

一种 Web 应用程序切片方法

赵丽艳^{1,2}, 何志学²

(1 中国科学院 研究生院, 北京 100040; 2 北华航天工业学院 计算机科学与工程系, 河北 廊坊 065000)

摘 要: 针对 Java Web 开发方法中关键技术的实现机制, 定义了 Web 应用程序中存在的依赖关系, 提出了一种构造 Web 程序系统依赖图, 并采用两步遍历图可达性算法计算程序切片的方法. 把程序切片技术运用到 Web 应用开发中, 为其调试、测试和维护等软件过程提供了新的支持方法.

关键词: Web 应用; 程序切片; 依赖性分析; 系统依赖图

中图分类号: TP311.56

文献标识码: A

文章编号: 1000-7180(2010)02-0176-04

An Approach to Web Application Program Slicing

ZHAO Li-yan^{1,2}, HE Zhi-xue²

(1 Graduate School, Chinese Academy of Sciences, Beijing 100040, China; 2 Department of Computer Science & Engineering, North China Institute of Astronautic Engineering, Langfang 065000, China)

Abstract: The dependences between statements in the Java Web application program are defined, and the method of constructing the system dependence graph based on the analysis of dependence is introduced. The program slice can be computed by using the two-pass slicing algorithm on the system dependence graph. The program slicing technique can facilitate the debugging, testing and maintenance of the Web application program in the progress of development of the Web application software or system.

Key words: web application; program slicing; dependence analysis; system dependence graph

1 引言

Web 应用是一种以 Web 页面为界面发布信息、接受输入的特殊网络应用程序. 随着 Internet 技术和 WWW 技术的飞速发展, Web 应用涉及的领域越来越广. 程序切片是一种程序分析技术^[1], 通过提取程序中直接或间接影响某个特定程序点变量值的语句达到分解程序的目的. 把程序切片技术引入到 Web 应用程序的分析过程中, 可以使开发人员的关注点集中于仅与某些 Web 页面或某个功能实现有关的那些程序, 能提高其可靠性和正确性.

文中针对 Java Web 应用程序开发方法中关键技术的实现机制, 定义了客户端 JSP 程序、服务器端 Servlet 程序中语句间存在的一组依赖关系. 在依赖性分析的基础上构造 Web 应用程序的系统依赖图,

并根据所定义的依赖关系和给定的切片准则, 提出了计算程序切片的方法^[2-5].

2 Web 应用程序的依赖关系

Web 应用程序的执行原理如图 1 所示. 一个典型的 HTML 文件与请求的 JSP 页面交互过程如下: HTML 文件包含一个表单, 当用户在表单中输入数据后, 单击【提交】按钮时, 浏览器会把用户的数据和请求发送到该页请求的 JSP 页面, 服务器端的 Servlet 容器会在第一次调用这个 JSP 时自动编译它. 编译完成后, 就会调用这个 JSP, JSP 从服务器获得请求表单的数据, 根据这个数据执行 Scriptlet 并产生一些输出, 这些输出包含的就是 HTML 文件, 然后由 Servlet 容器把这些输出发送到客户端.

程序中语句间的依赖性分析是构造系统依赖图

收稿日期: 2009-01-06; 修回日期: 2009-03-08

基金项目: 江苏省高校自然科学基金项目(05KJB520119); 重庆市自然科学基金项目(CSTC2006BB2259)

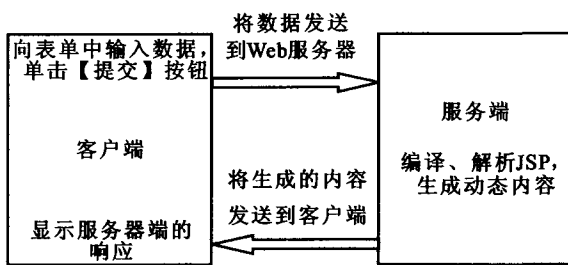


图1 客户端与服务器端交互

和计算程序切片的基础.从以上的执行过程可以看到,与一般程序相比,由于Web应用程序的交互性,使得它的依赖性分析更加复杂.在Web应用程序执行过程中引起的依赖关系,即包括了一般程序中的数据依赖、控制依赖,还包括了HTML标签与标签内部文档元素之间的控制依赖,表单元素与调用页面之间、页面与Servlet程序之间的调用依赖等.

2.1 数据依赖

数据依赖由变量的定义和引用引起,在Web应用程序中,存在两种情况的数据依赖形式,一是存在于JavaScript程序、Servlet程序的方法内语句间,另一种情况是存在于HTML程序与JSP页面内嵌的Java程序语句间.JSP程序中的脚本元素中的Java代码用来定义在产生的类文件中的类的属性和方法,这些声明后的变量和方法可以在JSP的任意地方使用.JSP为了简化页面的开发还提供了一些内部对象.这些内部对象不需要有JSP开发人员实例化,它们由容器实现和管理,在所有JSP页面中都能使用内部对象.在JSP的内建对象中,如request、session等可以对参数进行操作:request.getParameter()、session.getAttribute()、session.setAttribute()等.如果这些对参数的操作语句与定义对应实参的语句在同一应用程序内,则它们之间也存在数据依赖关系.与此类似的情况还有JSP中动作元素<jsp:setProperty>、<jsp:getProperty>等.

定义1 数据依赖:设 m 与 n 为给定程序中两条不同的语句, m 定义的变量集合为 $\text{def}(m)$, n 引用的变量集合为 $\text{ref}(n)$, n 数据依赖于 m ,记作 $DD(m, n)$,当且仅当:(1)存在由 m 到 n 的路径 $P:s_1 = m, s_2, \dots, s_n = n$,且存在一个变量 v ,满足 $v \in \text{def}(m) \cap \text{ref}(n)$;(2)对于路径 P 上的其他语句 $s' \in P - \{m, n\}$, $v \notin \text{def}(s')$.

定义1中的变量定义包含了各种形式对变量的赋值或初始化操作,变量引用包含了所有使用该变量的操作.在图2所示的程序中,语句20引用了语句17定义的变量count,且从语句17到语句20的

执行路径上count没有被重新定义,所以语句20数据依赖于语句17.

2.2 控制依赖

在一般程序中,控制依赖关系由条件语句或循环语句中的控制谓词引起.在Web应用程序中,对于JavaScript程序和Servlet程序,这种情况的依赖关系依然存在.同时,在页面HTML程序中,标记元素与其中的内容之间形成一种树形关系,以及JSP指令元素和动作元素与其属性之间形成一种属于关系,这些在本质上也是一种控制依赖关系.

定义2 控制依赖:(1)设 m 与 n 为给定程序中两条不同的语句,当且仅当:存在由 m 到 n 的路径 $P:s_1 = m, s_2, \dots, s_n = n$,对于路径 P 上的其他语句 $s' \in P - \{m, n\}$, n 是 s' 的后必经节点,即由 s' 到程序终止语句的路径必经过 n ,且 n 不是 m 的后必经节点, n 控制依赖于 m ;(2)设 m 为HTML程序中标记, n 为标签之间的部分(不包含标签本身),则 n 控制依赖于 m ;(3)设 m 为JSP指令或动作元素, n 为其属性,则 n 控制依赖于 m .

语句 n 控制依赖于语句 m ,记作 $CD(m, n)$.在图2中,语句7中含有判断谓词,则语句9、10控制依赖于语句7;语句20、21为语句19<form>标签中的内容,则语句20、21控制依赖于语句19.

```

1<html>
2  <head>
3    <title>例子程序</title>
4    <script type="text/javascript" language="javascript">
5      function valide()
6      {
7        if (document, forms[0].elements("data").value == "
7")
8        {
9          alert("保持缺省值吗?");
10         return true;
11        }
12      }
13  </script>
14 </head>
15 <body>
16   <%
17     int count=7;
18   %>
19   <form action="goser" method="post">
20     <input type="text" name="var" value="<%=count%>"
21     <input type="submit" value="提交" onclick="return
valide()">
22   </form>
23 </body>
24</html>
  
```

图2 客户端JSP页面

2.3 调用依赖

以上定义的数据依赖和控制依赖表示了一个方

法内部语句间的依赖关系,在实际应用程序中,总是多个方法间的调用,这时用调用依赖和参数依赖(输入和输出)表示调用关系.对于 Web 应用程序,不仅存在同一开发语言程序中方法之间的调用,还存在 HTML 程序与 JavaScript 程序中方法之间的调用、页面之间的调用、客户端程序对服务器端程序中方法的调用等.由于 Web 应用程序的复杂性和其固有的交互性,程序中形成的调用依赖关系比较复杂,这里仅考虑以下几种开发中常见的情况:

(1) HTML 中锚点元素 `<a href = "URL">` 产生超链接调用,其中的 href 属性指定了超链接文档的统一资源定位符 URL,如果用户单击与锚点对应的超链接,就会通过 HTTP GET 方法请求它.当浏览器在对这个请求的 HTTP 响应中接收到一个文档时,它默认会显示这个新文档,来代替包含超链接的文档,则原页面与被调用页面之间存在调用依赖关系;

(2) HTML 程序的表单(form)允许用户在 Web 页面上输入数据.其中必须的 action 属性的值指定了一个 URL,当用户提交表单时,把表单上收集到的信息发送到这个 URL,则表单所在页面与被调用 URL 之间存在调用依赖关系;

(3) JSP 指令元素 include 通知容器将当前 JSP 页面中的内嵌的、在指定位置上的资源内容包含. include 指令如下:

```
<%@ include file = "filename"%>
```

其中,filename 为要包含的文件名. include 指令将在 JSP 编译时插入一个包含文本或代码的文件,则 include 指令所在页面与被包含文件之间存在调用依赖关系;

(4) JSP 动作元素 `<jsp:include>` 操作允许在请求时间内在现成的 JSP 页面里包含静态或者动态资源文件,则同 include 指令形成的关系类似, `<jsp:include>` 指令所在页面与被包含文件之间形成调用依赖关系;

(5) JSP 动作元素 `<jsp:forward>` 操作允许将请求转发到另一个 JSP、Servlet 或者静态资源文件,则该指令所在页面与被请求对象之间形成调用依赖关系;

(6) 开发人员在编写 Servlet 程序时,一般直接继承 HttpServlet,HttpServlet 封装了编写基于 Http 协议 Servlet 的大部分功能.在一个 Servlet 实现类中有两个方法:doGet() 处理 Http Get 请求、doPost() 处理 Http Post 请求.这两个方法可以把相应请

求进行转发,此时,该方法与转发对象之间形成调用依赖关系.

Web 应用程序中存在多种形式的交互形式,形成的其他情况下的调用依赖关系与以上几种情况类似,不再列出.在图 2 所示的程序中,语句 21 调用了 JavaScript 程序中的 valide() 方法,则语句 21 与语句 4 之间存在调用依赖边;表单的 action 属性调用 goser 对应的 Servlet 类对象,则语句 21 与 doPost() 方法所在的语句 2 存在调用依赖关系.

3 构造系统依赖图

通过分析语句间的数据依赖和控制依赖关系^[6],可以把这些依赖关系表示为数据依赖图、控制依赖图.在程序控制流图的基础上,组合数据依赖图和控制依赖图,把单个方法的程序表示为程序依赖图,然后通过添加调用依赖边和与之相随的参数依赖边,由程序依赖图组成系统依赖图.

定义 3 控制流图:程序 P 的控制流图是一个有向图,可以用四元组 $\langle S, E, S_{\text{entry}}, S_{\text{exit}} \rangle$ 表示,其中 S 为图中结点的集合,程序中的每条语句都对应于控制流图上的一个结点; E 为图中的边集, $E = \{ \langle s_1, s_2 \rangle \mid s_1, s_2 \in S, \text{且语句 } s_1 \text{ 执行之后, } s_2 \text{ 可能成为下一条执行语句} \}$; S_{entry} 表示程序的入口结点; S_{exit} 表示程序的出口结点.

定义 4 数据依赖图:数据依赖图 DDG 是一个有向图 (S, D) ,其中结点集 S 是程序中所有语句对应的结点集合,如果有语句 s_2 数据依赖于语句 s_1 ,也就是有 $DD(s_1, s_2)$,则把边 $s_1 \rightarrow s_2$ 加入到边集 D 中.

定义 5 控制依赖图:控制依赖图 CDG 是一个有向图 (S, C) ,其中结点集 S 是程序中所有语句对应的结点集合;如果有语句 s_2 直接控制依赖于语句 s_1 ,也就是有 $CD(s_1, s_2)$,则把边 $s_1 \rightarrow s_2$ 加入到边集 C 中.

定义 6 程序依赖图:程序依赖图 PDG 是一个二元组 (S, E) , S 是程序中所有语句的结点集合,边集 $E = E_1 \cup E_2$,其中: $E_1 = \{ \langle s_1, s_2 \rangle \mid DD(s_1, s_2) \}$ 为数据依赖边集; $E_2 = \{ \langle s_1, s_2 \rangle \mid CD(s_1, s_2) \}$ 为控制依赖边集.

Web 应用程序的系统依赖图由用于页面表示的 JSP 程序和 Servlet 程序的程序依赖图通过调用边连接组成.调用请求对应的 Servlet 程序,如图 3 所示.整个应用程序对应的系统依赖图如图 4 所示.

```

1 public class ServletExample extends HttpServlet {
2     protected void doPost(HttpServletRequest request,
3                           HttpServletResponse response)
4     {
5         //获取 example 页面输入值
6         string value = request.getParameter("var");
7         System.out.println("the value is:" + value);
8         request.getRequestDispatcher("/exam.jsp")
9         .forward(request, response);
10    }
11 }

```

图3 服务器端 Servlet 程序

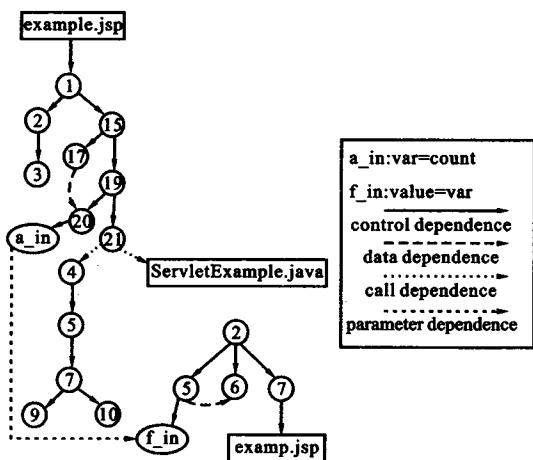


图4 系统依赖图

4 计算程序切片

基于系统依赖图可以采用过程间两步遍历图可达性算法^[7]计算给定切片准则 $\langle s, v \rangle$ 对应的程序切片:

(1) 从切片准则中 s 对应的结点开始, 沿着数据依赖边、控制依赖边、调用边和参数输入边(不沿参数输出边)反向遍历, 并标记所有到达的结点和边;

(2) 从(1)中到达的所有结点开始沿数据依赖边、控制依赖边和参数输出边(不沿参数输入边和调用边)反向遍历, 标记到达的结点和边。

最后的切片就是两步遍历过程中所有标记的结点代表的语句的集合。

对于图4中例子程序, 给定切片准则为 Servlet

程序中 $\langle 6, \text{value} \rangle$, 则计算切片结果为 example.jsp 中语句集合 $\{1, 15, 17, 19, 20, 21\}$ 与 ServletExample.java 中语句集合 $\{2, 5, 6\}$ 的并集^[8]。

5 结束语

文中主要讨论了 Java Web 应用程序开发中的典型的操作和实现方法。如何分析更加全面的程序以及在客户端与服务器端起到连接作用的 xml 文件; 如何定义语句间更为精确的依赖关系是下一步工作的重点。

参考文献:

- [1] Weiser M. Program slicing[J]. IEEE Transaction on Software Engineering, 1984, 10(4):352-357.
- [2] 肖建宇, 张德运, 陈海论, 等. 一种改进的用于并发程序静态切片的程序依赖图[J]. 微电子学与计算机, 2006, 23(1):6-10.
- [3] 周捷, 慕晓东, 王杰. 一种基于程序切片算法的软件故障诊断策略[J]. 微电子学与计算机, 2006, 23(8):110-116.
- [4] Ricca F, Tonella P. Construction of the system dependence graph for web application slicing[C]// Proceedings of the 2th IEEE International Workshop on Source Code Analysis and Manipulation. Italy, Trento, 2002:123-132.
- [5] Tonella P, Ricca F. Web application slicing in presence of dynamic code generation[J]. Automated Software Engineering, 2005, 12(2):259-288.
- [6] Xu L, Xu B, Chen Z, et al. Regression test for web applications based on slicing[C]// Proceedings of the 27th Annual International Computer Software Applications Conference. China: Nanjing, 2003:652-656.
- [7] Wu J, Xu B, Jiang J. Slicing web application based on hyper graph[C]// Proceedings of the 3th International Conference on Cyberworlds. Japan: Tokyo, 2004:177-181.
- [8] He Z, Zhang G, Rong M. Dynamic slicing concurrent Java programs using variable cache table[C]// Proceedings of the 2nd International Conference on Computer Science & Education. China: Beijing, 2007:363-369.

作者简介:

赵丽艳 女, (1979-), 助教. 研究方向为软件工程、工作流。
何志学 男, (1982-), 硕士, 助教. 研究方向为软件工程、程序分析与验证技术。