(19)中华人民共和国国家知识产权局

(12)发明专利申请

(10)申请公布号 CN 103473105 A
(43)申请公布日 2013.12.25

(21)申请号 201310445229.1

(22)申请日 2013.09.25

(71)申请人 北京大学
地址 100871 北京市海淀区颐和园路 5 号

(72)发明人 吴凌 张化劲 杨楠 王千祥

(51)Int.Cl.
G06F 9/445(2006.01)

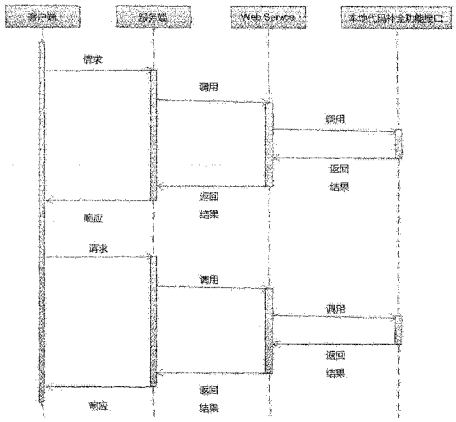
权利要求书1页 说明书5页 附图1页

(54)发明名称

一种在线代码补全功能的实现方法

(57)摘要

本发明公开了一种在线代码补全功能的实现方法。该方法包括步骤:1. 将本地集成开发环境的代码辅助功能,封装成可以被调用的本地接口;2. 实现代码辅助功能的 web service,该 web service 接受来自在线集成开发环境代码操作相关的数据,调用第一步得到的接口,获得本地集成开发环境代码辅助功能生成的结果,将该结果作为 web service 调用的结果;3. 实现在线集成开发环境客户端与服务器端的交互。在线集成开发环境客户端编辑区,记录用户在编辑代码过程中的数据,将该数据发送到服务器端,服务器端调用代码辅助功能web service,并将得到的结果返回给客户端显示。采用本发明的方法,可以为用户在在线集成开发环境中编写代码,提供类似本地集成开发环境的代码辅助功能。



CN 103473105 A

1. 一种在线代码补全功能的实现方法，为用户使用在线集成开发环境编程过程中，提供类似于本地集成开发环境的代码补全功能，其特征在于，包括如下步骤：

(1) 将本地集成开发环境的代码补全功能，封装成可以被调用的本地接口；

(2) 实现代码补全功能的 web service，该 web service 接受来自在线集成开发环境代码操作相关的数据，调用第一步得到的接口，获得本地集成开发环境代码补全功能生成的结果，将该结果作为 web service 调用的结果；

(3) 实现在线集成开发环境客户端与服务器端的交互。在线集成开发环境客户端编辑区，记录用户在编辑代码过程中的数据，将该数据发送到服务器端，服务器端调用代码补全功能 web service，并将得到的结果返回给客户端显示。

2. 如权利要求 1 所述的在线集成开发环境代码补全功能的实现方法，其特征在于，所述步骤 (1) 中，对于可以通过命令行方式进行调用的代码补全功能，接口的实现直接通过命令行调用语句实现。

3. 如权利要求 1 所述的在线集成开发环境代码补全功能的实现方法，其特征在于，所述步骤 (1) 中，对于无法通过命令行方式进行调用的代码补全功能，需要首先抽取出功能相应的代码，然后将该代码封装成接口。

4. 如权利要求 1 所述的在线集成开发环境代码补全功能的实现方法，其特征在于，所述步骤 (2) 中的 web service，既可以是符合标准规范的 web service，也可以是广义上的任何通过 http 方式访问的 web 服务。

一种在线代码补全功能的实现方法

技术领域：

[0001] 本发明涉及一种在线代码补全功能的实现方法，适用于为在线集成开发环境提供代码补全功能，以便使得用户在使用在线集成开发环境时，拥有和本地集成开发环境一样的代码补全功能，属于软件技术领域。

背景技术：

[0002] 随着云计算的提出，各种桌面程序逐步被迁移到云中。而桌面集成开发环境，作为一个与开发人员密切相关的程序，也开始被迁移到云中，我们称之为在线集成开发环境。在线集成开发环境有着很多特有的优势，如随时随地开发，无需搭建开发环境等。然而，本地集成开发环境经过数十年的发展已经拥有完备的功能体系。其中本地集成开发环境的代码补全功能，已经成为开发者在开发过程中不可或缺的功能，该功能可以大大提高开发者在开发过程中的效率。与本地集成开发环境相比，现有在线集成开发环境提供的功能比较简单，特别是未能提供开发过程中的代码补全功能，这严重影响了在线集成开发环境的发展。

发明内容：

[0003] 针对上述问题，本发明的目的是提供一种在线代码补全功能的实现方法。该方法能够为在线集成开发环境增加代码补全功能，从而为开发者在开发过程中带来便利。

[0004] 为解决上述技术问题，本发明的技术方案如下：

[0005] 一种在线代码补全功能的实现方法，为用户使用在线集成开发环境编程过程中，提供类似于本地集成开发环境的代码补全功能，其特征在于，包括如下步骤：

[0006] (1) 将本地集成开发环境的代码补全功能，封装成可以被调用的本地接口；

[0007] (2) 实现代码补全功能的web service，该web service接受来自在线集成开发环境代码操作相关的数据，调用第一步得到的接口，获得本地集成开发环境代码补全功能生成的结果，将该结果作为web service调用的结果；

[0008] (3) 实现在线集成开发环境客户端与服务器端的交互。在线集成开发环境客户端编辑区，记录用户在编辑代码过程中的数据，将该数据发送到服务器端，服务器端调用代码补全功能web service，并将得到的结果返回给客户端显示。

[0009] 2. 如权利要求1所述的在线集成开发环境代码补全功能的实现方法，其特征在于，所述步骤(1)中，对于可以通过命令行方式进行调用的代码补全功能，接口的实现直接通过命令行调用语句实现。

[0010] 3. 如权利要求1所述的在线集成开发环境代码补全功能的实现方法，其特征在于，所述步骤(1)中，对于无法通过命令行方式进行调用的代码补全功能，需要首先抽取出功能相应的代码，然后将该代码封装成接口。

[0011] 4. 如权利要求1所述的在线集成开发环境代码补全功能的实现方法，其特征在于，所述步骤(2)中的web service，既可以是符合标准规范的web service，也可以是广义上的任何通过http方式访问的web服务。

[0012] 本发明的技术效果在于：充分利用了现有本地集成开发环境的代码补全功能，避免重新开发代码补全功能，同时可以保证为在线集成开发环境提供稳定的代码补全功能，因为本地的集成开发环境的代码补全功能经过十多年的发展，功能趋于稳定，最后使得用户在享有在线开发的便利同时，也享有本地开发过程的高效。

附图说明

[0013] 图 1 表示在本发明中在线集成开发环境代码补全功能实现之后的效果图

[0014] 图 2 表示在本发明中在线集成开发环境代码补全功能的工作流程图

具体实施方式：

[0015] 附图 1 示意了在线集成开发环境代码补全功能实现之后的效果图。

[0016] 附图 2 示意了在线集成开发环境代码补全功能的工作交互图。

[0017] 整个在线集成开发环境代码补全功能的实现如下：

[0018] 首先，我们需要设计本地代码补全功能与 web service 交互的接口，该接口输入为需要补全的代码片段，输出为推荐的方法和属性列表。设计完接口之后，我们开始实现本地代码补全功能和 web service 交互的接口。当本地代码补全功能提供我们所需的子接口时，我们可以直接调用本地代码补全功能的一系列子接口，来实现与 web service 交互的接口。当无法直接调用本地代码补全功能的子接口时，我们需要抽取本地代码补全功能相关的子功能的实现代码，并且将这些代码整合到 web service 交互接口的实现中。通过该方式，我们可以充分复用本地代码补全功能已有子功能实现，获得我们所需要的与 web service 进行交互的一个完整功能的实现。

[0019] 然后，我们需要设计代码补全功能的 web service 部分。该部分的输入是需要补全的代码片段，输出为推荐的方法和属性列表。该部分是服务器端与本地代码补全功能之间交互的一个媒介。该部分的实现是，接受服务器端发送的需要补全的代码片段的请求，然后调用已经实现好的本地代码补全功能与 web service 进行交互的接口，计算出推荐的方法和属性列表，将该列表作为 web service 处理的结果，返回给服务器端。Web service 与服务器的交互可以是通过 Soap 协议交互，也可以通过 Http 协议交互。

[0020] 接着，我们需要设计代码补全功能的服务器端。服务器端负责接收客户端的请求，然后根据请求的不同类别，调用相应的业务逻辑功能，获得结果，并将结果返回给客户端显示。我们事先代码补全功能的服务器端时，加入代码补全功能的请求判断，如果符合请求的要求，我们将需要补全的代码片段发送给 web service，并将 web service 处理的结果返回给客户端显示。

[0021] 最后，我们需要设计代码补全功能的客户端。客户端的作用是收集用户在客户端的操作，当遇到用户的操作需要响应代码补全列表时，客户端收集各种用户编写的代码片段信息，然后向服务器端发送代码补全请求，服务器端处理完毕之后，将结果返回给客户端。客户端需要以列表框的形式，展现给用户，用户可以在列表框中进行选择，然后完成代码补全功能。

[0022] 其中的关键技术如下：

[0023] (1) 交互数据格式

[0024] 最常见的数据交互格式是字符串格式，但是由于在线集成开发环境客户端与服务器交互，服务器与 web service 交互，web service 与本地代码补全功能交互都需要传送代码等各种数据，不仅数据量大，而且类型复杂，所以需要采用 xml 或者 JSON 格式作为数据交互的格式。下面列举说明数据交互格式。

[0025] 以代码补全为例，我们想获得对象 a 中所有方法的列表，从主程序服务发给代码补全服务的 JSON 消息格式如下所示：

[0026]

```
{
mainClass : "  cn.edu.pku.TestCode ",
offset : 32,
length : 300,
command : "codeComplete"
}
```

[0027] 其中 mainClass 的值“cn. edu. pku. TestCode”是对象 a 所在 Java 文件的名称，length 的值 300 是整个 java 文件的代码长度。offset 的值 32 是 a 在整个 java 文件中的偏移量，最后 command 的值 codeComplete 表示，希望代码补全服务返回 a 的所有方法和属性。

[0028] 代码补全服务计算毕，将结果返回给主程序，主程序再以列表的形式显示出对象 a 的所有属性和方法。从代码补全服务返回的数据，也是以 JSON 格式传回主程序，JSON 消息中的内容如下：

[0029]

```
{
status : "OK",
ret : {
        number : 2,
        completion : [ {
                name : "method1",
                visibility : "public",
                type : "method"
        }, {
                name : "field1",
                visibility: "public",
                type : "field"
        } ]
}
}
```

[0030] 其中 status 值 ok 表示代码补全服务计算的结果正常。具体的返回的方法和属性的的值，由 ret 中的值提供。

[0031] 通过 JSON 格式，我们可以将足够多的信息在不同的层次上进行数据交互。(2) 本地代码补全功能提取

[0032] 由于本地集成开发环境功能稳定而且强大，如果重新开发一遍功能，代价十分巨大，所以我们可以将本地集成开发环境的功能提出出来，做成服务，以服务的形式，提供给在线集成开发环境使用。要想将本地代码补全功能做成服务，最重要的两个问题就是：(1)如何将本地代码补全功能从其他功能中分离出来。(2)如何将这些功能组织在一起，形成可以和 web service 进行交互的接口。

[0033] 对于这两个问题，分两种情况：如果本地代码补全功能每个子功能都是以接口的形式提供，我们就无需分离，直接可以调用这些子功能，按照一定得逻辑将这些子功能组织在一起，作为将来被 web service 调用的接口的实现。如果本地代码补全功能的子功能和其他无关的代码混杂在一起，我们需要分以下三步提取子功能，并且组合成供 web service 调用的接口的实现。

[0034] 1) 找到功能的入口点。所谓功能的入口点，就是指该功能实施的第一段代码。

[0035] 2) 根据代码依赖，逐步扩展代码。从入口点开始，根据代码之间的数据依赖关系，将属于该子功能的代码，都包含到该子功能的实现中。

[0036] 3) 将所有提取出的子功能组合。即按照一定得业务逻辑，将各个子功能组合在一起，从而实现一个完整的代码补全功能。

[0037] 下面展示的是一段提供给web service直接调用的本地代码补全功能的子接口的实现。

[0038] 这段代码的输入为 java 文件名 (javafile)，java 文件内容 (fileContent)，需要进行代码补全的位置偏移 (offset)，输出为推荐的代码补全相关的方法和属性列表。

[0039]

```
public JsonEntry codeComplete(String javafile, String fileContent, int
offset){
        workspace = ResourcesPlugin.getWorkspace();
        root = workspace.getRoot();
        project = root.getProject(projectName);
        if (project.exists()) {
            javaProject = JavaCore.create(project);
        } else {
            javaProject = this.createNewJavaProject();
        }
       IPath path_ = new Path("/src/" + javafile.replace('.', '/')
                        + ".java");
        IFile file = project.getFile(path_);
        cu = JavaCore.createCompilationUnitFrom(file);
        RET = new JsonEntry();
        String javaCode = fileContent;
        project.refreshLocal(IResource.DEPTH_INFINITE, null);
    MyCompletionRequestorAdapter my = new MyCompletionRequestorAdapter();
        cu.codeComplete(offset, my);
        jt = my.topTuple;
        RET.setAttribute("op", "complete");
        RET.setAttribute("value", jt);
       Return RET;
.....
}
```

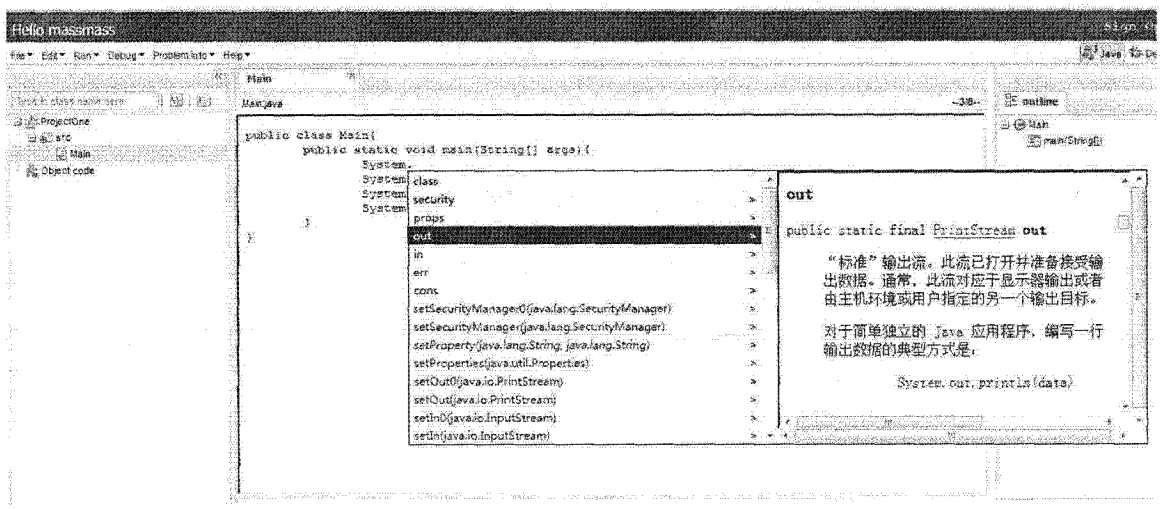

Hello massmass
Main.java
public class Main{
public static void main(String[] args){
System.
System.
System.
System.
}
}
class
security
props
out
in
err
cons
setSecurityManager0(java.lang.SecurityManager)
setSecurityManager(java.lang.SecurityManager)
setProperty(java.lang.String, java.lang.String)
setProperties(java.util.Properties)
setOut0(java.io.PrintStream)
setOut(java.io.PrintStream)
setIn0(java.io.InputStream)
setIn(java.io.InputStream)
out
public static final PrintStream out
“标准”输出流。此流已打开并准备接受输出数据。通常，此流对应于显示器输出或者由主机环境或用户指定的另一个输出目标。
对于简单独立的 Java 应用程序，编写一行输出数据的典型方式是：
System.out.println(data)

图 1

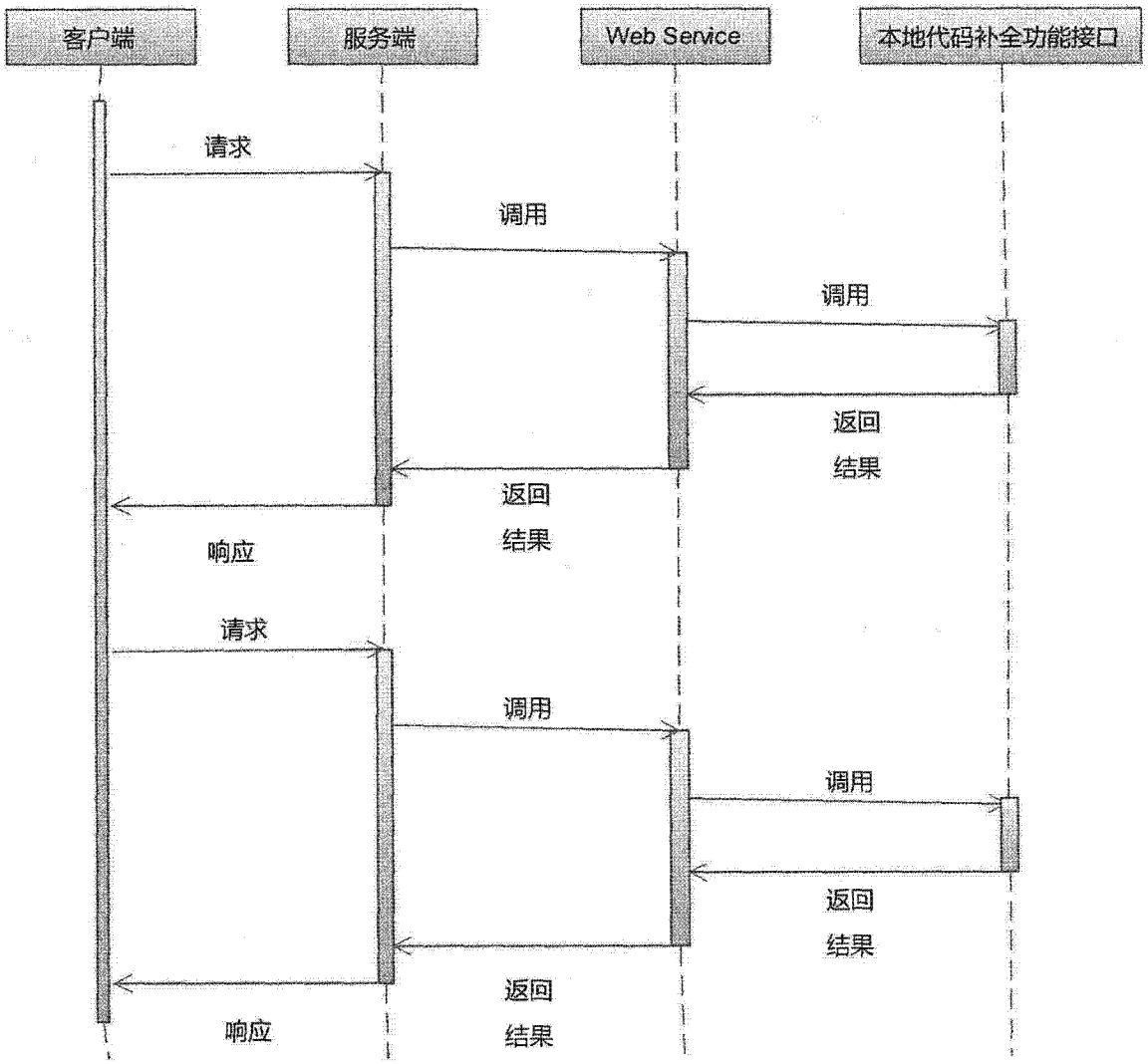

客户端
服务端
Web Service
本地代码补全功能接口
请求
调用
调用
返回
结果
返回
结果
响应
请求
调用
调用
返回
结果
返回
结果
响应

图 2