# 软件分析技术大作业书面报告

# ——Css-Tracker：样式作用域定位工具

## 概述

网络应用前端开发中，CSS、SCSS样式作用域不明确给开发带来困难，增加出错的可能性。因此本小组设计一个工具，可以在开发中定位样式具体的作用域，并针对前端开发框架Marionette作了实现，开发了一个eclipse插件。经测试，该工具能有效定位样式的作用域到文件中，帮助前端开发人员进行高效、便捷的开发。

项目托管于Github，地址是：<https://github.com/swordghost/Css-tracker>。

## 组织和分工

本组成员和分工如下表：

|  |  |  |  |
| --- | --- | --- | --- |
|  | 学号 | 姓名 | 分工 |
| 组长 |  | 史默臻 | 算法实现 |
| 组员 |  | 吕广利 | eclipse插件 |
|  | 1601110646 | 吴恺东 | 算法设计、测试用例 |

## 问题背景与案例

网络应用前端开发中，网页的显示效果会使用各式各样的样式（style）进行控制。一般而言，样式会分类放置在不同的CSS文件中，以使前端的文件功能单一、结构条理。但因为CSS文件只包含简单的样式定义，SCSS文件被开发出来，作为具有一定编程功能的CSS使用。标准的CSS文件本身就符合SCSS格式，而SCSS文件经过SASS编译器编译可以生成标准的CSS文件。

网络应用前端开发人员在开发过程中修改SCSS或CSS文件后，因为HTML结构的相似性，可能出现不期望的网页显示效果变化。但一个网络应用的网页可能涉及成百上千个之多，难以准确定位，也就给样式的修改产生了困难。

我们以前端开源开发框架Backbone构建的SPA（单页Web应用，Single Page Web Application）为例，来看看这种问题的具体表现：

在开发中，实际使用的多是Backbone的扩展框架Marionette（backbone.marionette）。Marionette比Backbone提供更多的功能和更便利的开发方式。借助打包器Webpack和ECMAScript 6翻译插件，可以在开发中使用ECMAScript 6（可以理解为一种javascript，但是语法等方面支持类似于java的风格）开发，经过打包后产生可以在浏览器中显示的SPA。

Marionette中，页面元素由多个子元素组成，每个子元素由View生成，而View由controller.js负责加载到主页面中。每个View定义在自己的view.js文件中，继承自Marionette提供的View类；View可以用.hbs文件（由HTML的标签等组成，就是一个局部的HTML文档）定义基本页面元素，并通过绑定.scss文件来控制样式。Webpack打包后，.scss文件会被编译为css，放在<style>标签中加入<head>内，因此其作用范围是全局。理论上，每个.scss都可能作用到任意View中。

接下来看一个实际案例。这个工程由两个View组成，controller.js中showPage()被调用后负责将两个View处理并加入页面中去。每个View都有自己的view.js、.hbs和.scss：

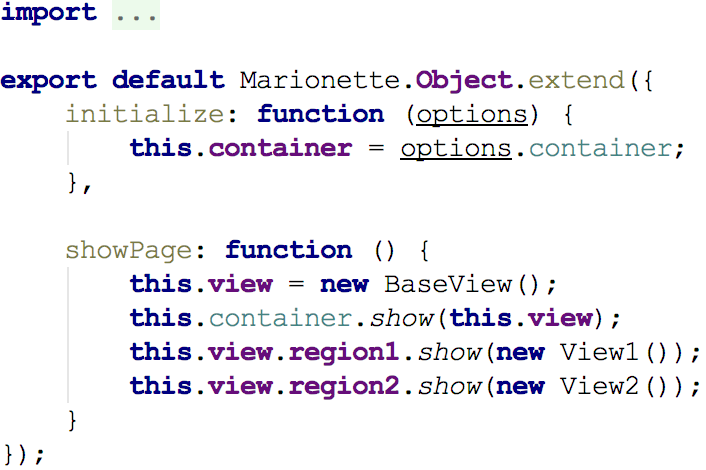


图1 controller.js

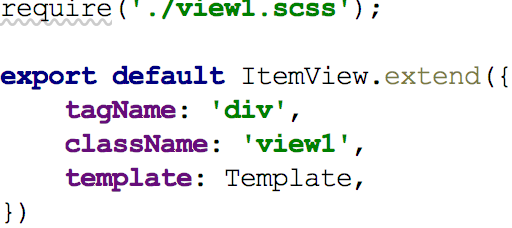


图2 view1.js

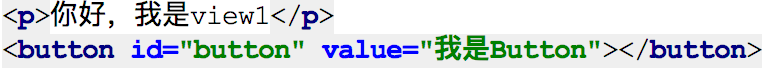


图3 view1.hbs



图4 view1.scss

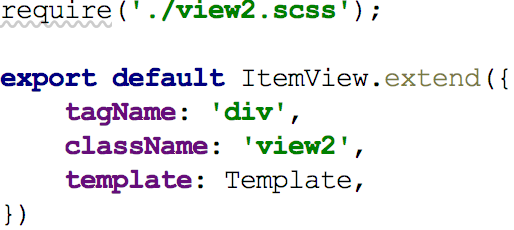


图5 view1.js

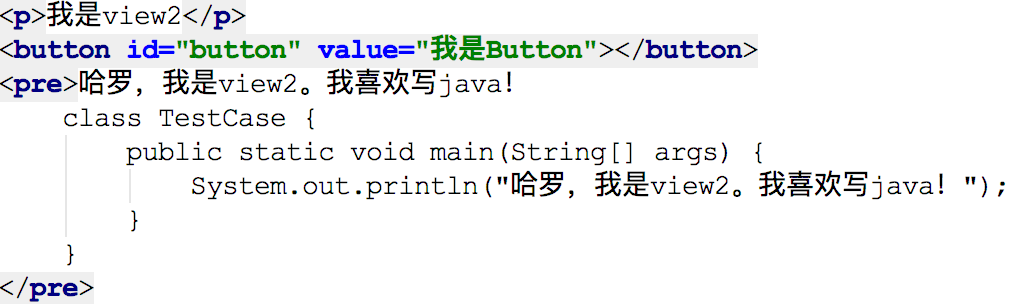


图6 view2.hbs

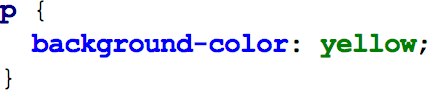


图7 view2.scss

注意这里的.scss绑定。开发人员的本意是让view1.scss只控制View1，让view2.scss只控制View2，也就是View1内的<p>不变黄，View2内的<button>不变蓝。然而，该工程打包后显示结果如下所示：

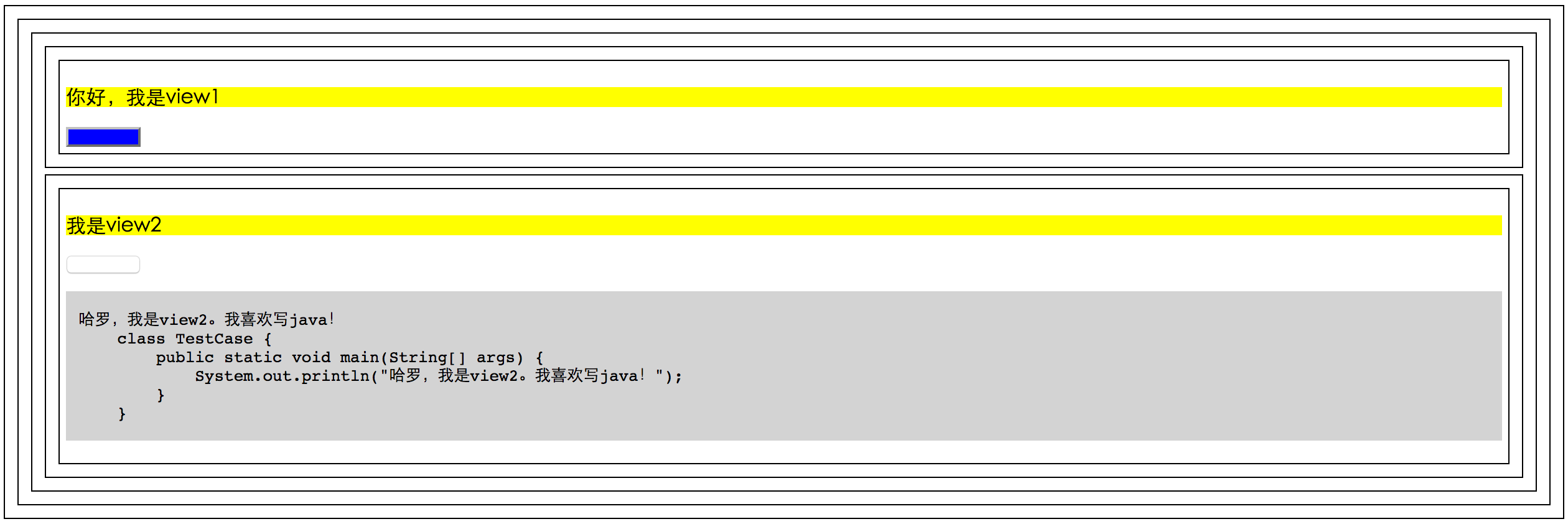


图8 打包后显示的页面

可见View2中的<p>错误地变黄了。

上述只是一个例子，错误很容易被定位然后解决。然而，真实场景中发生的错误会复杂很多，定位并解决错误所需的时间可能会千百倍延长，所耗费的人力也不可同日而语。

## 目标与实现

### 目标

根据上述，我们的目标就是实现一个工具，能定位.scss实际影响的页面文件。该文件能准确定位页面文件的位置，不受开发框架的干扰。最好是有一定的可视化UI，进一步方便用户的使用。

最终我们实现了一个eclipse插件显示.scss中样式实际影响的页面文件，在IDE中显示，并能借此跳转到对应的文件中。项目地址：

<https://github.com/swordghost/Css-tracker>

### 实现

实现分为两个层面，算法设计与分析，以及eclipse的插件开发。

#### 算法设计

要根据前述对想了解各个SCSS文件具体影响到了哪些view和哪些view中的哪些元素，就要模拟Webpack打包的过程，生成简化版的html代码。之后，我们对用户选中的SCSS中的语句作分析，首先获得它的文件位置和对应位置的Style类型，之后找到相应的Selector，从DOM树中找到对应元素并显示。

#### 算法实现

本次项目的算法部分代码在buildTreeForCase2.java中实现。

1. 建立DOM树

网页由许多View组成，统一由controller.js控制。View由Backbone.View派生。每个View都可能有一个.hbs指定其HTML模板。例如：

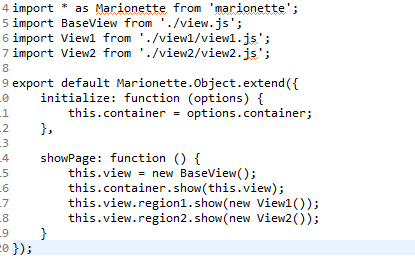


图9 controller.js

上面的Controller例子首先引入了view1.js和view2.js两个文件，并且在后面将二者show到region1和region2中。其中，region1和region2在BaseView中定义，controller.js将相应的View渲染后放到对应的标签中。

具体在实现时，我们使用getController函数通过entry.js找到项目中的controller路径，之后通过BuildTree函数完成建树过程。

BuildTree函数的工作过程是这样的。首先通过findImport函数找到引用的javascript文件的路径，之后调用shell函数解析各个对应hbs并包“壳”。

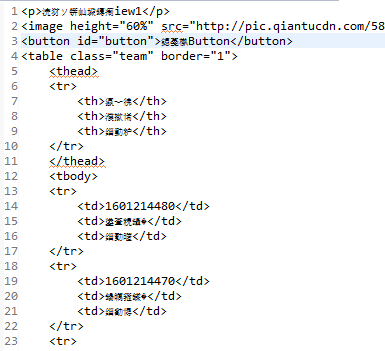


图10 view1.hbs

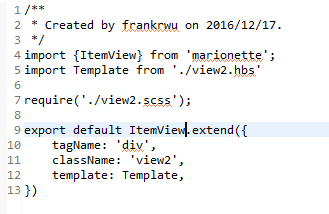


图11 view1.js

如上面实例所示，每个View的.hbs的结构和一般HTML文件的结构是一致的。view2.js如上所示，为View包上了一层“壳”，比如上面添加的就是classname为view2的div标签。因此套“壳”就是在解析.hbs文件后，将其加上一层标签作为DOM树中的节点返回；另外为了得到各个文件对应的地址，将路径作为一个属性加入节点中。这部分工作就是在shell函数中完成的。

总的来说，上面的各个过程汇总起来，我们就完成了建立DOM树的过程。大概流程如下：

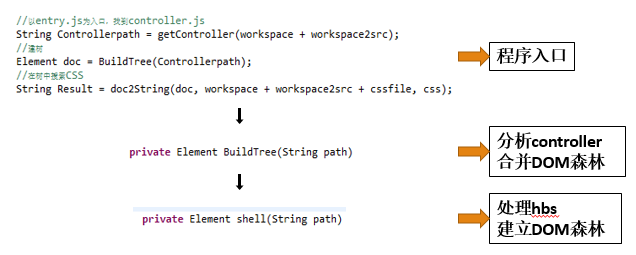


图12 建立DOM树的流程

获得的DOM树如下：



图13 建立好的DOM树

1. 分析SCSS

这部分的工作主要是解析插件返回的SCSS文件，在这些文件中找到插件返回的SCSS语句。之后找到对应的selector，从DOM树中找到selector对应的所有子节点，这些子节点都可能被相关的SCSS语句影响。

这部分功能在findSelector函数中实现。首先通过正则匹配大括号递归找到每个style和其对应的selector，比如在下图的SCSS代码中，background-color的selector就是view1中的button标签。我们记录下这些selector对并返回。

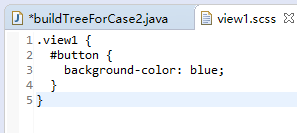


图14 view1.scss

1. 返回结果

这部分我们得到了刚才的DOM树和Selector，功能就比较简单了，只需要从树中找到selector对应的所有子节点就可以了。这部分功能在doc2String函数中实现，最后将所有的结果整理成给出规则的字符串，规则为：对每一个selector进行统计每一行是一个tag和其路径，用逗号隔开，实例如下：

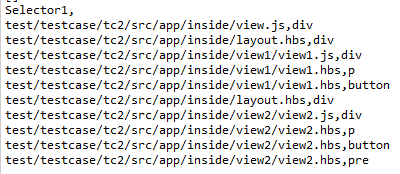


图15 算法输出

#### 插件开发

在eclipse中建立一个插件工程CssTrackerPlugin，在此基础上进行插件开发。

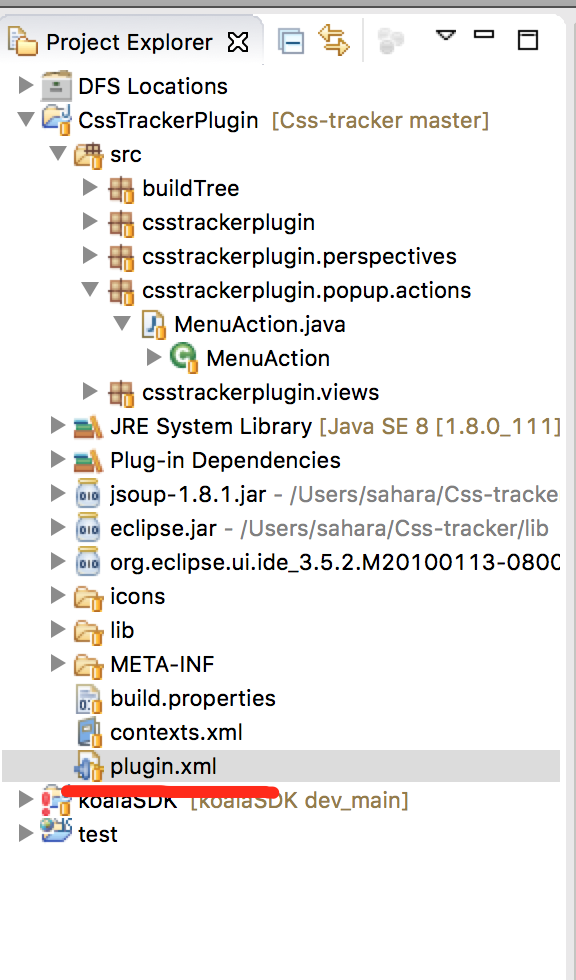


图16 工程结构

首先添加右键菜单。打开plugin.xml：

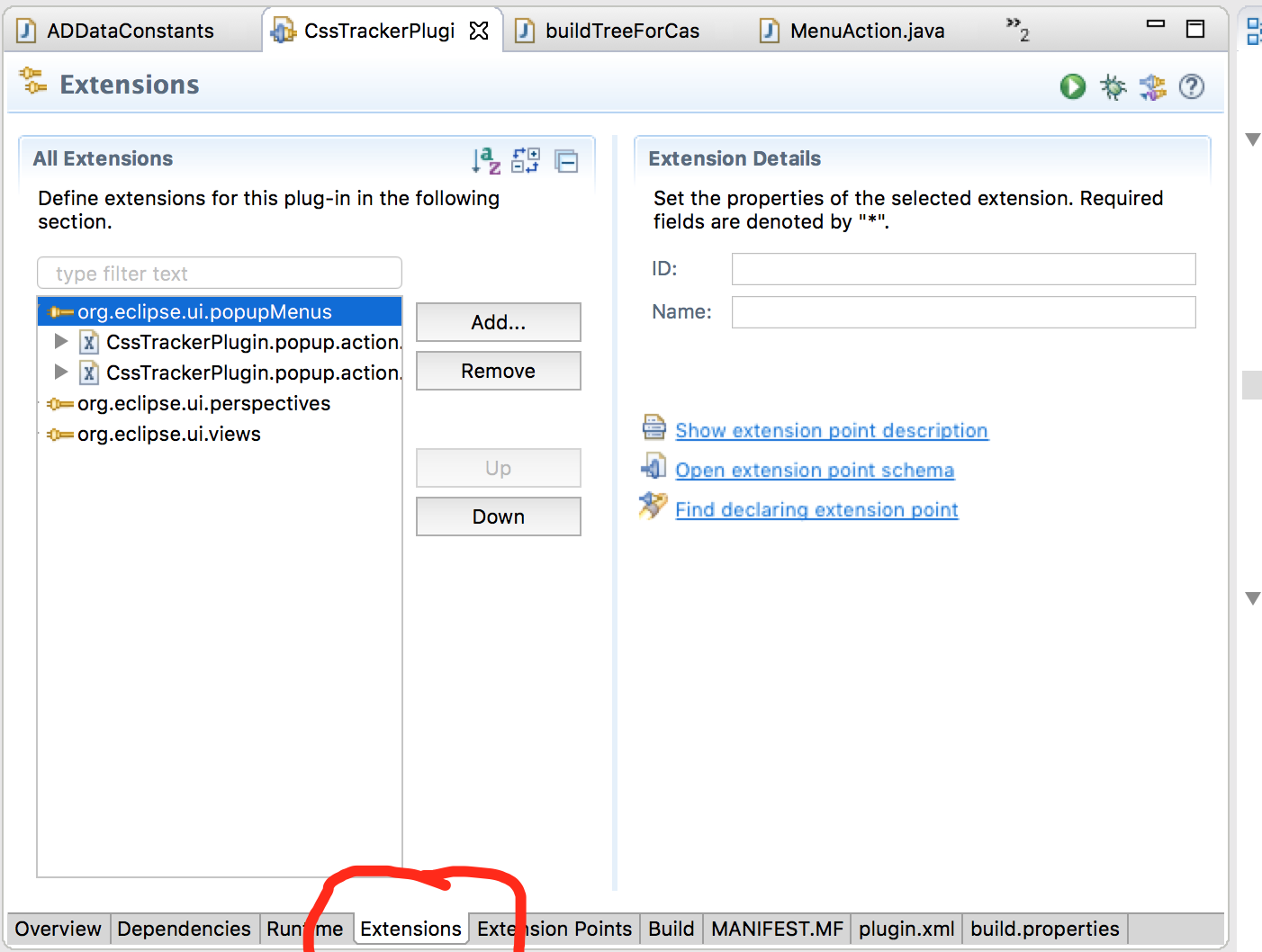


图17 plugin.xml

点击Add：

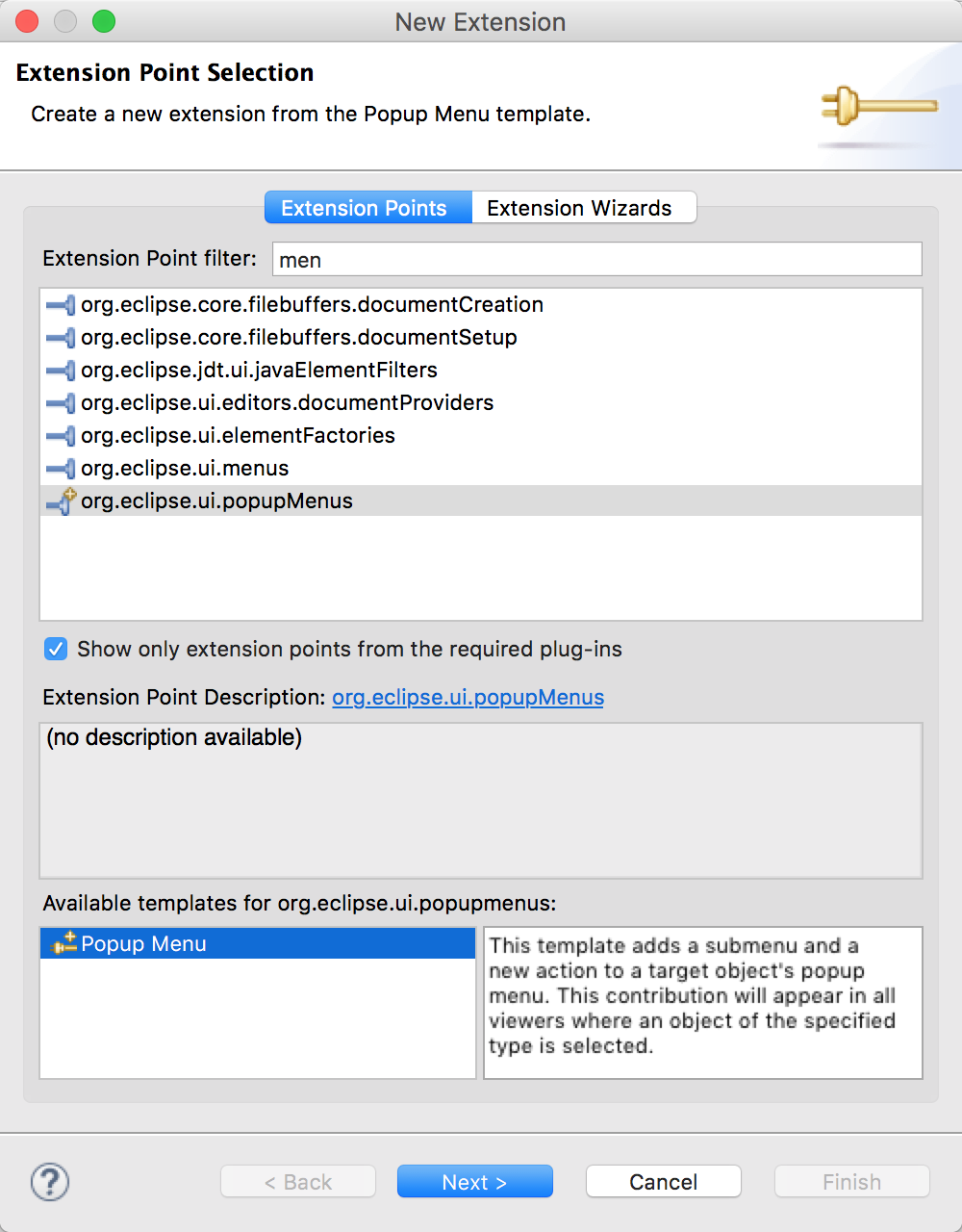


图18 Extension的Add界面

选中org.eclipse.ui.popupMenus（表示右键菜单），然后点Next设置控件属性：

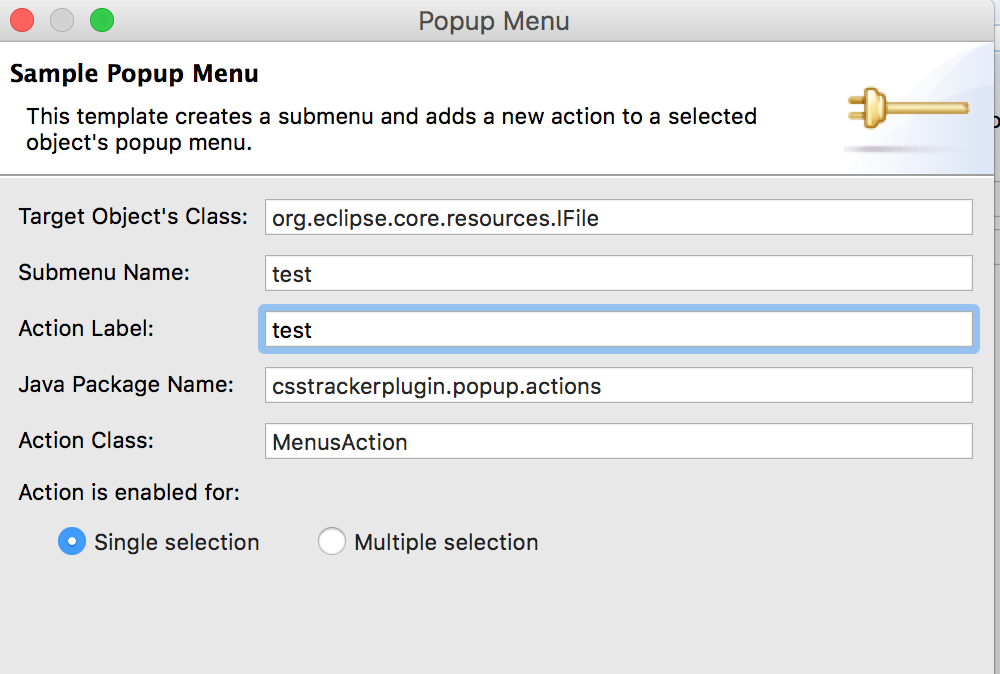


图 19 Popup Menu控件属性

第一个参数Target Object’s Class是该菜单的功能，我们的插件涉及的取值有：

|  |  |
| --- | --- |
| org.eclipse.core.resources.IFile | Java文件的右键点击菜单事件 |
| org.eclipse.jdt.core.IJavaElement | Java元素的右键点击菜单事件 |
| org.eclipse.core.runtime.IAdaptable | 任意元素的右键点击菜单事件 |

Submenu Name和Action Label分别是一级菜单和二级菜单的显示文字。

之后eclipse会自动生成实现类（需填充）：

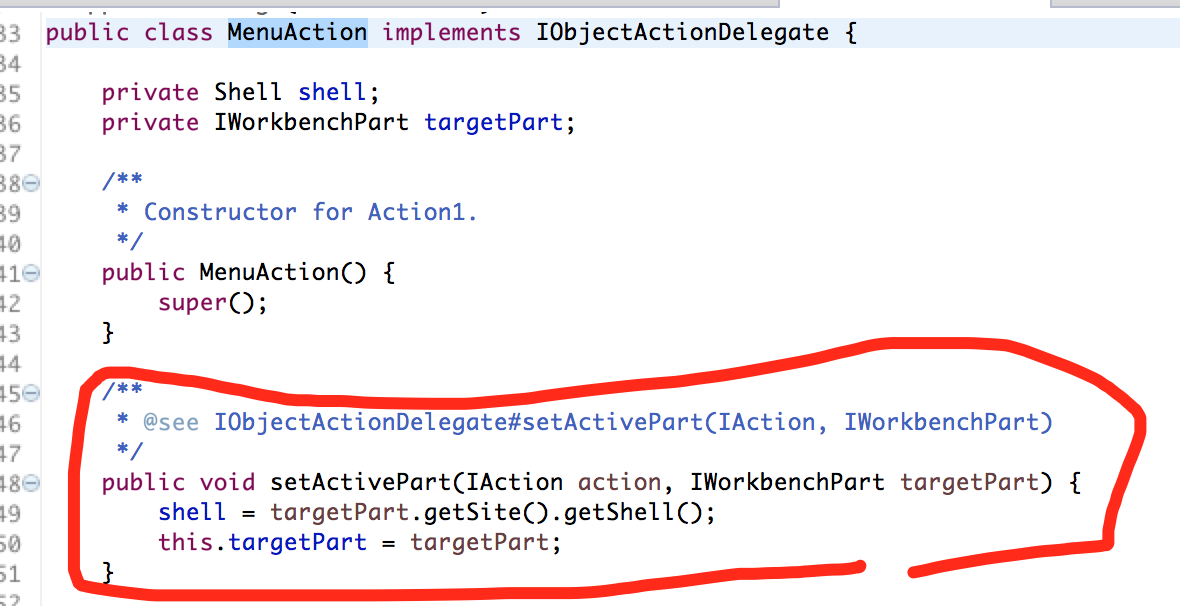


图20 自动生成的实现类

这个类中有一些重要的方法：

1. setActivePart()：获取全局参数，类似初始化方法。这里得到一个part，用于获取工程名和获取编辑器窗口。
2. run()：传递action（即事件）。这里即是选中元素右键并点击弹出菜单中相应项的事件。

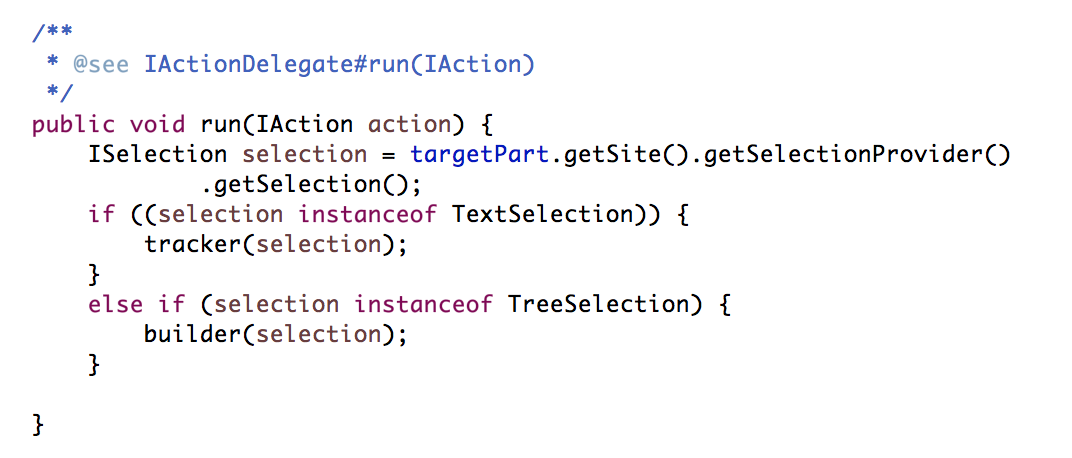


图21 run方法

首先判断是否选中了文件内的部分内容，若是则执行Css-Tracker的分析。



图22 tracker方法

检查选中的内容，如果没有内容则清空插件窗口并返回；若有，则将内容传递给算法，生成一个结果集并显示出来。

插件窗口也需要在plugin.xml中注册，并需要另外注册一个透视图。操作与上述类似：

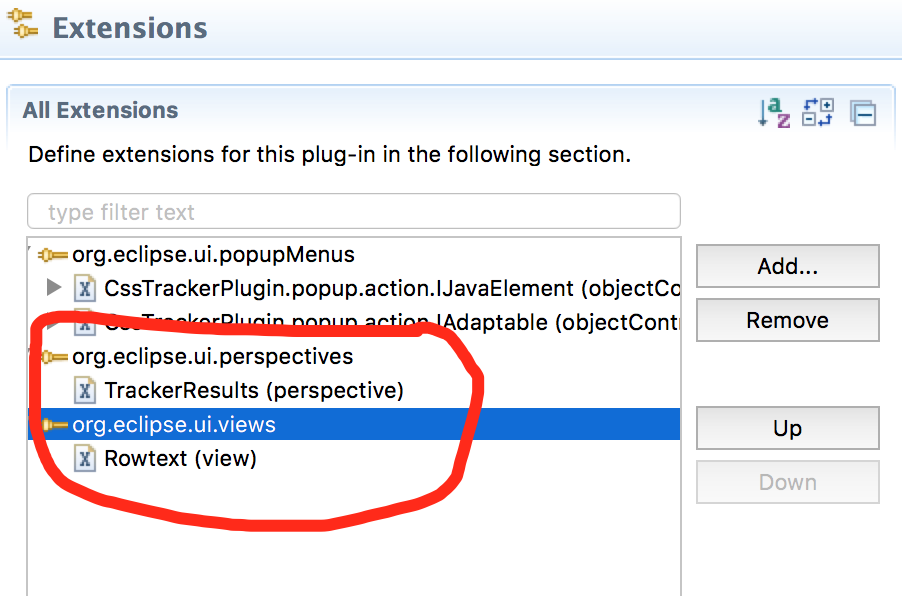


图 23 注册插件窗口（view）

eclipse自动生成代码：

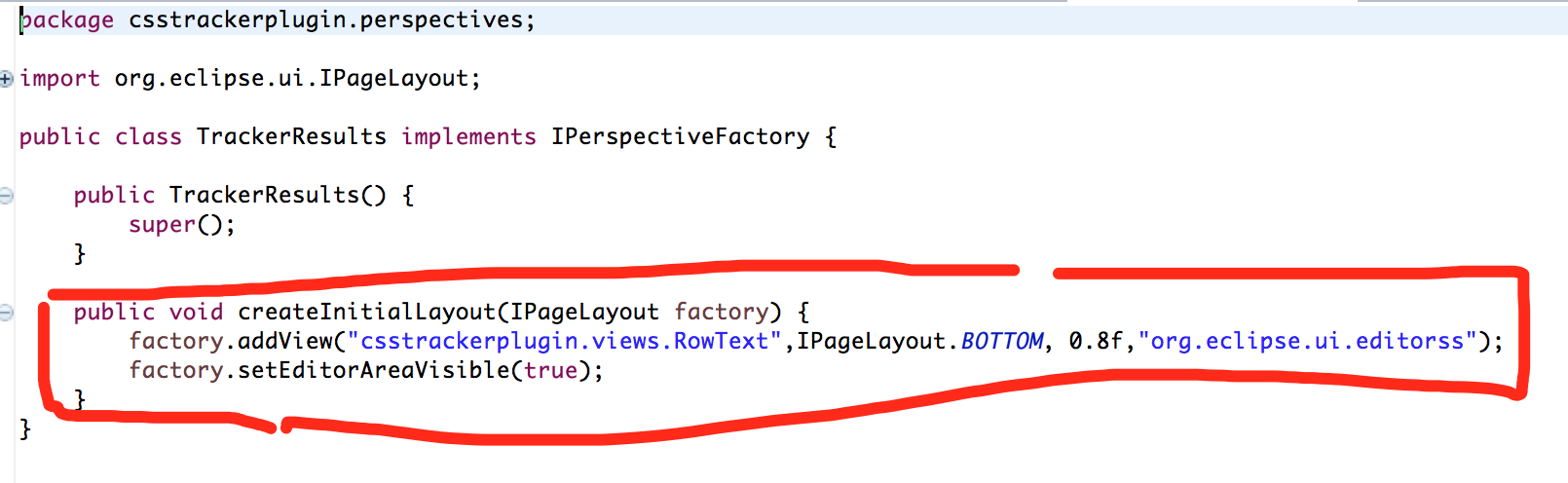


图24 自动生成的view代码

上图中标记出来的代码的作用就是透视图位移，设置view的显示位置位于编辑器的下方。

这个窗口如图所示：

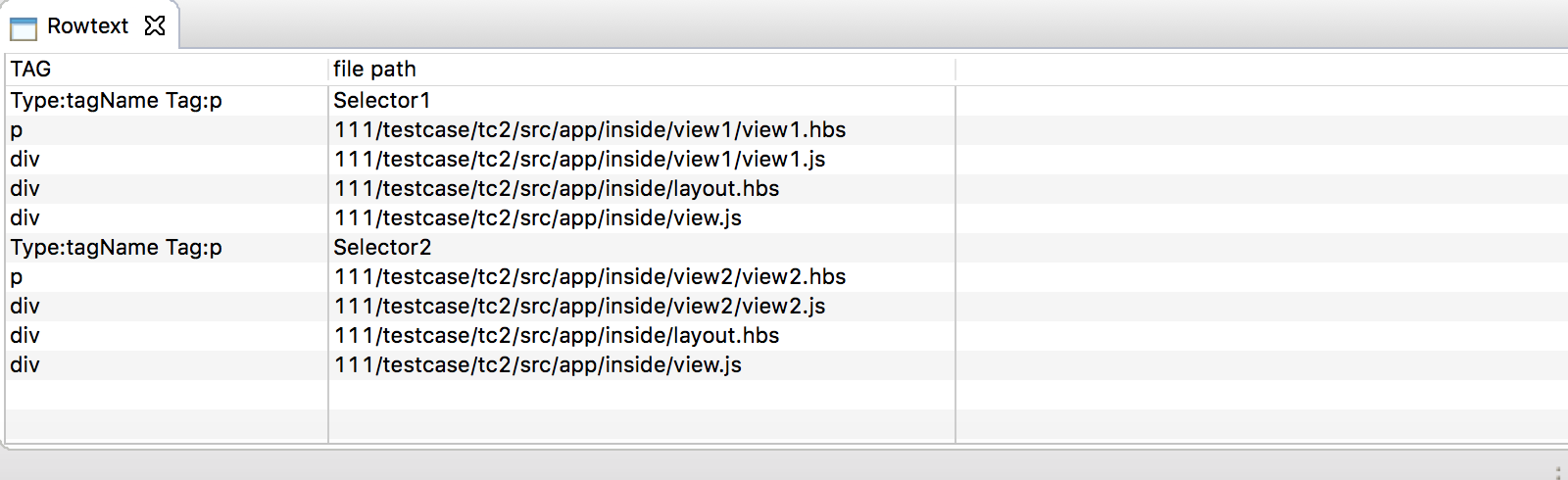


图25 生成的插件窗口

如图所示，窗口内有两列，是一个表格（Table）。因此需要实现TableView来展示数据。

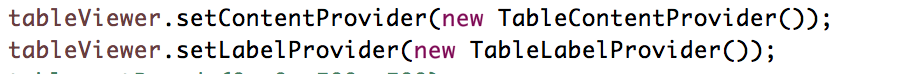


图26 TableView的具体实现

TableView的实现需要实现如上的两个类。这两个类提供了TableView自动解析数据的能力。我们的数据是简单的String[]形式，因此实现如下：

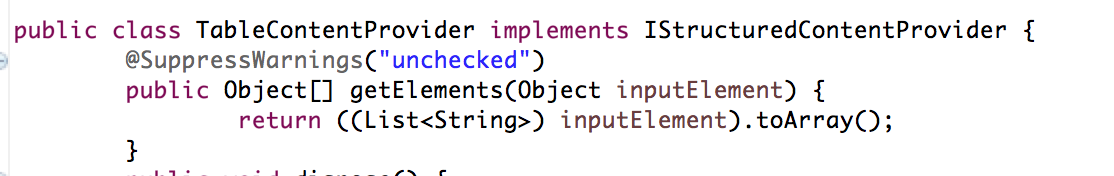


图27 ContentProvider的实现

这个类将String[]转化为一个list。

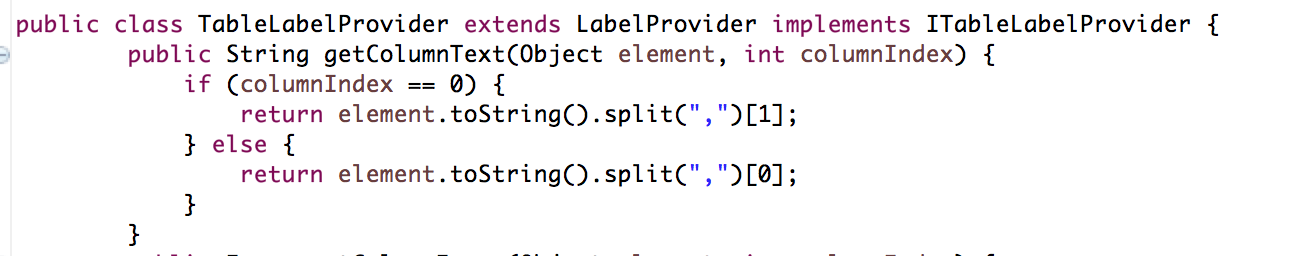


图28 LabelProvider的实现

这个类将list的每一列都读出来，因为原始数据和结果显示的顺序不一样，所以要处理一下。

接下来设置TableView的参数：

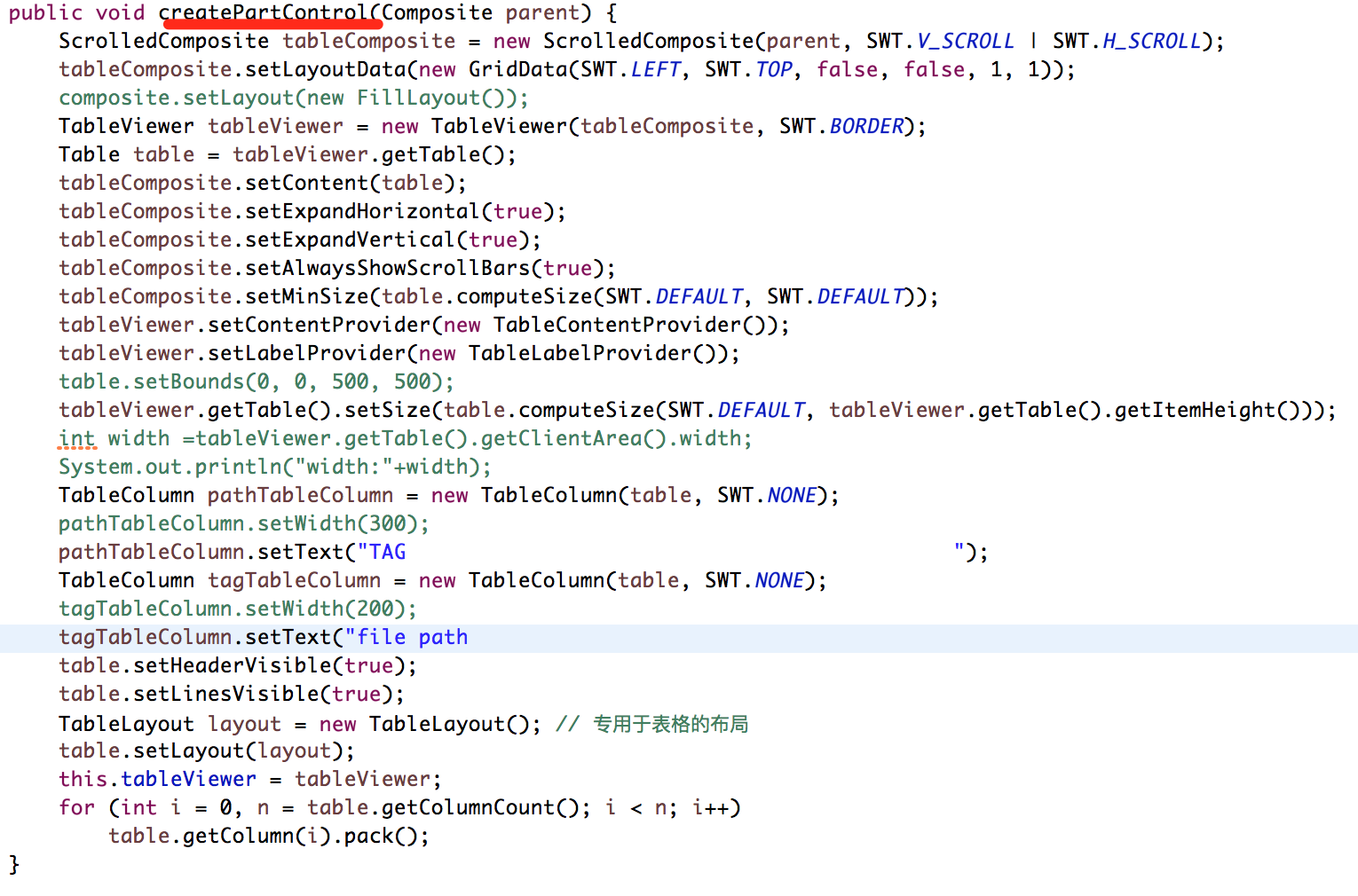


图29 createPartControl和TableView的参数设置

代码实现在方法createPartControl()中，因此可以在打开透视图的时候立刻被创建。参数配置涉及TableView的自动扩充、列自动适应、对齐、鼠标滚动。

接下来是双击行打开对应文件功能的实现。



图30 双击行打开对应文件功能的实现

获取到Table，然后监听Table的鼠标事件（通过listener），响应双击事件。获取到工程所在目录，并解析算法返回的数据内容的格式，生成最终路径。之后判断路径是否存在，存在的话就调用IDE接口打开编辑器视图；如果不存在，说明双击事件选中的不是文件，那就可以不响应。

于是插件功能实现完毕。

实现过程中遇到了不少问题，几乎都是eclipse提供的接口功能性不足而导致的。另外就是eclipse版本自带的支持包的不同导致的跨版本兼容性问题。这些问题都通过查找资料（主要是QA网站的资料）一一解决。

## 测试与验证

将插件安装到eclipse中后，将前端工程文件在eclipse中打开，然后在.scss文件中选中任意样式并右键点选Css-Tracker，就会显示结果。双击结果项即可打开对应文件。如下图：

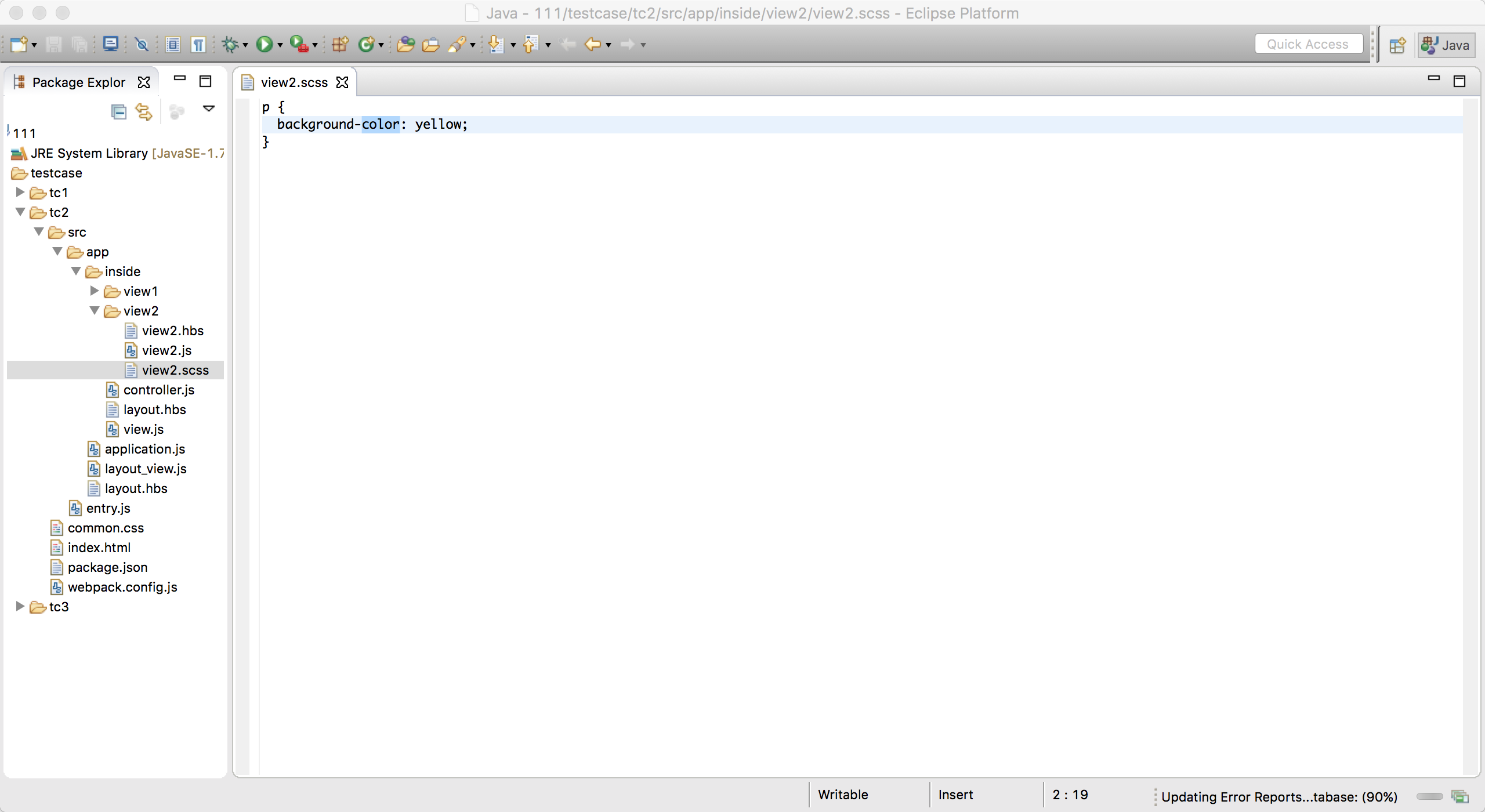


图31 选中样式（不要求全部选中）

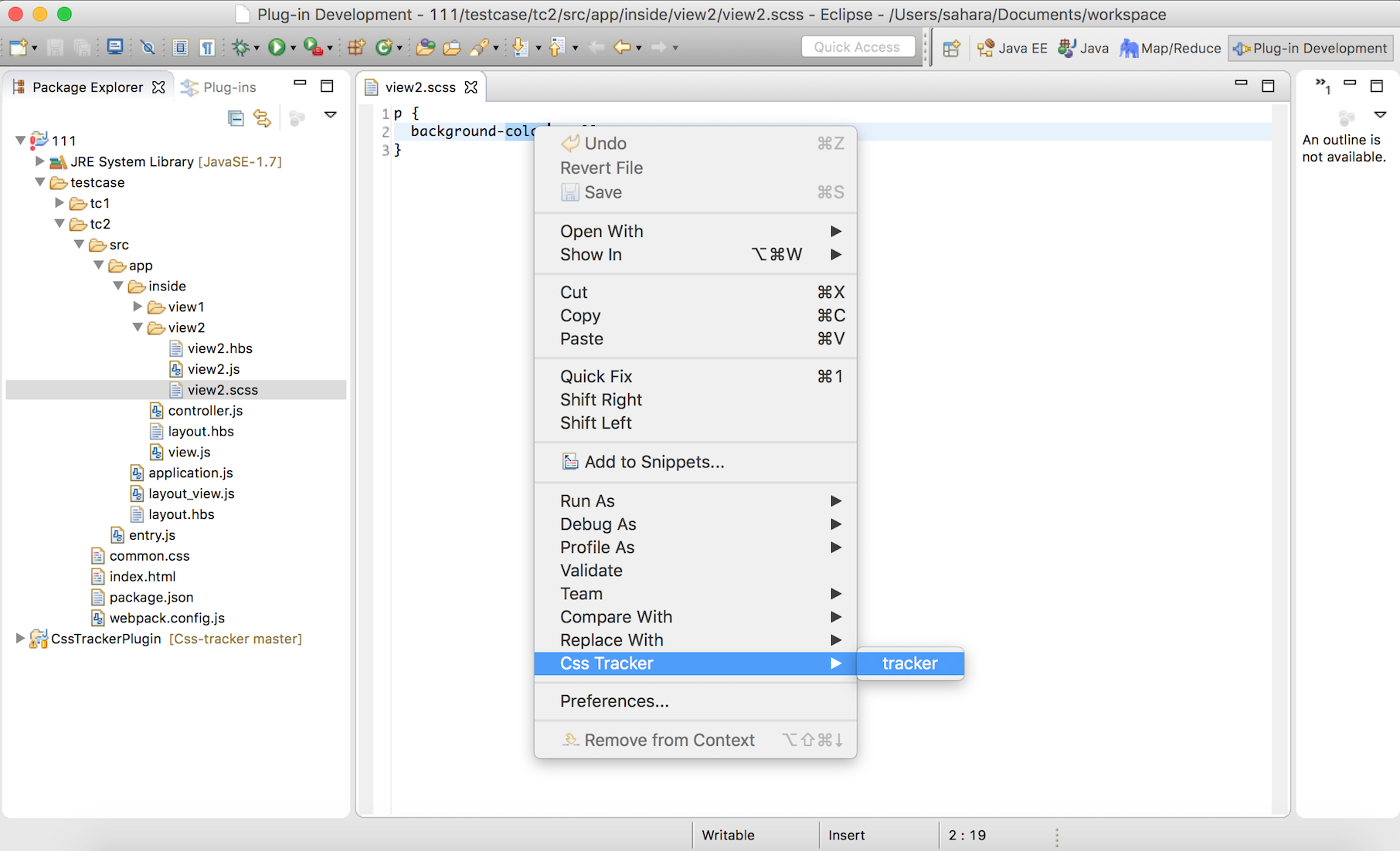


图32 点选Css-Tracker



图33 在结果窗口中双击行

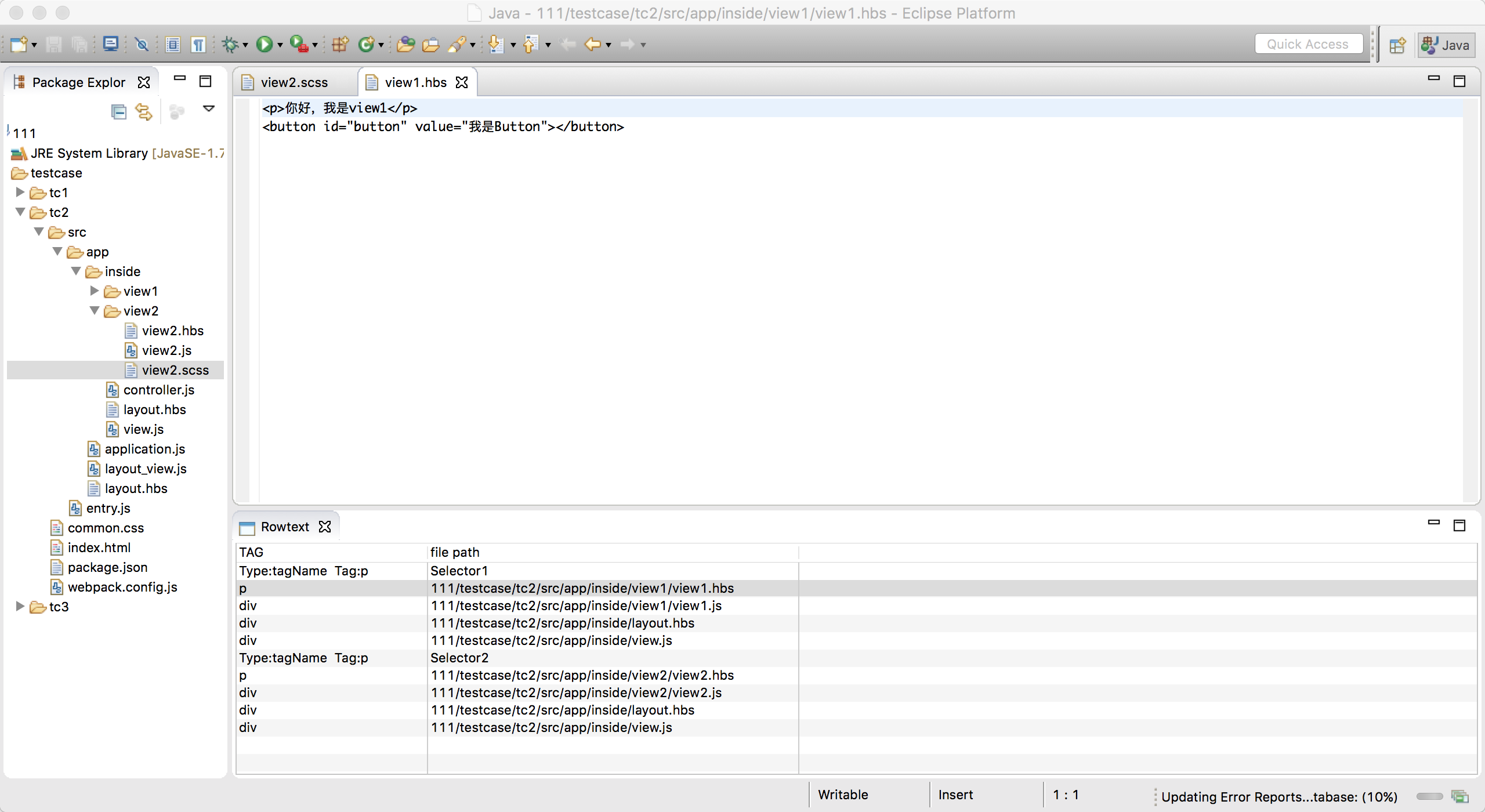


图34 编辑器自动打开对应文件

## 结论与收获

我们设计并实现了一个拥有可视化界面的、帮助前端开发人员跟踪样式对应页面元素所在位置的eclipse插件，能够正确地返回DOM树中样式实际影响到的所有节点，并能够跳转到文件中。这让我们对软件分析技术有了更深的了解，并锻炼了实际的工程能力和协作开发能力，获益匪浅。