基于模版的客户端申报方式的实现

郎海鹏　（湖南大学）
孙　芙　（中国科学院软件研究所 100080）

摘要：电子政务中申报数据的采集方式多种多样，本文介绍一种可支持断点续传的客户端软件，它和服务器端之间传输的申报数据文件，是基于一种用户预先定义好的模版格式。当申报业务的数据项发生变化时，通过对数据库和模版的修改，即可实现系统的升级，而无需对服务器端和客户端软件进行任何改动。

关键词：电子政务 xml 模版 断点续传 数据申报 客户端软件

1 引言

随着全球经济一体化的日益明显，以电子政务为代表的政府管理服务职能的电子化、自动化、无纸化目前正在一些国家尤其是发达国家中快速发展。可以说政府信息化是社会信息化的先导，电子政务是信息化社会发展的必然。在我国，信息化建设也成为当前产业结构升级和实现工业化、现代化的关键环节，为了完成用信息技术改造和提升传统产业的任务，政府部门的信息化管理水平必须有更大的飞跃[1]。

1.1 申报数据的一般采集方式

政府部门各种网上审批职能的实现，是电子政务中不可或缺的重要组成部分。网上审批解决方案中，对于申报数据的采集方式，也是百家争鸣，各有千秋。总的说来，从实现方法分类，大致有以下两种：一是通过网页提供数据格式，由用户登陆 Internet 网，在线填写申报资料；二是通过客户端的分发，用户在客户端进行数据输入，客户端软件通过 Internet 网采用某种网络协议将数据传输到服务器上。

第一种方式的优点是：表现格式灵活多变，可以根据需要通过改变服务器端程序修改要采集的数据项。但这种方式也有不足之处：不支持断点续传，当数据量很大时，如果由于线路的原因发生中断则需要重新传送，造成传送效率降低，同时容易形成垃圾数据；不能离线填写，当用户接入 Internet 网资源有限时，这种缺点最为突出。

第二种方式没有第一种方式的缺点，可以实现断点续传，当网络出现故障时，已上传的数据不会丢失。客户端软件可以采用离线方式填写申报数据，填写完毕后再接入 Internet 网上报数据。

1.2 改进方式

在信息产业部通信行业规划建设管理信息系统中，外网用户需要上传的数据量很大，因此需要支持断点续传，另外为了保密，不希望通过浏览器来上传重要数据，这种情况下非常适合采用客户端软件来实现资料上报。但根据审批的业务需要，上传的数据项可能会发生变化，传统的客户端软件无法满足频繁变动的需求，因此需要对客户端软件的实现方式加以改进，来适应实际项目的需求。

在系统设计中，我们采用模版格式对申报数据进行搜集，系统管理员可以灵活编辑变更模版格式，而外网用户可以通过客户端在线更新来获得最新的模版格式。客户端软件和服务器端之间传输的申报数据，都基于这种模版格式。当申报业务的数据项发生变动时，只需对数据库表或模版进行修改即可。

2 模版的设计

XML 技术是互联网国际标准组织 W3C 提出的表示数据语义信息的标准。它允许定义数量不限的标记来描述文档中的资料，允许嵌套的信息结构。

XML 带有一个 XML 语法分析器。XML 语法分析器使用 DTD 来确定一个文件是否规范化，即它应该包含正确定义的开始和结束标记。如果它是有效的，那么就说明完全遵守 DTD 规范。因此，XML 定义了更严格的数据结构。这样用户可以很容易将文件属性映射到数据结构或对象分级结构中，这就使客户端程序和服务器之间来回传输数据变得很可靠，也使用户可以使用结构化的 XML 文件作为一种中介体让数据在两种数据库之间灵活地进行转移[2]。因此选用 XML 文件作为模版的表现形式是一个比较好的解决办法。

2.1 模版格式

模版格式是指由服务器端提供的供用户下载并能由客户端软件所识别解析的一种 XML 文件格式，它规定了申报数据的格式、内容以及组织形式。

模版格式的 XML 中的使用的标记及其属性：

(1) 根节点

<root version = " xxx" height = " xxx" width = " xxx" target = "xxx"></root>；

version 属性标识了本模版的版本号，height 和 width 属性规定了模版生成后的界面的高和宽，target 属性标识了本 XML 元素，每个需要上传其内容的 XML 元素的 target 均不相同，以便服务器端能够正确识别解析，根节点的 target 属性表明了本模版的类型。

(2) 第一层子节点

＜firstLayerNode caption＝"xxx" target＝"xxx"＞……＜/firstLayerNode＞；

一个模版可能会包括很多个页面，第一层子节点即代表一个页面，caption 属性为本页面的标题，target 对应于数据库中的某一个表。

(3) 第二层子节点

第二层子节点规定了模版中每一个页面中可能会使用到的页面基本元素，元素可能还会包含子节点，比如表格元素。如果元素中有需要申报的数据，那么该元素的 target 属性就是数据在数据库中的字段名称。这些元素的种类包括：文本标签、输入框(单行/多行)、下拉列表框、表格、表格单元格、图片、黑线框、直线等。

2.2 子节点例子

文本标签格式的子节点可以定义为：

＜Label classtype＝"TLabel" caption＝"企业基本情况" left＝"224" top＝"16" width＝"185" height＝"20" fontsize＝"12"/＞

单行输入框格式的子节点可以定义为：

＜ComName classtype＝"TLabeledEdit" caption＝"企业名称" left＝"128" top＝"56" width＝"457" height＝"20" target＝"ComName" type＝"text" allownull＝"true"＞xx 公司＜/ComName＞

其中：classtype＝"TEdit"(元素类别，值不可更改) caption(标签文字，字符串) left(位置－左，整型) top(位置－顶，整型) height(位置－高，整型) width(位置－宽，整型) target(本元素标识)fontsize(文字大小，整型)。

3 客户端程序设计

客户端软件的作用是从服务器端下载最新版本的申报模版，根据模版定义格式，动态生成填写申报数据界面，并提供保存和上传申报数据功能。

下面的程序是用 DELPHI 实现的。

3.1 动态生成申报界面

每当用户选择了某项要申报的业务，客户端程序首先读取最新的申报模版，并以此为依据自动构造出申报界面。

模版由 xml 格式的文件构成，通过 delphi 中的 XMLDocument 控件，我们可以方便地读取标准 xml 文件。对 xml 文件中的各级节点，经过处理即可在一个 form 中生成对应的申报界面。示例代码如下：

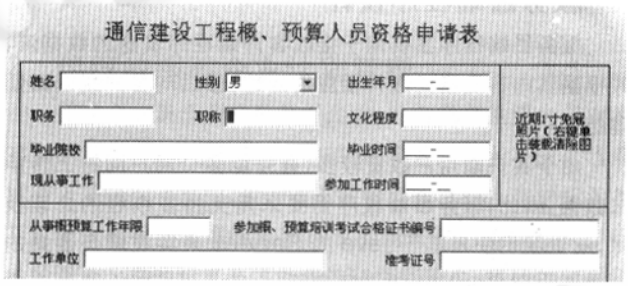
```
//根据 xml 模版定义，动态创建控件
function CreateModuleByXML(XMLDoc : TXMLDocument; pageControl : TPageControl): boolean;
var
  targets : array [0..500]of string; //保存 xml 元素和 form 中元素的对应关系
  i,j,k,controlCount : integer; //临时变量
  node,node1,node2,node3 : IXMLNode; //xml 文件中各层节点
  label1 : TLabel; //标签元素，没有 target
  memo : TMemo; //多行文本元素
  tabSheet : TTabSheet; //容器页元素，可包含其他元素，对应于 xml 文件的第一层节点
  shape : TShape; //线条元素
begin
  result := false;
  XMLDoc.Active := true;
  node := XMLDoc.DocumentElement;
  if node.ChildNodes.Count < 3 then exit;
  targets[0] := node.Attributes['version']; //保存模版版本
  targets[1] := node.Attributes['target']; //保存模版标识名
  TGroupBox(pageControl.Parent).Caption := node.Attributes['caption'];
  controlCount := 2;
  pageControl.Parent.Parent.Height := node.Attributes['height']+98;
  …… //对 form 的高、宽、字体等的设置
  for i := 0 to node.ChildNodes.Count-3 do //第一层节点的构建
  begin
    node1 := node.ChildNodes[i];
    tabSheet := TTabSheet.Create(pageControl);
    tabSheet.Parent := pageControl;
    tabSheet.Tag := controlCount;
    targets[controlcount] := node1.Attributes['target'];
    if Length(targets[controlcount]) = 0 then exit;
```

```
        controlCount := controlCount+1;
        tabSheet.Caption := node1.Attributes['caption'];
        tabSheet.PageControl := pageControl;
        for k := 0 to node1.ChildNodes.Count-1 do
        begin
          node3 := node1.ChildNodes[k];
          if UpperCase(node3.Attributes['classtype']) = 'TLABEL' then //标签节点构建
          begin
              label1 := TLabel.Create(pageControl);
              label1.Parent := tabSheet;
              label1.Caption := node3.Attributes['caption'];
              label1.Left := Node3.Attributes['left'];
          end
           else if UpperCase(node3.Attributes['classtype']) = 'TMEMO' then //多行文本节点构建
          begin
            memo := TMemo.Create(pageControl);
            memo.Parent := tabSheet;
            memo.Lines.Text := node3.Text;
            memo.Tag := controlCount;
            targets[controlCount] := node3.Attributes['target'];
            if Length(targets[controlcount]) = 0 then exit;
            controlCount := controlCount+1;
          end
          ……. //其他类型的元素依此类推,动态创建的原理一样
        end;
      end;
      XMLDoc.Active := false;
      result := true;
    end;
```

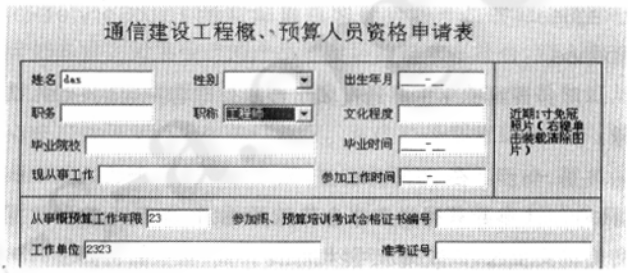
保存用户填写申报数据的过程,正好和动态创建申报界面过程相反,需要从申报界面的 form 中读取所有表单元素,根据表单元素的 Tag 属性和模版中的 target 相对应的关系,按照模版格式,生成可供上传的 xml 文件。

当申报数据项发生变化时,传统做法需要对客户端软件修改后重新分发,而我们采用的动态创建申报界面方式只需对服务器端的模版进行调整即可。如,申报数据中的职称字段原先是由用户手动输入,类型为单行文本框(TLabeledEdit);现在需要改为由用户选择输入,类型为下拉选择框(TComboBox)。这时我们只需将模版中的职称节点由 <RFGENDER classtype="TLabeledEdit" left="235" top="73" width="89" height="20" target="RFGENDER" type="text" ></RFGENDER> 改为 <RFGENDER classtype="TComboBox" left="235" top="73" width="89" height="20" target="RFGENDER" itemStrings="tc_dic_gender"></RFGENDER>

效果如下:



变动前



变动后

3.2 上传申报数据

客户端通过发送 web 请求上传申报数据的过程如下:

客户端填写完申报数据后,需填入用户名和密码,点击提交按钮进行提交,这时客户端程序执行以下过程:将用户名、密码以及申请类别和模版的版本号发送到服务器端验证,服务器端验证通过则返回成功信息并返回一个唯一标识作为客户端待上传申报数据的文件名,若验证不通过则返回失败信息。验证通过后,客户端把唯一标识的文件名作为包头信息的一部分,以数据流的格式将申报数据发送到服务器端,同时在本地建立跟踪文件,记录该申报数据的传输进度。服务器端根据唯一标识文件名对接收到的申报数据进行处理,然后向客户端返回结果。

下面简单介绍一下客户端上传数据流的包头格式：

前 200 个字节为本次文件传送的包头信息，根据包头信息可对整个数据流进行处理。包头信息可包括如下各种数据：

- 请求类型(如：版本验证、文件传输、取消传输等)
- 用户名和密码
- 版本标识、版本号
- 文件包在整个文件中的开始位置
- 文件包的大小
- 本次传送文件规定的名称，该名称在服务器端唯一
- 其他信息

4 服务器端程序设计

服务器端程序主要用于接收客户端程序发送的数据流，并根据其内容及格式进行相应的处理，最后向客户端返回处理结果。

由于客户端用户数量较多，有可能多个用户同时上传申报数据，因此，服务器端程序需要采用支持多线程的处理技术，这里我们使用基于 Java 的 Servlet 来实现服务器端的接收。一个 Servlet 类必须实现 doGet、doPost 这两个方法，其中 doPost 方法用于处理 Post 请求，该方法有两个参数，第一个参数为来自客户端数据的 HttpServletRequest，第二个参数为客户端响应的 HttpServletResponse。

服务器端程序主要通过如下步骤处理客户端的 Post 请求：

第一步，在 doPost 方法里，根据 HttpServletRequest 参数获得客户端上传的数据流，如果考虑到数据传输的安全性，该数据流可以加密。

第二步，分析数据流信息头(前 200 字节)，根据信息头内容，做出相应处理，如果是请求验证的，则把验证结果通过 HttpServletResponse 参数的 write 方法，返回给客户端程序；如果是取消上传的，则根据信息头中指定的文件名，删除服务器上传目录中的相应文件，然后把操作结果通过 HttpServletResponse 参数的 write 方法，返回给客户端程序；如果是请求上传的，根据信息头中包含的各项信息验证本次上传的数据包是否合法，如果合法，接着执行下一步，否则返回给客户端出错信息。

第三步，对于合法的上传请求，如果信息头中指定的文件名在服务器上传目录中不存在，则先按信息头中规定的文件名在上传目录中创建指定文件(文件扩展名为 jc，表明此文件为未续传完毕的文件)，然后在文件中写入本数据包的数据；如果指定的文件名在服务器上传目录中已存在，则把数据包按规定位置写入到服务器上传目录中的相应文件中。当整个文件的数据包都已写入完毕后，自动将文件的扩展名变为 xml，表明此文件为已续传完毕的文件，继续执行下一步。

注意：当网络出现故障或服务器端出现异常时，客户端将无法接收到服务器端返回信息，这种情况下，客户端认为数据传输出现问题，在本地跟踪文件中记录下文件上传的进度，并结束本次上传。当客户端通过自动或手动方式再次启动上传进程时，根据跟踪文件中记录的上传进度，继续给服务器端发送未上传的申报数据，直至整个文件传输完毕。

第四步，对于已传输完毕的文件，除了变更文件的扩展名外，还要根据该 xml 文件的各层节点所包含的数据以及它们对应的 target，构造生成插入数据库的 sql 语句，并把这些 sql 语句作为一个事务来提交。如果以上步骤都正确执行了，则返回给客户端上传成功的信息，否则返回相应的错误信息。

5 结束语

这种基于模版的客户端申报方式已在信息产业部通信行业规划建设管理信息系统中得到应用，大大减少了企业、个人申报时的工作量，成功解决了申报数据量大、掉线所带来的问题，同时满足了用户保密性方面的要求。该系统采用了这种灵活的模版方式后，将业务变动对系统的影响降到最低程度，只需对相应的模版和数据库表进行调整，而无需对客户端程序做任何变动，即尽量满足了用户变化的需要，又避免了修改客户端程序给用户带来的不便。该模式可以用于任何需要支持断点续传，且用户既对保密性要求较高，又希望系统具有较高灵活性的任何场合。

参考文献

1 孙和平、秦宏，电子政务概况，计算机世界报，第 11 期，2003。

2 李大成、陈苹萌，Java 与 XML 的结合应用，计算机应用，2002 年 2 月。

3 Goldfarb C F, Prescod P. The XML Handbook. Prentice Hall PTR, 1998。

4 李景春、王强 等，基于 DOM 的 XML 文档支持系统[J]，南京大学学报，2000－09。