



X-Series

可视化大规模软件开发工具集

赫
杰
辉

Agenda

- ▲ 大规模开发的挑战
- ▲ 大规模开发的出路
- ▲ 最值得抽象的模型
- ▲ X-Series工具集
- ▲ X-Series相关资源
- ▲ X-Series安装
- ▲ QA

**关于开发的一点
感性认识**

别人PPT里的系统架构



咱们的系统架构



面试时HR的承诺



被骗进来之后



系统开发很难

大规模开发的挑战

你
不
知
道
你
不
知
道

▲ 开发其实是个翻译的过程

需求文档 → 设计文档 → 代码实现

哪里有翻译，哪里就有误解

抽象层间存在细节的增强和丢失

▲ 不靠谱的文档

需求文档缺乏最新需求和关键实现细节

设计文档无法完全描述需求

设计文档与代码实现之间的关系被割裂

▲ 难以理解的源码

文档与代码不存在自动的关联性

软件编码的方式50年不变，过去的给现在的挖坑，现在的给将来的挖坑

逐行理解百万，千万代码行级别的系统是不可能完成的任务

怎么破？

大规模开发的出路

▲ 基于模型而不是代码开发

开发并不仅仅意味着写代码

将业务模型和数据模型从代码里面解放出来

专用工具解决专门的问题

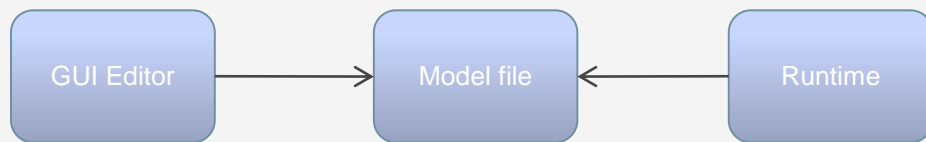
▲ 对工具的要求

简单易懂，直接明了

开发和运行阶段使用同一个模型

可视化的编辑方式

杜绝代码生成



最值得抽象的模型

▲ 流程图

一个系统包括哪些功能，每个功能包含哪些步骤

消灭粘合代码

▲ 决策树

一个决定受哪些因素影响，每个因素按照什么顺序考虑

取代复杂嵌套的if/else

▲ 状态机

一个实体具有哪些状态，状态之间如何转移

代替hard-code的状态判断和动作触发

X-Series为此而生

X-Series概览

▲ 一套轻量级的框架

易于学习

易于使用

易于测试

易于交流

▲ 解决大规模软件开发难题

沟通不畅

文档不新

分工不当

进度不明

工欲善其事
必先利其器

X-Series组件集

▲ Xross Unit

用流程图描述服务如何按步骤完成
服务级别

▲ Xross Decision tree

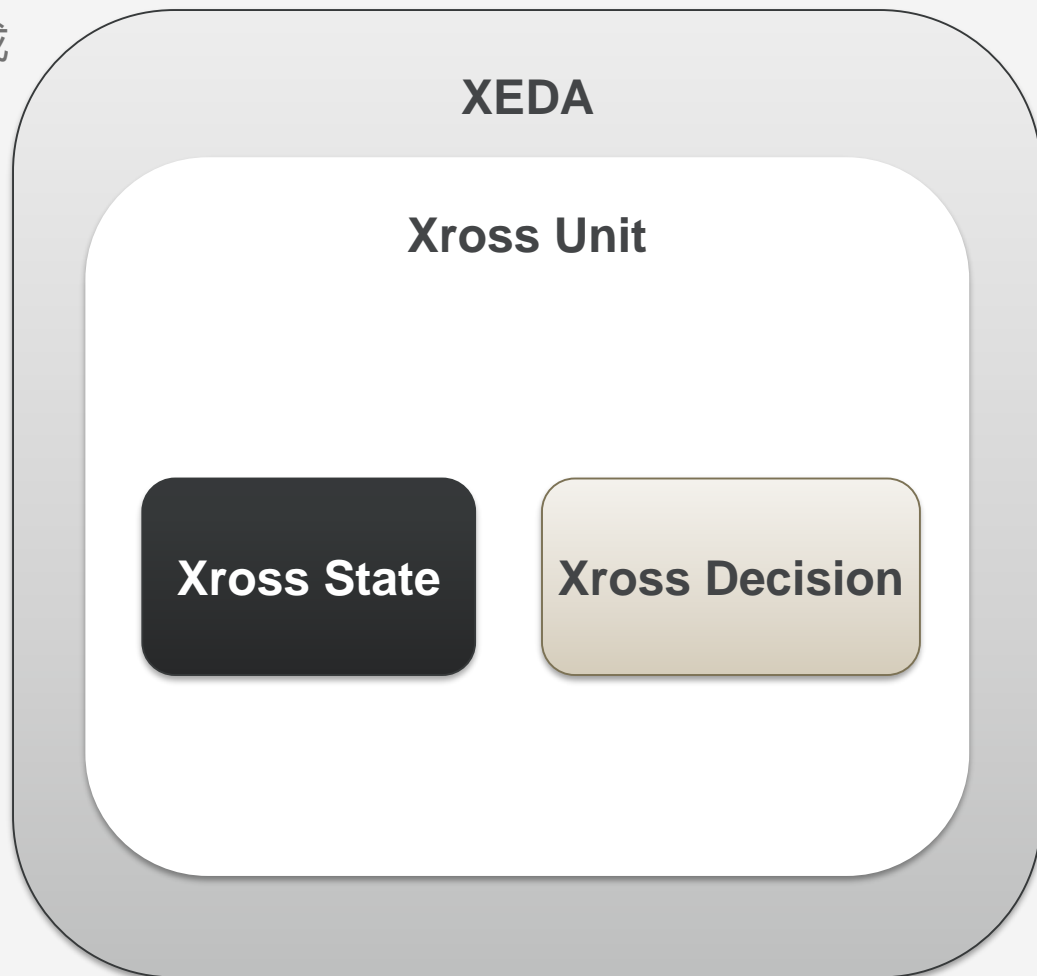
用决策树为复杂决策建模
模块级别

▲ Xross State

用状态机管理业务状态变迁
领域级别

▲ Xeda [WIP]

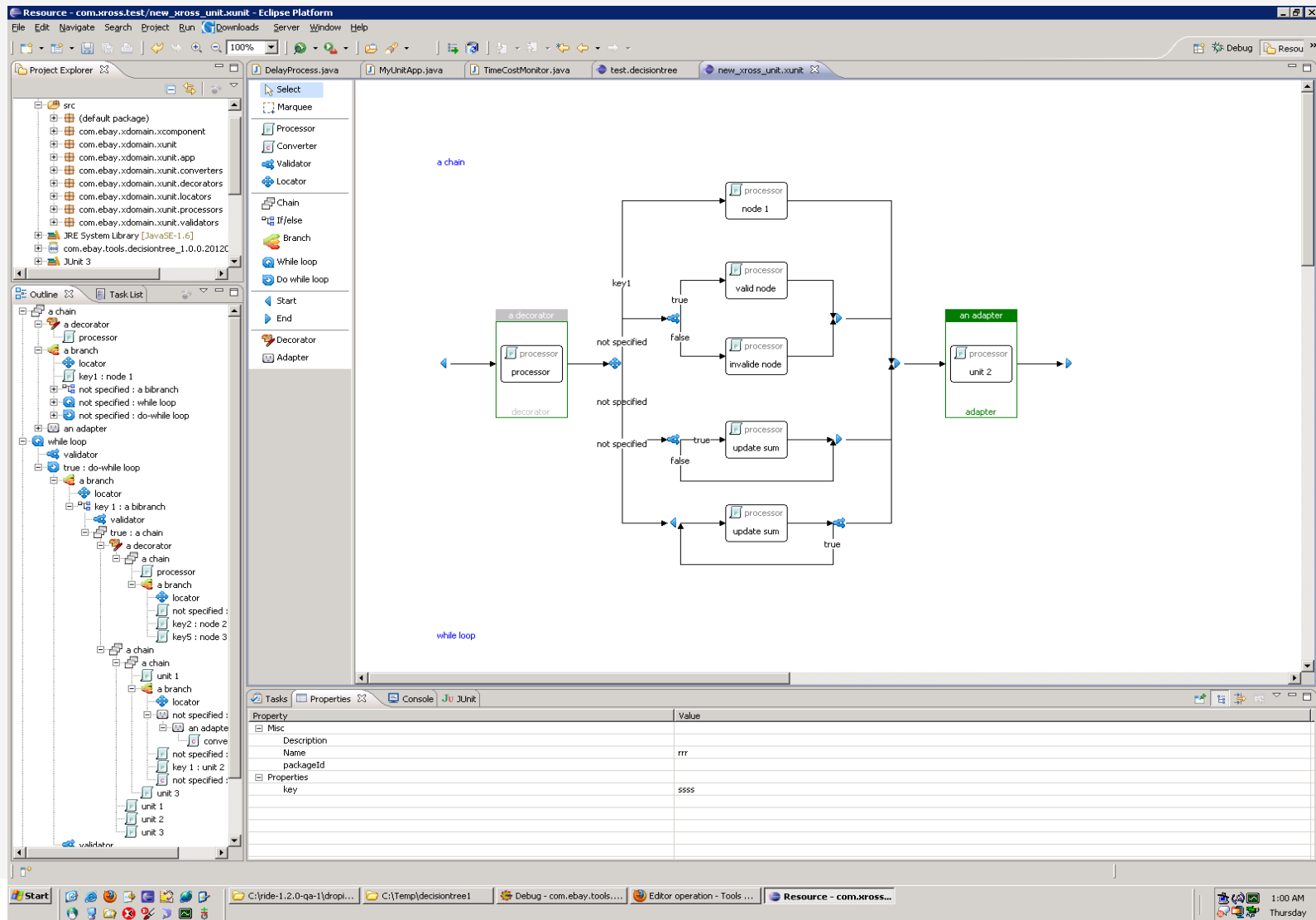
基于SEDA模型的微服务架构
运行平台级别



Xross Unit

Xross Unit

基于流程图的灵活的系统构建器



Xross Unit

- ▲ 提供丰富的行为组件

超精简接口 – processor, converter, validator, locator

- ▲ 提供丰富的结构组件

chain, if-else, branch, while, do while loop, **decorator**, **adapter**

- ▲ 编辑方法自然

简单对象组合为复杂结构 – E.g. Validator + Unit = if/else structure

- ▲ 可配置

可以在应用或构建单元层次上面配置参数，方便复用

- ▲ 模型与代码相关联

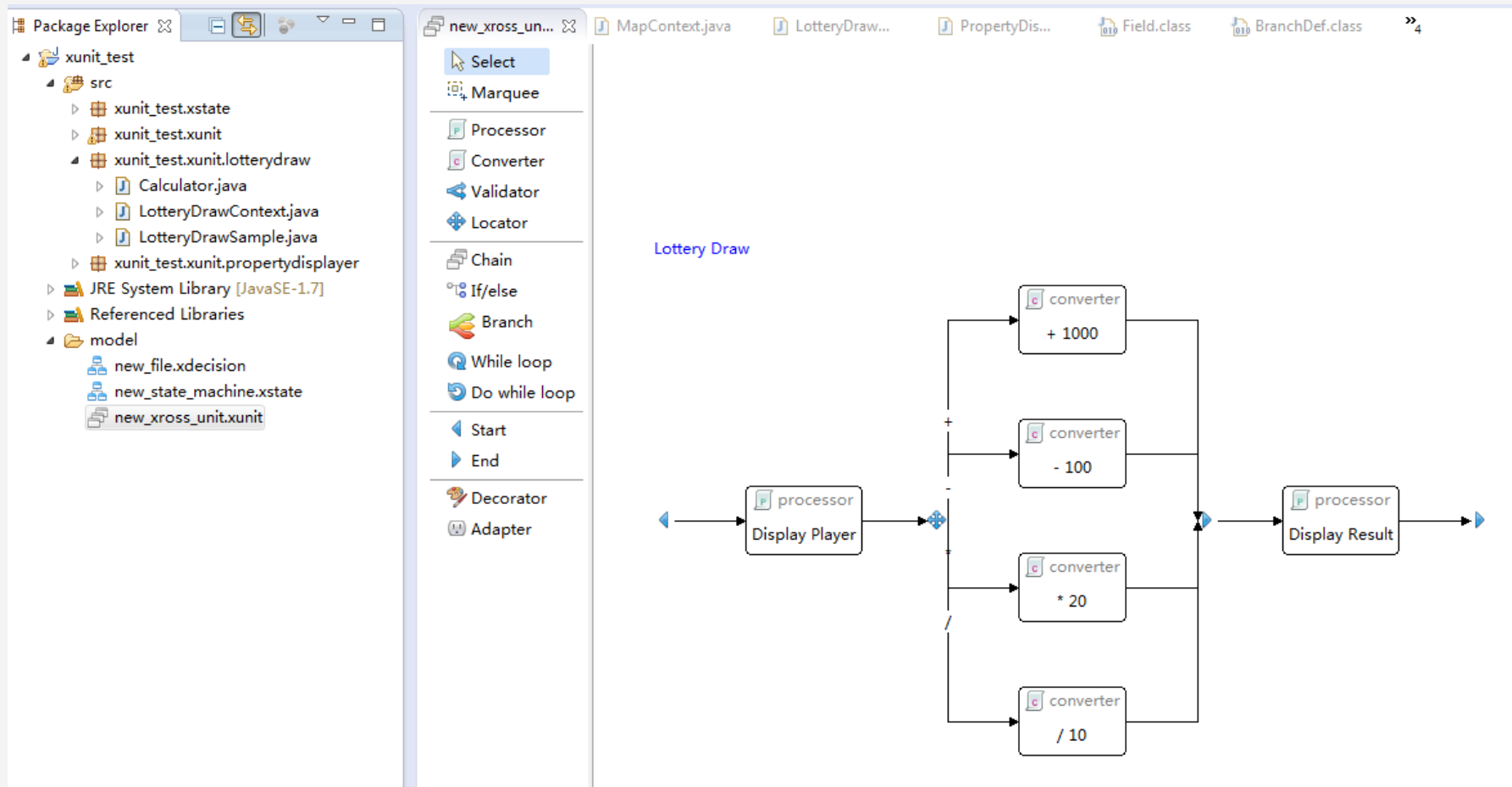
模型归模型，代码归代码，查看代码仅需双击进入

Xross Unit使用方式

- ▲ 构建系统蓝图
- ▲ 创建组件单元
- ▲ 关联蓝图和代码
- ▲ 生成实例并运行

构建系统蓝图

▲ 你可以一直和PM, PD, QA一起优化修改讨论



创建组件单元

▲ 函数式接口易于实现和测试

```
package xunit_test.xunit.lotterydraw;

import java.util.Map;

public class Calculator implements Converter, UnitPropertiesAware {
    private double delta;
    private String operation;

    @Override
    public Context convert(Context arg0) {
        LotteryDrawContext ctx = (LotteryDrawContext)arg0;

        double value = ctx.quantity;

        switch(operation){
            case "+": value+=delta; break;
            case "-": value-=delta; break;
            case "*": value*=delta; break;
            case "/": value/=delta; break;
        }

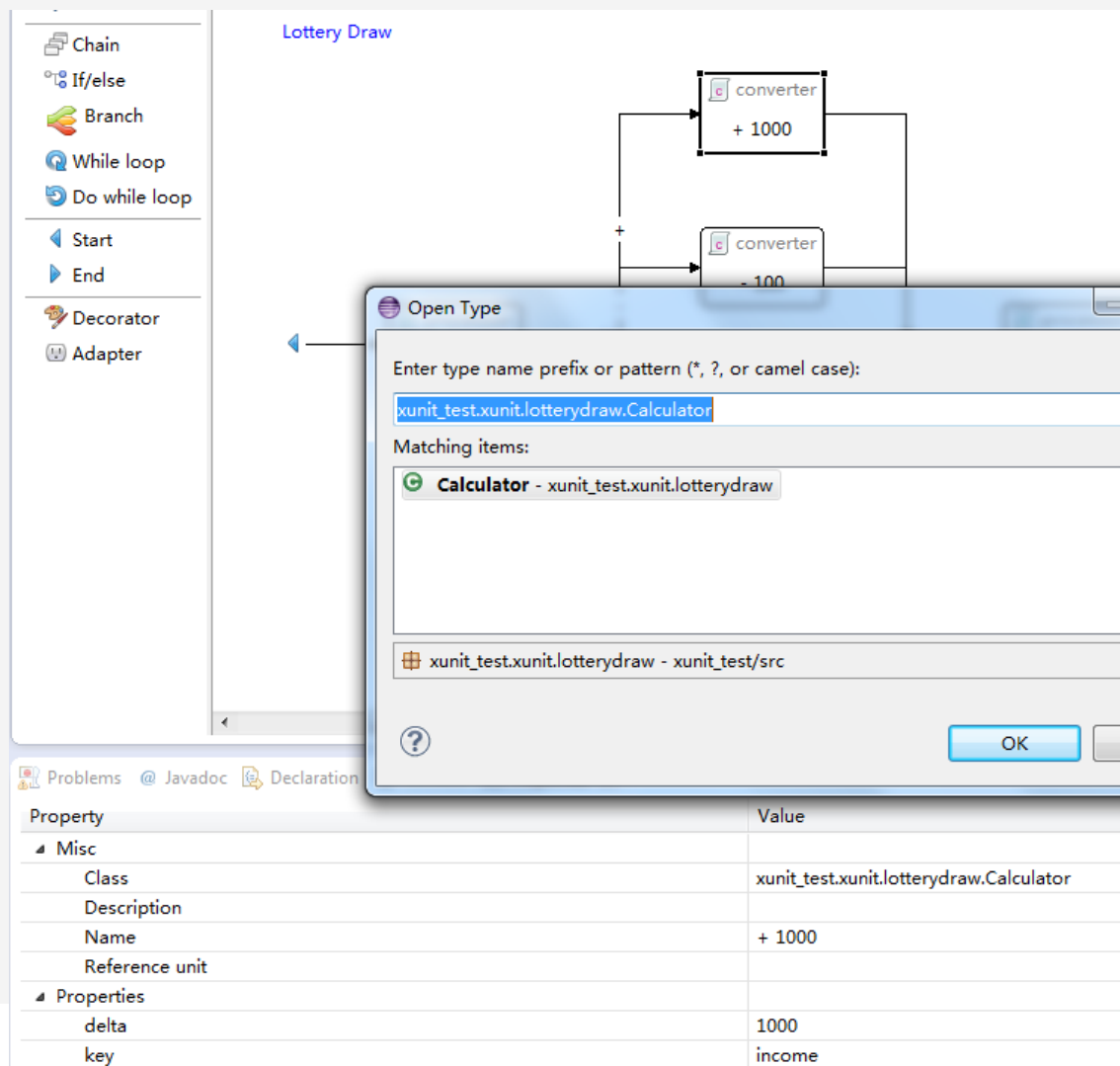
        ctx.quantity = value;

        return ctx;
    }

    @Override
    public void setUnitProperties(Map<String, String> arg0) {
        delta = Double.parseDouble(arg0.get("delta"));
        operation = arg0.get("operation");
    }
}
```

关联蓝图和代码

▲ 每个unit都可配置

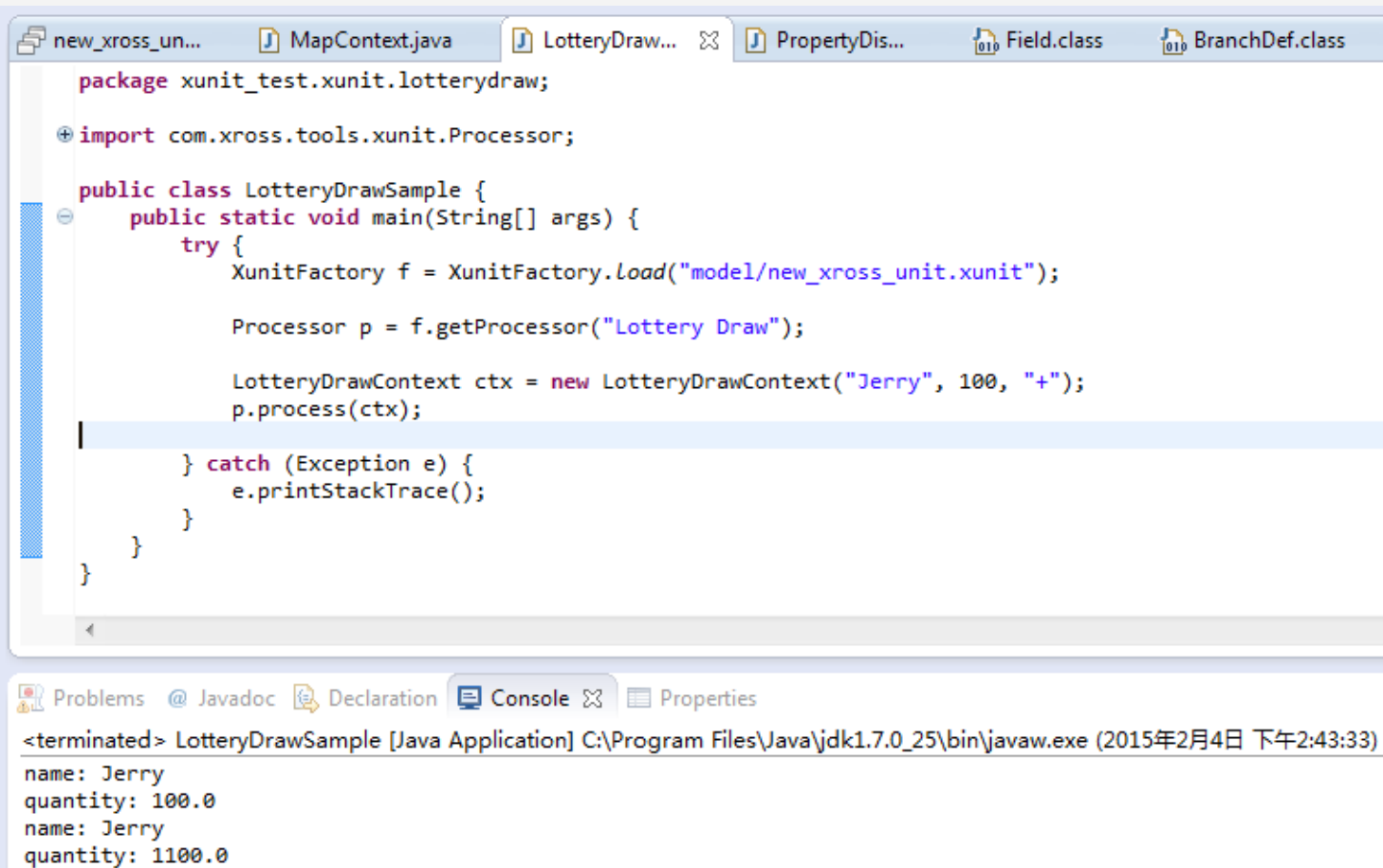


The screenshot illustrates the configuration of a unit in a diagram. The diagram, titled "Lottery Draw", shows a flow with two "converter" blocks. The top block is labeled "+ 1000" and the bottom block is labeled "- 100". An "Open Type" dialog is open, prompting the user to enter a type name prefix or pattern. The text "xunit_test.xunit.lotterydraw.Calculator" is entered in the search field. The dialog lists matching items, including "Calculator - xunit_test.xunit.lotterydraw" and "xunit_test.xunit.lotterydraw - xunit_test/src".

Property	Value
▲ Misc	
Class	xunit_test.xunit.lotterydraw.Calculator
Description	
Name	+ 1000
Reference unit	
▲ Properties	
delta	1000
key	income

生成实例并运行

- ▲ 创建模型实例
- ▲ 处理Context



The screenshot shows an IDE with several tabs: 'new_xross_un...', 'MapContext.java', 'LotteryDraw...', 'PropertyDis...', 'Field.class', and 'BranchDef.class'. The 'LotteryDraw...' tab is active, displaying the following Java code:

```
package xunit_test.xunit.lotterydraw;

import com.xross.tools.xunit.Processor;

public class LotteryDrawSample {
    public static void main(String[] args) {
        try {
            XunitFactory f = XunitFactory.Load("model/new_xross_unit.xunit");

            Processor p = f.getProcessor("Lottery Draw");

            LotteryDrawContext ctx = new LotteryDrawContext("Jerry", 100, "+");
            p.process(ctx);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Below the code editor, the 'Console' tab is active, showing the execution output:

```
<terminated> LotteryDrawSample [Java Application] C:\Program Files\Java\jdk1.7.0_25\bin\javaw.exe (2015年2月4日 下午2:43:33)
name: Jerry
quantity: 100.0
name: Jerry
quantity: 1100.0
```


Xross Unit的优势

▲ 快速组建系统

自顶向下分解，组件化设计，流水线式开发

最优化设计复用

快速切换开发焦点

▲ 高内聚，低耦合

通过名字描述功能

通过配置调整行为

通过Context限定数据

每个unit仅仅完成明确描述的功能

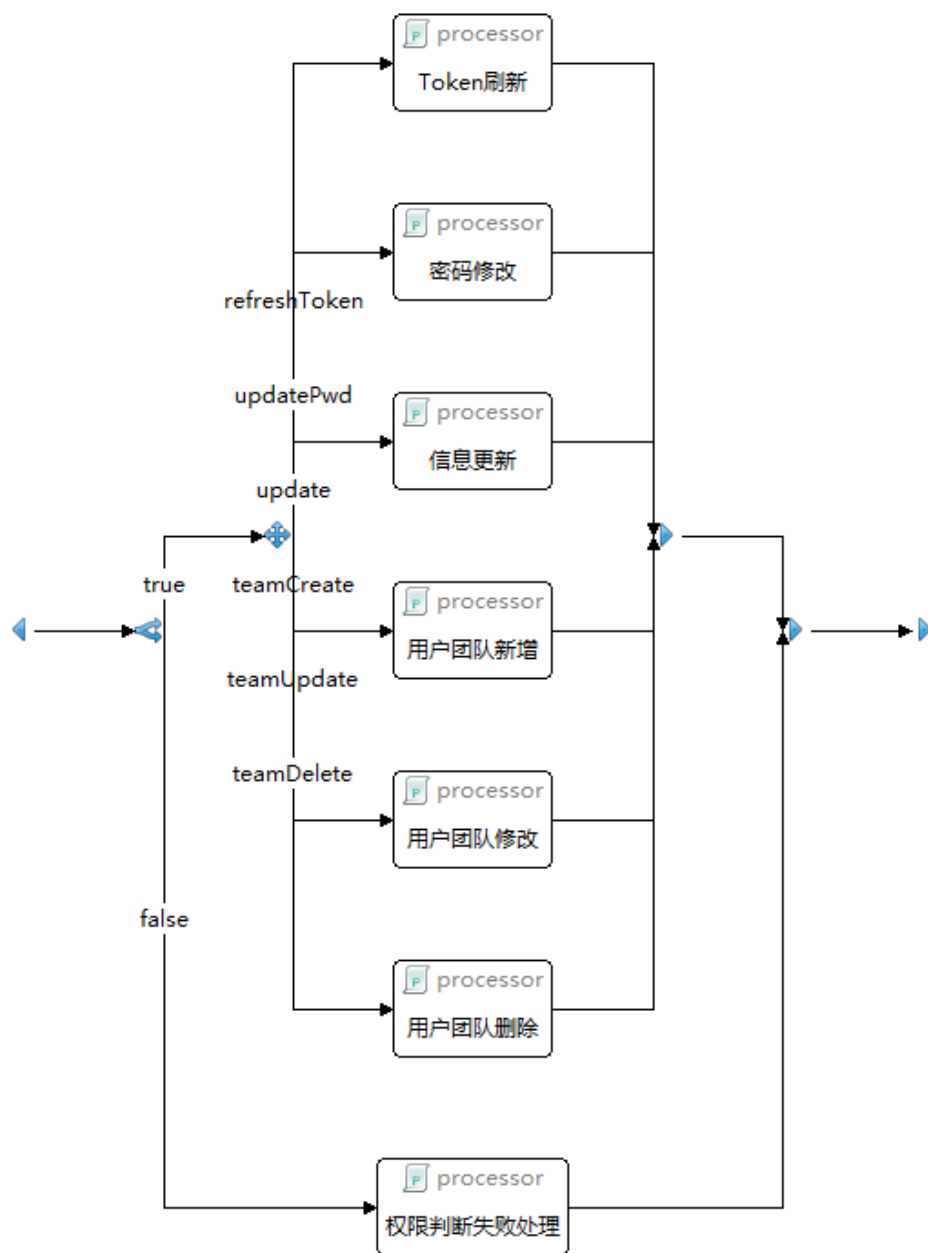
▲ 易于单元化测试

单接口设计，无选择，无歧义的实现

通过构造Context，轻松模拟测试数据

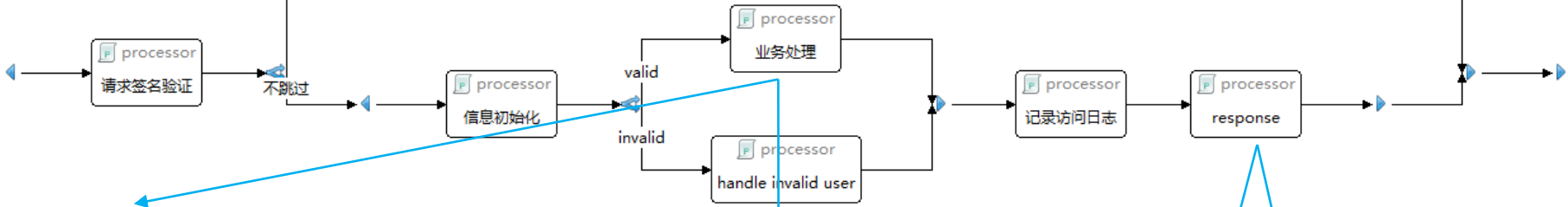
实际案例

security check chain

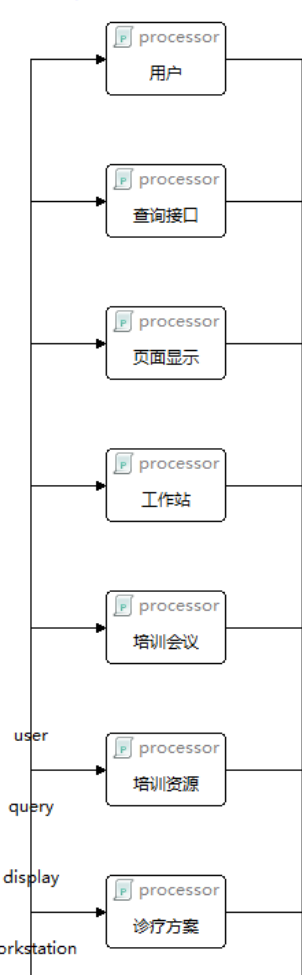


同一模型文件的子图引用

main



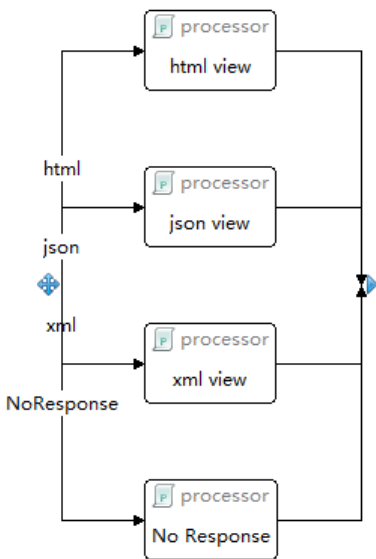
module process branch



内置功能

Property	Value
Class	
Description	
Name	response
Reference unit	view branch

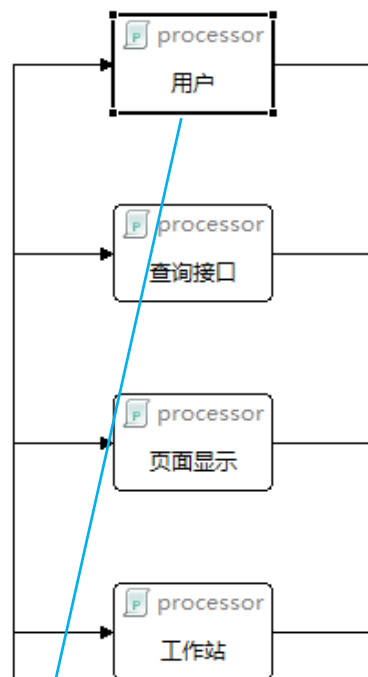
view branch



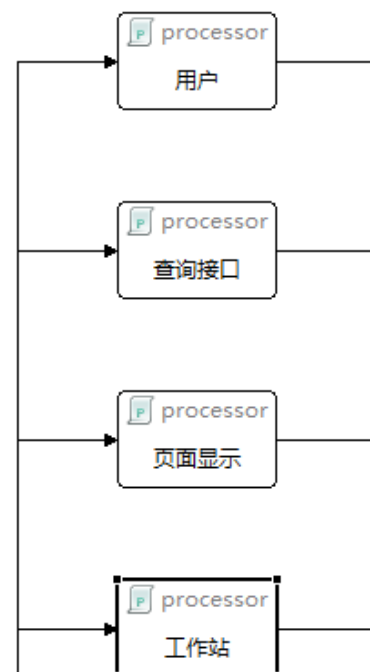
Property	Value
Class	
Description	
Name	业务处理
Reference unit	module process branch

不同模型文件的子图引用

module process branch



module process branch



Property	Value
▲ Misc	
Class	com.medicalmaster.web.control.entry.DistributeProcessor
Description	
Name	用户
Reference unit	
▲ Properties	
module	user
processor	user process

用户的创意用法

Property	Value
▲ Misc	
Class	com.medicalmaster.web.control.entry.DistributeProcessor
Description	
Name	工作站
Reference unit	
▲ Properties	
module	workstation
processor	workstation process

关于 Xross Unit更多信息

▲ 不是又一个Spring

Spring: 从整体如何由局部构成的观点构建系统

Xunit: 从请求如何被处理的行为观点构建系统

▲ 不是工作流

工作流处理多角色在多请求之间的任务/路径管理

Xunit 管理一个请求的响应路径/处理单元

▲ 不是一个可视化的编程语言

可视化的编程语言解释和生产代码

Xunit 在业务层组装行为和结构单元

Workflow

Xunit

Spring

Visual
Language

关于 Xross Unit更多信息

▲ 为什么使用单元来完成代码也能做的事情？

因为问题的大小决定手段的选择，想象下面工作的复杂度

- “Hello World”
- 一个Web Service
- 一个小的Web App
- 一个淘宝，ebay，ctrip规模的网站

▲ 为什么不用现有的命令框架

缺乏管理单元的内部细节表示

- Servlet – Command at URL level
- JEE: Session Bean, Entity Bean, Message Bean – Command at bean id level

尽管有大量的小的仅仅只有一页代码的command

但是还是会有少量但是非常重要的command是非常的复杂[80/20原则]

Xross Decision

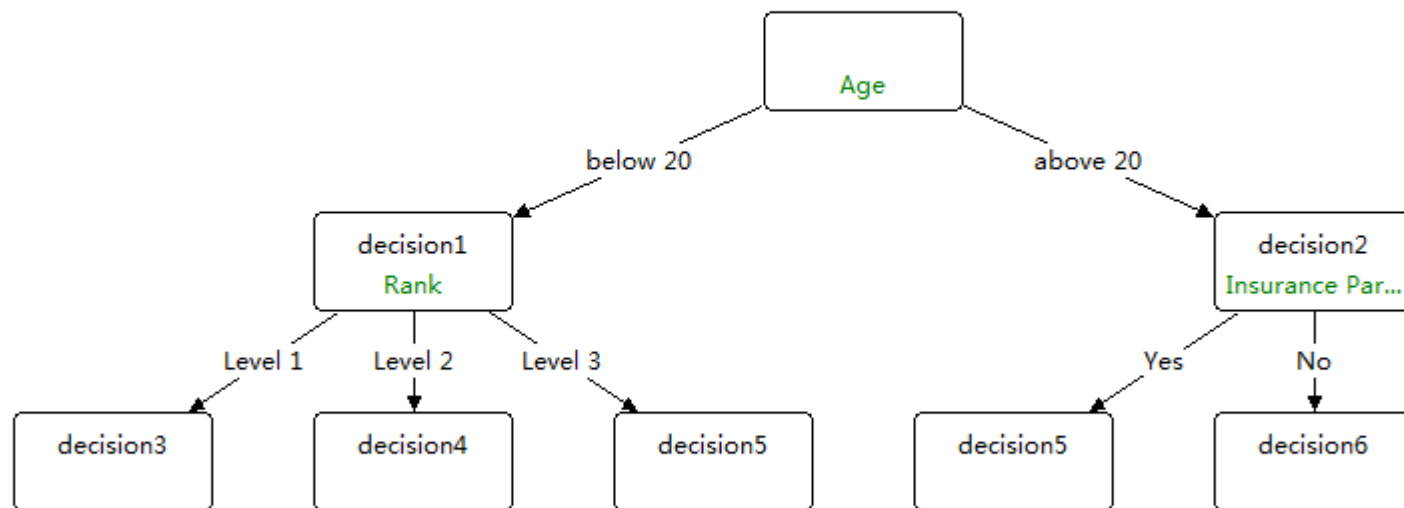
Xross Decision

▲ 图形化的决策树编辑器

以所见即所得的方式表达复杂逻辑判断的过程

依据模型生成单元测试的验证代码

替代if/else，极大的简化代码



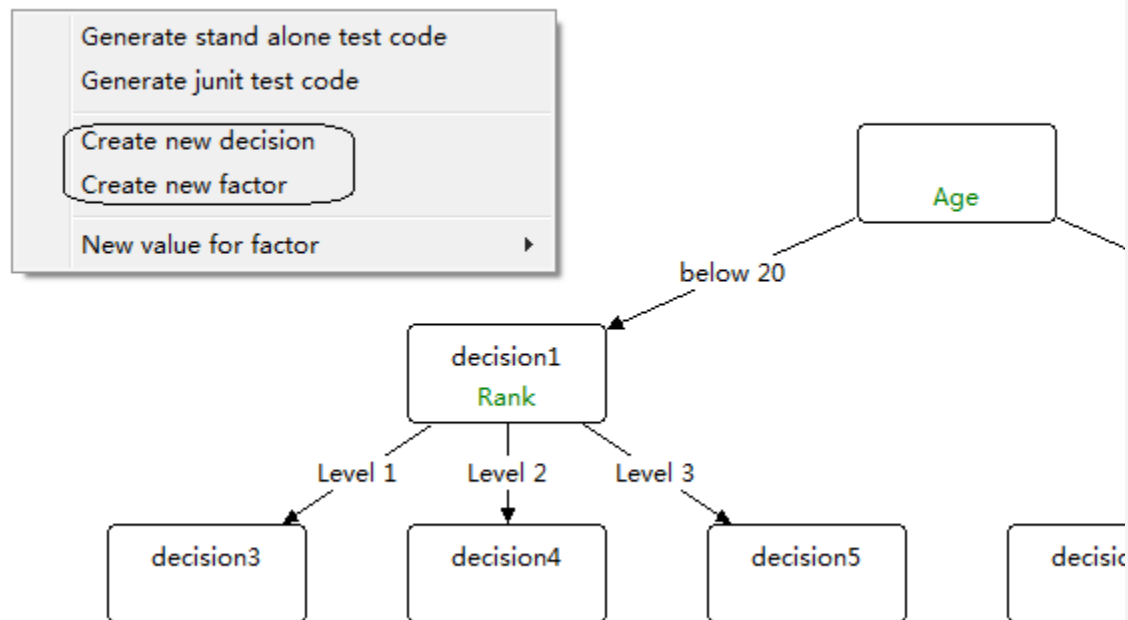
Xross Decision建模

▲ 定义决策的考虑因素

因素是包括多个可能取值的变量

▲ 定义决策

代表特定选择的的常量



Xross Decision测试

▲ 生成单元测试

检验模型是否正确运行，示范如何使用工具

The screenshot shows the Xross Decision IDE interface. At the top, a menu is open with options: "Generate stand alone test code", "Generate junit test code" (highlighted), "Create new decision", "Create new factor", and "New value for factor". Below the menu, a decision tree diagram is visible with a node labeled "Level 1" and a box labeled "decision3". A status bar indicates "Finished after 0.061 seconds" and "Runs: 1/1", "Errors: 0", "Failures: 0". The bottom panel shows the generated unit test code in a text editor:

```
package com.ebay.xdomain.xcomponent;

import org.junit.Test;

import static org.junit.Assert.*;

import com.xross.tools.xdecision.MapFacts;
import com.xross.tools.xdecision.XDecisionTree;
import com.xross.tools.xdecision.XDecisionTreeFactory;

public class SimpleDecisionTest {
    @Test
    public void testGetDecision(){
        XDecisionTree<String> tree;
        try {
            tree = XDecisionTreeFactory.create
        } catch (Exception e) {
            e.printStackTrace();
            fail();
        }
    }
}
```

The bottom right panel shows the "Failure Trace" section, which is currently empty.

```
package xunit_test.xdecision;

import org.junit.Test;

public class SimpleDecisionTest {
    @Test
    public void testGetDecision(){
        XDecisionTree<String> tree;
        try {
            tree = XDecisionTreeFactory.create("model/SimpleDecision.xdecision");
        } catch (Exception e) {
            e.printStackTrace();
            fail();
            return;
        }

        //Verify tree
        MapFacts test;

        test = new MapFacts();
        test.set("Age", "below 20");
        assertEquals("decision1", tree.get(test));

        test = new MapFacts();
        test.set("Age", "above 20");
        assertEquals("decision2", tree.get(test));

        test = new MapFacts();
        test.set("Age", "below 20");
        test.set("Rank", "Level 1");
        assertEquals("decision3", tree.get(test));

        test = new MapFacts();
        test.set("Age", "below 20");
        test.set("Rank", "Level 2");
        assertEquals("decision4", tree.get(test));

        test = new MapFacts();
        test.set("Age", "above 20");
        test.set("Insurance Participation", "Yes");
```

Xross State

Xross State

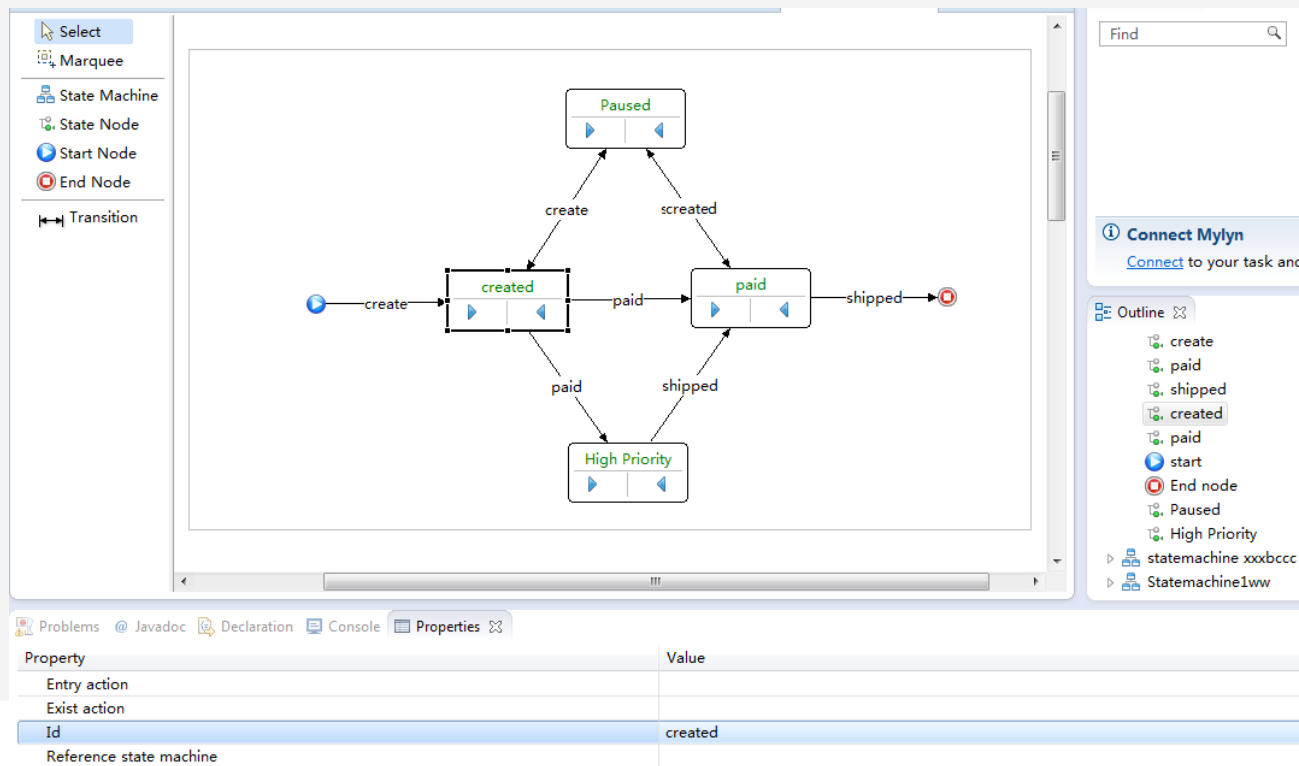
▲ 可视化创建状态机的编辑器

用处极其广泛

▲ 结合模型和代码

可以创建仅包含状态和变迁的状态机

也可以提供状态变迁时的触发器



状态机是系统的核心

Xross State扩展元素

▲ 状态转移触发器

EntryAction

ExitAction

TransitionAction

▲ 状态转移校验

TransitionGuard

```
package xunit_test.xstate;

import com.xross.tools.xstate.EntryAction;

public class TestAction implements EntryAction, ExitAction, TransitAction, TransitionGuard {
    public void transit(String sourceStateId, String targetStateId, Event event) {
        System.out.println(String.format("Transit from %s to %s on %s", sourceStateId, targetStateId, event.getId()));
    }

    public void exit(String sourceStateId, Event event) {
        System.out.println(String.format("Exit from %s on %s", sourceStateId, event.getId()));
    }

    public void enter(String targetStateId, Event event) {
        System.out.println(String.format("Enter into %s on %s", targetStateId, event.getId()));
    }

    public boolean isTransitAllowed(String sourceId, String targetId, Event event) {return true;}
}
```


Xross State 示例

```
package xunit_test.xstate;

import com.xross.tools.xstate.Event;

public class StateTest {
    public static void main(String[] args) {
        try {
            StateMachineFactory f = StateMachineFactory.load("model/new_state_machine.xstate");
            StateMachine sm = f.create("Statemachine1ww");

            print(sm);
            sm.notify(new Event("e1"));

            print(sm);
            sm.notify(new Event("e2"));

            print(sm);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    static void print(StateMachine sm) {
        System.out.println("Current state Id: " + sm.getCurrentState().getId());
        System.out.println("Is ended: " + sm.isEnded());
    }
}
```

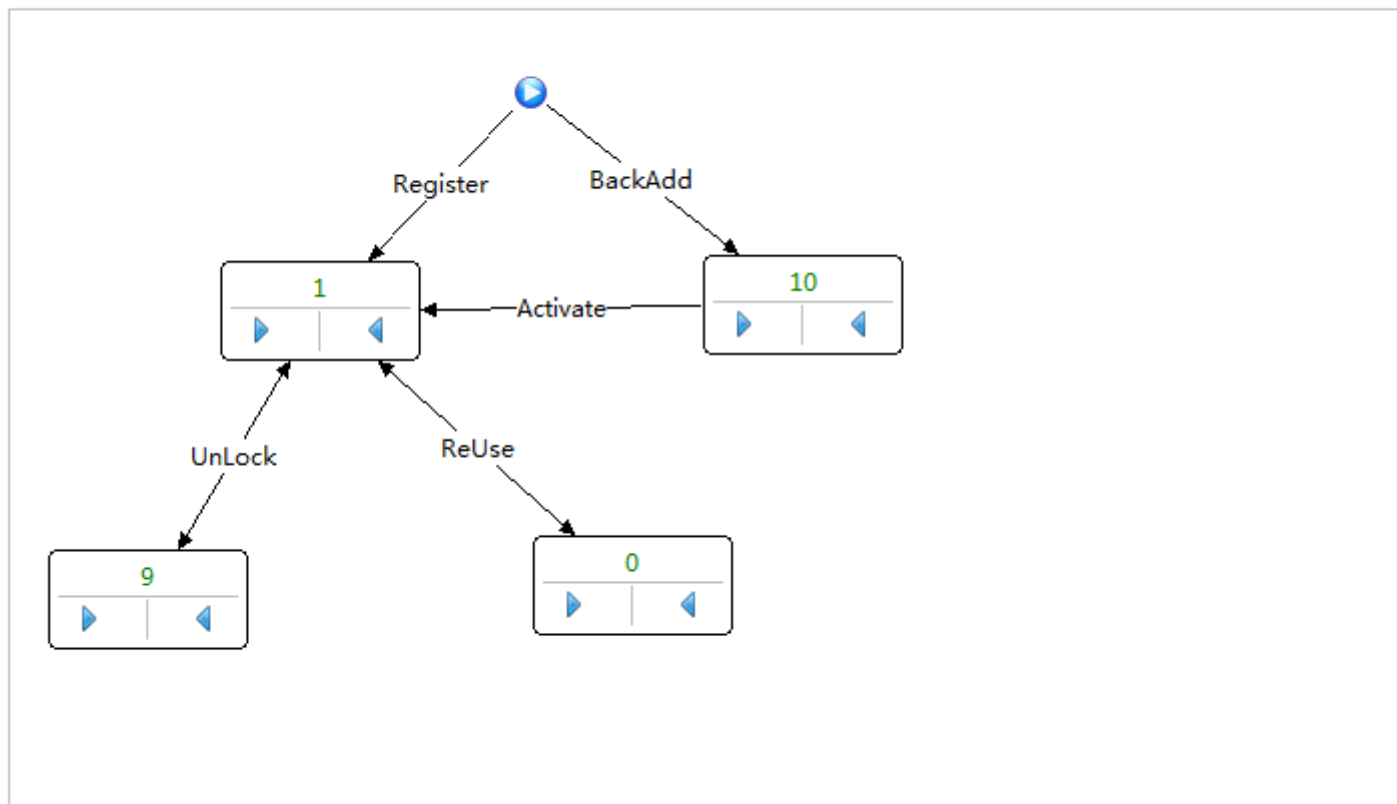
Problems @ Javadoc Declaration Console Properties

<terminated> StateTest [Java Application] C:\Program Files\Java\jdk1.7.0_25\bin\javaw.exe (2015年2月6日 下午12:27:35)

```
Current state Id: start
Is ended: false
Current state Id: S1
Is ended: false
Current state Id: end
Is ended: true
```

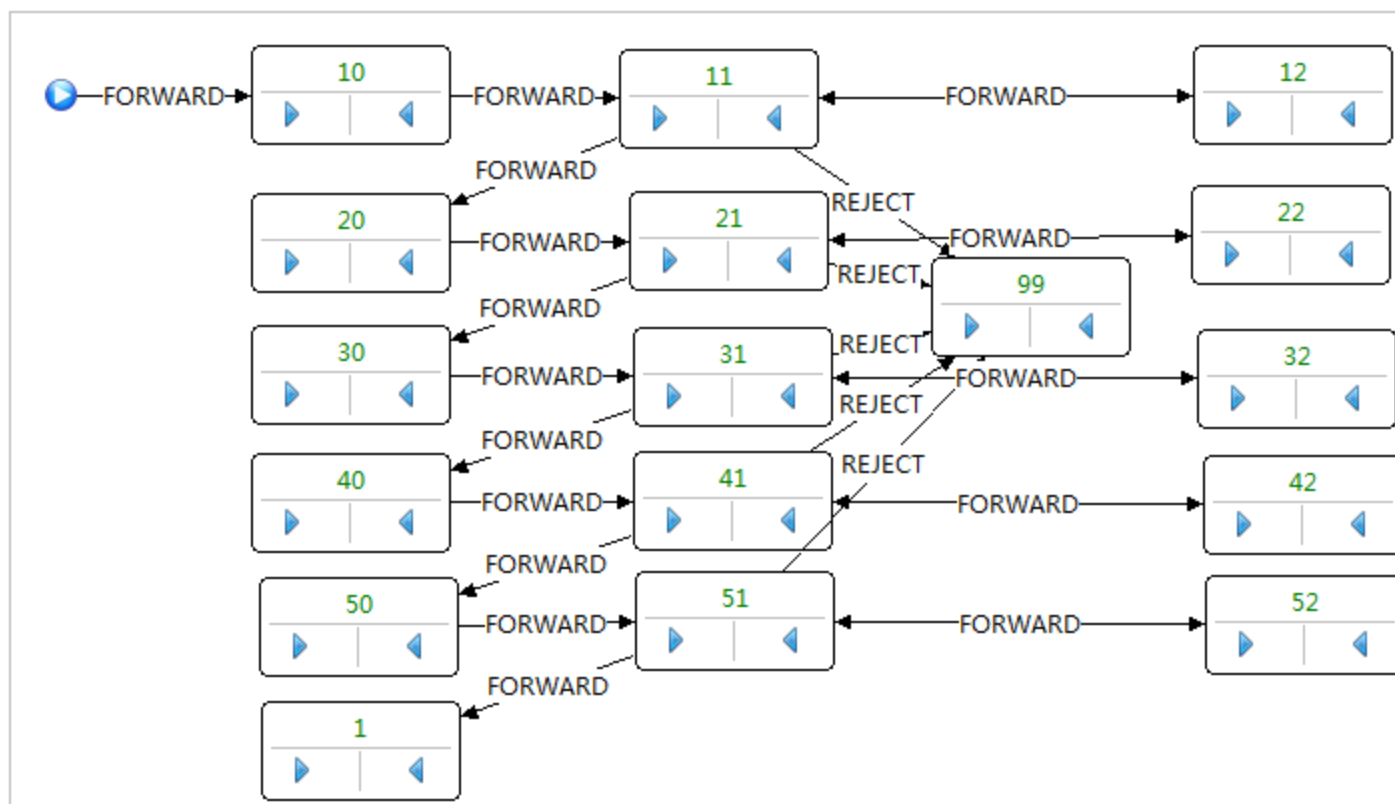
实际案例

user status



实际案例

researchJoinStatus



XEDA

The Next

▲ XEDA

SEDA/Microservice implementation

Service deployment

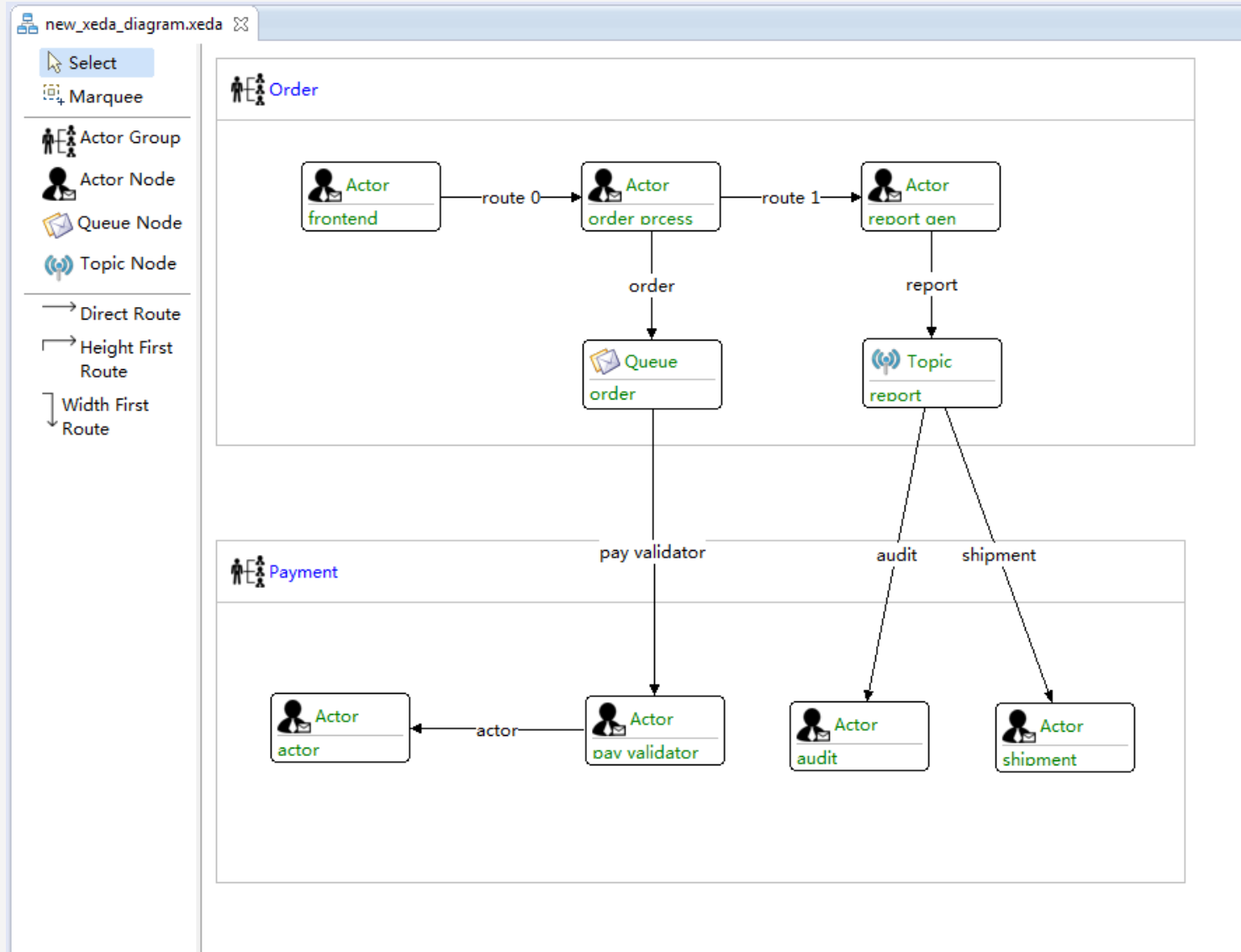
Managing/Monitoring

Distributed OS

You must be
this tall to use
microservices



XEDA Preview



X-series 资源

X-series 资源

▲ Codebase

<https://github.com/hejieshui/xUnit>

<https://github.com/hejieshui/xDecision>

<https://github.com/hejieshui/xState>

▲ Sample

Normal project sample

https://github.com/hejieshui/xross-tools-installer/blob/master/com.xross.tools.xunit.feature/installer/xunit_test.zip

Maven project sample

<https://github.com/hejieshui/xross-tools-installer/blob/master/com.xross.tools.xunit.feature/installer/x-series-sample.zip>

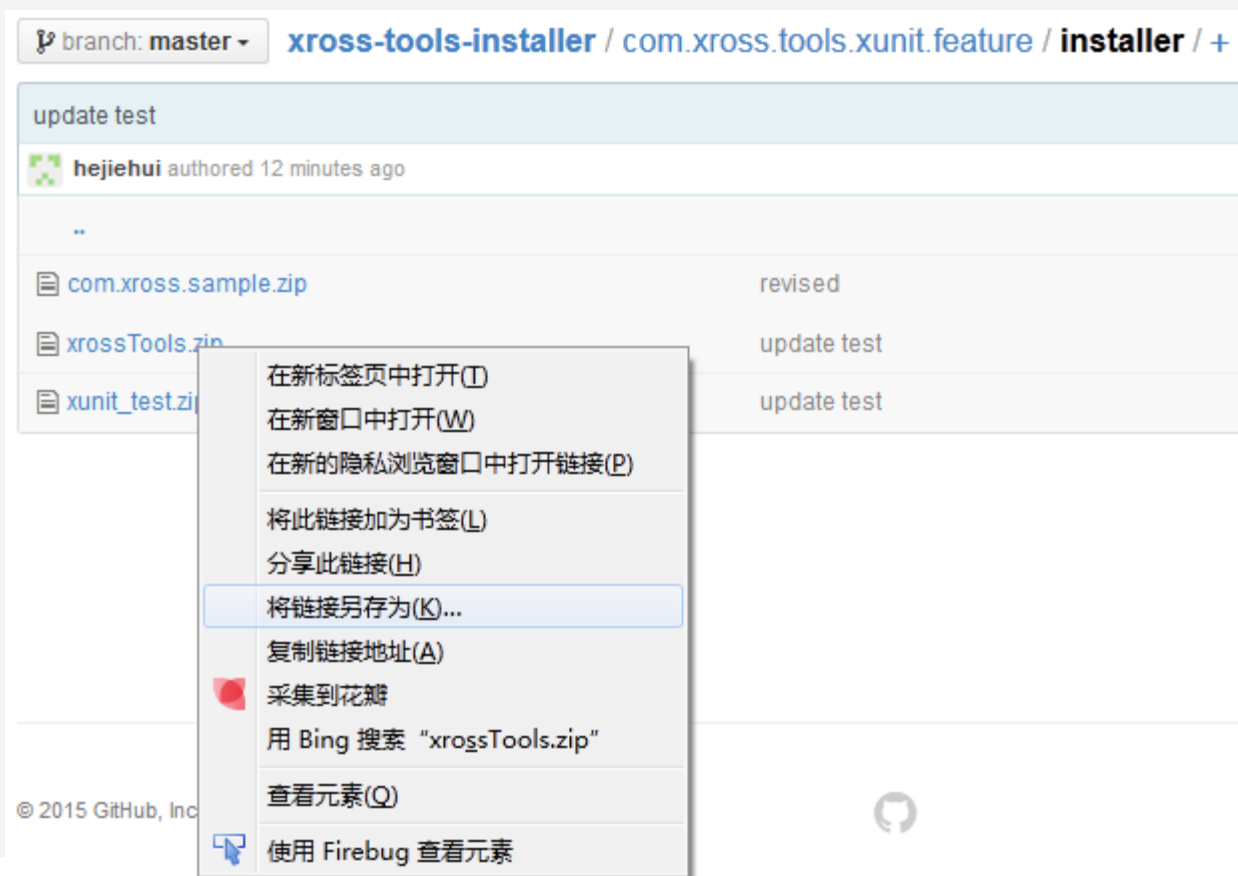
▲ All-in-one Installer

<https://github.com/hejieshui/xross-tools-installer>

安装 X-series

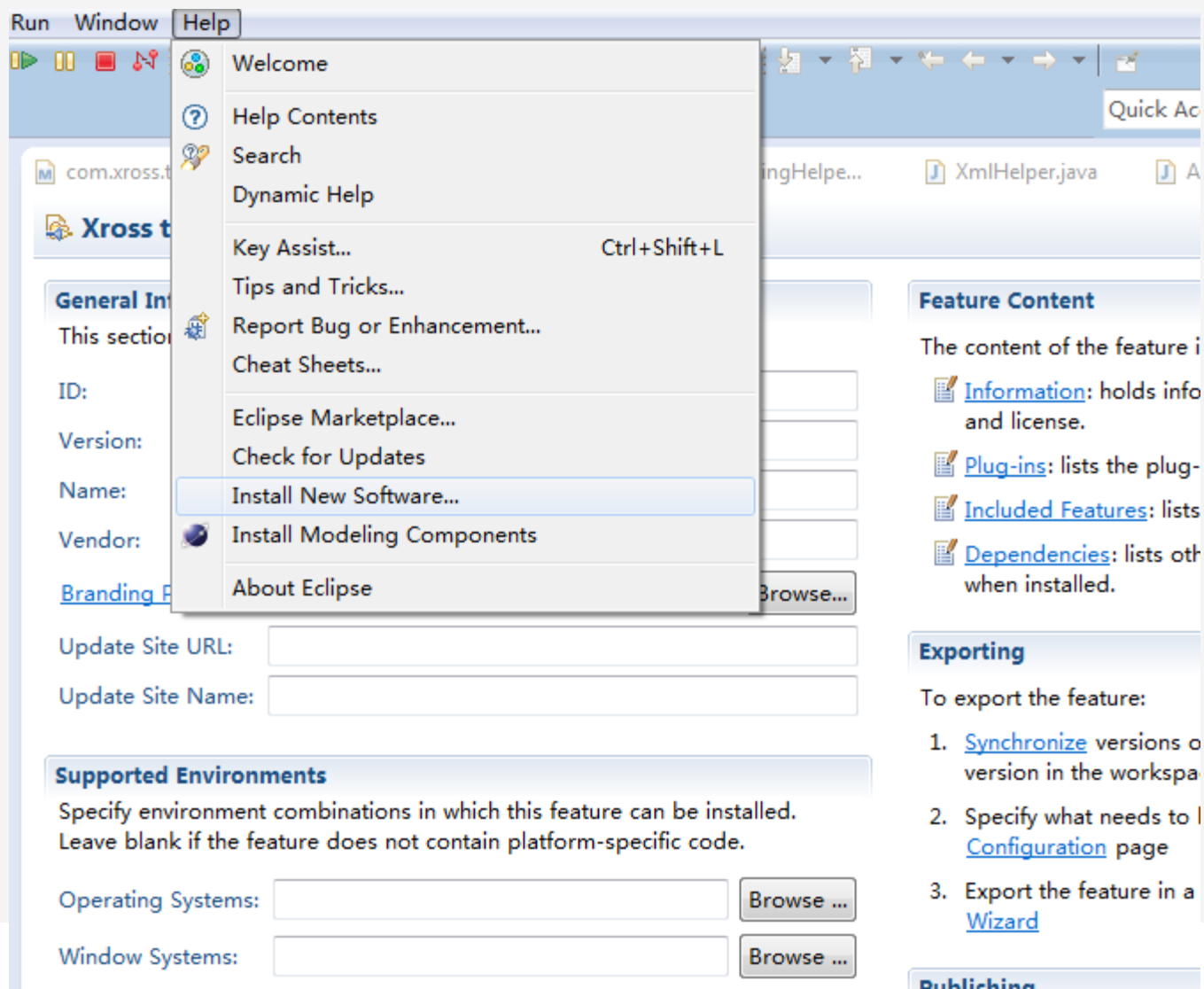
▲ 下载

<https://github.com/hejiehui/xross-tools-installer/blob/master/com.xross.tools.xunit.feature/installer/xrossTools.zip>



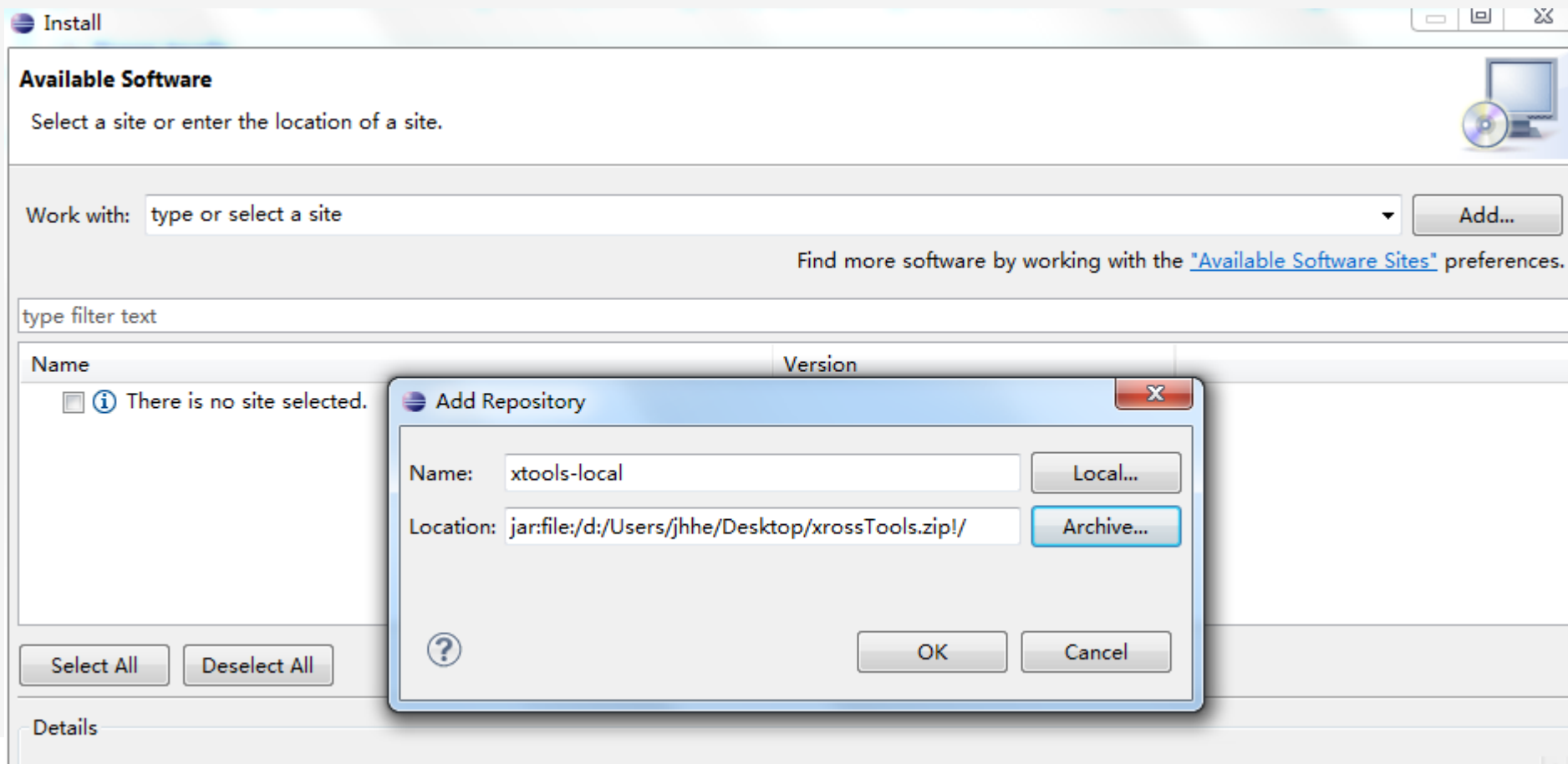
X-series安装

▲ 安装



安装 X-series

▲ 指定安装路径




安装 X-series

- ▲ In case you have installed GEF, you can uncheck the following

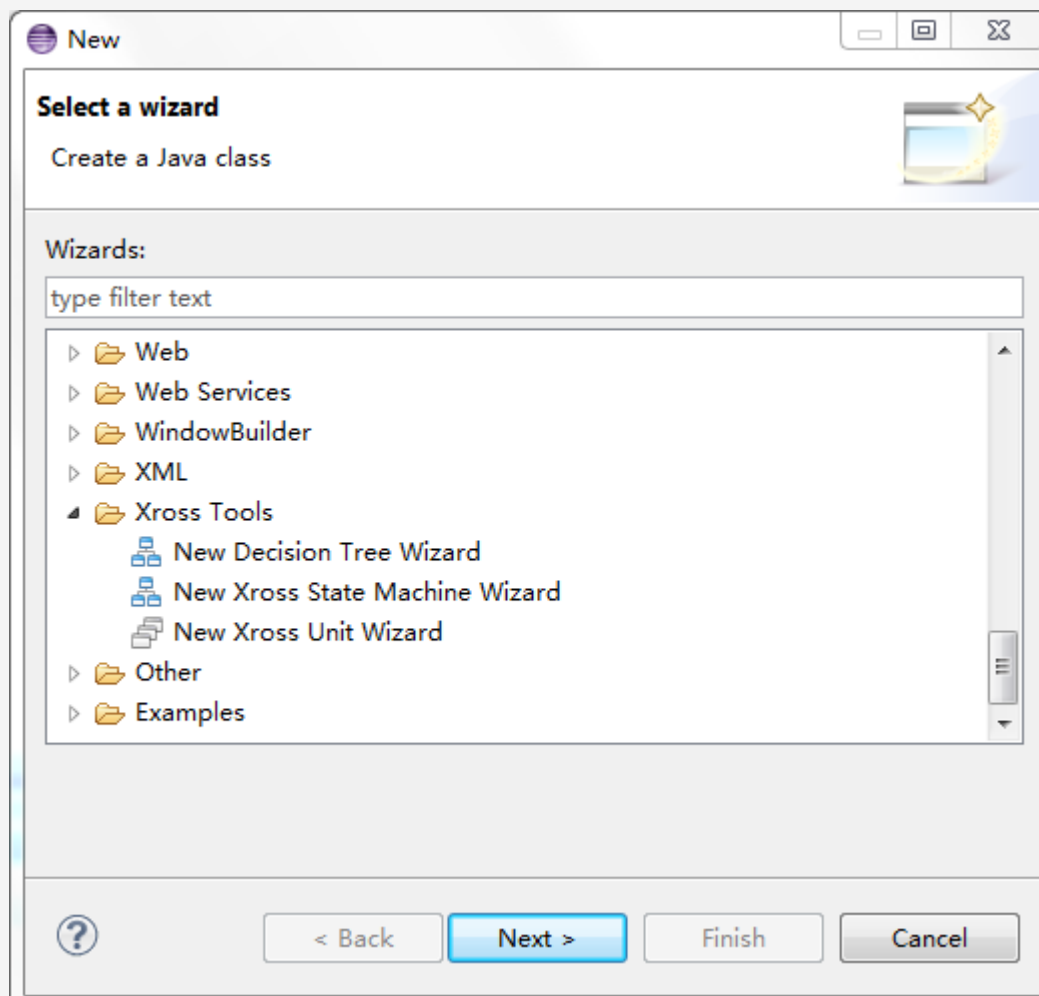
Details

<input type="checkbox"/> Show only the latest versions of available software	<input checked="" type="checkbox"/> Hide items that are already installed
<input type="checkbox"/> Group items by category	What is already installed?
<input type="checkbox"/> Show only software applicable to target environment	
<input type="checkbox"/> Contact all update sites during install to find required software	

 < Back Next >

安装 X-series

▲ New...Xross Tools



支持C#

- ▲ Xross unit C# runtime

https://github.com/hejiehui/xUnit/blob/master/doc/xunit_c%23.zip

- ▲ Xross State C# runtime

https://github.com/hejiehui/xState/blob/master/doc/xstate_c%23.zip

- ▲ Xross Decision C# runtime

https://github.com/hejiehui/xDecision/blob/master/doc/xdecision_c%23.zip

Before The End

- ▲ 在语言层面打转是没有出路的
- ▲ 大规模开发必须要求合适的工具
- ▲ 方向和眼光永远比速度重要
- ▲ 请开始使用X-Series

問題比答案更重要