependent module
- Component under test
- Service
- Test JVM
- Test
- DB

Prepare

Boris Brodski: Write cool scalable enterprise application tests with Xtend & embedded DSLs - EclipseCon Europe 2014
Integration test

- Test entire system

Diagram:

- App-Server JVM
  - Dependent module
  - Component under test
  - Service
  - Test
  - Test JVM
  - Prepare
  - DB
Unit test

- Test in isolation

Diagram:
- Test JVM
- Dependent module
- Component under test
- DB
- Test

Conventional tests
Three types of tests

Introduction
New Xtend-based technologies
Demo

Boris Brodski
Write cool scalable enterprise application tests with Xtend & embedded DSLs
EclipseCon Europe 2014 8/26
Unit test

- Test in isolation

Diagram:

- Test JVM
- Dependent module
- Component under test
- Mock
- DB
- Test

Boris Brodski
Write cool scalable enterprise application tests with Xtend & embedded DSLs
EclipseCon Europe 2014
Unit test

- Test in isolation

![Diagram showing unit test concept]

- Dependent module
- Component under test
- Test
- DB
- Test JVM
Persistence test

- Test with live database
- Hybrid between unit and integration tests

Diagram:

- Test JVM
- Dependent module
- Component under test
- Prepare
- Test
- DB

Write cool scalable enterprise application tests with Xtend & embedded DSLs

Boris Brodski

EclipseCon Europe 2014
Persistence test

- Test with live database
- Hybrid between unit and integration tests
Persistence test

- Test with live database
- Hybrid between unit and integration tests

Diagram:
- Dependent module
- Component under test
- Test JVM
- Mock
- DB
- Prepare
Outline

1. Introduction
   - Conventional tests
   - Three types of tests

2. New Xtend-based technologies
   - Xtend & test contrib
   - XFactory
   - JMockit-Xtend

3. Demo
   - Live-Demo
Java 10 today

**Xtend**

- Extension to Java
- Adds modern language features
- Compiles into readable Java
- 100% interoperable with Java
- With advanced IDE Support
- Java 8

- [http://xtend-lang.org](http://xtend-lang.org)
- [http://xtextcasts.org](http://xtextcasts.org)
Xtend Feature overview

- Type inference
  ```java
  for (item : list) {...}
  ``
  ```java
  "age".toFirstUpper
  ``

- Extension methods
  ```java
  btn.addActionListener[ println("Click!") ]
  ``
  ```java
  @Accessors String name
  ``
  ```java
  "abc" > "bcd"
  ``

- Lambda Expressions
  ```java
  btn.addActionListener[ println("Click!") ]
  ``

- ActiveAnnotations
  ```java
  @Accessors String name
  ``
  ```java
  "abc" > "bcd"
  ``

- Operator overloading
  ```java
  "abc" > "bcd"
  ``

- Powerful switch expressions
  ```java
  switch o {Set case o.size > 3: o.size}
  ``
  ```java
  msg = if (answer) "yes" else "no"
  ``

- No statements
  ```java
  msg = if (answer) "yes" else "no"
  ``
  ```java
  "abc" > "bcd"
  ``

- Template expressions
  ```java
  '''name
  «name
  score
  «score * 10
  '''
  ```

- Multiple dispatch
  ```java
  def dispatch op(Long s) {...}
  def dispatch op(Short l) {...}
  def dispatch op(Float f) {...}
  ```

Boris Brodski
Write cool scalable enterprise application tests with Xtend & embedded DSLs
EclipseCon Europe 2014 12/26
Simple JUnit test with Xtend

Java

```java
public class MyTest {

    @Test
    public void test1() {
        // test code
    }
}
```

Boris Brodski
Write cool scalable enterprise application tests with Xtend & embedded DSLs
EclipseCon Europe 2014
Simple JUnit test with Xtend

Java

public class MyTest {
    @Test
    public void test1() {
        // test code
    }
}

Xtend

class MyTest {
    @Test
    def void test1() {
        // test code
    }
}
Java

import static org.junit.Assert.*

assertEquals(f1(4), f2(1, 2, f3(3)))

assertTrue(f1(1, 2, f2(3)))
Assertions

**Java**

```java
import static org.junit.Assert.*

assertEquals(f1(4), f2(1, 2, f3(3)))

assertTrue(f1(1, 2, f2(3)))
```

**Xtend**

```xtend
import static extension org.junit.Assert.*

f1(4).assertEquals(f2(1, 2, f3(3)))
```

```
assertTrue(f1(1, 2, f2(3)))
```
### Assertions

#### Java

```java
import static org.junit.Assert.*

assertEquals(f1(4), f2(1, 2, f3(3)))

assertTrue(f1(1, 2, f2(3)))
```

#### Xtend

```xtend
import static extension org.junit.Assert.*

f1(4).assertEquals(f2(1, 2, f3(3)))

f2(1, 2, f3(3)) <=> f1(4)

f1(1, 2, f2(3)) <=> true
```

### Implementation

```java
public void operator_spaceship(Object o1, Object o2) {
    Assert.assertEquals(o2, o1);
}
```

```xtend
public void operator_spaceship(Object o1, Object o2) {
    Assert.assertEquals(o2, o1);
}
```
Assertions

Java

```java
import static org.junit.Assert.*

assertEquals(f1(4), f2(1, 2, f3(3)))

assertTrue(f1(1, 2, f2(3)))
```

Xtend

```xtend
import static extension org.junit.Assert.*

f1(4).assertEquals(f2(1, 2, f3(3)))

f2(1, 2, f3(3)) <=> f1(4)

f1(1, 2, f2(3)) <=> true
```

Implementation

```java
public void operator_spaceship(Object o1, Object o2) {
    Assert.assertEquals(o2, o1);
}
```
Java

```java
Calendar now = Calendar.getInstance();
date.setTime(now.getTime());
date.add(Calendar.YEAR, -2);
dto.setTimestamp(date.getTime());
```

```java
public long years(final int years) {
    Calendar date = Calendar.getInstance();
    Date now = date.getTime();
    date.add(Calendar.YEAR, years);
    return date.getTime().getTime() - now.getTime();
}

public Date ago(final long timeInMillis) {
    return new Date(System.currentTimeMillis() - timeInMillis);
}
```
Java

```java
Calendar now = Calendar.getInstance();
date.setTime(now.getTime());
date.add(Calendar.YEAR, -2);
dto.setTimestamp(date.getTime());
```

Xtend

```xtend
dto.timestamp = 2.years.ago
```
Java

```java
Calendar now = Calendar.getInstance();
date.setTime(now.getTime());
date.add(Calendar.YEAR, -2);
dto.setTimestamp(date.getTime());
```

Xtend

```xtend
dto.timestamp = 2.years.ago
```

Implementation

```java
public long years(final int years) {
    Calendar date = Calendar.getInstance();
    Date now = date.getTime();
    date.add(Calendar.YEAR, years);
    return date.getTime().getTime() - now.getTime();
}

public Date ago(final long timeInMillis) {
    return new Date(System.currentTimeMillis() - timeInMillis);
}
```
Java

```
Calendar now = Calendar.getInstance();
date.setTime(now.getTime());
date.add(Calendar.YEAR, -2);
dto.setTimestamp(date.getTime());
```

Xtend

```
dto.timestamp = 2.years.ago
```

Implementation

```
public long years(final int years) {
    Calendar date = Calendar.getInstance();
    Date now = date.getTime();
    date.add(Calendar.YEAR, years);
    return date.getTime().getTime() - now.getTime();
}

public Date ago(final long timeInMillis) {
    return new Date(System.currentTimeMillis() - timeInMillis);
}
```
Java

AuthorDTO author = new AuthorDTO();
author.setFirstName("Erle Stanley");
author.setLastName("Gardner");

GenreDTO genre = new GenreDTO();
genre.setName("Detective");

BookDTO b = new BookDTO();
b.setAuthor(authorDTO);
b.setGenre(genreDTO);
b.setTitle("The Case of the Velvet Claws");
b.setIsbn("0884114015");
Filling structures

**Java**

```
AuthorDTO author = new AuthorDTO();
author.setFirstName("Erle Stanley");
author.setLastName("Gardner");

GenreDTO genre = new GenreDTO();
genre.setName("Detective");

BookDTO b = new BookDTO();
b.setAuthor(authorDTO);
b.setGenre(genreDTO);
b.setTitle("The Case of the Velvet Claws");
b.setIsbn("0884114015");
```

**Xtend**

```
val book = new BookDTO => [
  author = new AuthorDTO => [
    firstName = "Erle Stanley"
    lastName = "Gardner"
  ]
  genre = new GenreDTO => [
    name = "Detective"
  ]
  title = "The Case of the Velvet Claws"
  isbn = "0884114015"
]
Filling structures

Java

```java
AuthorDTO author = new AuthorDTO();
author.setFirstName("Erle Stanley");
author.setLastName("Gardner");

GenreDTO genre = new GenreDTO();
genre.setName("Detective");

BookDTO b = new BookDTO();
b.setAuthor(authorDTO);
b.setGenre(genreDTO);
b.setTitle("The Case of the Velvet Claws");
b.setIsbn("0884114015");
```

Xtend

```xtend
val book = new BookDTO => [
  author = new AuthorDTO => [
    firstName = "Erle Stanley"
    lastName = "Gardner"
  ]
  genre = new GenreDTO => [
    name = "Detective"
  ]
  title = "The Case of the Velvet Claws"
  isbn = "0884114015"
]
```
Outline

1. Introduction
   - Conventional tests
   - Three types of tests

2. New Xtend-based technologies
   - Xtend & test contrib
   - XFactory
   - JMockit-Xtend

3. Demo
   - Live-Demo
XFactory

- Create and persist entities
- Embedded DSL (into Xtend/Jnario)
- Designed for unit and persistence tests
- Open Source:
  https://github.com/borisbrodski/xfactory
Using XFactory

Get instance of a valid entity

```scala
val book = xbuild(new XFactoryBook)
```
Using XFactory

**Get instance of a valid entity**

```scala
val book = xbuild(new XFactoryBook)
```

**Persist an entity**

```scala
val book = xpersist(new XFactoryBook)
```
Using XFactory

Get instance of a valid entity

```scala
val book = xbuild(new XFactoryBook)
```

Persist an entity

```scala
val book = xpersist(new XFactoryBook)
```

Change default values

```scala
val b = xpersist(new XFactoryBook) [ minimal set [ title = "Eclipse IDE - kurz & gut" ] ]
```
Using XFactory

Get instance of a valid entity

```scala
val book = xbuild(new XFactoryBook)
```

Persist an entity

```scala
val book = xpersist(new XFactoryBook)
```

Change default values

```scala
val b = xpersist(new XFactoryBook) [ minimal set [ title = "Eclipse IDE - kurz & gut" ] ]
```

Use predefined methods

```scala
val b = xpersist(new XFactoryBook) [ minimal(author) makeBestSeller ]
```
Implementing XFactory

Building an author

class XFactoryAuthor extends AbstractXFactory<Author> {

    extension XtendTestContrib = new XtendTestContrib

    override minimal() {
        set [
            firstName = "John"
            lastName = "Doe"
            birthday = 14.\(\text{\textit{april}(1967)}\)

            genre = xpersistBefore(new XFactoryGenre)
        ]
    }

    def kill() {
        set [
            dayOfBirth = birthday + 40.\(\text{\textit{years}}\)
        ]
    }
}
Implementing XFactory

Building an author

class XFactoryAuthor extends AbstractXFactory<Author> {
    extension XtendTestContrib = new XtendTestContrib

    override minimal() {
        set [
            firstName = "John"
            lastName = "Doe"
            birthday = 14. april(1967)

            genre = xpersistBefore(new
        ]
    }

    def kill() {
        set [
            dayOfDeath = birthday + 40. years
        ]
    }
}
Outline

1. Introduction
   - Conventional tests
   - Three types of tests

2. New Xtend-based technologies
   - Xtend & test contrib
   - XFactory
   - JMockit-Xtend

3. Demo
   - Live-Demo
Unit test

- Test in isolation

Diagram:

- Dependent module
- Component under test
- Database
- Test JVM
- Test

Boris Brodski
Write cool scalable enterprise application tests with Xtend & embedded DSLs
EclipseCon Europe 2014
JMockit-Xtend

- Add JMockit support to Xtend/Jnario
- http://jmockit.github.io/
- http://github.com/borisbrodski/jmockit-xtend

Using JMockit-Xtend

```java
@Mocked
DependentModule module

@Test
def void test() {
    ...
    stub [
        module.validName("test")
        result = true
    ]
    ...
}
```
JMockit-Xtend

- Add JMockit support to Xtend/Jnario
- http://jmockit.github.io/
- http://github.com/borisbrodski/jmockit-xtend

```java
@Mocked DependentModule module;

@Test
def void test() {
    ...  
    stub [  
        module.validName("test")  
        result = true  
    ]  
    ...  
}
```
Outline

1. Introduction
   - Conventional tests
   - Three types of tests

2. New Xtend-based technologies
   - Xtend & test contrib
   - XFactory
   - JMockit-Xtend

3. Demo
   - Live-Demo
Boris Brodski

Write cool scalable enterprise application tests with Xtend & embedded DSLs

EclipseCon Europe 2014 25/26
Questions?

- Conventional Testing
- IT Test
- Unit Test
- Persistence Test

- Xtend
- Xtend-Contrib
- XFactory
- JMockit-Xtend