基于 *WebService* 技术的 *J2ME* 和 *.NET* 互连

徐翔斌

摘　要　使用实现 JSR172 规范的 J2ME 开发包，可以快速实现 J2ME 客户端同 WebService 服务器端应用程序通信和数据交换，为企业构建移动电子商务系统提供了一种快速、有效的解决方案。

关键词　WebService，J2ME，JSR172

一、引言

随着移动通信网络的发展和支持 J2ME 的个人移动通信设备的普及、用手机，PDA 和其他嵌入式设备从网络中获取信息已经成为现代人生活的一部分，同时传统意义上的电子商务活动也开始向移动通信领域内扩展，为此，企业必须改造和升级现有的电子商务系统，构建自己的移动电子商务系统。利用无线互联技术，企业可以将电子商务的终端用户从企业内部的 IntraNet 或广域的 InterNet 中的客户端扩展到不受空间位置限制的移动用户，从而拓展了企业的需求。目前绝大多数企业级应用都是构建在 .NET 或者 J2EE 平台上，笔者在为某企业的移动电子商务系统改造过程中，利用 WebService 技术成功地实现 J2ME 移动终端设备和 .NET 企业级应用的互连，取得了良好的效果。

二、.NET 企业级应用程序

1. 系统功能设计

该企业级应用程序是为互联网上的用户提供天气预报服务，用户登录该系统，输入感兴趣的城市和日期，系统就会将相应的最高和最低温度、湿度、阴晴状况、出行提示等天气信息告诉用户。系统采用的是微软的 SQLServer2000 数据库来存储城市天气状况。该系统的数据表设计如图 1 所示（注：为节约篇幅，笔者对系统的数据表和业务流程进行了适当的简化）。

相应的 SQLSERVER2000 的数据库脚本如下：

```
create table tblcity (
  cityId    int      identity,
  cityName  varchar(20)  not null,
  cityOther  varchar(30)  null,
```

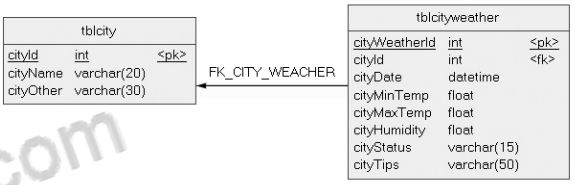

图 1　服务器端应用程序数据表设计图

```
  constraint PK_TBLCITY primary key (cityId)
)
create table tblcityweather (
  cityWeatherId  int  identity,
  cityId  int  not null,
  cityDate  datetime  not null,
  cityMinTemp  float  not null,
  cityMaxTemp  float  not null,
  cityHumidity  float  not null,
  cityStatus  varchar(15)  not null,
  cityTips  varchar(50)  not null,
  constraint PK_TBLCITYWEATHER primary key (cityWeatherId)
)
create  index Relationship_1_FK on tblcityweather (cityId ASC)
alter table tblcityweather  add constraint FK_CITY_WEACHER
foreign key (cityId)
   references tblcity (cityId)2.
```

2. 构建 .NET WebService 应用

打开 Microsoft Visual Studio .NET 2003，新建一个基于 Visual C# 语言的 ASP.NET Web 服务项目，Web 服务的名字为 weatherforcast，并将 Web 服务文件的名字修改为 wetherforcast.asmx，添加一个提供天气服务的 WebService 方法，该文件主要内容如下：

```
using System;
using System.Collections;
```

```
using System. ComponentModel;
using System. Data. SqlClient;
using System. Data;
using System. Diagnostics;
using System. Web;
using System. Web. Services;
namespace wetherforcast
{
    [WebService(Namespace = ″  http: //ecjtu. jx. cn/WebService″)]
    public class WeatherForcast : System. Web. Services. WebService
    {
        //这里省略系统自动生成的代码
        [WebMethod]
        public string getWeatherByCity(string cityName, string cityDate)
        {
string weather = null;
string connStr = ″server = localhost; uid = sa; pwd = 128ecjtu; database = weatherservice″;
SqlConnection sqlConn = new SqlConnection(connStr);
SqlDataReader sqlDataReader = null;
try
    {
    sqlConn. Open();
    string cmdString = ″select cityName, CONVERT(varchar(10), cityDate, 120), cityMinTemp, cityMaxTemp, cityHumidity, cityStatus, cityTips from tblcity inner join tblcityweather on tblcity. cityId = tblcityweather. cityId where cityName = ′″ + cityName + ″′″ + ″ and cityDate = ′″ + cityDate + ″′″;
    SqlCommand cmd = new SqlCommand(cmdString, sqlConn);
    sqlDataReader = cmd. ExecuteReader();
    while (sqlDataReader. Read())
    {
    weather = sqlDataReader[0] + ″#″ + sqlDataReader[1] + ″#″ + sqlDataReader[2] + ″#″ + sqlDataReader[3] + ″#″ + sqlDataReader[4] + ″  #  ″ + sqlDataReader[5] + ″  #  ″ + sqlDataReader[6] + ″#″;
    }
}
finally
{
    if (sqlDataReader ! = null)
    {
    sqlDataReader. Close();
    }
    if (sqlConn ! = null)
    {
    sqlConn. Close();
    }
    }
        return weather;
    }
}
}
```

注意：读者可以根据自己的 SQLServer 服务器来设置 SqlConnection 连接字串 connStr 参数。

该 Web 服务为用户提供一个 API 接口函数：

```
public string getWeather(String strCity, string strDate){}
```

它将客户端输入的城市和日期作为输入参数，从 weatherservice 数据库中的 tblcity 和 tblcityweather 两个表中进行连接查询，检索出符合条件的天气信息，然后将相关信息返回给 WebService 的客户端。为减少在 WebService 调用过程中，SOAP 的网络传输字节数量，同时也为了方便客户端对 WebService 返回的信息进行解析，在这里将返回的天气信息编码成 string 格式，其中采用“#”符号隔开记录中相应字段的信息，这样客户端就可以根据“#”分隔符进行解析，然后提取天气情况相应的具体资料。

三、J2ME 客户端程序设计

1. J2ME 客户端工作流程

该 J2ME 客户端工作流程比较简单，用户启动程序后，首先显示的是一个有关系统信息主屏幕，按“进入系统”键进入一个工作屏幕，在这里用户可以输入相应的城市名称和选择日期，提交后，系统同 WebService 服务器端进行通信和数据交换，当 WebService 服务器端的信息返回后，系统将查询的天气信息显示给用户；若是网络出现故障，不能连接到 WebService 服务器，系统会显示一个网络故障的提示屏幕。整个工作流程如图 2 所示。

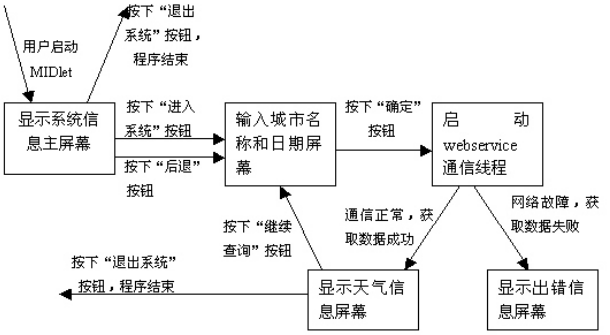

图 2 J2ME 客户端工作流程

2. 工具准备

首先，从 sun 的官方网站 www. sum. com 上下载 J2ME 开发工具包，注意要下载提供支持 JSR172 规范类库（也就是对

WebService 提供支持）的开发包，笔者推荐下载 Windows 环境下的开发包 j2me_wireless_toolkit－2_2－windows. exe。下载后双击 j2me_wireless_toolkit－2_2－windows. exe 可执行文件就可以安装，笔者建议安装在默认路径下，安装过程比较简单，一直回车就可以了，安装完毕后得到 C: \WTK22 目录，J2ME 开发的应用程序就位 C: \WTK22\apps 的子目录中。

3. 生成 WebService 的 Stub

j2me_wireless_toolkit－2_2 为 J2ME 的开发者准备了一个根据 WebService 的 WSDL 描述文件自动生成 J2ME 调用客户端 Stub 的工具，方便了开发基于 WebService 的 J2ME 应用程序。该工具的使用简单：从系统程序组的 J2me Wireless Toolkit2. 2 的子菜单中找到 Utilities 并且启动它，点击 Stub Generator 按钮，弹出 Stub Generator Dialog 工作界面，工作界面如图 3 所示。

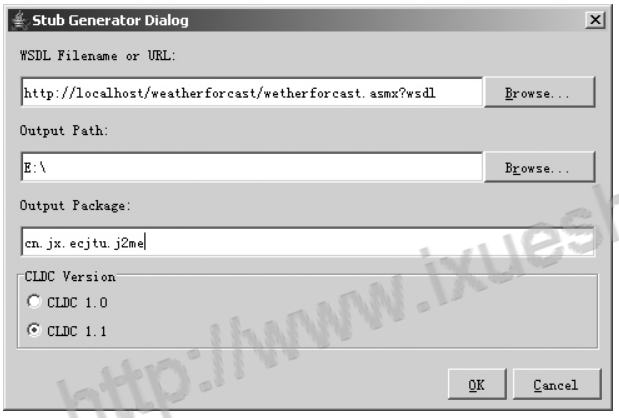

图 3 Stub Generator 工作界面

在 WSDL Filename or URL 中输入前面用 C#语言构建的 WebService 的 WSDL 描述文件，选择一个 Output Path 作为生成 stub 文件的目录，并且在 Output Package 中指定一个 Java 包名，笔者的设置情况如图 3 所示。设置完毕后，点击 OK 按钮，该工具就会在指定的路径下生成 J2ME 环境下同该的 WebService 服务器通信和交换数据的接口文件和相应的 Stub 文件，这些文件将被用在 J2ME 系统中，作为调用 WebService 的基础设施，从 JSR172 规范中复杂的 WebService 调用 API 中解脱出来，专心处理应用程序的界面设计和业务逻辑。

4. J2ME 程序设计

J2ME 客户端采用典型 MVC 设计模式，由一个屏幕控制器（Navigator 类），和数据获取组件（GetWeatherByWebService Thread 类，WetherforcastSoap 接口和 WetherforcastSoap_Stub 类）和几个屏幕显示组件（MainScreen 类，GetWeatherScreen 类和 ShowWeatherDate 类）构成，该设计具有良好的扩展性；系统的设计类如图 4 所示。

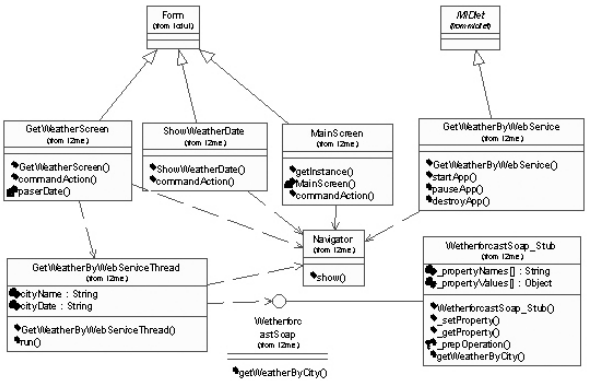

图 4 J2ME 客户端设计类图

启动 J2ME Wireless Toolkit2. 2 的 KToolbar 工具，新建一个名为 GetWeatherByWebService 的 J2ME 项目，MIDlet 文件名为 GetWeatherByWebService。内容如下：

```
package cn. jx. ecjtu. j2me;
import javax. microedition. lcdui. *;
import javax. microedition. midlet. *;
public class GetWeatherByWebService extends MIDlet {
public GetWeatherByWebService() {
    super();
    Navigator. dispaly = Display. getDisplay(this);
    Navigator. midlet = this;
}
public void startApp() throws MIDletStateChangeException {
    Navigator. show(Navigator. MAIN_SCREEN, null);
}
public void pauseApp() {    }
public void destroyApp(boolean arg0) throws MIDletStateChangeException {
    Navigator. midlet. notifyDestroyed();
}
}
```

该 MIDlet 简单，程序启动后屏幕控制器 Navigator 将系统的主屏幕 MainScreen 显示出来，MainScreen 类显示系统的基本信息，代码如下：

```
package cn. jx. ecjtu. j2me;
import javax. microedition. lcdui. *;
public class MainScreen extends Form implements CommandListener{
    private static Displayable instance = new MainScreen();
    private StringItem label;
    private Command exitCmd;
    private Command enterCmd;
    public static Displayable getInstance(){
    return instance;
}
```

```
    private MainScreen(){
        super("移动天气服务");
        label = new StringItem("基于 WEB 服务的天气查询系统","", Item. PLAIN);
        append(label);
        exitCmd = new Command("退出系统", Command. EXIT, 1);
        enterCmd = new Command("进入系统", Command. OK, 2);
        addCommand(exitCmd);
        addCommand(enterCmd);
        setCommandListener(this);
    }
public void commandAction(Command c, Displayable d){
        if(c = = enterCmd){
          Navigator. show(Navigator. GET_WEATHER_
SCREEN, null);
        }else{
          Navigator. midlet. notifyDestroyed();
        }
    }
}
```

MainScreen 被设计成为单例模式，因系统只有一个主屏幕，这样可以防止系统实例化多个 MainScreen 对象而浪费手机内存。按下“进入系统”按钮之后，Navigator 类将 GetWeatherScreen 屏幕显示出来，用户可以在这个屏幕上输入城市名称，并且选择日期，GetWeatherScreen 类的代码如下：

```
package cn. jx. ecjtu. j2me;
import javax. microedition. lcdui. *;
import javax. microedition. midlet. *;
import java. util. *;
public class GetWeatherScreen extends Form implements
CommandListener {
    private TextField cityName;
    private DateField df;
    private Command okCmd;
    private Command backCmd;
    public GetWeatherScreen() {
    super("移动天气服务");
    cityName = new TextField("请输入城市名:","", 8,
TextField. ANY);
    backCmd = new Command("后退", Command. BACK, 1);
    okCmd = new Command("确定", Command. OK, 2);
    df = new DateField("请选择日期:", DateField. DATE);
    df. setDate(new Date());
    append(cityName);
    append(df);
    addCommand(okCmd);
    addCommand(backCmd);
    setCommandListener(this);
    }
public void commandAction(Command c, Displayable d) {
    if (c = = okCmd) {
  new GetWeatherByWebServiceThread(cityName. getString
(), paserDate(df. getDate())). start();
    }else{
        Navigator. show(Navigator. MAIN_SCREEN);
    }
}
private String paserDate(Date date) {
    Calendar calendar = Calendar. getInstance();
    calendar. setTime(df. getDate());
    int day = calendar. get(Calendar. DATE);
    int year = calendar. get(Calendar. YEAR);
    int month = calendar. get(Calendar. MONTH) + 1;
    StringBuffer sbfDate = new StringBuffer();
    sbfDate. append(year) . append("-") . append(month)
. append("-"). append(day);
    return sbfDate. toString(). trim();
}
}
```

用户在完成相关信息的输入按“确定”按钮后提交数据，系统则启动一个后台线程 GetWeatherByWebServiceThread 同 WebService 服务器通信，GetWeatherByWebServiceThread 代码如下：

```
package cn. jx. ecjtu. j2me;
import java. rmi. RemoteException;
import javax. microedition. lcdui. *;
public class GetWeatherByWebServiceThread extends Thread
{
private String cityName;
private String cityDate;
public GetWeatherByWebServiceThread(String cityName,
String cityDate) {
    this. cityName = cityName;
    this. cityDate = cityDate;
}
public void run() {
    try {
    WetherforcastSoap wetherforcastSoap = new
WetherforcastSoap_Stub();
    String weatherInfo = wetherforcastSoap.
getWeatherByCity(cityName, cityDate);
    Navigator. show(Navigator. WEATHERINFO_SCREEN,
weatherInfo);
} catch (RemoteException e) {
    Alert alert = new Alert("从服务器接受数据错误","错
误", null, AlertType. ALARM);
    alert. setTimeout(2000);
    Navigator. dispaly. setCurrent(alert);
    }
}
}
```

在一个新的线程中进行 WebService 服务连接和传输数据是必要的，可以防止在同服务器通信故障的时候用户无法干预，甚至死机。在获取 WebService 服务器返回的数据后，系统将数据以一定的格式显示出来；要是出现了网络故障，系统将无法从 WebService 中获取数据，则显示一个系统通信故障的友好界面。而用于显示查询结果数据显示类，对 WebService 服务端传送过来的数据进行解码（以“#”为分隔符）和显示，代码如下：

```
package cn. jx. ecjtu. j2me;
import javax. microedition. lcdui. *;
public class ShowWeatherDate extends Form implements
CommandListener {
    String[] itemName = {"城  市:", "日  期:",
                "最低温度:", "最高温度:",
                "湿  度:", "天气状况:", "出行提示:"};
    private StringItem[] items =new StringItem[7];
    private Command cmdContinue;
    private Command cmdExit;
    public ShowWeatherDate(String weatherInfo) {
        super("您查询的结果如下");
        cmdExit = new Command("退出系统", Command.
EXIT, 1);
        cmdContinue = new Command("继续查询", Command.
OK, 2);
        addCommand(cmdExit);
        addCommand(cmdContinue);
        setCommandListener(this);
        int k =0;
        for (int i = 0; i <7; i + +) {
            k = weatherInfo. indexOf("#");
            String cityN = weatherInfo. substring(0, k);
            items[i] =new StringItem(itemName[i], cityN,
Item. PLAIN);
            this. append(items[i]);
            weatherInfo = weatherInfo. substring(k + 1);
        }
    }
    public void commandAction(Command c, Displayable d) {
        if (c = = cmdContinue) {
            Navigator. show(Navigator. GET_WEATHER_
SCREEN, null);
        }else{
            Navigator. midlet. notifyDestroyed();
        }
    }
}
```

最后是 Navigator 类，它作为一个屏幕流程控制器， 根据程序的工作流程将适当的屏幕显示出来，这样做符合 MVC 模式，也便于程序的扩展。Navigator 类的代码如下：

```
package cn. jx. ecjtu. j2me;
import javax. microedition. midlet. *;
import javax. microedition. lcdui. *;
public class Navigator {
    final public static int MAIN_SCREEN = 1;
    final public static int GET_WEATHER_SCREEN = 2;
    final public static int WEATHERINFO_SCREEN = 3;
    public static MIDlet midlet;
    public static Display dispaly;
    public static int current;
    public static void show(int nextScreen, Object o) {
    switch (nextScreen) {
    case MAIN_SCREEN:
        dispaly. setCurrent(MainScreen. getInstance());
        break;
    case GET_WEATHER_SCREEN:
        dispaly. setCurrent(new GetWeatherScreen());
        break;
    case WEATHERINFO_SCREEN:
        dispaly. setCurrent(new ShowWeatherDate
((String)o));
        break;
    }
}
}
```

将以上的类文件放置在 C:\WTK22\apps\GetWeatherWebService\src 目录下，同时把前面生成的 WebService 的 Stub 文件也拷贝到该目录下，在 KToolbar 中对项目进行编译和运行就可以了。

四、结语

无线移动设备作为一种屏幕和内存等资源受限设备，无法像 PC 机那样有很强的数据计算和图像处理的能力，键盘输入快捷性不强，在系统的设计和升级的过程中，应该考虑将数据计算和图像处理放在服务器端进行，然后将计算和处理的结果经过一定编码通过网络传输给无线移动设备。考虑到无线移动设备屏幕受限等因素，需要对传输和显示的数据和图像进行全面分析，在有限的显示空间内尽可能地提供有效、直观、丰富的文字和图片信息。

参考文献

1. Roger Riggs. J2ME 无线设备编程[M]. 机械工业出版社, 2001

2. 焦祝军, 张威. J2ME 无线通信技术应用开发[M]. 希望电子出版社, 2002

3. JOSEPH BUSTOS, KARLI WATSON. NET WEB 服务入门经典——C#编程篇[M]. 清华大学出版社, 2003

(收稿日期：2006 年 6 月 20 日)

阅读此文的还阅读了：

1. 双卡平台不再设限 S3 Chrome S27双卡测试
2. 实战4显卡SLI 技嘉GA-8N-SLI-QUAD Royal主板
3. 基于.NET平台WebService的应用集成与性能优化
4. 基于WebService技术的J2ME和.NET互连
5. CMOS ER开关可避免潜在的砷化镓问题
6. 基于.NET WebService的XML Web服务构建
7. 基于．Net＋WebService混合技术校园门户系统设哥的思考
8. 基于.Net Webservice虚拟教研平台的构建
9. 基于.NET平台的WebService构建与调用技术
10. 基于J2ME的WebService掌上系统的设计与实现