

数据集成作业二文档

1 院系服务端（教务系统ABC服务器）+ 院系客户端

1.1 需求

1. A、B、C院系使用原有的 LocalServer 端处理与本院系客户端的通信，如登录验证，课程信息传送等消息，使用自定义 socket 文本消息格式或使用 XML 标准格式表示查询或响应消息。
2. 新增加的 XMLServer 端处理数据过程如下：通过读取本地数据库数据，将请求数据转化为 XML 文件，并将在与集成服务器的通信中使用。在处理 XML 文件时，用相应的 schema 进行验证。

1.2 Local Server

1. 对学生用户：
 - a. 登陆验证：对不同年级的学生设置不同的选课权限
 - b. 个人信息管理：个人信息查询更新
 - c. 个人课程管理：课程查看，选择，删除和更新
2. 对管理员用户：
 - a. 登陆验证：对不同级别的管理员设置相应的管理权限
 - b. 个人信息管理：个人信息查询更新
 - c. 统计信息查看：查看本院系学生、课程、课程被选情况等统计信息
 - d. 教务信息管理：对课程，开放课程，学生，选课等数据的设置操作
 - e. 后台管理：增删管理员，启动关闭服务器，查看在线情况等等

以上是一个教务系统所包含的基本功能。基于这个集成系统的特殊性，我们对功能进行了补充和精简，如下：

1. 对学生用户：
 - a. 登录验证：在 LocalServer 处理登录验证，并根据学生的权限展示不同的课程选项
 - b. 查看课程：
 - i. 本院系课程：由LocalServer直接进行相应
 - ii. 其他院系课程：通过HTTP请求与集成端交互，解析传输的XML文件获取课程信息
 - c. 选择、退选课程：
 - i. 本院系课程：由LocalServer直接进行相应
 - ii. 其他院系课程：通过XML和集成端交互，集成端请求其他院系的LocalServer完成选课/退选

2. 对管理员用户

- a. 查看本院系学生选课情况、课程情况、课程被选情况
- b. 开放课程供其他院系学生选择

1.3 XML Server

1. 使用dom4j生成xml

```
1 import org.dom4j.Document;
2 import org.dom4j.DocumentHelper;
3 import org.dom4j.Element;
4 import org.dom4j.io.OutputFormat;
5 import org.dom4j.io.XMLWriter;
6
7 import java.io.File;
8 import java.io.FileOutputStream;
9 import java.sql.ResultSetMetaData;
10
11 public class XMLTest {
12     public static void main(String[] args) {
13         try {
14             // 创建document对象
15             Document document = DocumentHelper.createDocument();
16             // 创建根节点bookRoot
17             Element StudentRoot = document.addElement("studentA");
18             // 向根节点中添加第一个节点
19             Element book1 = StudentRoot.addElement("student");
20             // 向子节点中添加属性
21             book1.addAttribute("id", "1");
22             // 向节点中添加子节点
23             Element name = book1.addElement("name");
24             // 向子节点赋值
25             name.setText(".....");
26         }
27         .....
28     }
29 }
```

2. 使用xsd解析xml

1.4 实现思路

1. 连接数据库

因为ABC院系使用不同的服务器，所以采用不同的方法连接数据库。

- A院系使用SqlServer
- B院系使用Oracle
- C院系使用mysql

以A院系为例：

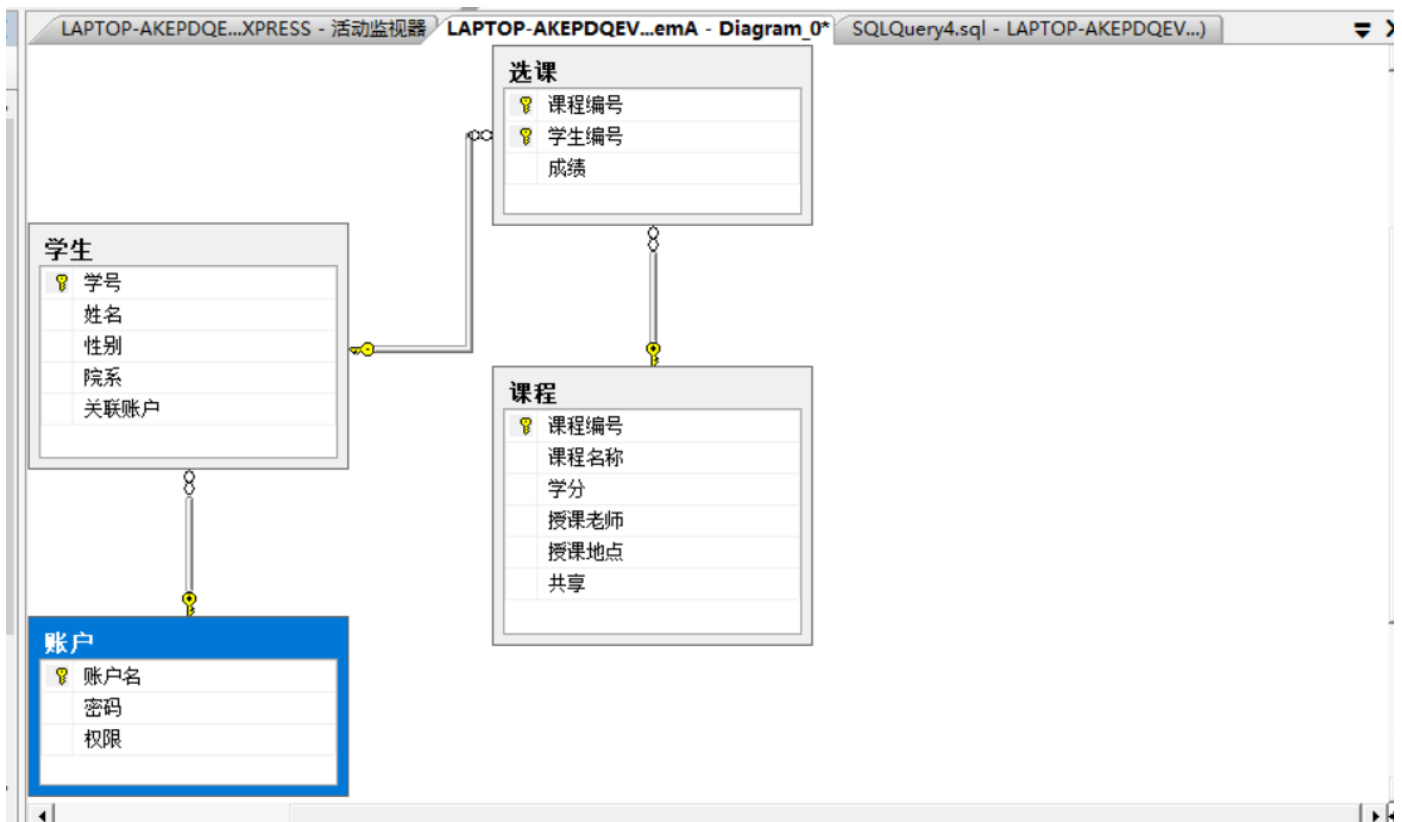
- 安装Sql Server 2008 (Microsoft® SQL Server® 2008 R2 SP2 - Express Edition)
- Java连接Sql Server2008

```
1 package org.systemA;
2
3 import java.sql.Connection;
4 import java.sql.DriverManager;
5 import java.sql.PreparedStatement;
6 import java.sql.ResultSet;
7
8 public class AConnection {
9     public static Connection getConnection() {
10         String dbURL = "jdbc:sqlserver://localhost:1433;DatabaseName=systemA"
11         Connection con = null;
12         try {
13             //1.加载驱动
14             Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");
15             System.out.println("加载驱动成功！");
16             //2.连接
17             con = DriverManager.getConnection(dbURL, "sa", "123456");
18             System.out.println("连接数据库成功！");
19         } catch (Exception e) {
20             e.printStackTrace();
21             System.out.println("连接数据库失败！");
22         }
23         return con;
24     }
25
26     public static void main(String[] args) {
27         AConnection.getConnection();
28     }
29 }
```

- 建表脚本

```
1 -- 创建院系A数据库
```

```
2 IF NOT EXISTS(SELECT name FROM master.dbo.sysdatabases WHERE name = 'systemA')
3 CREATE DATABASE systemA;
4
5 -- 切换到院系A数据库
6 USE systemA;
7
8 -- 创建账户表
9 IF NOT EXISTS(SELECT * FROM sysobjects WHERE name='账户' AND xtype='U')
10 CREATE TABLE 账户 (
11     账户名 VARCHAR(10) PRIMARY KEY,
12     密码 VARCHAR(6),
13     权限 CHAR(4)
14 );
15
16 -- 创建学生表
17 IF NOT EXISTS(SELECT * FROM sysobjects WHERE name='学生' AND xtype='U')
18 CREATE TABLE 学生 (
19     学号 VARCHAR(12) PRIMARY KEY,
20     姓名 VARCHAR(10),
21     性别 VARCHAR(2),
22     院系 VARCHAR(10),
23     关联账户 VARCHAR(10) FOREIGN KEY REFERENCES 账户(账户名)
24 );
25
26 -- 创建课程表
27 IF NOT EXISTS(SELECT * FROM sysobjects WHERE name='课程' AND xtype='U')
28 CREATE TABLE 课程 (
29     课程编号 VARCHAR(8) PRIMARY KEY,
30     课程名称 VARCHAR(10),
31     学分 VARCHAR(2),
32     授课老师 VARCHAR(10),
33     授课地点 VARCHAR(20),
34     共享 CHAR(1)
35 );
36
37 -- 创建选课表
38 IF NOT EXISTS(SELECT * FROM sysobjects WHERE name='选课' AND xtype='U')
39 CREATE TABLE 选课 (
40     课程编号 VARCHAR(8),
41     学生编号 VARCHAR(12),
42     成绩 VARCHAR(3),
43     CONSTRAINT PK_选课 PRIMARY KEY (课程编号, 学生编号),
44     CONSTRAINT FK_选课_课程 FOREIGN KEY (课程编号) REFERENCES 课程(课程编号),
45     CONSTRAINT FK_选课_学生 FOREIGN KEY (学生编号) REFERENCES 学生(学号)
46 );
```



构造数据

```
1  -- 插入账户数据
2  INSERT INTO 账户 (账户名, 密码, 权限)
3  VALUES ('user1', '123456', 'admi'),
4          ('user2', '123456', 'stud'),
5          ('user3', '123456', 'stud'),
6          ('user4', '123456', 'stud'),
7          ('user5', '123456', 'stud');
8
9  -- 插入学生数据
10 INSERT INTO 学生 (学号, 姓名, 性别, 院系, 关联账户)
11 VALUES ('20210001', '张三', '男', '计算机', 'user3'),
12         ('20210002', '李四', '男', '信息管理', 'user4'),
13         ('20210003', '王五', '女', '数学', 'user5');
14
15 -- 插入课程数据
16 INSERT INTO 课程 (课程编号, 课程名称, 学分, 授课老师, 授课地点, 共享)
17 VALUES ('C001', '计组', '3', '张老师', 'A101', '0'),
18         ('C002', '数据结构', '4', '李老师', 'B201', '1'),
19         ('C003', '数据库', '3', '王老师', 'C301', '0');
20
21 -- 插入选课数据
22 INSERT INTO 选课 (课程编号, 学生编号, 成绩)
23 VALUES ('C001', '20210001', '88'),
24         ('C002', '20210001', '95'),
25         ('C003', '20210001', '76');
```

```
26      ('C002', '20210002', '90'),
27      ('C003', '20210002', '85'),
28      ('C002', '20210003', '92');
```

2. 与其他系统交互

使用HTTP请求访问集成端，获得课程信息，以及跨选课程；同时构造HTTP请求，传输xml文件。

- 构造Http Server用于相应集成端的请求，如选课和退课

```
1 // 创建HttpServer服务器
2 HttpServer httpServer = HttpServer.create(new InetSocketAddress(8080), 10)
3 //将 / 请求交给MyHandler处理器处理
4 httpServer.createContext("/", new MyHandler());
5 httpServer.start();
```

- 构造Http Client用于向集成端发送请求，如选修其它院系共享的课程

- 具体通过apache common封装好的HttpClient实现请求功能

httpClient的get或post请求方式步骤：

- 生成一个HttpClient对象并设置相应的参数；
- 生成一个GetMethod对象或PostMethod并设置响应的参数；
- 用HttpClient生成的对象来执行GetMethod生成的Get方法；
- 处理响应状态码；
- 若响应正常，处理HTTP响应内容；
- 释放连接。

```
1 public class HttpClientUtil {
2     .....
3 }
```

3. 实现系统功能

系统端使用纯JAVA实现，客户端使用JAVA SWING实现

- 登录验证

A院系教务系统

用户名


密 码

身 份 ☐ 管理员 ☒ 学生

功能选择

A院系教务系统

查看个人信息

 A院系教务系统

学号: 20210001

姓名: 张三

性别: 男

院系: 计算机

关联账户: user3

返回

- 查看已选课程

 A院系教务系统

课程编号	学生编号	成绩
C001	20210001	88
C002	20210001	95
C003	20210001	76

返回

- 选择和退选本院系课程

- 选课本院系课程

A院系教务系统

—□×

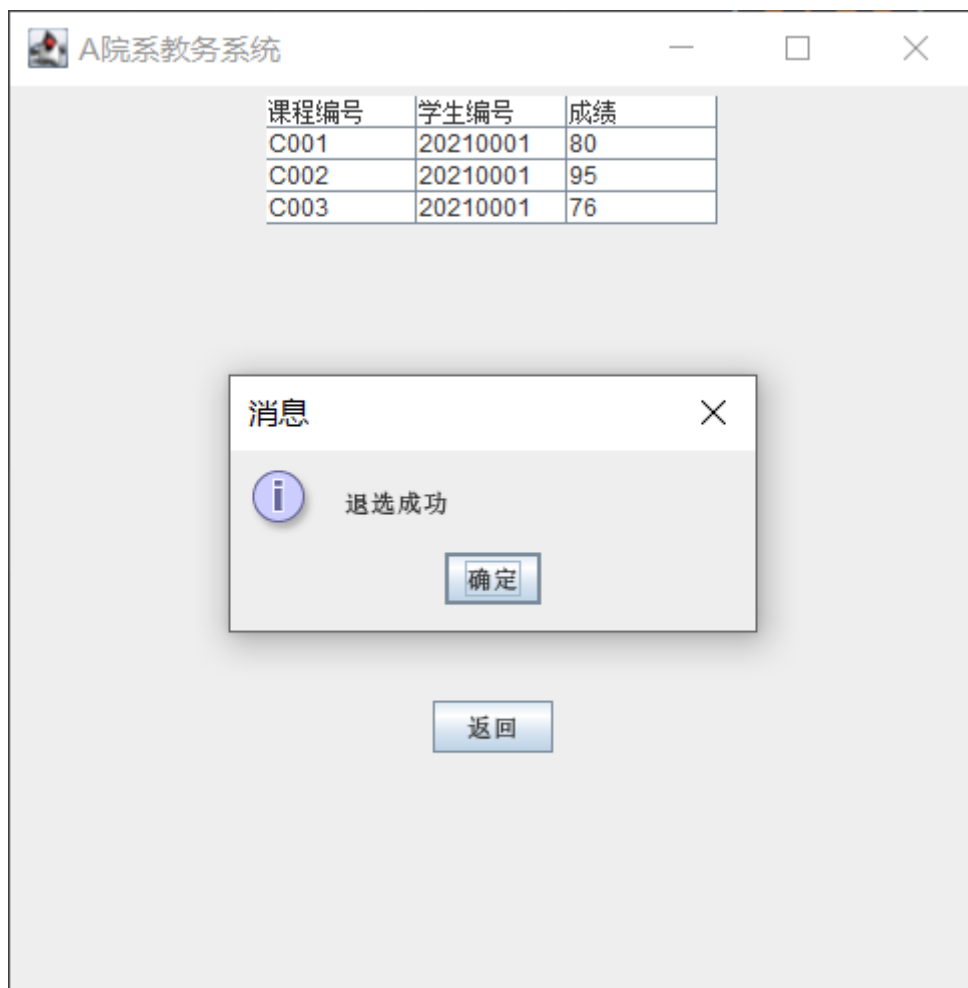
课程编号	课程名称	学分	授课老师	授课地点
C001	计组	3	张老师	A101
C002	数据结构	4	李老师	B201
C003	数据库	3	王老师	C301

计组▼

选课

返回

- 退选



- 查看跨院系的课程

2 集成服务端

2.1 需求

集成服务器主要负责响应处理 系统间的通信消息和命令，例如课程选择和更新，和接受与转发封装数据的 XML 文件，并 进行 XML 文件的验证，转换。

2.2 实现思路

- 格式化XML
 - 使用xslt将外部传入的XML格式化
 - classA.xml

```

<?xml version="1.0" encoding="utf-8"?>
<Classes>
  <class>
    <课程编号>123456789</课程编号>
    <课程名称>class1</课程名称>
    <学分>2</学分>
    <授课老师>teacher</授课老师>
    <授课地点>A</授课地点>
  </class>
  <class>
    <课程编号>223456789</课程编号>
    <课程名称>class2</课程名称>
    <学分>2</学分>
    <授课老师>teacher2</授课老师>
    <授课地点>B</授课地点>
  </class>
</Classes>

```

- class.xml

```

<?xml version="1.0" encoding="utf-8"?>
<Classes>
  <class>
    <id>123456789</id>
    <name>class1</name>
    <teacher>teacher</teacher>
    <location>A</location>
  </class>
  <class>
    <id>223456789</id>
    <name>class2</name>
    <teacher>teacher2</teacher>
    <location>B</location>
  </class>
</Classes>

```

- formatClass.xsl

```
<Classes>
  <xsl:for-each select="class">
    <class>
      <id>
        <xsl:value-of select="课程编号"/>
        <xsl:value-of select="编号"/>
        <xsl:value-of select="Cno"/>
      </id>
      <name>
        <xsl:value-of select="名称"/>
        <xsl:value-of select="课程名称"/>
        <xsl:value-of select="Cnm"/>
      </name>
      <xsl:choose>
        <xsl:when test="./Cpt=''>
          <score>
            <xsl:value-of select="学分"/>
          </score>
        </xsl:when>
        <xsl:when test="./Cpt!=''>
          <score>
            <xsl:value-of select="Cpt"/>
          </score>
        </xsl:when>
      </xsl:choose>

      <teacher>
        <xsl:value-of select="授课老师"/>
        <xsl:value-of select="老师"/>
        <xsl:value-of select="Tec"/>
      </teacher>
    </class>
  </xsl:for-each>
</Classes>
```

```

        <teacher>
            <xsl:value-of select="授课老师"/>
            <xsl:value-of select="老师"/>
            <xsl:value-of select="Tec"/>
        </teacher>

        <location>
            <xsl:value-of select="授课地点"/>
            <xsl:value-of select="地点"/>
            <xsl:value-of select="Pla"/>
        </location>
    </class>
</xsl:for-each>
</Classes>
</xsl:template>

```

- 将标准XML转化为服务器需要的XML

- choice.xml

```

<?xml version="1.0" encoding="gb2312"?>
<Choices>
    <choice>
        <sid>12345678</sid>
        <cid>123456789</cid>
        <score>2</score>
    </choice>
</Choices>

```

- choiceA.xml

```

<?xml version="1.0" encoding="utf-8"?>
<Choices>
  <choice>
    <学号>12345678</学号>
    <课程编号>123456789</课程编号>
    <成绩>2</成绩>
  </choice>
</Choices>

```

- choiceToA.xsl

```

<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:output method="xml" encoding="gb2312" />
  <xsl:template match="/">
    <xsl:apply-templates />
  </xsl:template>
  <xsl:template match="Choices">
    <Choices>
      <xsl:for-each select="choice">
        <choice>
          <学号>
            <xsl:value-of select="sid" />
          </学号>
          <课程编号>
            <xsl:value-of select="cid" />
          </课程编号>
          <成绩>
            <xsl:value-of select="score" />
          </成绩>
        </choice>
      </xsl:for-each>
    </Choices>
  </xsl:template>
</xsl:stylesheet>

```

- 使用上述功能即可完成跨院系选课
 - 将院系服务器发送的选课xml格式化，根据目标服务器重新转换为目标服务器所需格式

```
@RequestMapping("/classChoice")
public String chooseClass(@RequestParam("from") String from,
                          @RequestParam("to") String to,
                          @RequestBody String classChoice){
    String formatChoiceXML=Trans.doXsl( xslPath: basePath+formatChoice, classChoice).getToContent();
    String toChoiceXML=Trans.doXsl( xslPath: basePath+transChoice+to+suffix, formatChoiceXML).getToContent();
    //向目标服务器发送选课请求,返回值保存在res
    String toUrl=getToUrl(to);
    toUrl=toUrl+choiceSuffix;
    String res= null;
    try {
        res = HttpHelper.sendPost(toUrl,toChoiceXML);
    } catch (IOException e) {
        e.printStackTrace();
    }
    return res;
}
```