工具集成的柔性接口机制

张 彪,陈 平,胡圣明
(西安电子科技大学 软件工程研究所,陕西 西安 710071)

摘 要:工具集成中的一个很重要的问题是如何能够灵活方便地将工具集成到一起。文中阐述了一种方法,利用 Java 反射机制,将接口描述在接口描述文件中,无需硬编码调用接口的代码,增强了动态扩展的能力,当接口变化时仅需改变接口描述文件的映射关系,而程序代码无需做任何改变,提高了集成的灵活性和扩展性,达到了接口柔性化的目的。

关键词:集成;反射;XML;序列化
中图分类号:TP311 **文献标识码**:A **文章编号**:1673-629X(2007)09-0082-03

Flexible Interface Mechanism of Tools Integration

ZHANG Biao, CHEN Ping, HU Sheng-ming
(Software Engineering Institute, Xidian University, Xi'an 710071, China)

Abstract: Effective integration of tools becomes more and more important. Discussed a method to make interfaces more flexible. By writing the declarations of functions into files and using Java reflection mechanism, functions can be invoked without writing any hard codes. If function interfaces changed, only change their declarations in files and invoking codes do not need any changes. So the extensibility and flexibility of integration is enhanced in this way.

Key words: integration; reflection; XML; serialization

0 引 言

大多数企业内部往往要实现信息共享,将各应用程序配合完成目标。但由于这些系统大多是独立开发而未考虑与其他系统的互操作性,几乎没有提供方法来访问其内部数据和商务流程,因此将这些系统与企业内其他系统进行集成便成为关键所在。为此提出了一种可方便进行工具集成的接口柔性化的方法。解释引擎动态加载并执行描述在接口描述文件中的 API 而无需硬编码,增强了灵活性和扩展性。

1 目前已有的解决方案

面向服务的架构(SOA)是一个组件模型,它将应用程序的不同功能单元——服务(Services)通过服务间定义明确的接口和契约联系起来。接口采用中立的方式定义,独立于具体实现服务的硬件平台、操作系统和编程语言,使得在这样的系统中所构建的服务可使用统一和标准的方式进行通信。目前 CORBA 和 Web Services 都比较接近 SOA 的目标。

1.1 CORBA

CORBA 是由 OMG 制定的著名的分布对象计算规范,是目前业界支持最多的分布计算标准,主要解决异构平台上分布对象互操作和可移植问题,其核心机制是对象请求代理(Object Request Broker, ORB),ORB 提供的软件总线机制包括了确定和定位对象、进行连接管理和收发数据所有必需的通信设施。客户对象只要获得了服务对象的引用,就可以透明地发送请求和接收响应,就可以调用对象提供的服务,无需关心服务的位置、实现语言和运行平台。

1.2 Web Services

Web Services[1]由 W3C 机构制定的一系列标准组成,用来促进跨平台的程序间通信,力求应用获得最佳的开放互操作性。它主要基于 HTTP 等网络传输协议,采用结构化的信息建模语言 XML,定义了信息交换协议 SOAP、服务描述语言 WSDL、服务发布注册机制。

Web Services 为 Internet 带来了灵活的、基于开放标准的分布式计算,它的目的是使应用程序能够在网络上进行无缝集成,而不必考虑它的编程语言和运行环境。这样可使得 Web 服务的提供者和 Web 服务的使用者解耦,增强了灵活性和配置性。

收稿日期:2006-11-13
基金项目:研究生创新基金(05009)
作者简介:张 彪(1981-),男,内蒙古锡盟人,硕士研究生,从事面向对象技术与系统设计工具集成研究;陈 平,博士生导师,从事面向对象技术的研究。

2　利用 XML 扩展集成的灵活性

在使用一组工具时通常需要将不同工具的部分功能进行特定方式的组合以提供新的功能。目前有很多框架规范的实现可以做到这一点,如 CORBA 和 Web Services 都提供了接口和实现的分离,但它们规模较大,因此文中参照 Web Services 架构提出了一套简洁适用的方案,给定一个配置文件描述各个功能调用的信息(如方法名、参数列表等信息)和流程(参照 BPEL 语言)[2],然后程序读入这个文件并按流程执行,达到功能重组的目的。如果接口发生变化,则仅需改变接口描述文件,调用程序代码不需作任何改变,降低了调用者和被调用者的耦合关系。因为方法的签名(signature)存放在文件中,所以从此文件读入方法名和参数列表后,需要根据方法名和参数列表调用对应的方法。文中提出一种思路,通过 API 的描述和相应的 Class 文件,利用 Java 的反射机制[3]实现接口的动态调用。主要使用到以下两个类的方法:Class 和 Method。

Class 中的 forName(String)方法:给定一个类的名字,返回此类的 class 属性;

Class 中的 getMethod(String methodName, Class[] parameterTypes)方法,返回此类的名为 methodName 的方法,返回值类型为 Method;

Method 中的 invoke(Object obj, Object[] args)方法:调用此 Method 所代表的方法,obj 是调用此方法的对象,args 是实参列表。

XML[4,5]是一种具有可扩展性和灵活性的语言,允许使用者创建和使用自己的标记,并且提供了一种结构化的数据表示方式,使得用户界面分离于结构化数据。XML 是被设计用来存储数据、携带数据和交换数据的,由于其结构跟对象的结构类似,这里采用 XML 文件来表示和存放对象。

2.1　程序中需要使用的各种文件

描述方法的 FunctionDescription 文件:此文件包含所需调用方法的所有信息的描述,包括方法所属的类名、方法名、方法的形式参数列表(每个形参包括类型名和变量名)和返回值的类型。如图 1 所示。

```
<?xml version="1.0" encoding="UTF-8" ?>
- <descriptions xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  - <description methodName="printStudentInfo">
      <class className="StudentManager" />
      <returnType returnTypeName="void" />
    - <parameterList>
        <parameter paraName="s" paraTypeName="Student" />
      </parameterList>
    </description>
  </descriptions>
```

图 1　printStudentInfo 方法的 FunctionDescription 文件

作为实参数据的 InputData 文件:此文件包含调用方法时的输入数据,即一组实参,对应于 FunctionDescription 文件中所描述方法的形参列表,供调用时使用,所以它依赖于 FunctionDescription 所描述的形参列表。如图 2 所示。

```
<?xml version="1.0" encoding="UTF-8" ?>
- <inputcatalist xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  - <inputdata className="dynamicInvoker.StudentManager" methodName="p
    - <parameters>
      - <s>
          <sNo>0421121294</sNo>
        - <name>
            <firstName>zhang</firstName>
            <lastName>biao</lastName>
          </name>
```

图 2　一个 Student 对象作为实参

描述类的结构的 ClassDescription 文件:由于在 FunctionDescription 文件中描述的方法包含的形参列表中形参可以是复杂类型,所以 ClassDescription 文件需要包含有对这些复杂类型的描述(描述复杂类型的各个字段)。在对象序列化存储时,需要用到复杂类型的各字段名字,所以需要加以描述供查询使用。图 3 描述了 Student 类的结构。

```
<?xml version="1.0" encoding="UTF-8" ?>
- <classDescriptions xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  - <classDescription className="Student">
    - <fieldList>
        <field fieldName="sNo" fieldType="String" />
        <field fieldName="name" fieldType="String" />
      </fieldList>
    </classDescription>
  </classDescriptions>
```

图 3　Student 类的结构

描述方法流程的 FunctionFlow 文件:FunctionFlow 文件中描述了各方法调用的先后顺序以及参数传递的关系。图 4 是一个 FunctionFlow 文件。

```
<?xml version="1.0" encoding="UTF-8" ?>
- <functionCalls>
    <functionCall functionDescripFile="functionofStudentManager.xml" inputDataFile=
      classDescripFile="ClassDescription.xml" useLastResult="false" />
    <functionCall functionDescripFile="function1.xml" inputDataFile="" classDescripFile=
    <functionCall functionDescripFile="function2.xml" inputDataFile="" classDescripFile=
  </functionCalls>
```

图 4　一个 FunctionFlow 文件

每个 XML 文件都有相应的 XML schema 文件描述其格式,可用来规范 XML 文件的格式。例如图 5 是描述 FunctionDescription 文件格式的 XML schema 文件。

2.2　对象的序列化存储与恢复

由于各模块之间需要通过文件传递数据,这些数据通常以对象的形式存在,所以如何将对象存储到文件中以及如何把存在文件中的对象恢复到内存中显得很重要。对象的存储是将一对象的名字,各字段的名字和值储存到文件中。这里遇到的困难是需要判断一个对象是否是为组合类型对象,如果是,要把组合它的各部分都序列化,如果是简单类型对象

直接把名字和值写入文件。

```
<?xml version="1.0" encoding="UTF-8" ?>
- <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault=
    "qualified" attributeFormDefault="unqualified">
  - <xs:element name="descriptions">
    - <xs:complexType>
      - <xs:sequence>
          <xs:element ref="description" minOccurs="0" maxOccurs="unbounded" />
        </xs:sequence>
      </xs:complexType>
    </xs:element>
  - <xs:element name="description">
    - <xs:complexType>
      - <xs:sequence>
          <xs:element ref="class" />
          <xs:element ref="returnType" minOccurs="0" />
          <xs:element ref="parameterList" minOccurs="0" />
        </xs:sequence>
        <xs:attribute name="methodName" type="xs:string" use="required" />
      </xs:complexType>
    </xs:element>
  - <xs:element name="class">
    - <xs:complexType>
        <xs:attribute name="className" type="xs:string" use="required" />
      </xs:complexType>
    </xs:element>
  - <xs:element name="returnType">
    - <xs:complexType>
        <xs:attribute name="returnTypeName" type="xs:string" use="required" />
      </xs:complexType>
    </xs:element>
  - <xs:element name="parameterList">
    - <xs:complexType>
      - <xs:sequence>
          <xs:element ref="parameter" minOccurs="0" maxOccurs="unbounded" />
        </xs:sequence>
      </xs:complexType>
    </xs:element>
  - <xs:element name="parameter">
    - <xs:complexType>
        <xs:attribute name="paraTypeName" type="xs:string" use="required" />
        <xs:attribute name="paraName" type="xs:string" use="required" />
      </xs:complexType>
    </xs:element>
  </xs:schema>
```

图 5 描述 FunctionDescription 文件格式的 XML schema 文件

这一步的主要做法如下：

0. 得到此对象的各子字段，设字段数为 n；

1. 置 i 等于 0；

2. 如果 i 大于等于 n，则结束；

3. 检查第 i 字段是否为简单类型，如果是，则输出此字段的名字和值；否则对此字段递归使用本算法；

4. i 自增 1，返回第 2 步。

对象的恢复是对象存储的逆过程，读入存储在文件中的对象各字段的值，并将其恢复成一个具有各字段值的对象。对象恢复的困难是如何动态建立一个反映当前文件所存数据结构的对象，经研究后，笔者决定采用广义表作为存储结构（即图 6 中的 SerializedObject 类），把存储在文件中的数据读入到一个 SerializedObject 对象中，然后再根据 SerializedObject 对象建立起 Java 对象的内存映像。建立工作主要是依据 Java 的反射机制，首先根据对象的名字查询出它的类名（在 FunctionDescription 文件中可以查到），然后用类加载器加载此类，建立此类的对象，并从文件中读入各字段的值，设置此对象相应的各字段（遇到私有字段可设置访问权限为 true，此算法与存储对象的算法类似），这样就得到了具有指定状态的对象。

2.3 程序流程的主要步骤

0. 置 i 为 0；

1. 如果不存在第 i 个 FunctionCall，程序结束；否则从 FunctionFlow 文件中找到它，并记录下这个 FunctionCall 所使用的 FunctionDescription 文件、InputData 文件、ClassDescription 文件的位置，供后面读取使用；

2. 从 FunctionDescription 文件中读出类名，方法名；

3. 如果此次调用有 InputData 文件，则从 InputData 文件读出实参列表（包括各个参数名和参数的值），如果 useLastResult 为 true，则把上一次调用返回结果的文件也读出实参列表；

4. 从 FunctionDescription 文件中读出各参数的类型，根据实参列表建立 object 数组并调用方法，其中使用 InputData 文件中参数的值来构造对象（如参数中有复杂类型，则需从 ClassDescription 文件中读取其各字段信息并从 InputData 文件中读取各字段的值，构造起各字段，从而构造起整个对象）；

5. 从 FunctionDescription 中读出返回值类型，将方法的返回值做成 XML 文件，可供后面的调用使用；

6. i 自增 1，返回第 1 步。

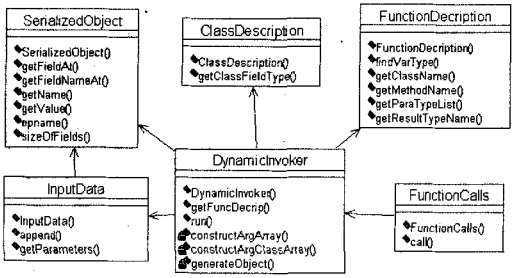

图 6 程序中的主要类

3 结束语

在笔者的工作中涉及到如何把描述在 XML 中的以数据形式存在的对象转换成真正的 Java 对象，以及把 Java 对象存储到 XML 文件中，这是比较复杂的地方。文中所阐述的方法尚存在一些不完善之处，只是

（下转第 91 页）

算法中产生的错位畸变。

对比图像 5 和图像 7 可知，汉字“粤”的孤立点畸变得到了纠正，即文中提出的改进算法纠正了基于图像像素的八邻域分析的孤立点畸变。

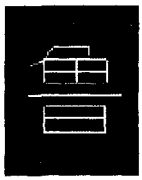

图 6　改进算法细化图像

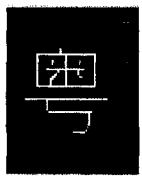

图 7　改进算法细化图像

5　结　论

由于传统的汉字字符图像细化主要是基于图像像素八邻域分析的算法，而没有考虑具体的汉字字符图像的骨架特征的经常出现笔划交叉的实际情况，这样就存在一定程度的细化误差。经过改进的细化算法，充分考虑到了汉字字符图像笔划经常的实际情况，对传统的算法进行了简单而有效的修正，该算法已经成功的应用到了车牌汉字字符识别系统的预处理环节上。综上所述，文中提出的改进算法，较好地纠正了传统的基于图像像素的八邻域分析的细化算法中产生的畸变，方法简单，容易实现，对汉字识别系统的图像预处理有一定的可借鉴性。

参考文献：

[1] 刘佐濂，邓荣标，孔嘉圆. 一种车牌识别算法的实现[J]. 中国科技信息，2005，23(2)：56－57.

[2] 刘桂雄，申柏华，冯云庆. 基于笔划趋势分析的二值图像细化方法[J]. 光学精密工程，2003，11(5)：527－530.

[3] 史绍强，王英健，唐贤瑛. 基于整形特征和模糊识别的手写体汉字识别[J]. 微机发展，2004，14(1)：114－116.

[4] 缪绍纲. 数字图像处理[M]. 成都：西南交通大学出版社，2001.

[5] 王晓丹，吴崇明. 基于 MATLAB 的系统分析与设计——图像处理[M]. 西安：西安电子科技大学出版社，2000.

(上接第 84 页)

简单的方法顺序调用，因此只能在运行前就安排好各调用的先后顺序，没有做到在执行过程中动态控制流程的转移。形参列表中的类型有些还不能处理，如数组类型、基类和派生类的接口问题，将来可考虑使用 JAXB 这样的软件做此工作。

参考文献：

[1] 柴晓路，梁宇奇. Web Services 技术、架构及应用[M]. 北京：电子工业出版社，2003.

[2] Andrews T，Curbera F. Business Process Execution Language for Web Services Version 1.1[EB/OL]. 2002－07－30. http://www－128.ibm.com/developerworks/library/specification/ws－bpel/.

[3] Halloway S d. Java 平台组件开发[M]. 北京：清华大学出版社，2004.

[4] W3C Recommendation. Extensible Markup Language(XML) 1.0 (Fourth Edition)[EB/OL]. 2006. http://www.w3.org/TR/REC－xml/#sec－intro.

[5] Skonnard A，Gudgin M. XML 精要快速参考手册[M]. 北京：人民邮电出版社，2002.

(上接第 87 页)

环境下用 Microsoft Visual C++ 6.0 实现。在改进的点方向图计算的基础上，采用合成分割法对指纹图像进行分割，从图 3 可以看到，该分割算法对噪声干扰具有较强的鲁棒性。从实验结果(见图 5)可以看到，采用文中所述的基于方向图的指纹图像预处理方法，可以有效地去除噪声干扰，并且能够保持纹线方向和纹线特征基本不变，为准确提取指纹特征提供了保证。

参考文献：

[1] Lam L，Lee S W，Suen C Y. Thinning Methodologies－comprehansive Survey[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence，1992，14(9)：869－885.

[2] 黄贤武，苏鹏程，柏培权. 基于方向滤波分割的指纹自动识别系统算法[J]. 中国图像图形学报，2002，7(8)：829－834.

[3] 韩伟红，黄子中，王志英. 指纹自动识别系统中的预处理技术[J]. 计算机研究与发展，1997，34(12)：914－920.

[4] 冯星奎，颜祖泉，肖兴明，等. 指纹图像合成分割法[J]. 计算机应用研究，2000(1)：76－77.

[5] Gorman L O，Nickerson F V. An Approach to Fingerprint Filter Design[J]. Pattern Recognition，1989，1(4)：381－385.

[6] Jain A K，Hong L，Bolle R. On－line fingerprint verification [J]. IEEE Transactions on Pattern Analysis and Machine Intelligence，1997，19(4)：302－314.

[7] 楚亚蕴，詹小四，孙兆才，等. 一种结合方向信息的指纹图像二值化算法[J]. 中国图像图形学报，2006，11(6)：855－860.

[8] 冯星奎，李林艳，颜祖泉. 一种新的指纹图像细化算法[J]. 中国图像图形学报，1999，4(10)：835－838.