# BrowerAdapt模块

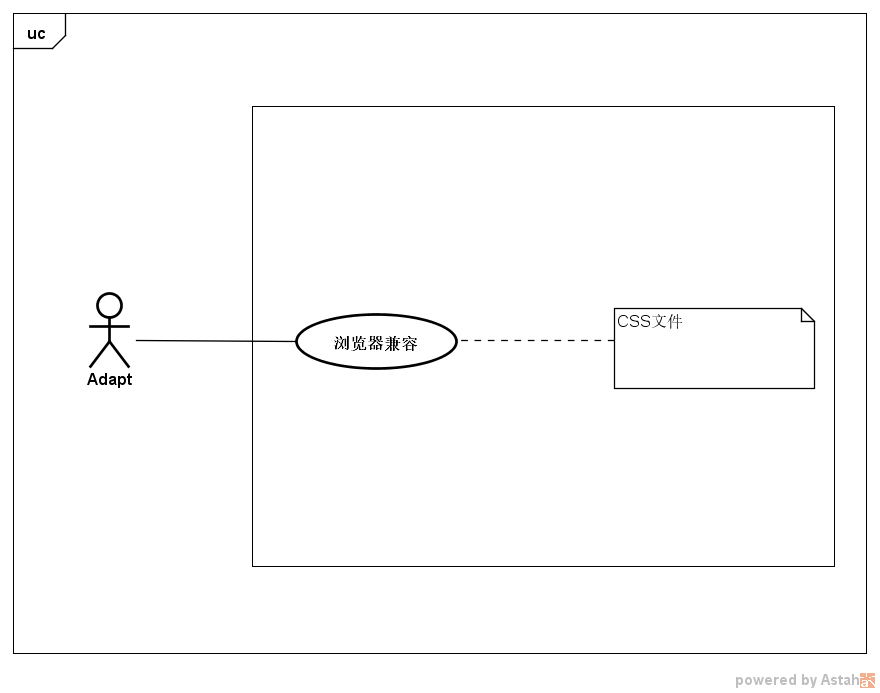
## 概述

### 承担的需求（主要秘密）

浏览器兼容模块的职责是根据用户浏览器类型及版本的不同，自动采取兼容措施，保证网页在不同浏览器上仍保持一致的显示效果。

**质量属性要求**：网页显示要求具有较高的兼容性，而变更的部分主要是用户浏览器类型及版本的变化。因此该模块的设计就要着重考虑网页CSS对不同浏览器的兼容性，可以灵活变更又不影响其他模块的调用，同时，为避免CSS进行兼容变更之后文件过大影响网站加载速度，采取按兼容版本不同编写相应的同名CSS文件，并由浏览器兼容模块统一调度。

**浏览器兼容用例图**



**用例描述**

1. 浏览器兼容

当用户访问网站时，系统自动获取用户的浏览器版本，并返回相应的CSS文件，

以达到兼容浏览器的目的。

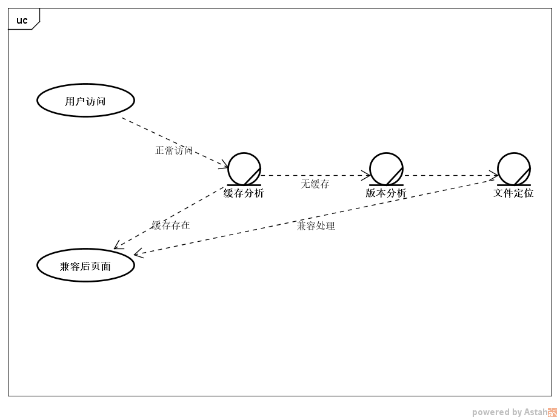
**可能会修改的实现（次要秘密）**

扩展兼容版本

**角色**

1. CSS缓存文件分析
2. 浏览器版本分析
3. CSS文件重定位

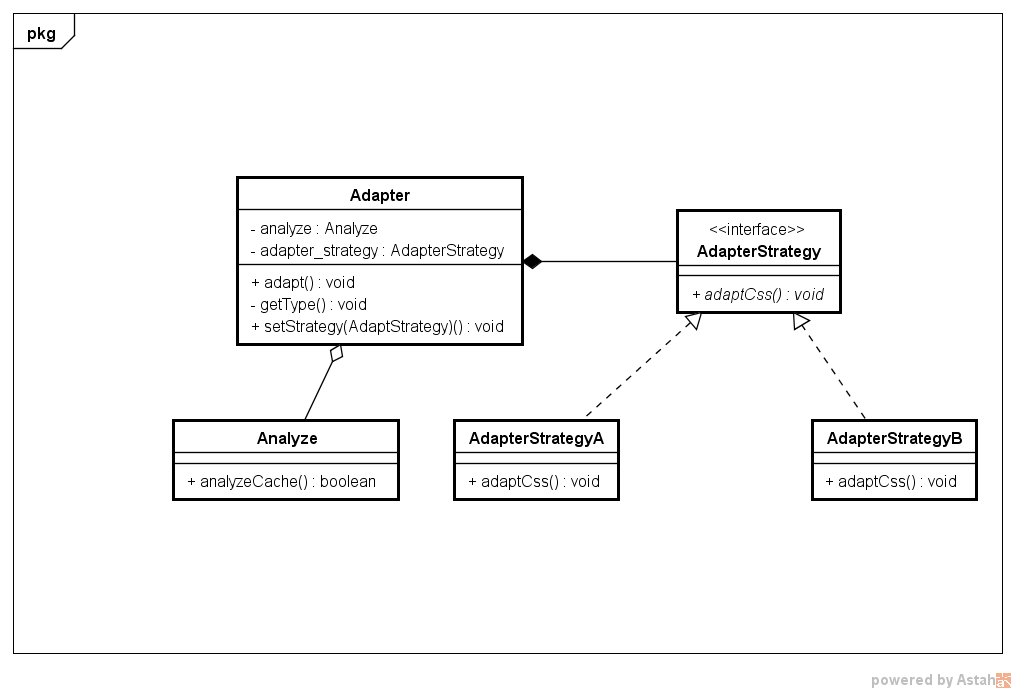
**角色流程图**

****

**向外提供接口**

|  |  |
| --- | --- |
| **接口名** | **职责** |
| Public void adapt() | 根据浏览器类型，选择合适的CSS文件 |

## 类图



## 类描述

### Analyze类

该类的职责是分析用户浏览器是否缓存了相应的CSS，以便

Adapter类判断是否需要进行兼容操作

**类方法：**

|  |  |
| --- | --- |
| 方法名 | Public GoodsPriceList findNamesake(Goods goods) |
| 前置条件 |  |
| 后置条件 |  |
| 作用 |  |

Boolean analyzeCache();

检查是否有网站缓存，并返回结果

前置条件：用户正常访问

后置条件：无

### Adapter类

该类的职责是根据Analyze类的结果判断是否需要进行兼容处理，如果需要，则根据兼容策略进行兼容处理

**类方法：**

Void setStrategy(AdaptStrategy)

组合模型表明在Adapter类的构造函数中其实就已经指定了兼容算法(AdaptStrategy),但为了更好地动态指定不同的兼容算法，我们可以使用setStrategy方法来重设Adapter对象的adapt\_strategy成员变量。

前置条件:Adapter对象已经被初始化；

后置条件: Adapter对象的adapt\_strategy成员变量被重新设置为新的算法对象

Void adapt()

Adapter类的主要方法，负责根据缓存分析结果和兼容策略，对用户浏览器进行兼容

BrowerType getType()

该方法获取用户浏览器类型版本等信息并返回

### AdapterStrategy接口

为了使得不同的兼容算法能够更好地复用，以及避免将来算法的变更对现有行为类

造成影响，选择策略模式将搜索算法抽象成接口，在接口中申明adaptCss方法，并由

具体的AdapterStrategy类实现

### AdapterStrategyA类、AdapterStrategyB类

这些类，是AdapterStrategy接口的具体实现类，实现了接口中申明的方法，并且数量在未来可能增加

**类方法：**

Void adaptCss(BrowerType type)

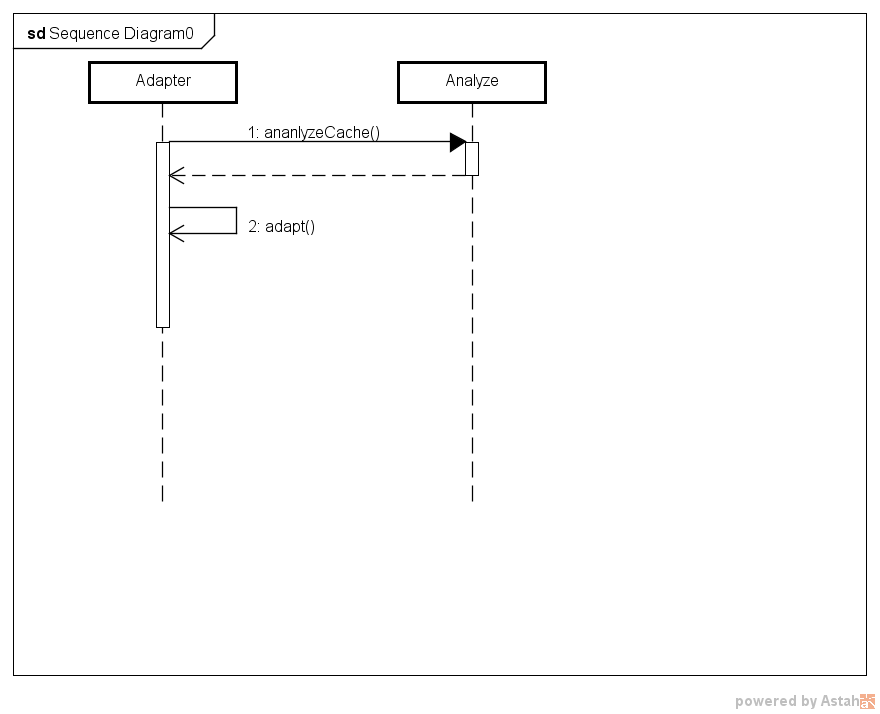
该方法是实现了接口中定义的adaptCss方法，参数type为用户浏览器类型信息

前置条件：getType方法以被执行

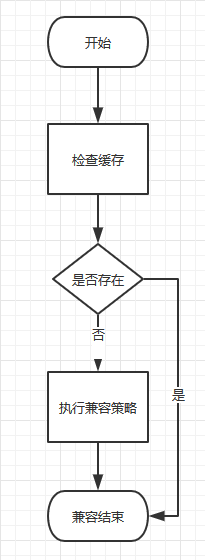
后置条件：无

## 重要协作

### 顺序图



### 流程图



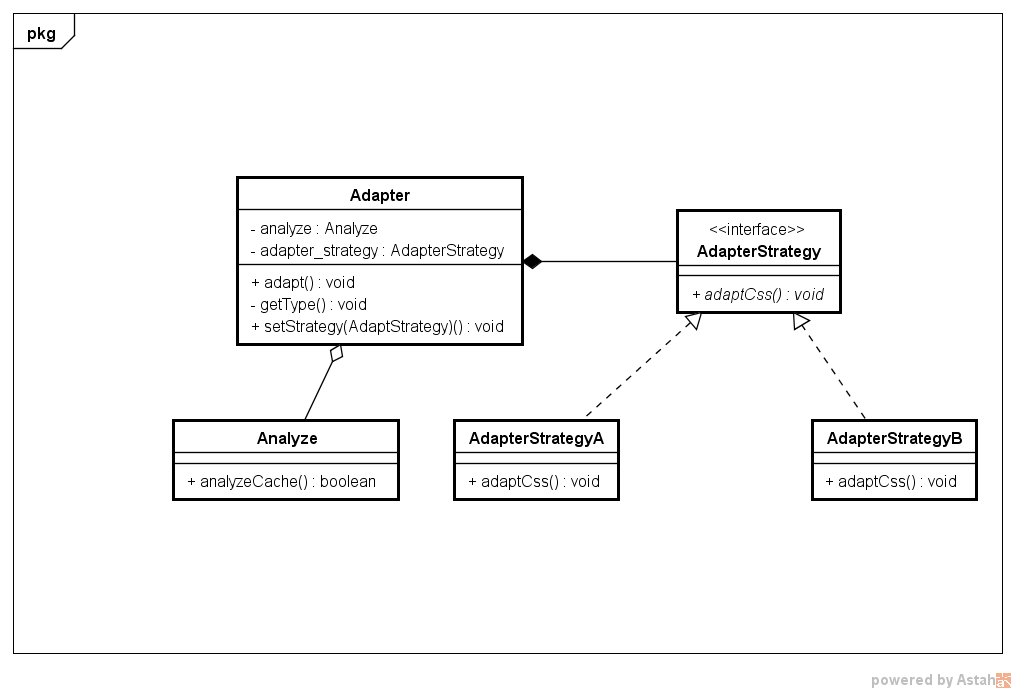
## 设计模式

### 策略模式

策略模式定义了一系列的算法，并将每一个算法封装起来，而且使它们之间可以相互替换。策略模式让算法独立于使用它的客户而独立变化。它提供了一种替代继承的方法，而且既保持了继承的优点(代码重用)，还比继承更灵活(算法独立，可以任意扩展)。同时避免程序中使用多重条件转移语句，使系统更灵活，并易于扩展。策略模式还遵守大部分GRASP原则和常用设计原则，高内聚、低偶合。

在浏览器兼容模块中，由于浏览器兼容算法以后可能变更，因此把该算法封装起来，实现接口adaptCss（），以便将来可以方便地修改。以便以后可以方便地修改。如果要新增算法，增加一个算法类，实现上述接口，并在声明的时候将引用指向新的算法类即可；如果要移除算法，移除对应的算法类并更新声明的引用即可；如果要修改算法，可在不修改接口的前提下修改算法类的内部即可，保证了高内聚和低耦合。

**类图体现：**

****

# CommentCheck模块

## 概述

### 承担的需求（主要秘密）

评价检查模块主要有四个职责：发布评价、敏感评价处理、水军检测以及商家定制候选词汇检查，如下：

1. 发布评价

用户提交评价后，系统根据敏感词汇表自动检查评价内容是否属于敏感评价

1. 若检查结果为正常评价，则发布该评价并保存至数据库
2. 若判断为敏感评价，则标志为待审核评价并保存至数据库，不发布评价的同时通知管理员进行审核
3. 敏感评价处理

管理员审批标志为待审批的评价，判断需要对评价进行哪些处理

1. 若管理员判定评价合法，则审批通过，系统按照正常流程发布评价
2. 若判定该评价含有敏感词汇不宜发布，则审批不通过，系统将评价标志为待修改评价，并通知用户修改评价并重新发布
3. 水军检测

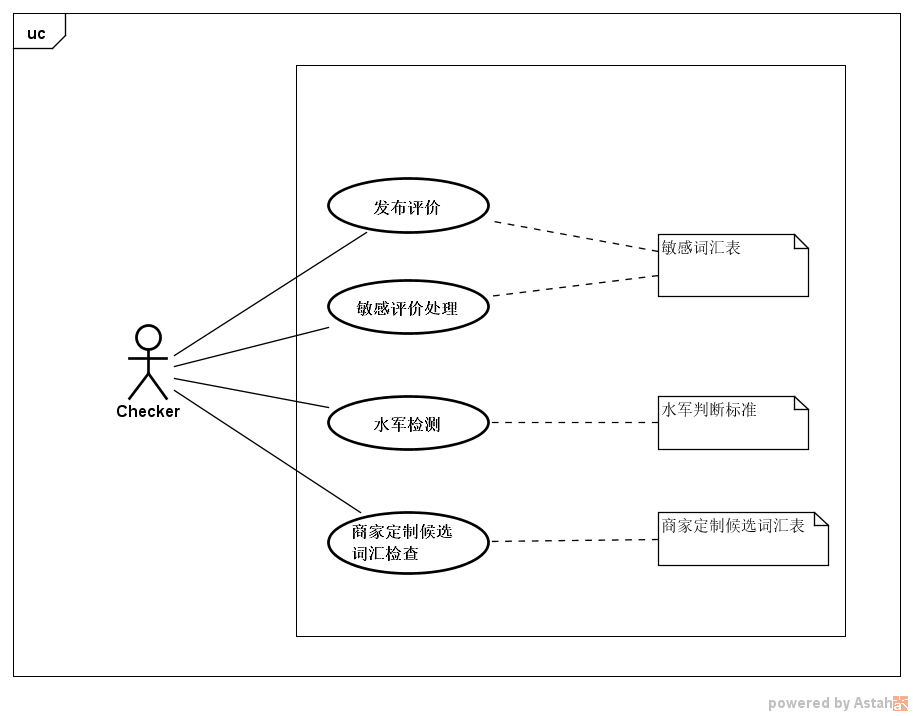
当评价被判定为敏感评价之后，系统自动进行水军检测以判断该用户是否为水军。

若判定为水军，则将该用户加入数据库的水军列表中，同时通知管理员

1. 商家定制候选词汇处理

当用户的评价通过所有检查并成功发布后，系统自动进行候选词汇检查，判断评价中是否含有候选词汇，若有，则记录该用户和评价，并通知相应的商家

**评价检查用例图**



**用例描述**

发布评价

当用户提交评价后，检查评价内容并发布合法评价，记录含敏感词汇的评价

敏感评价处理

管理员审批被标志为待审批的评价，决定评价是否真的含有敏感词汇

水军检测

系统依据水军判定标准，自动检查发布被管理员判定为非法评论的评价的用户，

记录并禁止疑似水军用户

商家定制候选词汇检查

评价正常发布后，系统检查该评价内容，若含有候选词汇，记录并通知相应商家

**可能会修改的实现（次要秘密）**

1. 水军判定标准的可扩展性
2. 敏感评价处理方式的可扩展性
3. 候选词汇、敏感词汇的存储格式和方式可能发生变化

**角色**

1. 检查用户评价，并发布通过检查的评价
2. 处理被判定为敏感评价的评价，包括评价的处理、用户水军身份的检测和用户相应处理
3. 检查评价是否含有商家候选词汇并进行后续处理

**向外提供接口**

|  |  |
| --- | --- |
| **接口名** | **职责** |
|  |  |
|  |  |
|  |  |
|  |  |

## 类图

## 类描述

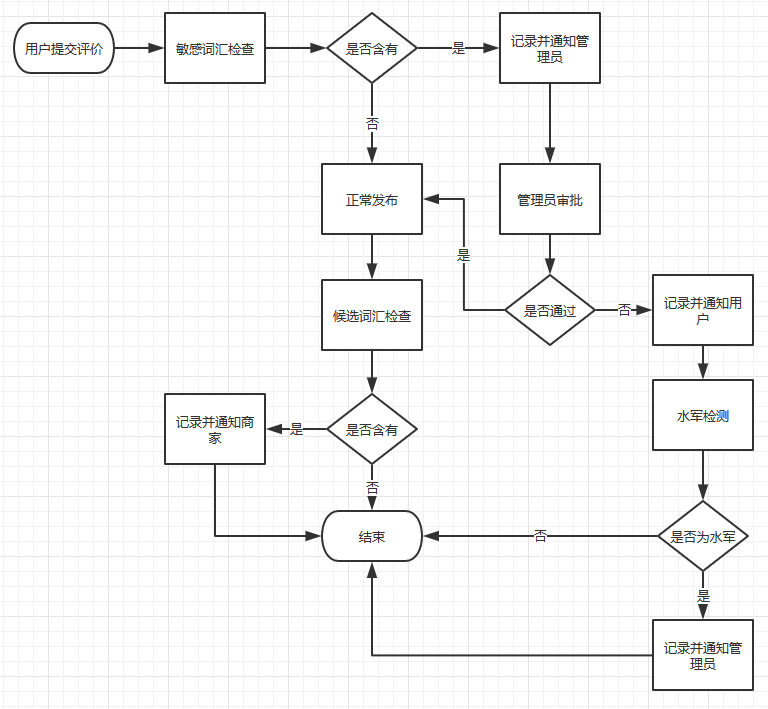
### PriceParity接口

|  |  |
| --- | --- |
| 方法名 | Public GoodsPriceList findNamesake(Goods goods) |
| 前置条件 |  |
| 后置条件 |  |
| 作用 |  |

### PriceParityImpl类

|  |  |
| --- | --- |
| 方法名 |  |
| 前置条件 |  |
| 后置条件 |  |
| 作用 |  |

## 重要协作



## 设计模式