软件开发工具实例教程

（jQuery+EasyUI 版）

浙江理工大學
ZHEJIANG SCI-TECH UNIVERSITY
管理科学与工程系

2015 年 9 月

目 录

第 1 章、jQuery 及其开发环境 1
1.1 J2EE 开发环境简介 1
1.2 安装 Tomcat 服务器 1
1.3 安装 MyEclipse 6
1.4 导入和建立工程文件 jQDemos 10
1.5 添加和配置 MyEclipse 中的 Tomcat6.x 服务器 14
1.6 部署 jQDemos 工程文件 16
1.7 SQL Server2008 的网络配置 19
1.8 安装和生成 emlab 数据库 22
第 2 章、EasyUI 基本表单控件 23
实例 1. 创建 EasyUI 表单及常用控件。 23
实例 2. EasyUI 表单及其控件的 jQuery 语句设计。 25
实例 3. 利用绝对坐标实现表单控件的位置布局。 28
实例 4. 利用 append()方法在 jQuery 中动态定义和生成 EasyUI 控件。 31
实例 5. 构造控件自定义函数，简化控件的使用。 32
实例 6. 控件定义函数的综合应用。 36
实例 7. 使用 tabs 标签页控件分页显示多个表单控件。 38
实例 8. 静态组合框 combobox 控件及其事件和初值的设置。 39
实例 9. 利用滑块（slider）控件实现图片的等比例缩放。 40
实例 10. 工具栏、菜单和消息框控件的使用。 43
实例 11. 窗口控件 window 的应用及其函数构造。 46
实例 12. 利用布局（layout）控件实现页面布局。 46
实例 13. 获取表单中可编辑控件的名称、类型及其值。 48
实例 14. 表单的键盘控制与控件聚焦。 49
实例 15. 利用文件框控件 file 实现文件的上传。 51
实例 16. 服务器端文件下载的实现。 54
第 3 章、EasyUI 数据库控件 57
实例 17. 构造连接数据库 Java 类，实现数据查询操作。 57
实例 18. 服务器端数据更新语句的执行。 59
实例 19. 服务器端存储过程和用户定义函数的调用。 61
实例 20. 动态 combobox 组合框控件的定义与初值设置。 62

实例 21. 动态 combobox 组合框控件之间联动效果的实现。 64
实例 22. 服务器端取数据库中数据赋值到表单。 65
实例 23. 客户端和服务器端表单数据验证。 67
实例 24. 服务器端数据库记录的增删改操作。 69
实例 25. 服务器端 SQL 脚本文件.sql 的运行。 74
第 4 章、树与数据网格及其应用 78
实例 26. 静态树 Tree 控件及其基本操作。 78
实例 27. 基于 JSON 数据的静态树加载与操作。 80
实例 28. 从数据库一次性加载数据到树节点。 84
实例 29. 动态树节点的分层逐级加载。 87
实例 30. 动态组合树 ComboTree 控件及其应用。 91
实例 31. 静态 JSON 数据网格控件基础。 93
实例 32. 动态数据库网格控件及其分页处理。 95
实例 33. 数据库网格中数据的过滤与定位。 98
实例 34. 可编辑数据网格及其数据增删改操作的实现。 103
实例 35. 数据网格的表单关联扩展。 107
实例 36. 树形网格（TreeGrid）的综合应用。 110
第 5 章、图表统计及其应用 119
实例 37. FusionCharts 单序列图表及应用。 119
实例 38. FusionCharts 多序列图表及应用。 122
实例 39. FusionCharts 图表向下钻取。 124
实例 40. 带多层表头和汇总行的数据网格及其 Excel 文件输出。 130
附录 1、jQDemos 自定义控件函数 136
附录 2、JavaScript 常用函数 144
版权所有，未经许可不得复制
..... 146
网盘下载地址：http://pan.baidu.com/s/1dDuHMa1 ，密码：65h4
..... 146

第 1 章、jQuery 及其开发环境

jQuery 是继 prototype 之后又一个优秀的 Javascript 库。它是轻量级的 JS 库 ，兼容 CSS3 和各种浏览器（IE 6.0+, FF 1.5+, Safari 2.0+, Opera 9.0+），jQuery2.0 及后续版本将不再支持 IE6/7/8 浏览器。jQuery 能更方便地处理 HTML 和为网站提供 AJAX 交互。jQuery 另一个比较大的优势是其文档说明齐全，同时还有许多成熟的插件可供选择。

jQuery 的核心理念是 write less、do more（写得更少，做得更多），最早在 2006 年 1 月由美国人 John Resig 发布。如今，jQuery 已经成为最流行的 javascript 库，在世界前 10000 个访问最多的网站中，有超过 55%在使用 jQuery。

jQuery 是免费、开源的，使用 MIT 许可协议。jQuery 的语法设计可以使开发者更加便捷，例如操作文档对象、选择 DOM 元素、制作动画效果、事件处理、使用 Ajax 以及其他功能。除此以外，jQuery 提供 API 让开发者编写插件。其模块化的使用方式使开发者可以很轻松的开发出功能强大的静态或动态网页。

jQuery，顾名思义，也就是 JavaScript 和查询（Query），即辅助 JavaScript 开发的库。

1.1 J2EE 开发环境简介

- 服务器：Tomcat6.0 及以上。
- 数据库：SQL Server2008。
- 程序开发环境：MyEclipse6.5 及以上。
- 程序设计语言：客户端 Javascript（JS）、服务器端 JSP、JAVA。
- jQuery 插件库：EasyUI 等。

1.2 安装 Tomcat 服务器

1.2.1 Tomcat6.0 及安装步骤

将文件夹 apache-tomcat-6.0.30 复制到 C 盘根目录下，即可完成 Tomcat6.0 的安装过程，这时 C:\ apache-tomcat-6.0.30 文件夹下包括图 1-1 所示内容：

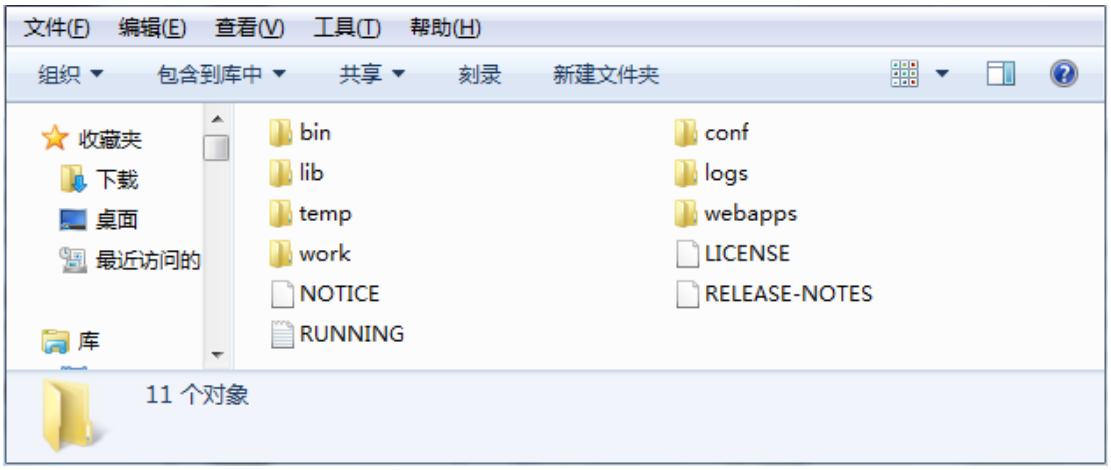


图 1-1　Tommcat6.0 文件目录

1.2.2 安装 JDK

打开 C:\Program Files\Java 这个文件夹，如果存在 jdk*和 jre*两个子文件夹，表明 Windows 系统已经安装了 JDK，如果不存在 C:\Program Files\Java 及其子文件夹，则需要安装 JDK。对没有安装 JDK 的操作系统，下面以 jdk-7 为例，阐述 JDK 安装过程。

①点击安装程序，开始安装过程。选择 JDK 默认安装路径（如图 1-2 所示），不作更改。

图 1-2　JDK 安装过程（选择默认安装路径）

②同样安装 JRE 时选择默认安装路径，即与 JDK 安装在同一个 java 文件夹下（如图 1-3 所示）。安装成功后关闭窗体。

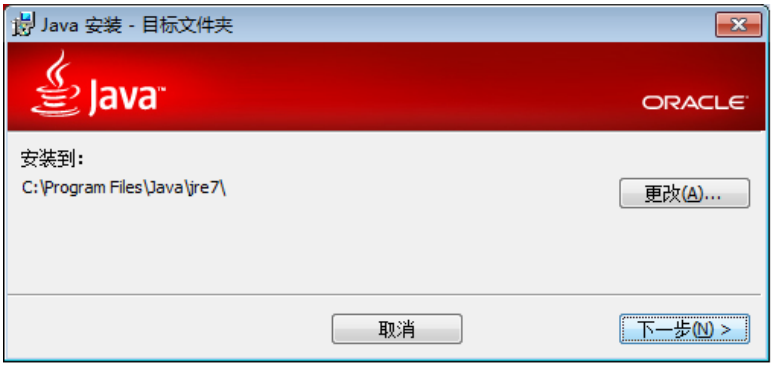


图 1-3　JRE 安装过程（选择默认安装路径）

图 1-4　JDK 安装结束界面

1.2.3 设置 JDK 环境变量

以 Win7 为例（如果已经安装 JDK，直接设置环境变量）：

①右击“我的电脑”，选择“属性”，选择“高级系统设置”，这时出现图 1-5 界面。

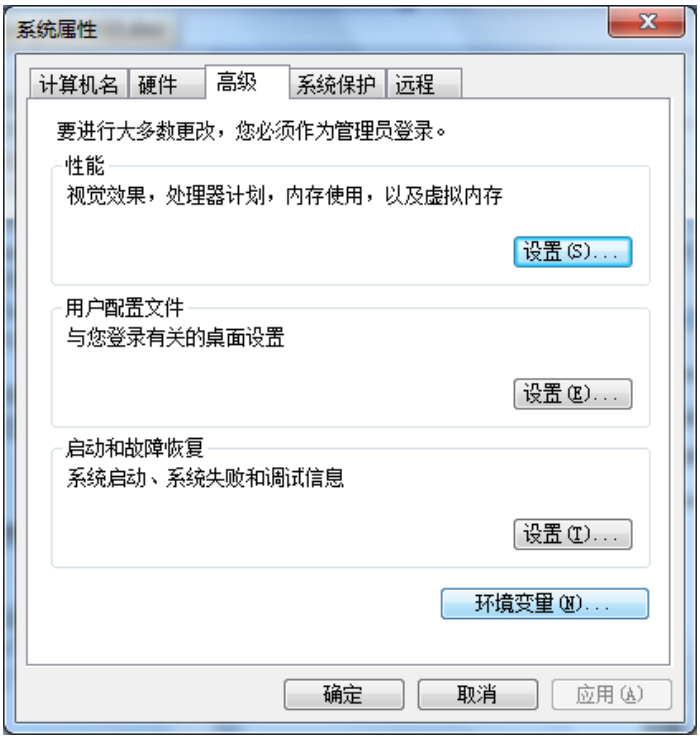


图 1-5　JDK 环境变量配置（选择高级系统设置）

②点击上图中的“环境变量”按钮，在“系统变量”中新建 JAVA_HOME 变量。变量值填写 JDK 的安装目录（例如 C:\Program Files\Java\jdk1.7.0_4⑤。

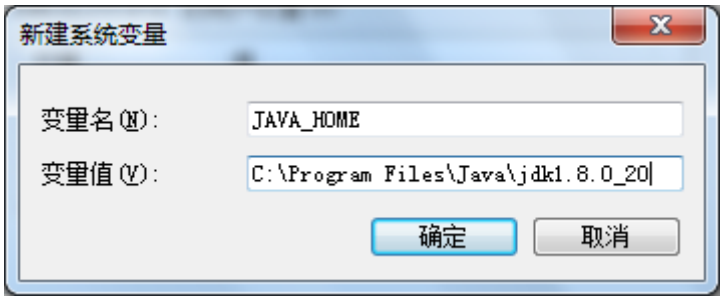


图 1-6　JDK 环境变量配置（新建 JAVA_HOME 变量）

③系统变量中找到 Path 变量，点击“编辑”（如图 1-6 所示）。在变量值这一栏中输入：

%JAVA_HOME%\bin;%JAVA_HOME%\jre\bin;

注意：最后一个分号必须输入。

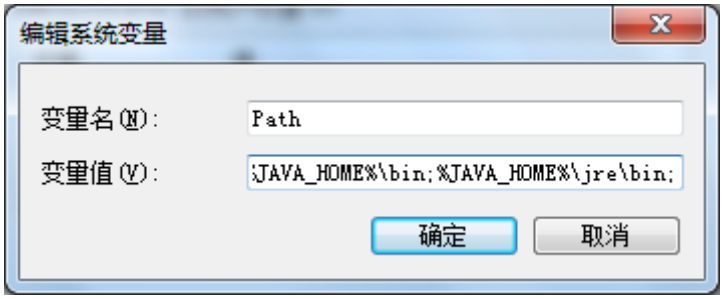


图 1-6　JDK 环境变量配置（编辑 PATH 变量）

④在“系统变量”中新建 CLASSPATH 变量（如图 1-7 所示）。变量值填写如下：

.;%JAVA_HOME%\lib;%JAVA_HOME%\lib\tools.jar

注意：最前面有一个点。系统变量配置完毕，按“确定”按钮退出。

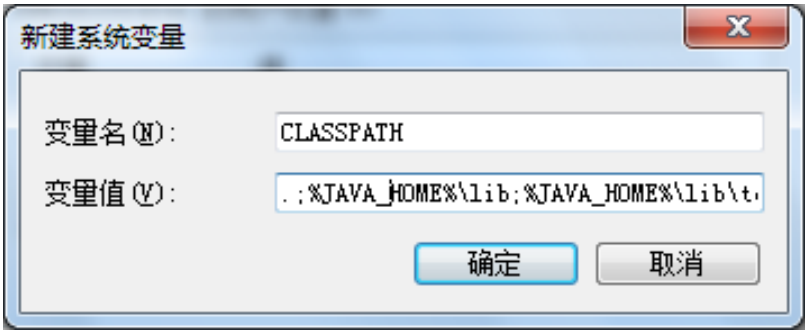


图 1-7　JDK 环境变量配置（新建 CLASSPATH 变量）

⑤检验是否配置成功。运行 cmd（或在命令行中）输入 java -version（注意：java 和 -version 之间有空格）。若出现图 1-8 所示结果，显示 JDK 版本信息，则说明 JDK 安装和配置成功。

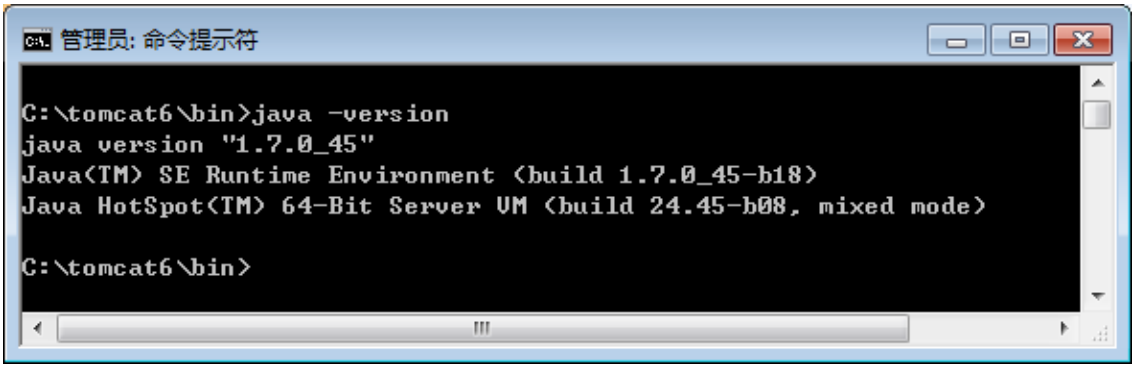


图 1-8　JDK 环境变量配置成功

⑥重启计算机，JDK 环境变量设置方可生效。

1.2.4 Tomcat6.0 的启动和停止

①打开 C:\ apache-tomcat-6.0.30\bin 文件夹，点击其中的 startup.bat 文件，即可启动 Tomcat6.0 服务器，这时屏幕上出现图 1-9 所示信息，表明 Tomcat 服务器已经开启并将一直保持运行状态。

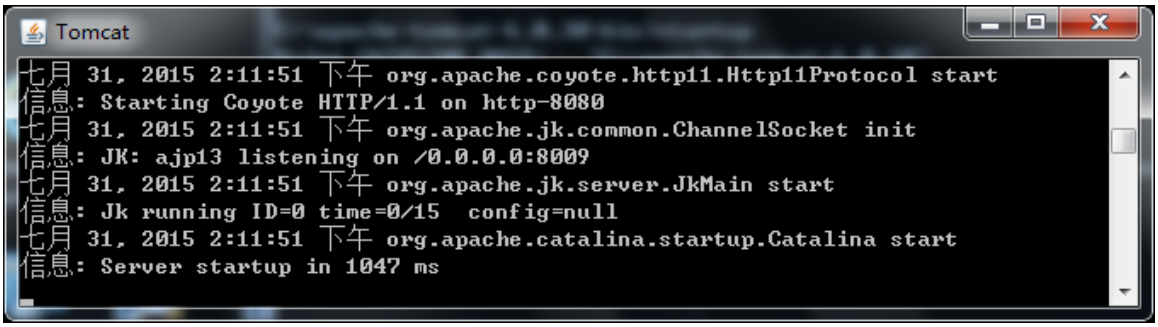


图 1-9　Tomcat6.0 启动运行界面

②也可以通过打开一个游览器（建议使用 Google 的 Chrome 游览器），在游览器地址栏输入下列地址：http://127.0.0.1:8080。如果出现图 1-10 所示页面，则表明 Tomcat 安装成功并已启动。

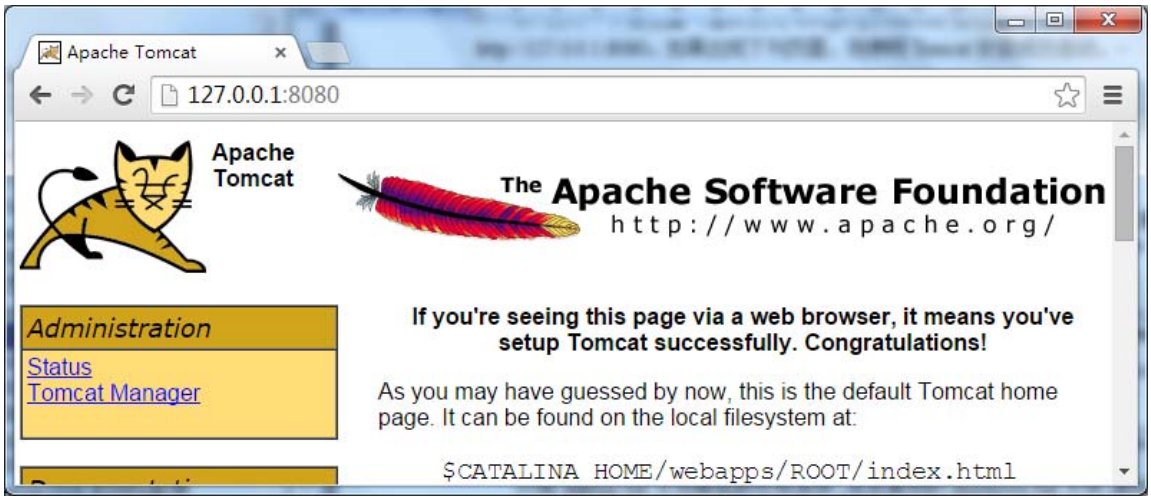


图 1-10　Tomcat 配置成功启动页面

③在 startup.bat 所在的同一文件夹中，点击其中的 shutdown.bat 文件，即可停止和关闭 Tomcat 服务器。注意：在安装 MyEclipse 之后，通常在 MyEclipse 开发环境中开启和关闭 Tomcat 服务器。这时需要按照上一步的操作方法，关闭 Tomcat 服务器，否则在 MyEclipse 无法开启服务器。

1.2.5 设置 Tomcat 端口号

在启动 Tomcat 的过程中，有些系统可能会出现错误，其中一种可能是端口号 8080 与其他应用程序发生冲突，这时需要修改或重新设置端口号。重新设置端口号的方法如下：

①在 C:\apache-tomcat-6.0.30\conf\文件夹中，用记事本打开编辑 server.xml 配置文件；

②在该文件中搜索 8080 字符串（如图 1-11 所示），将下列行中的 port="8080"改成 port="8088"：

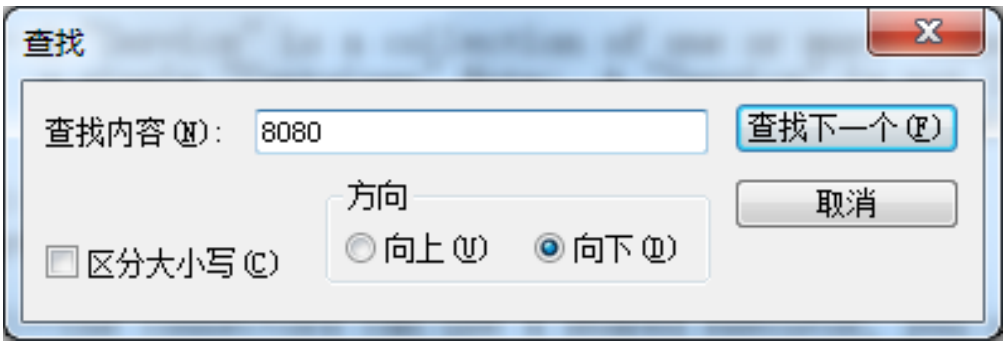


图 1-11　Server.xml 配置文件查找 8080 端口

<Connector port="8080" protocol="HTTP/1.1" connectionTimeout="20000" redirectPort="8443" /> 改为：
<Connector port="8088" protocol="HTTP/1.1" connectionTimeout="20000" redirectPort="8443" />
<Connector executor="tomcatThreadPool" port="8080" protocol="HTTP/1.1" connectionTimeout="20000" redirectPort="8443" />　改为：
<Connector executor="tomcatThreadPool" port="8088" protocol="HTTP/1.1" connectionTimeout="20000" redirectPort="8443" />

③保存 server.xml 文件后退出，重新启动 Tomcat，然后在游览器中输入下列地址：http://127.0.0.1:8088

1.3 安装 MyEclipse

这里以 MyEclipse6.5 为例，描述安装过程。

①点击打开安装文件，开始安装过程（如图 1-12 所示）。

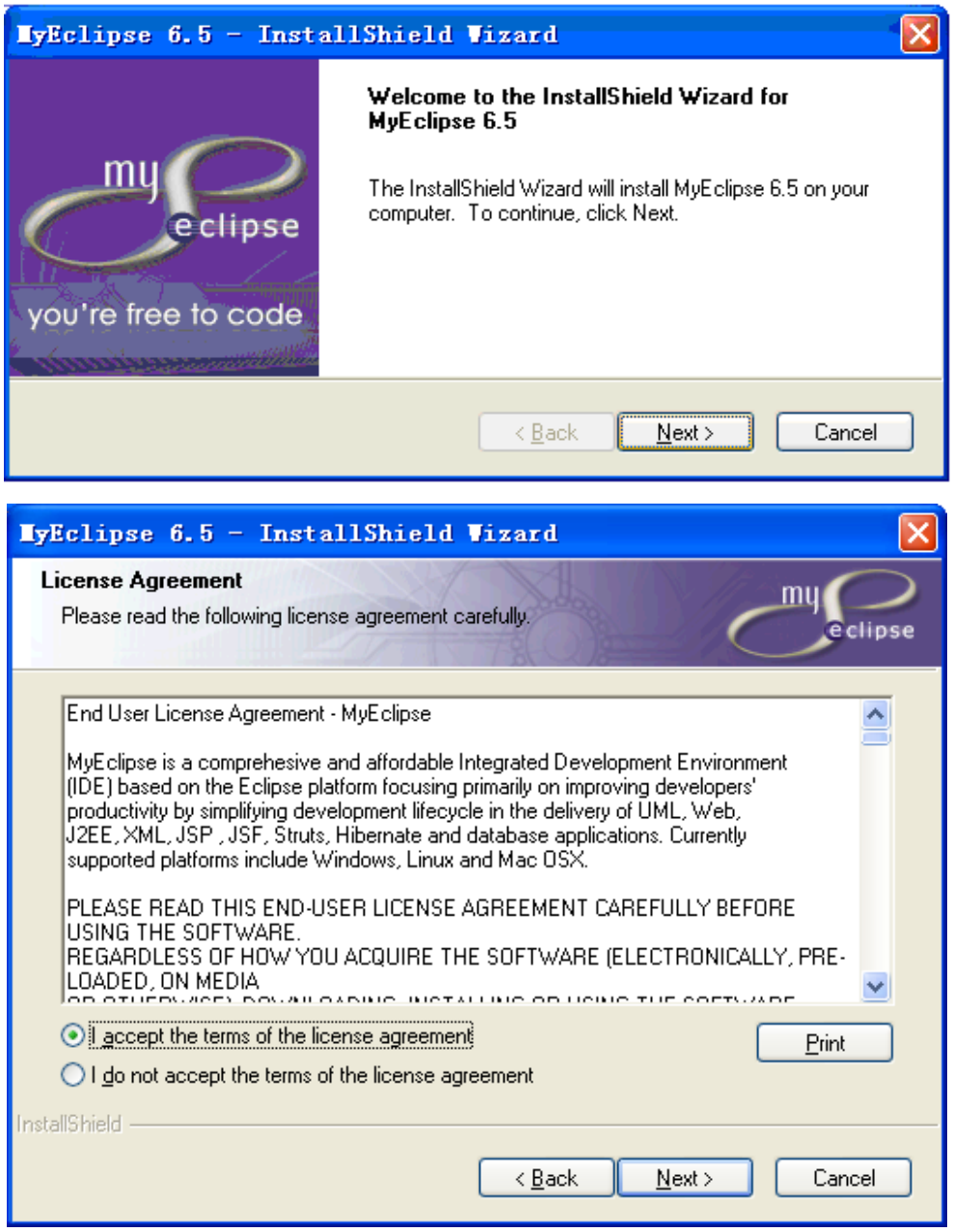


图 1-12　运行 myEclipse6.5 安装程序

②选择默认路径安装（如图 1-13 所示）。

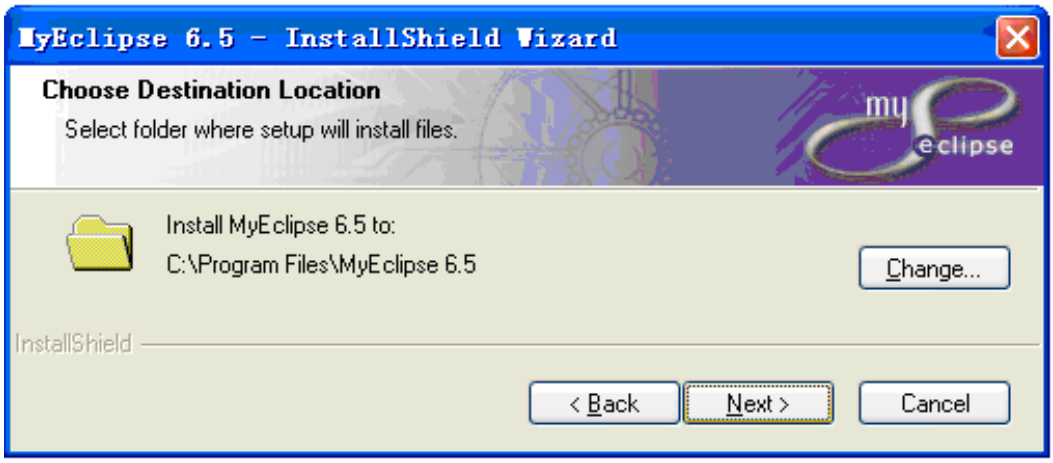


图 1-13　选择默认路径安装 myEclipse6.5

③点击“Install”正式开始安装（如图 1-14 所示）。

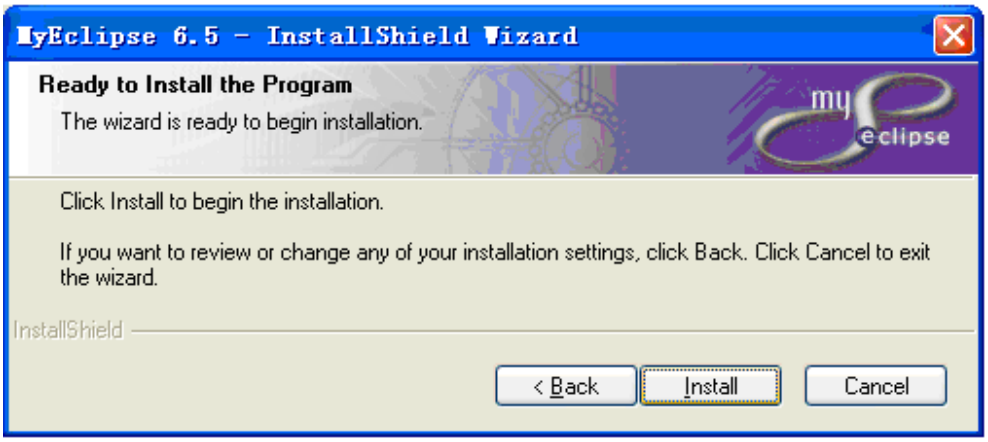


图 1-14　正式开始 myEclipse6.5 安装过程

④正在安装，请稍候（如图 1-15 所示）。

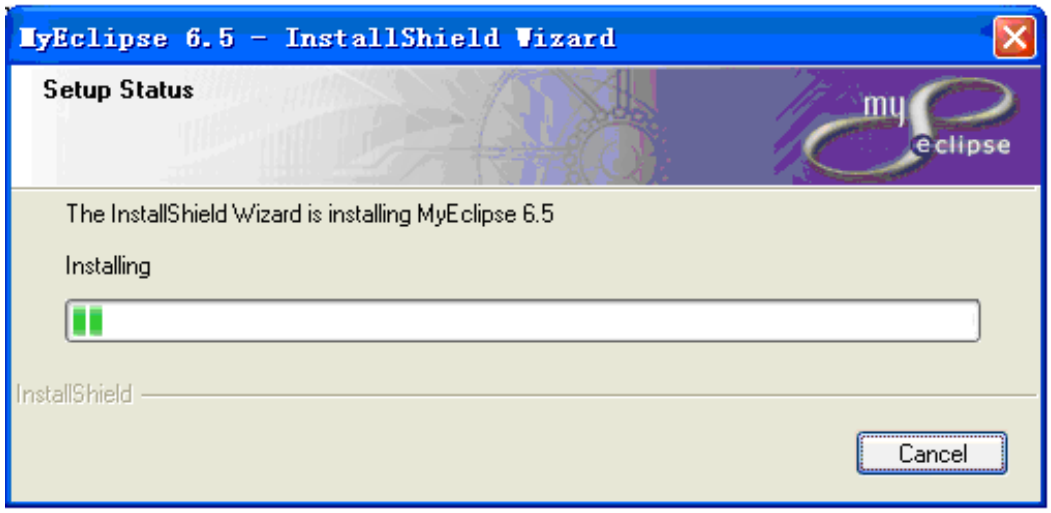


图 1-15　myEclipse6.5 安装过程进行中

⑤安装完毕后，选择“Launch MyEclipse6.5”，关闭“Open the release notes”，这时安装程序将打开和启动 MyEclipse（如图 1-16 所示）。

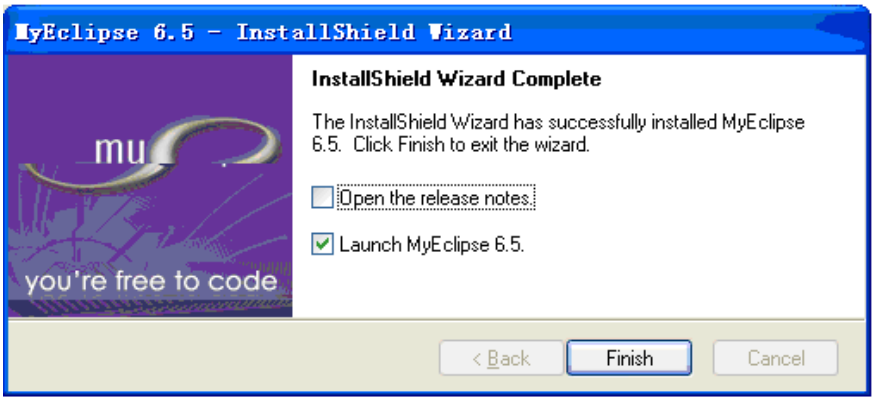


图 1-16　myEclipse6.5 安装成功结束

⑥启动进入 MyEclipse，需要几分钟（如图 1-17 所示，有点慢哦）。

图 1-17　启动打开 myEclipse6.5 程序

⑦输入 Workspace 工作目录（例如 d:\myjsp，如图 1-18 所示）；在后面复选框中（Use this as the default and do not ask again）打上勾，然后点击 OK 按钮。

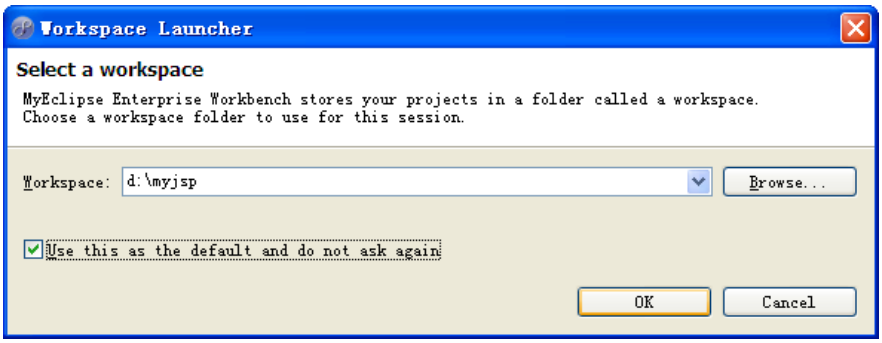


图 1-18　配置 myEclipse6.5 的工作文件目录

⑧这时出现 MyEclipse 编辑界面。稍后在屏幕会跳出一个子窗体，要求输入注册码。

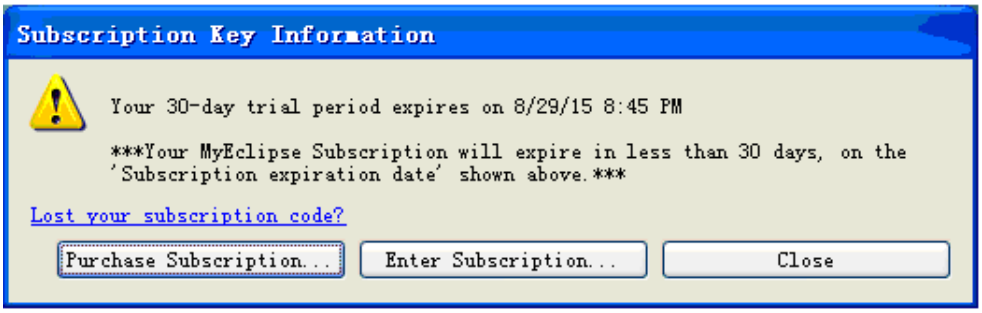


图 1-19　提示输入 myEclipse6.5 注册码和序列号

⑨点击“Enter Subscription …”，输入用户名和注册码如下（如图 1-20 所示）。

zxywolf

mLR8ZC-855355-6354665292178100

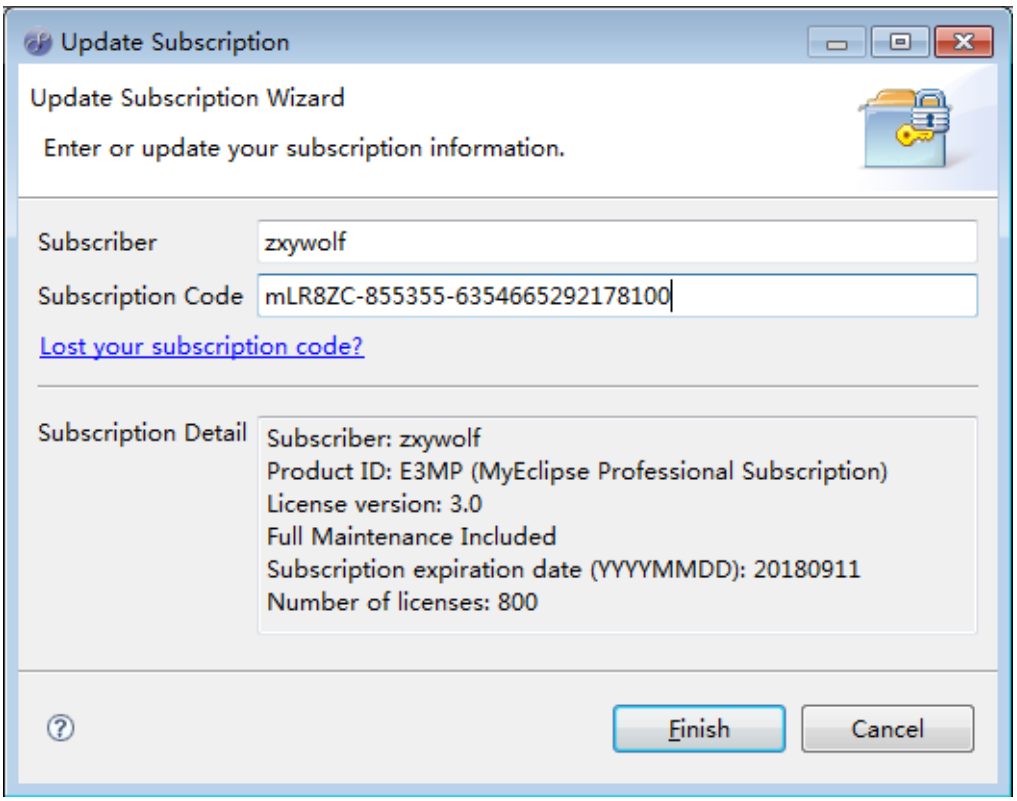


图 1-20　输入 myEclipse6.5 注册码和序列号

⑩如果需要重新注册，可在工具栏中点击 Window+Preferences 菜单，在菜单树中找到 MyEclipse Enterpise Workbence 之下的 Subscription 菜单（如图 1-22 所示），点击该菜单后在右侧屏幕中输入注册码。

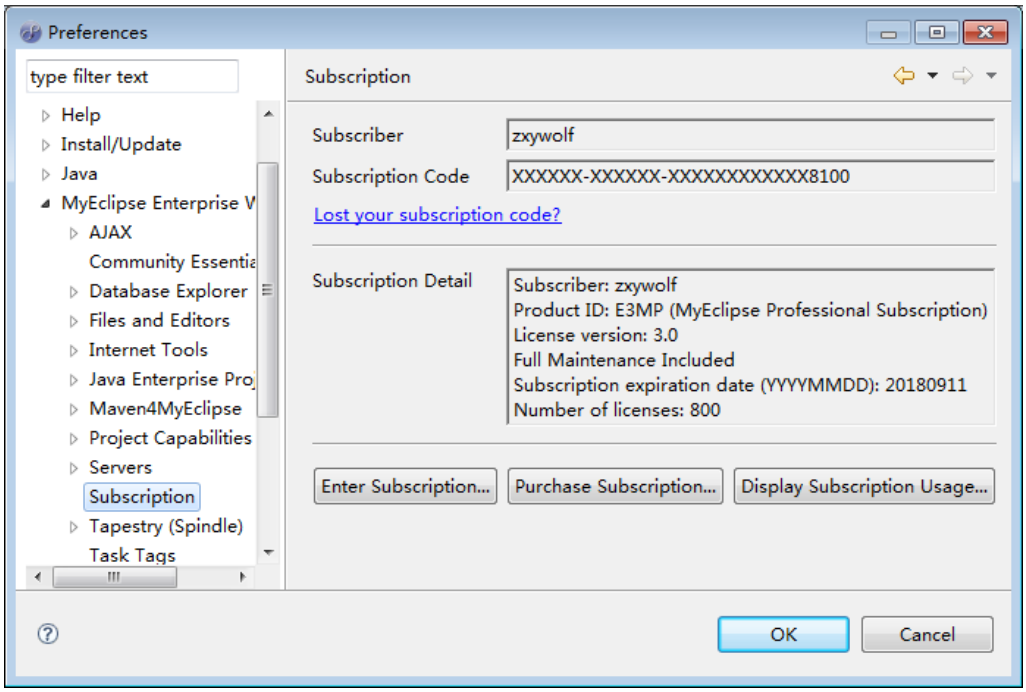


图 1-22　修改 myEclipse6.5 注册码和序列号

①关闭 MyEclipse 的 Welcome 栏目，出现下列 MyEclipse 工作环境，至此 MyEclipse 安装结束（如图 1-23 所示）。

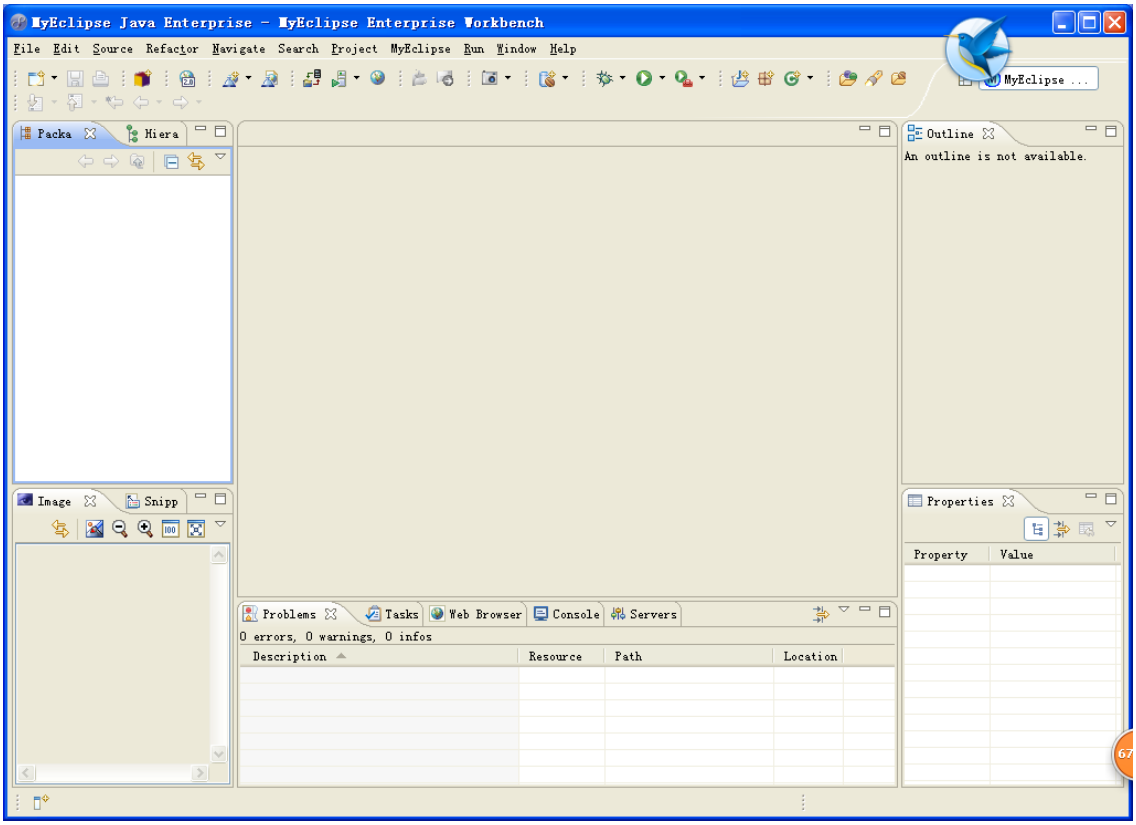


图 1-23　myEclipse6.5 程序设计与调试环境

1.4 导入和建立工程文件 jQDemos

①将 jQDemos.rar 文件复制到 d:\myjsp 文件夹中，这个文件夹中一开始时只有一个子文件夹和这个压缩文件（如图 1-24 所示）。

图 1-24　jQDemos 工程文件解压

②按右键将 jQDemos.rar 解压到当前文件夹（即 d:\myjsp 文件夹）下（如图 1-25 所示）。

图 1-25　jQDemos 工程文件解压后的文件夹

③返回到 MyEclipse 程序设计环境（如图 1-26 所示），点击工具栏中的 File+Import 菜单，导入工程文件夹 jQDemos。

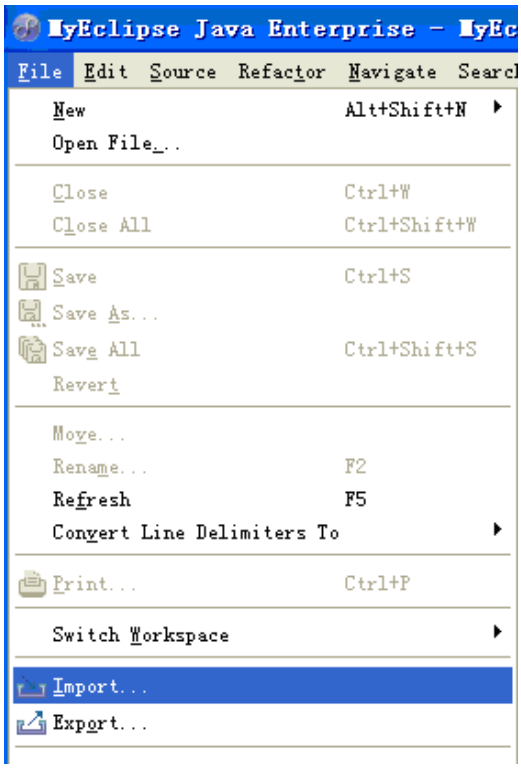


图 1-26　选择导入 jQDemos 工程

④在下图中选择 Existing Projects into Workspace 菜单，点击“Next”按钮（如图 1-27 所示）。

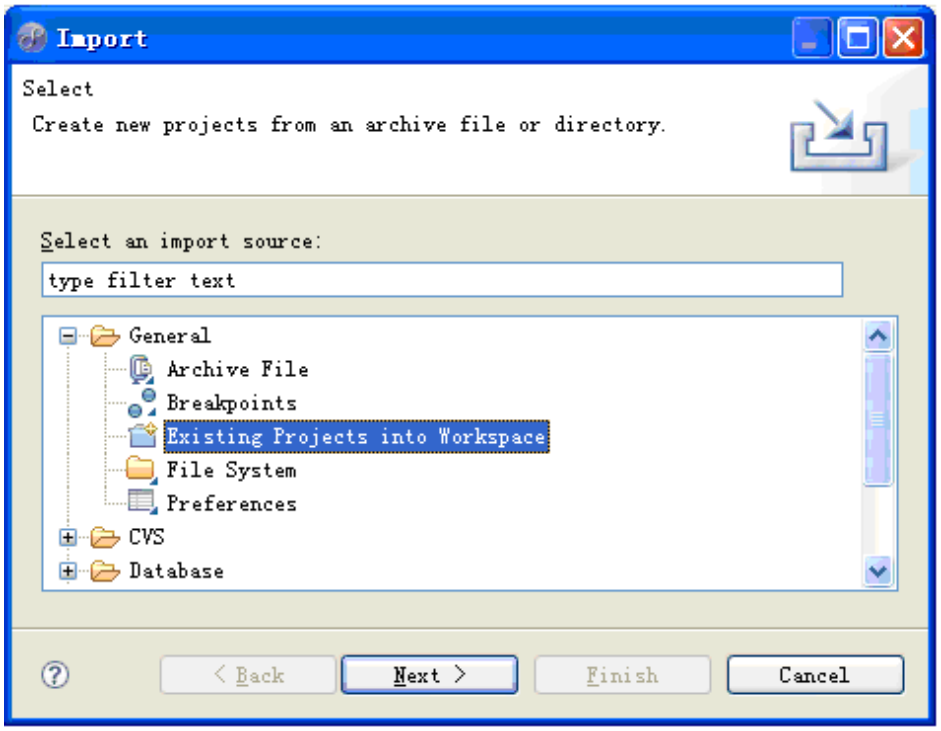
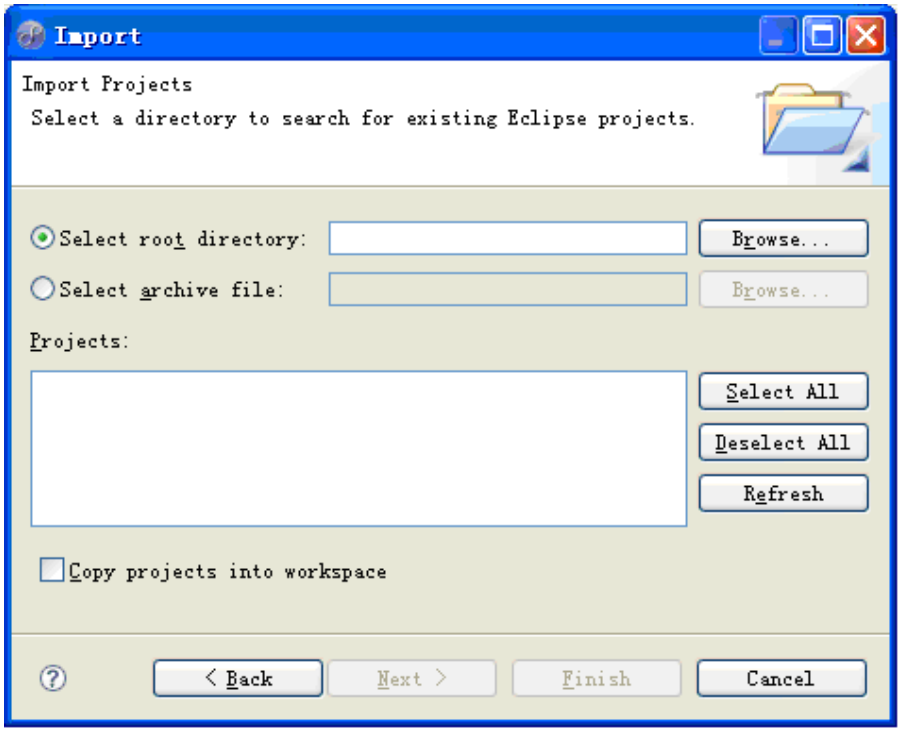


图 1-27　选择从现有的工程中导入 jQDemos 到工作文件夹

④点击下图中的 Browse 按钮，在弹出的子窗体中选择 d:\myjsp\jQDemos 文件夹（如图 1-28 所示，注意不要选择后面的子文件夹）



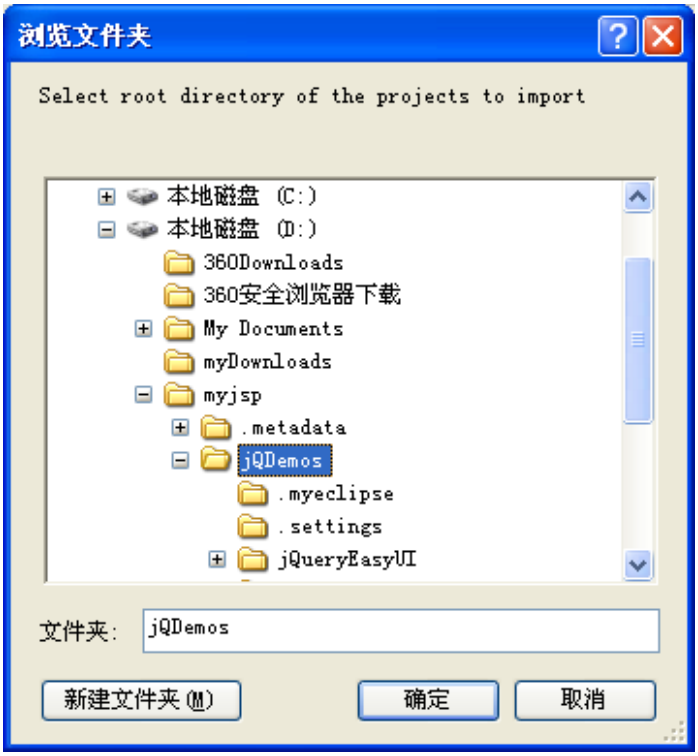


图 1-28　导入 jQDemos 工程时选择文件夹为 jQDemos

⑤点击上图 1-28 中的“确定”按钮，出现图 1-29 所示界面。在图 1-29 的 Copy projects into workspace 中打上钩；点击“Finish”按钮。

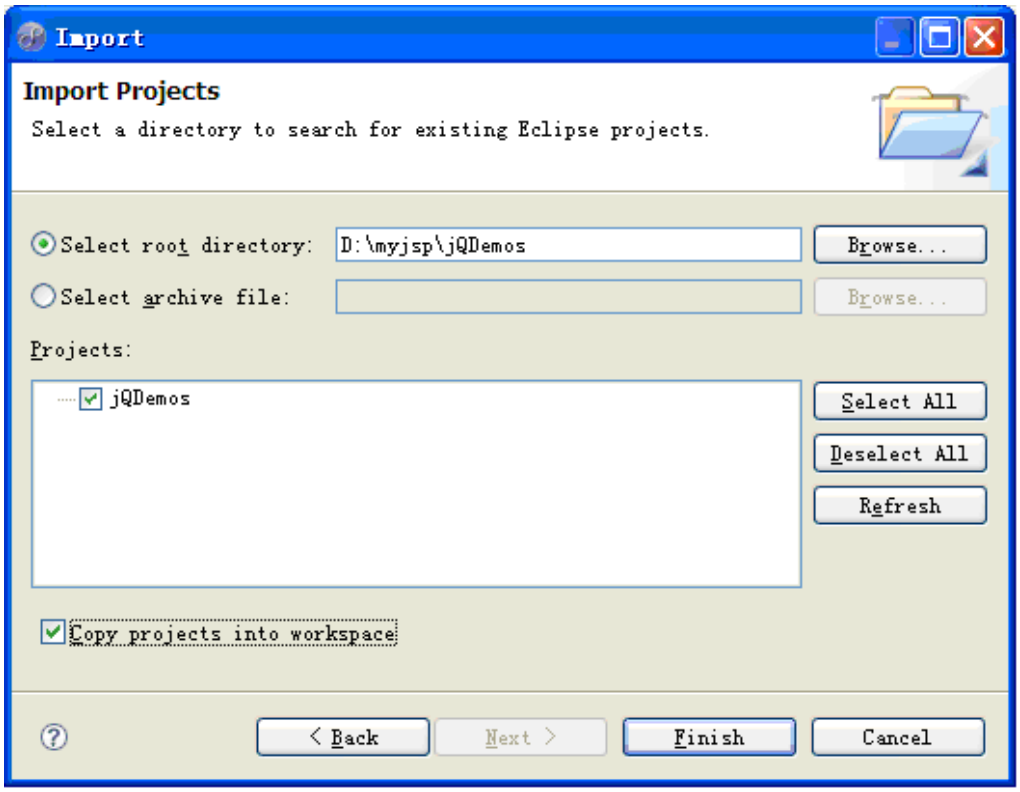


图 1-29　导入 jQDemos 工程结束

⑥这时在 MyEclipse 中出现下列界面（如图 1-30 所示），即已经存在一个工程文件 jQDemos。

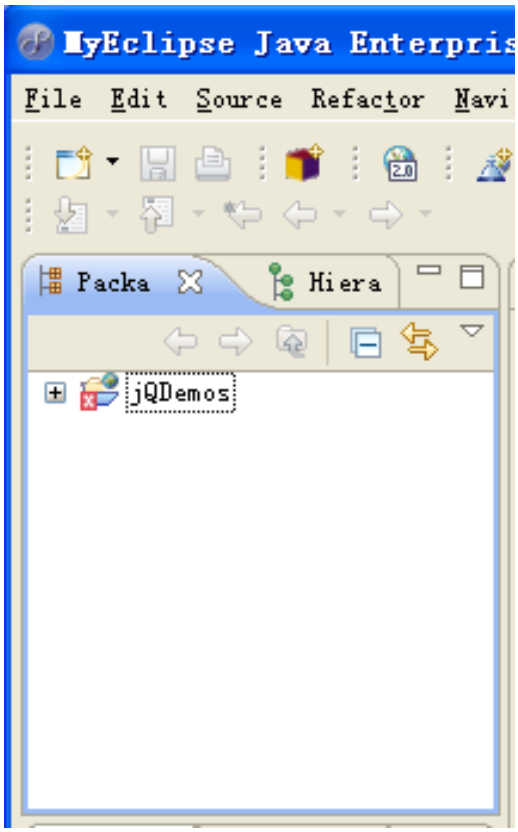


图 1-30　导入 jQDemos 工程成功

1.5 添加和配置 MyEclipse 中的 Tomcat6.x 服务器

①在 MyEclipse 中点击 Window+Preferences 菜单（如图 1-31 所示）。

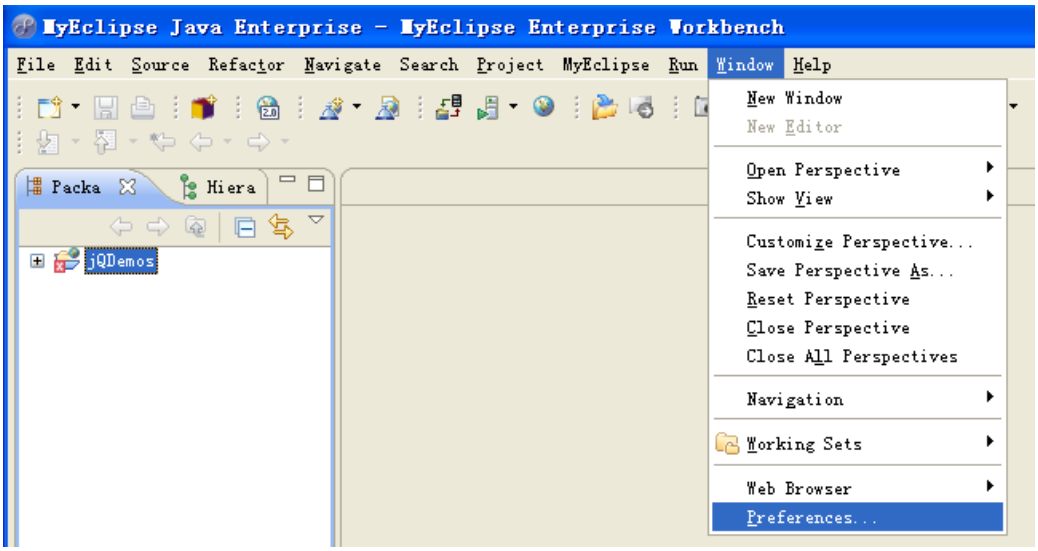


图 1-31　在 myEclipse6.5 中配置 Tomcat6.0

②在图 1-32 所示的左侧菜单树中输入框“type filter text”中输入“tomcat”这个字符进行过滤；在菜单树选中 Tomcat 6.x；点击右侧的第一个 Browse 按钮，选中 Tomcat6.0 所在的文件夹位置（不要点击到子文件夹哦），即 C:\apache-tomcat-6.0.30。点击“确定”按钮。

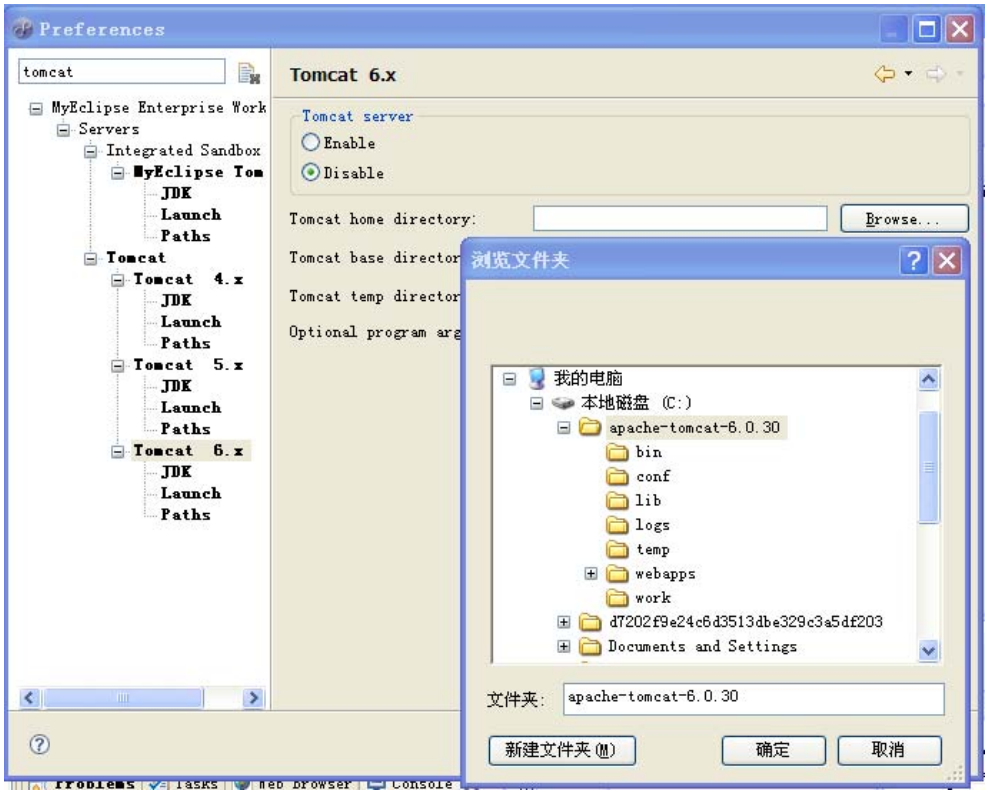


图 1-32　选择 Tomcat6.x 所在的文件夹

③这时屏幕出现图 1-33 所示界面。点中右上侧的无线按钮“Enable”，再点击 OK 后，屏幕关闭，返回到 MyEclipse 界面，这时 Tomcat 配置结束。

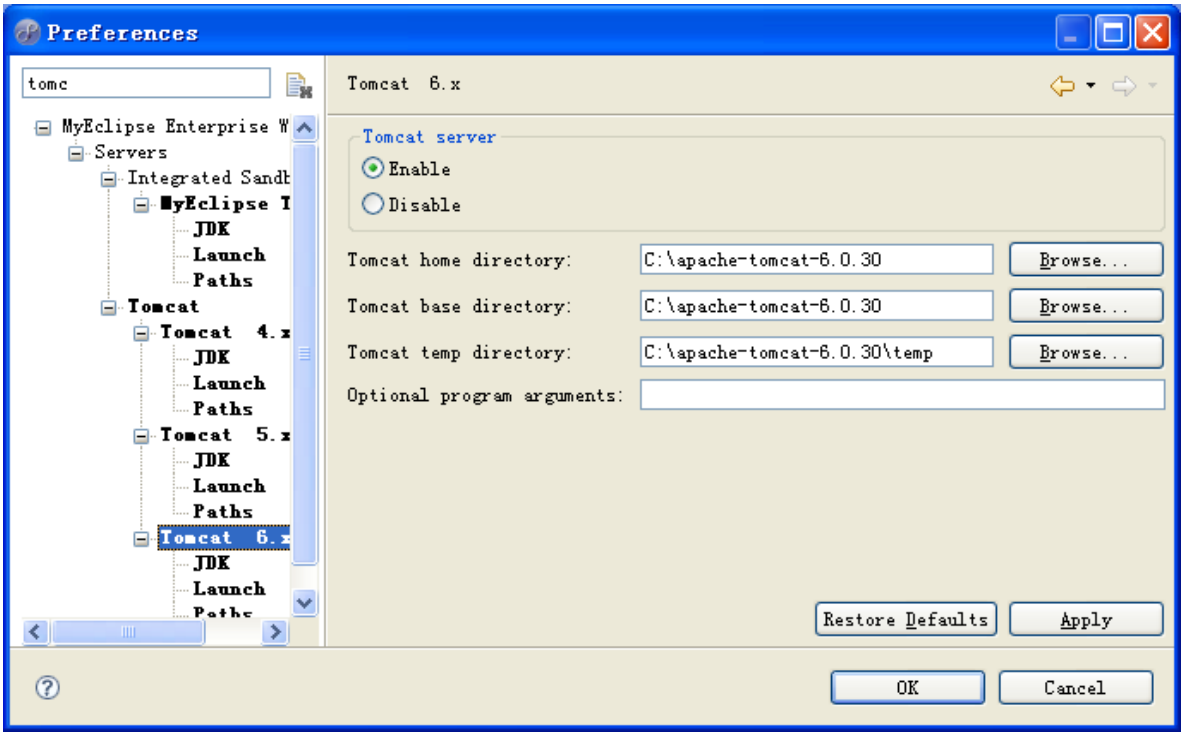


图 1-33　选中 Tomcat6.0 文件夹并激活

④开启 myEclipse 中的 Tomcat6.x 服务器。先使用 shutdown.bat 关闭命令行中的 Tomcat 服务器（方法见 1.2.4），点击 MyEclipse 工具栏中的“Start”小按钮，选择 Tomcat 6.x+Start 菜单（如图 1-34 所示）。

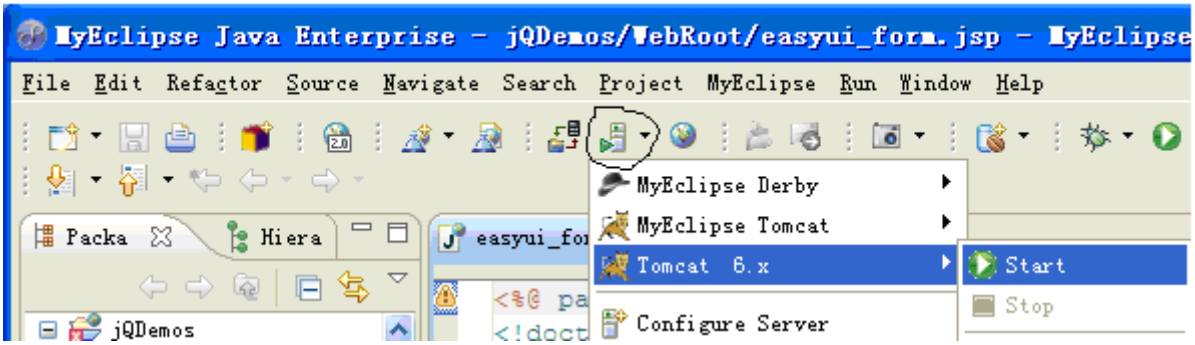


图 1-34　在 myEclipse6.5 中启动 Tomcat6.0

这时屏幕下方出现下列信息（如图 1-35 所示），至此 Tomcat 服务器在 MyEclipse 中配置结束。

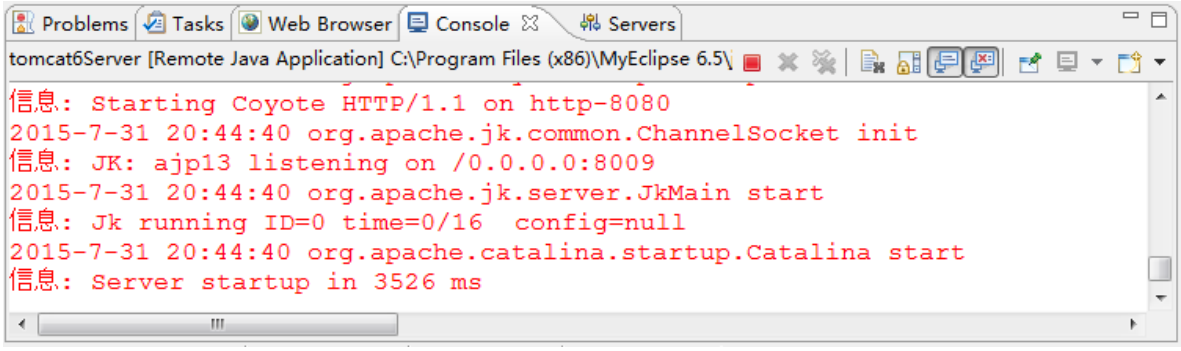


图 1-35　myEclipse6.5 中启动 Tomcat6.0 成功提示信息

1.6 部署 jQDemos 工程文件

①点击 MyEclipse 窗体工具栏中的一个“部署”（Deploy）小按钮（如图 1-36 所示）。

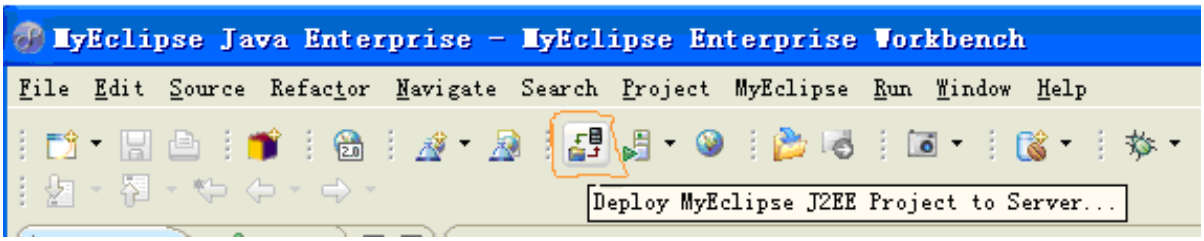


图 1-36　myEclipse6.5 中部署 jQDemos 工程

②这时屏幕出现图 1-37 界面。在 Project 组合框中选中 jQDemos（其实这时只有这个选项）。

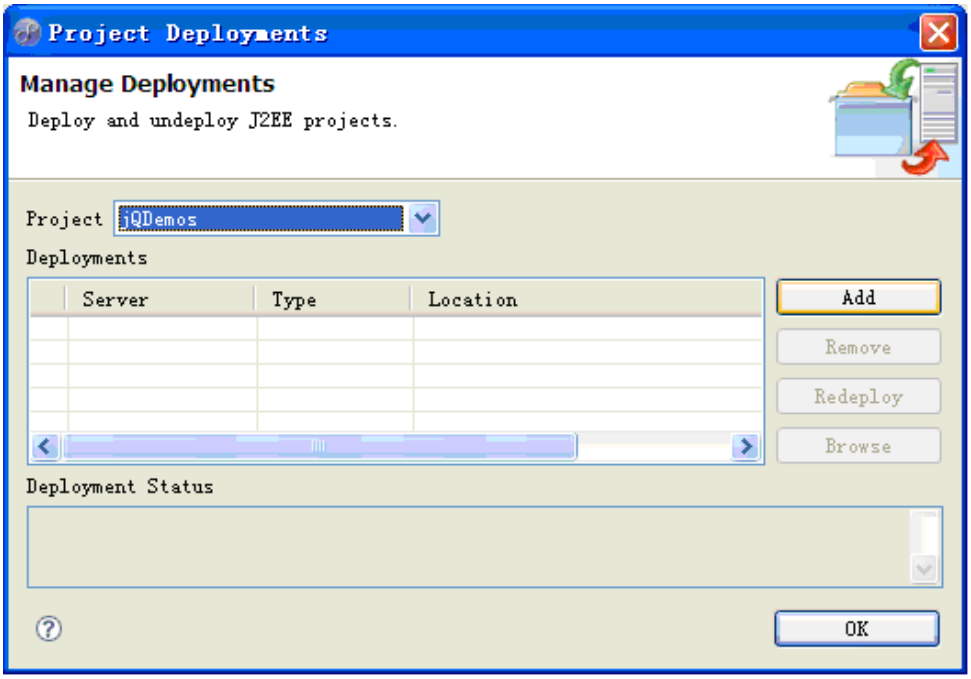


图 1-37　myEclipse6.5 中将 jQDemos 工程部署到 Tomcat6.0 中去

③点击上图 1-37 右侧的“Add”按钮，出现图 1-38 界面。在 Server 组合框中选中“Tomcat 6.x”。

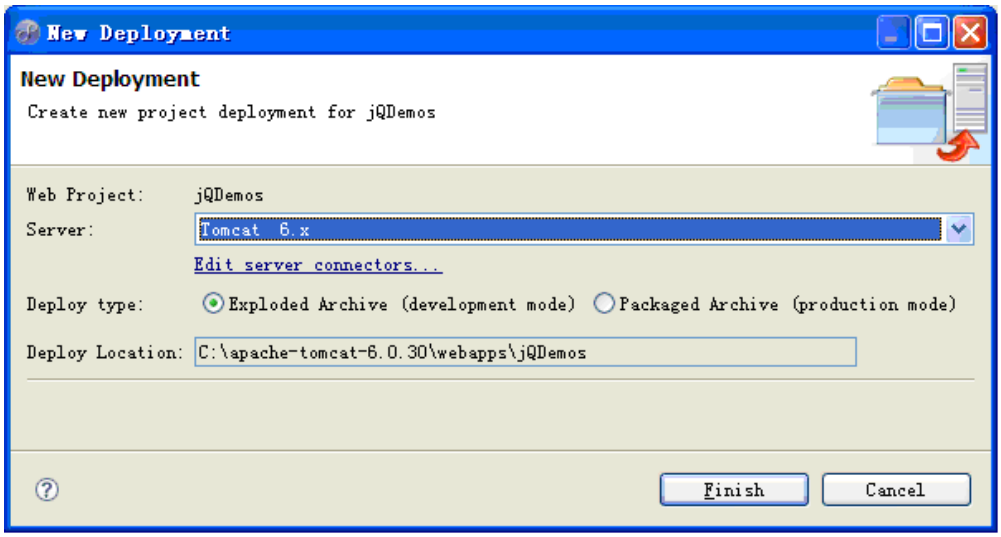


图 1-38　myEclipse6.5 中选择 Tomcat6.0 服务器

④点击图 1-38 中的“Finish”按钮，这时出现部署工程项目文件的过程（如图 1-39 所示）。

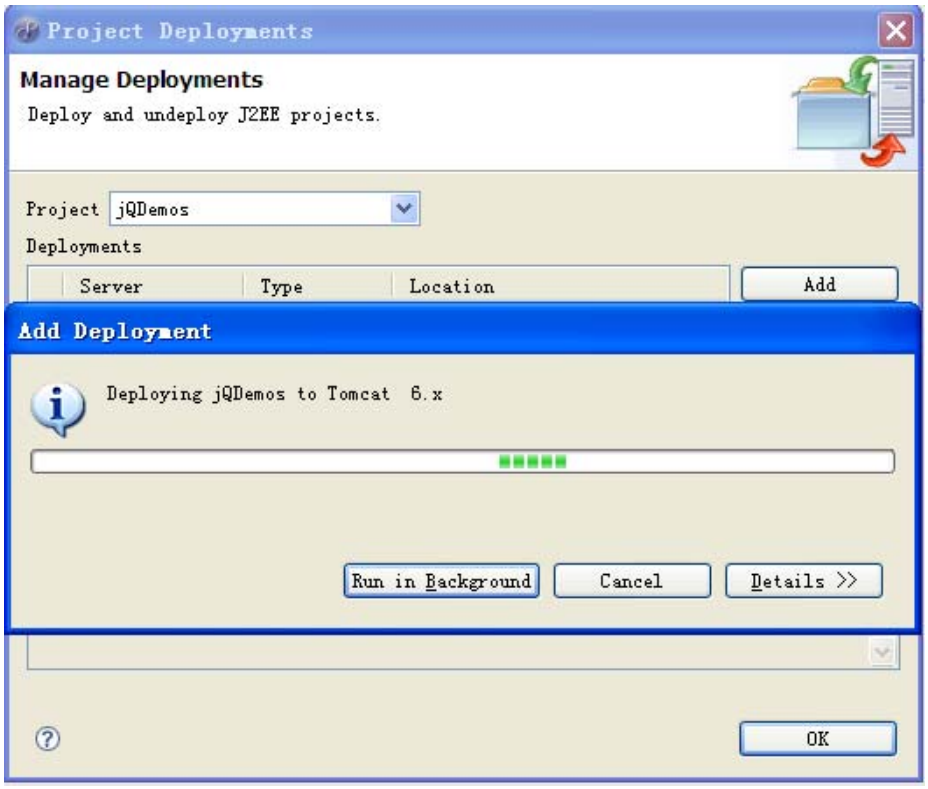


图 1-39　myEclipse6.5 提示正在部署 jQDemos 工程到 Tomcat6.0

⑤部署完成后，出现图 1-40 所示界面。此时，点击“OK”按钮，关闭部署过程。第二次部署某个工程时，可以点击“Redeploy”（重新部署）按钮。

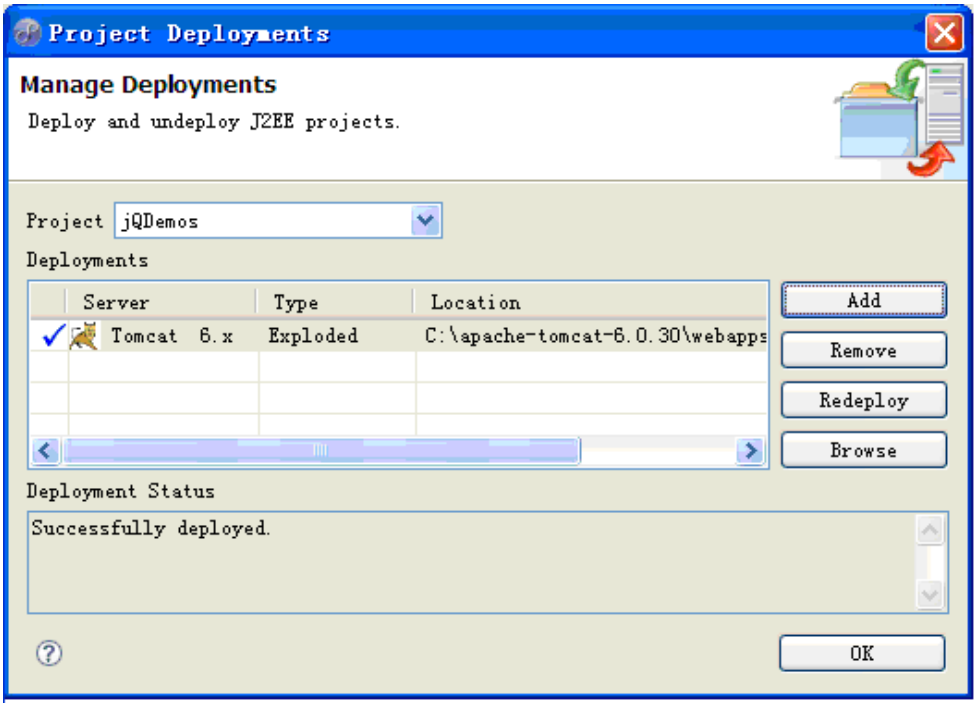


图 1-40　myEclipse6.5 中部署 jQDemos 工程成功

⑥程序编辑。打开 jQDemos 文件夹+WebRoot，找到 example01_basicform.jsp 程序。这时可能出下列屏幕（如图 1-41 所示），暂时无法看到该程序的源码。

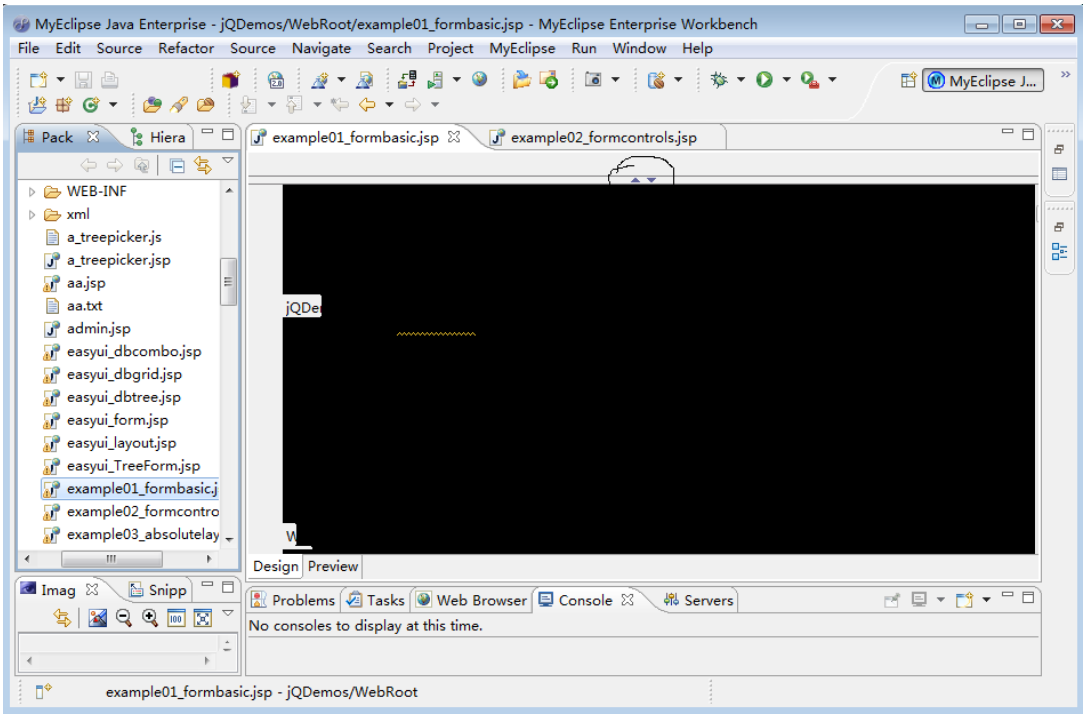


图 1-41　myEclipse6.5 中打开 jQDemos 工程文件

点击图 1-41 中上面的小图标按钮或关闭 example01_basicform.jsp 程序再打开，这时屏幕窗体正常，出现源程序代码如下（如图 1-42 所示）：

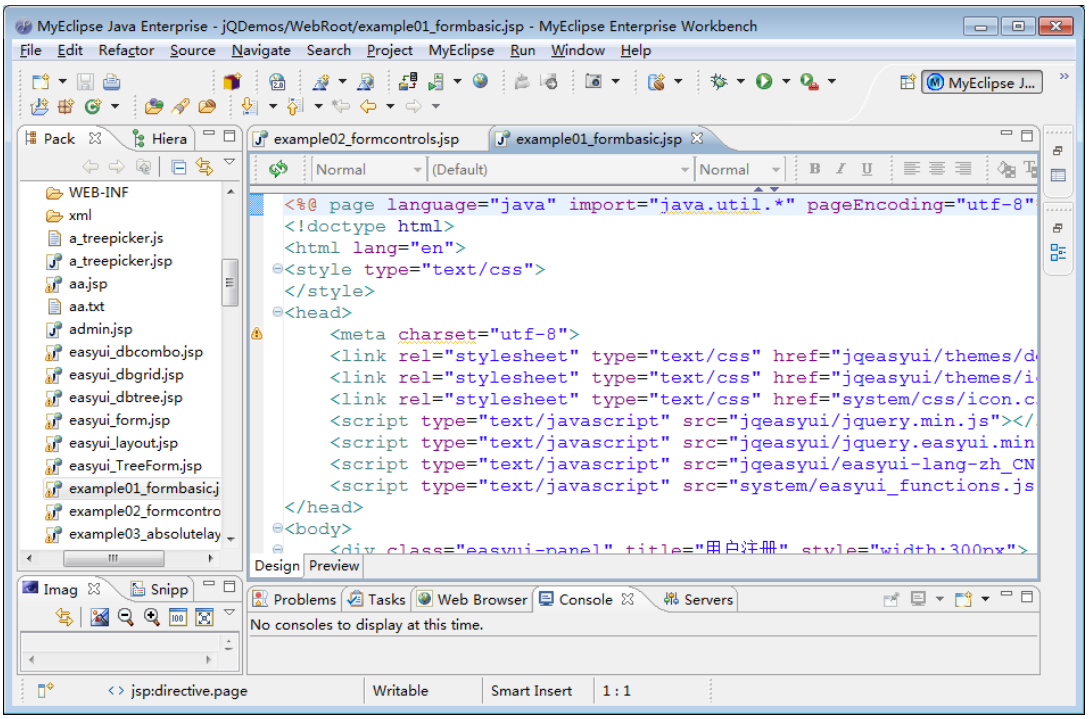


图 1-42　myEclipse6.5 中编辑 jQDemos 工程文件

⑦安装和打开 Google 的 Chrome 游览器；在游览器中输入下列地址：http://127.0.0.1:8080/jQDemos/example02_formcontrols.jsp.jsp。这时出现图 1-43 所示的程序运行界面页面，至此 J2EE 环境安装成功。

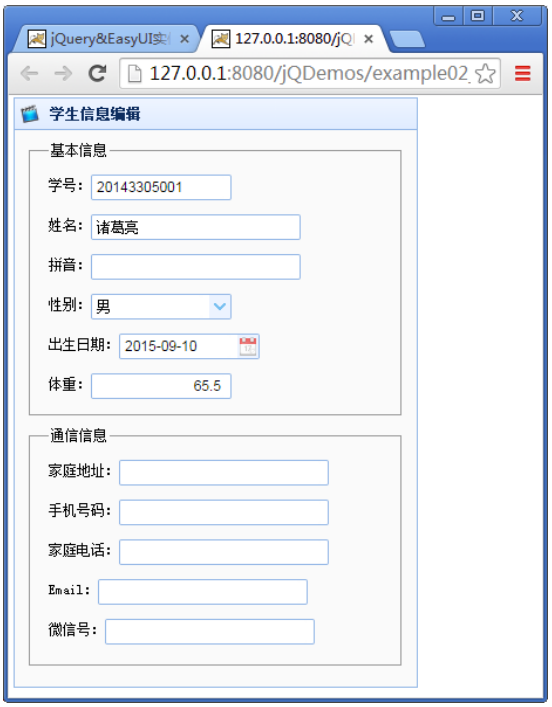


图 1-43　在 Chrome 游览器中成功运行 jQDemos 页面

1.7 SQL Server2008 的网络配置

①执行 Microsoft SQL Sever2008 菜单下的配置工具中找到“SQL Server 配置管理器”（如图 1-44 所示），点击该菜单。

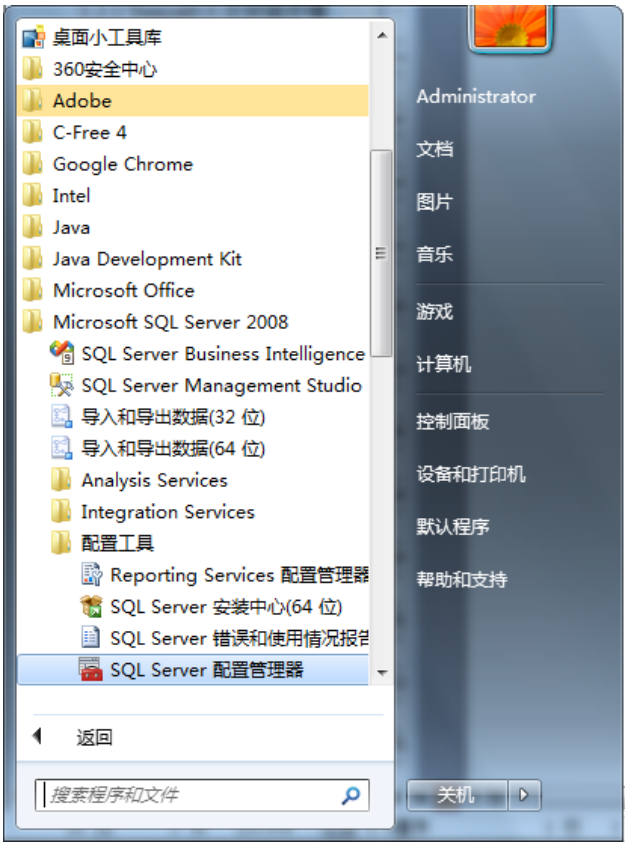


图 1-44　找到 SQLServer 数据库配置管理器

②展开菜单树，选中“MSSQLSERVER 的协议”（位于图 1-46 左侧），点击图 1-45 右侧的“TCP/IP”选项。

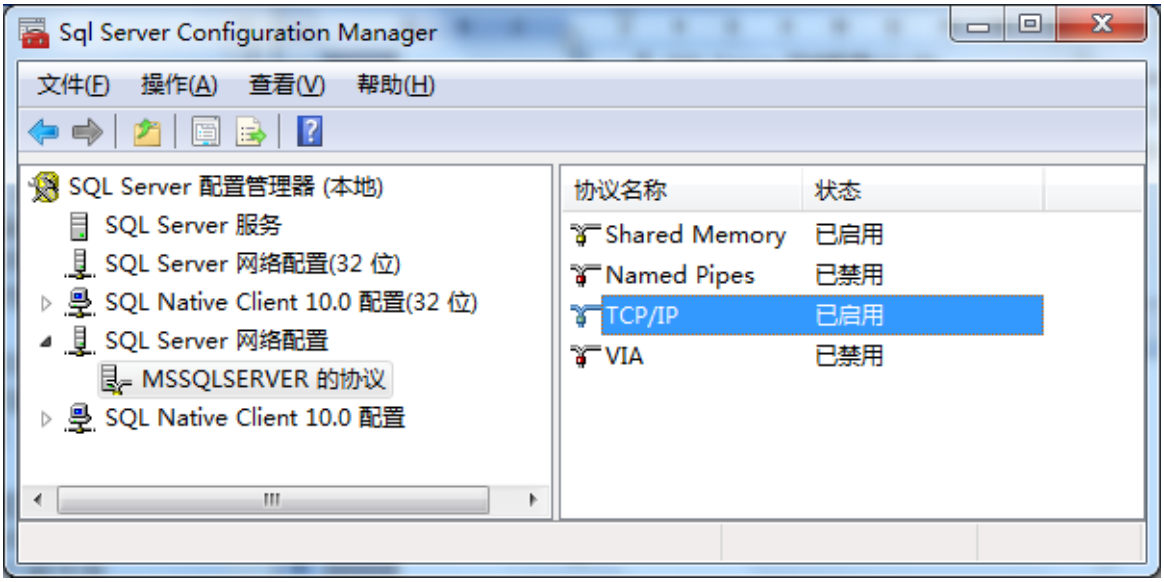


图 1-45　配置 MSSQLSERVER 协议中的 TCP/IP 选项

③在下图左侧的“协议”栏目中，将“TCP/IP”的“已启用”状态设置为“是”，然后在图 1-46 右侧的“IP 地址”栏目中，将第一个 IP1 和最后一个 IPALL 中的 TCP 端口值都设置成“1433”。点击“确定”按钮后返回到前一个窗口界面。

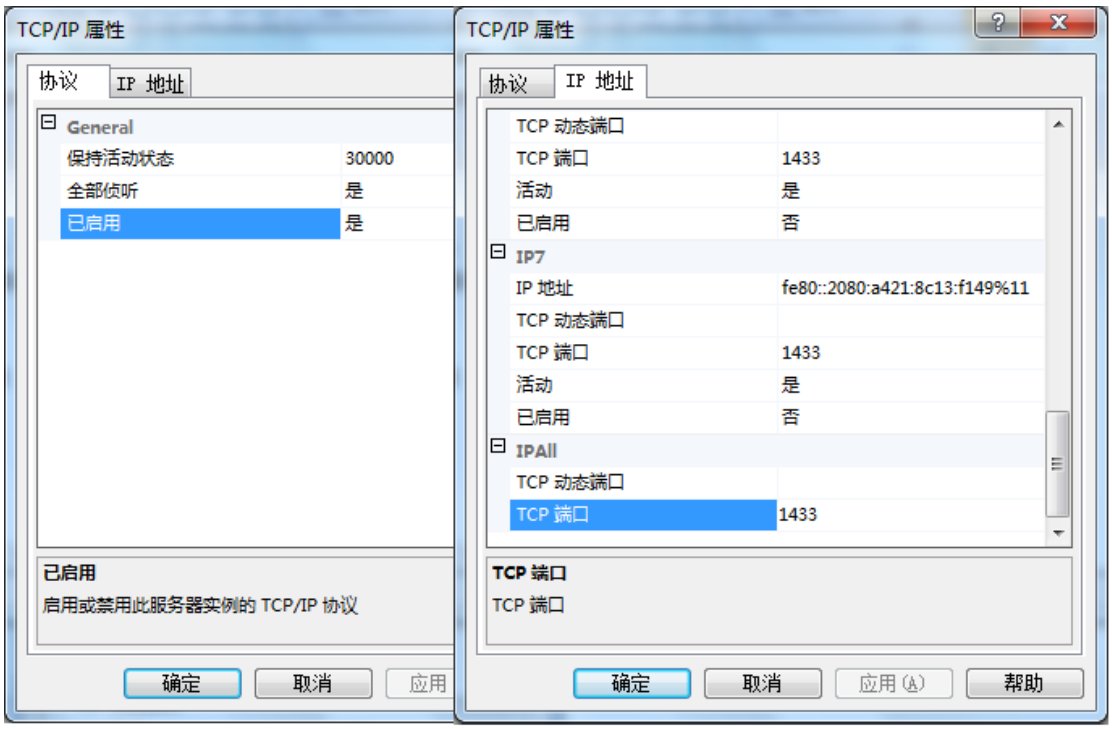


图 1-46　配置 MSSQLSERVER 协议中的 TCP 端口号

④点击下图 1-47 所示的屏幕左侧菜单树中的“SQL Server 服务”，在该图屏幕的右侧中找到 SQL Server（MSSQLSERVER 或 SQLEXPRESS），并双击该菜单。

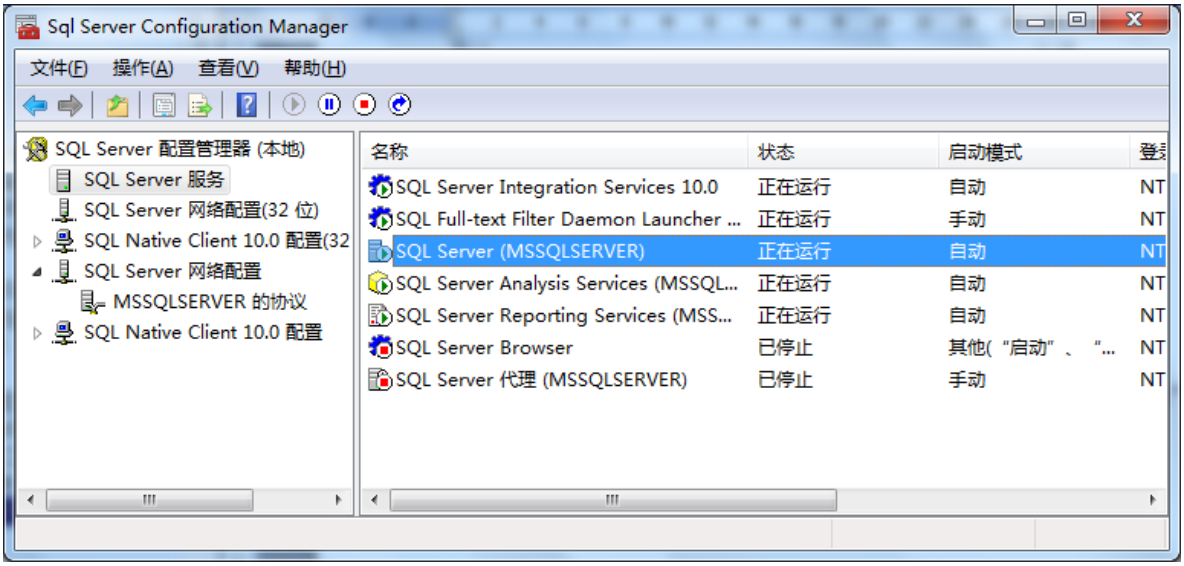


图 1-47　查找 MSSQLSERVER 服务

⑤这时，在下图 1-48 所示的屏幕中点击“重新启动”按钮，数据库服务被重新启动。启动结束后，返回至上一界面，这时显示 SQL Server（MSSQLSERVER）正在运行。至此在 J2EE 中数据库配置结束。

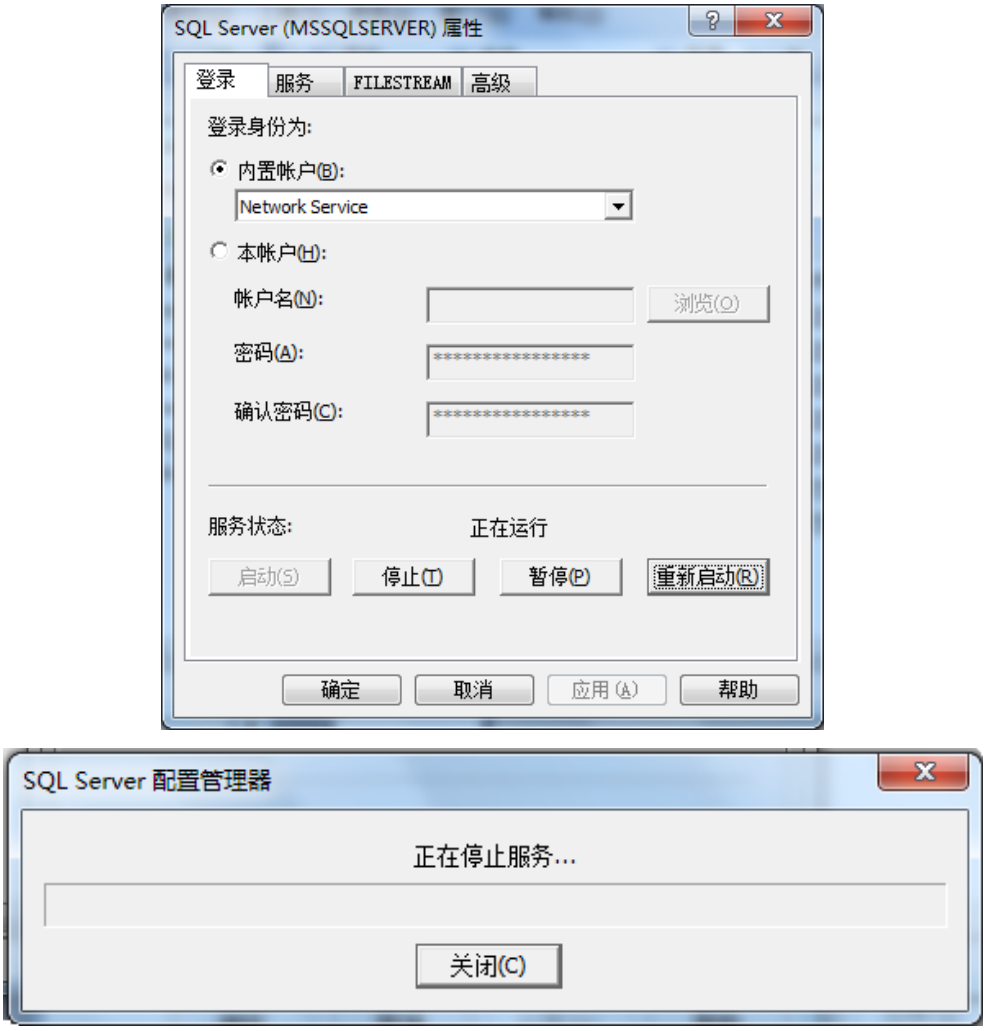


图 1-48 重新启动 MSSQLSERVER 服务

1.8 安装和生成 emlab 数据库

①第一种方法：将数据库备份文件 emlab64.bak 或 emlab32.bak 复制到 d:\myjsp 文件夹下；打开 SQL Server2008，输入下列命令，从备份中恢复数据库：

```
use master
if (db_id('emlab') is not null) drop database emlab
restore database emlab from disk='d:\myjsp\emlab64.bak' with replace
```

此方法只适用于Win7+64位操作系统。Win7+32位操作系统的数据库备份文件为emlab32.bak，数据恢复命令与上面相同。

②第二种方法：打开 d:\myjsp\jQDemos\WebRoot\文件夹下的一个子文件夹 sqlscript，在 SQL Server2008 数据库系统中执行这个文件夹下的所有.sql 脚本文件（如图 1-48 所示）。注意必须先运行 emlab.sql，创建所有数据表结构，然后关闭这个文件。这时，数据表结构已经存在，但都是没有记录的空表，执行其他脚本文件可生成模拟数据。emlab.sql 只能执行一次，如果再次执行，其他脚本文件生成的模拟数据会被删除，又需要重新运行这些脚本文件。

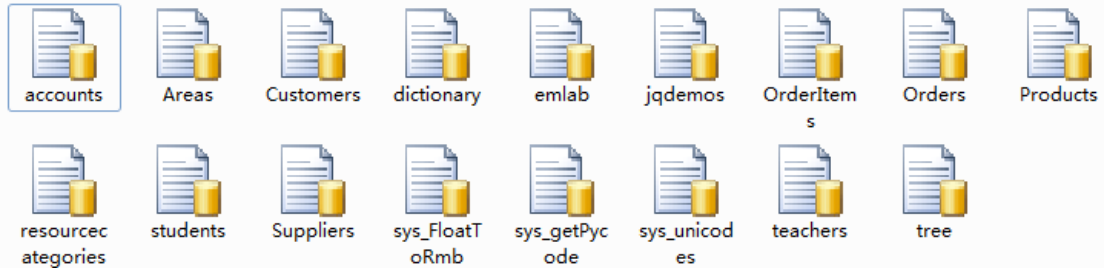


图 1-48　emlab 数据库脚本文件清单

③第三种方法：在 1.6 和 1.7 基础上，输入网址：http://127.0.0.1:8080/jQDemos/database.jsp。先点击“连接测试”提示成功后，再点击“win64”或“win32”按钮恢复数据库。若恢复数据库失败，则点击“运行脚本”按钮，等候生成数据库和模拟数据。

④打开游览器，在地址栏中输入：http://127.0.0.1:8080/jQDemos/。这时程序开始运行，如果出现下列界面（如图 1-50 所示），则表明 J2EE 已成功连接到 SQL Server 数据库 emlab。至此数据库连接和配置成功，J2EE 全部安装完毕！

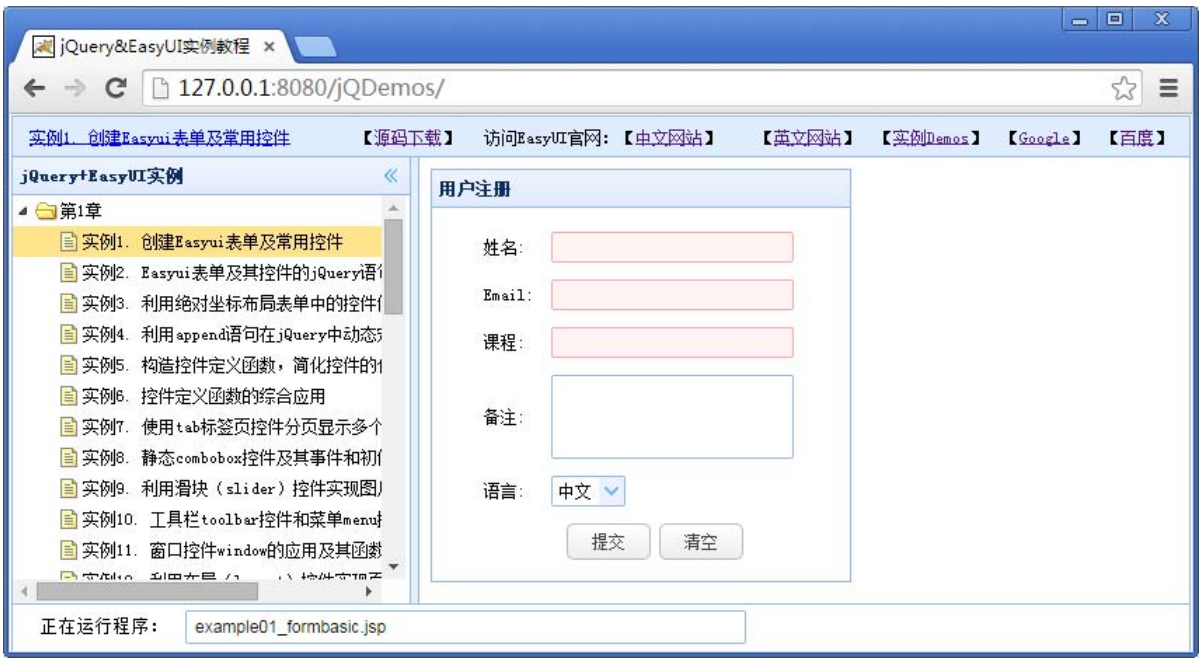


图 1-50　MSSQLSERVER 数据库配置连成功

第 2 章、EasyUI 基本表单控件

实例 1. 创建 EasyUI 表单及常用控件。

本例创建一个包含jQuery插件的jsp文件，阐述jQuery的页面程序的基本结构。该页面包含一个EasyUI的表单控件、四个textbox文本框、一个combobox组合框和两个linkbutton按钮控件。页面布局使用基本的HTML表格（Table）标签。页面程序在<head>…</head>之间使用<script>标签引用jQuery的几个样式文件和jQuery.min.js、jQuery.EasyUI.min.js、EasyUI-lang-zh_CN.js三个插件库文件。本实例程序运行界面如图2-1所示，完整程序代码见程序2-1。

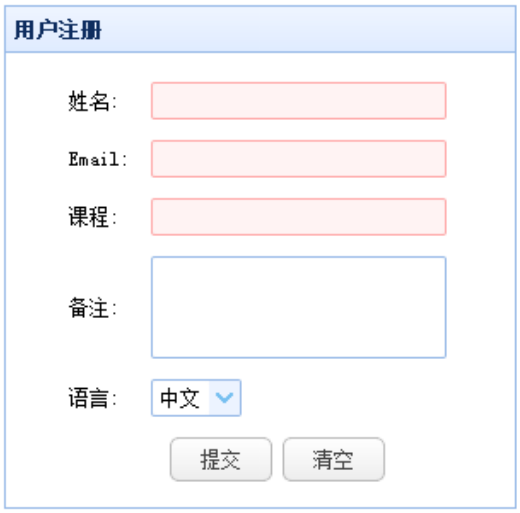


图2-1　EasyUI表单及常用控件程序运行界面

程序2-1. example01_formbasic.jsp

```
<%@ page language="java" import="java.util.*" pageEncoding="utf-8"%>
<!doctype html>
<html lang="en">
<style type="text/css">
</style>
<head>
    <meta charset="utf-8">
    <link rel="stylesheet" type="text/css" href="jqEasyUI/themes/default/EasyUI.me.css">
    <link rel="stylesheet" type="text/css" href="jqEasyUI/themes/icon.css">
    <link rel="stylesheet" type="text/css" href="system/css/icon.css">
    <script type="text/javascript" src="jqEasyUI/jQuery.min.js"></script>
    <script type="text/javascript" src="jqEasyUI/jQuery.EasyUI.min.js"></script>
    <script type="text/javascript" src="jqEasyUI/EasyUI-lang-zh_CN.js"></script>
    <script type="text/javascript" src="system/Easyui_functions.js"></script>
</head>
<body>
    <h4>EasyUI表单控件基础</h4>
    <div class="EasyUI-panel" title="注册" style="width:300px">
        <div style="padding:10px 10px 10px 30px">
        <form id="form1" method="post">
            <table cellpadding="5">
                <tr>
```

```
                <td>姓名:</td>
                <td><input class="EasyUI-textbox" type="text" name="name"
                        data-options="required:true"></input></td>
            </tr>
            <tr>
                <td>Email:</td>
                <td><input class="EasyUI-textbox" type="text" name="email"
                        data-options="required:true,validType:'email'"></input></td>
            </tr>
            <tr>
                <td>课程:</td>
                <td><input class="EasyUI-textbox" type="text" name="subject"
                        data-options="required:true"></input></td>
            </tr>
            <tr>
                <td>备注:</td>
                <td><input class="EasyUI-textbox" name="message"
                        data-options="multiline:true" style="height:60px"></input></td>
            </tr>
            <tr>
                <td>语言:</td>
                <td>
                    <select class="EasyUI-combobox" name="language">
                        <option value="cn">中文</option>
                        <option value="uk">英文</option>
                        <option value="jp">日文</option>
                    </select>
                </td>
            </tr>
        </table>
    </form>
    <div style="text-align:center;padding:5px">
        <a href="javascript:void(0)" class="EasyUI-linkbutton" style="width:60px;"
         onclick="submitForm()">提交</a>
        <a href="javascript:void(0)" class="EasyUI-linkbutton" style="width:60px;"
         onclick="clearForm()">清空</a>
    </div>
    </div>
</div>
<script>
    function submitForm(){
        $('#form1').form('submit');    //jQuery语句
    }
    function clearForm(){
        $('#form1').form('clear');    //jQuery语句
    }
</script>
</body>
</html>
```

主要知识点：

①jQuery+EasyUI插件库的引用及页面文件的总体结构。

②EasyUI控件在HTML中的定义及其class类的名称。

③样式CSS、层DIV的使用及利用表格Table进行页面布局（HTML知识点）。

④EasyUI表单控件中data-options和style选项的设置。例如：data-options="required:true"表示该输入项值不能为空，validType:'email'表示输入项内容必须符合email的基本格式要求。

⑤jQquery控件的表达方式：$('#form1')。

思考题：

①如何设置各个控件的宽度、高度等属性。

②如何设置组合框combobox的选项与组合框的高度。

③如何不使用table进行页面布局。

实例 2. EasyUI 表单及其控件的 jQuery 语句设计。

本例利用HTML语句定义一个EasyUI表单控件myForm1，在这个表单中添加多种EasyUI类型控件，其中包括textbox、numberbox、datebox、combobox等。jQuery程序代码在<scipt>…</scipt>之间定义，从$(document).ready(function(){});语句开始。

在本例中，表单及控件的高度、宽度等属性的设置均根据EasyUI规则由jQuery语句定义，而控件在页面中的位置布局则通过HTML的层 和<p>标记实现。关于EasyUI控件的属性将在以后实例中详细阐述，详细资料可参见http://www.jEasyUI.com/documentation/index.php、http://www.jEasyUI.com、和中文的http://www.jEasyUI.net/等网址。本实例程序运行界面如图2-2所示，完整程序代码见程序2-2。

图2-2　EasyUI表单及其控件的jQuery程序运行界面

程序2-2：example02_formcontrols.jsp

```
<%@ page language="java" import="java.util.*" pageEncoding="utf-8"%>
<html>
<style type="text/css">
</style>
<head>
<meta charset="utf-8">
<link rel="stylesheet" type="text/css" href="jqEasyUI/themes/default/EasyUI.me.css">
<link rel="stylesheet" type="text/css" href="jqEasyUI/themes/icon.css">
<link rel="stylesheet" type="text/css" href="system/css/icon.css">
    <script type="text/javascript" src="jqEasyUI/jQuery.min.js"></script>
    <script type="text/javascript" src="jqEasyUI/jQuery.EasyUI.min.js"></script>
    <script type="text/javascript" src="jqEasyUI/EasyUI-lang-zh_CN.js"></script>
<script type="text/javascript" src="system/Easyui_functions.js"></script>
</head>
<body style="margin: 2px 2px 2px 2px;">
<div id='main' style="margin:2px 2px 2px 2px;">
<div class="EasyUI-panel" id="myForm1" title=" 学生信息编辑"
    style="background:#fafafa;" data-options="iconCls:'panelIcon'" >
    <fieldset id="myFieldset1" style="position:relative; top:10px; left:10px; width:300px;
        height:230px; padding: 2px 0px 0px 16px; border:1px groove" >
        <legend>基本信息</legend>
        <p>学号：<input class="EasyUI-textbox" type="text" id="stuid"
            style="padding:0px 2px 0px 4px"></input></p>
        <p>姓名：<input class="EasyUI-textbox" type="text" id="stuname"
            style="padding:0px 2px 0px 4px"></input></p>
        <p>拼音：<input class="EasyUI-textbox" type="text" id="pycode"
            style="padding:0px 2px 0px 4px"></input> </p>
        <p>性别：<input class="EasyUI-combobox" type="text" id="gender"
            style="padding:0px 2px 0px 4px"></input> </p>
        <p>出生日期：<input class="EasyUI-datebox" type="text" id="birthdate"
            style="padding:0px 2px 0px 4px"></input> </p>
            <p>体重：<input class="EasyUI-numberbox" type="text" id="weight"
            style="padding:0px 2px 0px 4px;"></input> </p>
    </fieldset>
        <fieldset id="myFieldset2" style="position:relative; top:20px; left:10px;
            width:300px; height:200px; padding: 2px 0px 0px 16px; border:1px groove" >
        <legend>通信信息</legend>
        <p>家庭地址：<input class="EasyUI-textbox" type="text" id="address"
            style="padding:0px 2px 0px 4px"></input></p>
            <p>手机号码：<input class="EasyUI-textbox" type="text" id="mobile"
            style="padding:0px 2px 0px 4px"></input></p>
            <p>家庭电话：<input class="EasyUI-textbox" type="text" id="homephone"
            style="padding:0px 2px 0px 4px"></input> </p>
            <p>Email：<input class="EasyUI-textbox" type="text" id="email"
            style="padding:0px 2px 0px 4px"></input> </p>
        <p>微信号：<input class="EasyUI-textbox" type="text" id="weixin"
```

```
            style="padding:0px 2px 0px 4px"></input> </p>
        </fieldset>
    </div>
    </div>
    <script>
    $(document).ready(function(){   //jQuery从此代码开始
        //example02_formcontrols.jsp
        var gendersource=[{'gender':'男'},{'gender':'女'}];
        $("#myForm1").panel({width:345, height:505});
        $("#stuid").textbox({width:120});
        $("#birthdate").datebox({width:120});
        $("#weight").numberbox({width:120});
        $("#gender").combobox({width:120, data:gendersource, valueField:'gender', textField:'gender'});
        var items = $('#gender').combobox('getData');
        $("#gender").combobox('select', items[0].gender);
        $("#stuid").textbox('setValue','20143305001');
        $("#stuname").textbox('setValue','诸葛亮');
        $("#birthdate").datebox('setValue','2015-9-10');
        $("#weight").numberbox({precision:1, max:200, min:30});
        $("#weight").numberbox('setValue',65.5);
        $("#weight").numberbox('textbox').css('text-align','right');
        });   //jQuery代码从此结束
    </script>
    </body>
    </html>
```

主要知识点：

①表单常用控件的基本类型：textbox、numberbox、datebox、combobox、checkbox、tabs、tree、grid、treegrid、filebox、button、menu等。

②利用jQuery语句设置textbox、datebox、numberbox、combobox的属性，包括宽度、高度等。例如：$("#stuid").textbox({width:120});

③利用jQuery语句设置控件的初值。例如：$("#stuname").textbox('setValue','诸葛亮');

④利用jQuery语句设置numberbox数值框的属性，包括最大值、最小值、小数点位数等。例如：

$("#weight").numberbox({precision:1, max:200, min:30});

⑤利用jQuery语句设置numberbox数值框的右对齐。例如：

$("#weight").numberbox('textbox').css('text-align','right');

⑥利用jQuery语句设置combobox组合框的选项与初值。例如：

var gendersource=[{'gender':'男'},{'gender':'女'}]; //定义数据源

$("#gender").combobox({width:120, data:gendersource, valueField:'gender', textField:'gender'});

var items = $('#gender').combobox('getData'); //提取所有选项

$("#gender").combobox('select', items[0].gender); //将初值设置为第一个选项

⑦datebox控件的日期格式及本地化设置。引入jqEasyUI/EasyUI-lang-zh_CN.js。

⑧HTML中Fieldset的定义与使用。

⑨jQuery判断控件是否存在方法： if ($('#id').length>0)

思考题：

①不同控件在页面中的上下对齐问题，即如何设置文字与编辑框之间的距离。

②在页面中一行显示多列问题。例如：如何在“体重”文本框的右侧显示“KG”这个字符。

实例 3. 利用绝对坐标实现表单控件的位置布局。

本例采用绝对（坐标）地址法（position: absolute）确定控件在页面中的显示位置。在EasyUI中，控件绝对位置的设置采用top和left两个属性（标签），而与其他有些插件使用y和x有所不同。控件位置属性可以在HTML的style中设置，也可以通过样式CSS设置。本例通过jQuery语句设置EasyUI控件的位置、尺寸等属性。在EasyUI中，由于有些控件无法直接使用CSS布局定位，这时可在其外面先添加一个层 ，然后再对层进行绝对位置布局，以确定控件显示的位置。

本例将实例2中定义位置布局的 … 换成>div>… ，将EasyUI-panel类型的外层容器myForm1的position设置为relative（在style中设置），在这个容器内添加的控件其style中的position设为absolute。每个控件及其label采用jQuery代码设置，所有label实现右对齐。

本实例程序运行界面如图2-3所示，主要代码如下见程序2-3。

图2-3. 利用绝对坐标实现表单控件位置布局程序运行界面

程序2-3. example03_absolutelayout.jsp

```
<body style="margin: 2px 2px 2px 2px;">
<div id="main" style="margin:2px 2px 2px 2px;">
 <div class="EasyUI-panel" id="myForm1" title=" 学生信息编辑" style="position:relative;
        background:#fafafa;" data-options="iconCls:'panelIcon'" >
 <fieldset id="myFieldset1" style="position:absolute; top:10px; left:10px; width:300px; height:210px;
        padding: 2px 0px 0px 16px; border:1px groove" >
        <legend>基本信息</legend>
        <label id='stuid_label'>学号：</label>
       <div id="stuid_div"><input class="EasyUI-textbox" type="text" id="stuid"></input></div>
        <label id='stuname_label'>姓名：</label>
       <div id="stuname_div"><input class="EasyUI-textbox" type="text" id="stuname"></input></div>
```

```
        <label id='pycode_label'>拼音：</label>
        <div id="pycode_div"><input class="EasyUI-textbox" type="text" id="pycode"></input></div>
        <label id='gender_label'>性别：</label>
        <div id="gender_div"><input class="EasyUI-combobox" type="text" id="gender"></input></div>
        <label id='birthdate_label'>出生日期：</label>
        <div id="birthdate_div"><input class="EasyUI-datebox" type="text" id="birthdate"></input></div>
        <label id='weight_label'>体重：</label>
        <div id="weight_div"><input class="EasyUI-textbox" type="text" id="weight"></input></div>
        <label id='weightx_label'>KG</label>
  </fieldset>
  <fieldset id="myFieldset2" style="position:absolute; top:235px; left:10px; width:300px; height:180px;
        padding: 2px 0px 0px 16px; border:1px groove" >
        <legend>通信信息</legend>
        <label id='address_label'>家庭地址：</label>
        <div id="address_div"><input class="EasyUI-textbox" type="text" id="address"
              style="padding:0px 2px 0px 4px"></input></div>
        <label id='mobile_label'>手机号码：</label>
        <div id="mobile_div"><input class="EasyUI-textbox" type="text" id="mobile"
              style="padding:0px 2px 0px 4px"></input></div>
        <label id='homephone_label'>家庭电话：</label>
        <div id="homephone_div"><input class="EasyUI-textbox" type="text" id="homephone"
        style="padding:0px 2px 0px 4px"></input></div>
        <label id='email_label'>Email：</label>
        <div id="email_div"><input class="EasyUI-textbox" type="text" id="email"
              style="padding:0px 2px 0px 4px"></input></div>
        <label id='weixin_label'>微信号：</label>
        <div id="weixin_div"><input class="EasyUI-textbox" type="text" id="weixin"
              style="padding:0px 2px 0px 4px"></input></div>
  </fieldset>
  </div>
</div>
<script>
 $(document).ready(function(){
        //控件位置布局
        $("#stuid_label").css({position: "absolute", top:"23px", left:"10px", width:"65px",
        "text-align":"right", "z-index":2});
        $("#stuid_div").css({position: "absolute", top:"20px", left:"76px", "z-index":2});
        $("#stuname_label").css({position: "absolute", top:"53px", left:"10px", width:"65px",
        "text-align":"right", "z-index":2});
        $("#stuname_div").css({position: "absolute", top:"50px", left:"76px","z-index":2});
        $("#pycode_label").css({position: "absolute", top:"83px", left:"10px", width:"65px",
        "text-align":"right", "z-index":2});
        $("#pycode_div").css({position: "absolute", top:"80px", left:"76px", "z-index":2});
        $("#gender_label").css({position: "absolute", top:"113px", left:"10px", width:"65px",
        "text-align":"right", "z-index":2});
        $("#gender_div").css({position: "absolute", top:"110px", left:"76px", "z-index":2});
        $("#birthdate_label").css({position: "absolute", top:"143px", left:"10px", width:"65px",
        "text-align":"right", "z-index":2});
        $("#birthdate_div").css({position: "absolute", top:"140px", left:"76px", "z-index":2});
        $("#weight_label").css({position: "absolute", top:"173px", left:"10px", width:"65px",
       text-align":"right", "z-index":2});
        $("#weight_div").css({position: "absolute", top:"170px", left:"76px", "z-index":2});
        $("#weightx_label").css({position: "absolute", top:"173px", left:"190px", width:"25px",
        "text-align":"right", "z-index":2});
        $("#address_label").css({position: "absolute", top:"23px", left:"10px", width:"65px",
```

```
        "text-align":"right", "z-index":2});
        $("#address_div").css({position: "absolute", top:"20px", left:"76px", "z-index":2});
        $("#mobile_label").css({position: "absolute", top:"53px", left:"10px", width:"65px",
        "text-align":"right", "z-index":2});
        $("#mobile_div").css({position: "absolute", top:"50px", left:"76px", "z-index":2});
        $("#homephone_label").css({position: "absolute", top:"83px", left:"10px", width:"65px",
        "text-align":"right", "z-index":2});
        $("#homephone_div").css({position: "absolute", top:"80px", left:"76px", "z-index":2});
        $("#email_label").css({position: "absolute", top:"113px", left:"10px", width:"65px",
        "text-align":"right", "z-index":2});
        $("#email_div").css({position: "absolute", top:"110px", left:"76px", "z-index":2});
        $("#weixin_label").css({position: "absolute", top:"143px", left:"10px", width:"65px",
        "text-align":"right", "z-index":2});
        $("#weixin_div").css({position: "absolute", top:"140px", left:"76px", "z-index":2});
        $("#myForm1").panel({width:345, height:465});
        $("#stuid").textbox({width:120});
        $("#birthdate").datebox({width:120});
        $("#weight").numberbox({width:120});
        $("#stuid").textbox('setValue','20143305001');
        $("#stuname").textbox('setValue','诸葛亮');
        $("#birthdate").datebox('setValue','2015-9-10');
        $("#weight").numberbox({precision:1, max:200, min:30});
        $("#weight").numberbox('setValue',65.5);
        $("#weight").numberbox('textbox').css('text-align','right');
        //设置combobox
        var gendersource=[{'gender':'男'},{'gender':'女'}];
        $("#gender").combobox({width:120, data:gendersource, valueField:'gender', textField:'gender'});
        var items = $('#gender').combobox('getData');
        $("#gender").combobox('select', items[0].gender);
        $("#address").textbox({width:230});
      });
  </script>
  </body>
```

主要知识点：

①在 EasyUI 中绝对坐标定位采用 top 和 left 属性，而不是有些插件库中使用 y 和 x。

②数字框 numberbox 右对齐方式$("#weight").numberbox('textbox').css('text-align','right')。

③label 标签和编辑型控件的 CSS 设置，设置其 top、left、width 等属性值，当然还有 position 值为 absolute。

思考题：

1．如果在 jQuery 中不定义控件的 CSS，页面显示的效果会怎样？

2．控件（包括一些如 fieldset 之类的容器）采用绝对坐标的定位显示方式虽然效果较好，但程序书写比较繁琐，如何简化？

3．如何动态定义页面中的控件？即不在 HTML 中事先定义控件，而在<script>…</script>之间使用 jQuery 语句定义。

作业题：

使用EasyUI插件及绝对地址定位方式，编写程序设计图2-4所示页面。

图 2-4　使用绝对坐标定位作业题效果图

实例 4. 利用 append()方法在 jQuery 中动态定义和生成 EasyUI 控件。

前面几个实例采用HTML标签语句定义控件，但有时也有一些控件由于事先是不确定的，需要在程序运行过程中动态生成，为此本例介绍append()方法，在javascript语句中动态定义EasyUI控件，具体方法和程序如下。

①在页面预留一个层，取名为main。所有其他动态生成的控件均嵌套在这个层之内。

```
<div id="main" style="margin:2px 2px 2px 2px;">
```

②在<script>…</script>之间编写程序。先定义一个变量（如str），把需要生成控件的HTML语句保存到这个变量中，然后使用append()方法添加到相应的父类容器中去。

例如，在main层中添加一个表单类（panel）控件的语句如下：

```
var str='<div class="EasyUI-panel" id="myForm1" title=" 学生信息编辑"
style="position:relative; background:#fafafa;" data-options="iconCls:\'panelIcon\'" >';
$("#main").append($(str));   //在main容器下嵌入一个表单控件myForm1
```

同样地，在表单myForm1中添加一个fieldset类控件的语句如下：

```
str='<fieldset id="myFieldset1" style="position:absolute; top:10px; left:10px; width:300px;
height:210px; padding: 2px 0px 0px 16px; border:1px groove" ><legend>基本信息</legend></fieldset>';
$("#myForm1").append($(str));   //在myForm1容器下嵌入一个fieldset字段集控件
```

推而广之，各类控件都可以通过这种方式动态生成。又如在myFieldset1字段集容器中添加两个label和textbox文本框控件，其语句如下：

```
str='<label id="stuid_label">学号：</label>';
str+='<div id="stuid_div"><input class="EasyUI-textbox" type="text" id="stuid"></input></div>\n';
str+='<label id="stuname_label">姓名：</label><div id="stuname_div"><input class="EasyUI-textbox"
type="text" id="stuname"></input></div>\n';
$("#myFieldset1").append($(str));
```

③与实例3相同，在动态添加和生成控件之后，可以使用jQuery语句设置各个控件的属性。

本例完整代码见程序example04_appendcontrols.jsp中。

主要知识点：

动态添加控件的方法：$("#parentid").append($(str));。这里，parentid 为父类容器的 ID，str 变量为符合 HTML 语法的控件定义语句，str 需要用括号和$引导。

思考题：

1．在动态定义各个控件时，有哪些代码具有相似性或重复性？如何使用函数动态定义控件？

2．构造一个函数用于动态定义textbox控件，这个函数需要几个参数？参考下列API链接地址，查看相关控件的属性。

http://www.jEasyUI.com/demo/main/

http://www.jEasyUI.com/documentation/index.php

实例 5. 构造控件自定义函数，简化控件的使用。

从前面实例中可以看出控件的定义是比较繁琐的，如果将一些常用的控件通过函数形式加以定义和调用，这将大大简化控件的使用。函数可以将控件的许多标准属性加以固化，同时又允许用户自己定义一些属性，这些属性往往可被描述为参数。

jQDemos 工程创建一个名为 fn_function.js 的 javascript 公共文件，该文件中包含各类控件自定义函数，并在以后实例中调用这些函数。这里仅介绍表单 panel、字段集容器 fieldset、文本框 textbox 这三个控件的定义函数。

①myForm()函数：定义一个表单，其参数包括 id、parent、title、top、left、height、width 和 style 等，分别表示控件标识符、父类容器名、窗体标题、起始显示位置坐标、表单的高度和宽度、表单样式，style 参数选项指定表单是否可以最大化、最小化、收缩或关闭。

②myFieldset()函数。定义一个容器，在容器内分组显示各个字段，其参数包括 id、parent、title、top、left、height、width 等，各参数的含义与 myForm()类同。

③myTextField()函数。定义一个 textbox 文本输入框，其参数包括 id、parent、label、labelwidth、top、left、height、width、value、style 等。这里 label 为文本输入框之前的提示符，labelwidth 为该提示符的宽度，value 为文本框的初值，style 为文本框的数据验证方式，包括 readonly（只读）、email（邮件格式验证）、url（网址格式验证）、password（密码形式隐藏）、icon（图标显示）等。

控件自定义函数的引用方式与其他插件库 JS 文件相同，即在<head>…</head>之间添加下列代码：<script type="text/javascript" src="system/Easyui_function.js"></script>。jQDemos 控件自定义函数的主要参数如表 2-1 所示。

表 2-1. 控件自定义函数主要参数表

参数	说明	参数	说明
id	控件标识符	max	数字框输入数据的最大值
parent	父类容器的标识符	items	静态 combobox、checkbox、menu 等控件的可选项，各选项之间用分号分隔
title	标题（用于表单、标签页、树、数据网格等控件中）	cols	静态 combobox、checkbox、menu 等控件的可选项，各选项之间用分号分隔
top	控件在屏幕上显示位置的 Y 坐标值	textfield	checkbox 等控件中选项分行显示时的行高（行间距）
left	控件在屏幕上显示位置的 X 坐标值	masterfield	checkbox 等控件中每行显示的选项个数（即列数）
height	控件的高度	keyfield	动态 combobox 控件中选项文本列名称
width	控件的宽度	sortfield	多个 combobox 控件联动时依赖于主组合框的列名称
label	编辑型控件输入引导提示符的标签名	sql	数据表的关键字（通常为主键），用于与数据库连接的控件
labelwidth	引导提示符标签的宽度	buttons	数据表的排序列（缺省时为 keyfield 的值），用于与数据库连接的控件
value	文本框等控件的初值	tabs	动态 combobox、grid、tree 等控件从数据库中提取数据的查询语句

length	输入数据的最大长度（字符数）	style	定义 window 弹出式窗体控件的按钮，为 ok、cancel、save 的组合
decimal	数字框的小数位数		分页标签控件中各个分页的标题，用分号分隔
min	数字框输入数据的最小值		其他属性的选项值（包括最大化、最小化、收缩、关闭等）

下面以 myTextField 为例，阐述控件自定义函数的主要设计步骤。

①熟悉和掌握控件的常用属性，确定哪些属性是固定的、哪些是可变的（可设置的）；把可变的属性定义为参数。为此确定 textbox 的主要参数为 id, parent, label, labelwidth, top, left, height, width, value, style 等。

```
function myDefineTextField(id, parent, label, labelwidth, top, left, height, width, value, style){
```

②默认值设置。例如：如果父类容器标识符为空，则设置为 main，文本框的高度位 0，则设置为系统默认值（这里由 systext.height 变量给定，默认值为 24px）。

```
if (parent=='') parent='main';
if (height==0) height=systext.height;
```

③根据参数 label 的值，确定是否需要创建一个输入提示符 label 文字标签。当 label 参数值为非空时，调用 myFieldLabel()创建一个文字标签。如果 labelwidth 为 0 时，文字标签与文本编辑框分行显示。

```
if (label!='' && labelwidth==0){
    myFieldLabel(id,parent,label,labelwidth,top,left+2);   //创建文字标签
    top=1*(top+syslabel.fontsize+syslabel.topmargin+2); //换行显示文本框
    labelwidth=0;
    }else{
        myFieldLabel(id,parent,label,labelwidth,top,left); //创建文字标签
    }
}
```

④创建文本框控件，需要在其外面添加一个层（这个层也有 id），以帮助确定其绝对位置。创建文本框的 HTML 语句保存在一个变量(str)中。如果文本框的初值参数 value 非空，则在 HTML 定义中添加初值设置。之后使用 append 语句将控件添加到父类容器中去。

```
var str='<div id="'+id+'_div"><input ';
if (value!=undefined && value!='){   //设置初值
    str+=' value="'+value+'"';
}
str+=' class="EasyUI-validatebox EasyUI-textbox" type="text" id="'+id+'" style="padding:0px 6px 0px
6px;" ></div>';
$("#"+parent).append($(str));
```

④创建控件之后，根据函数中的参数值，设置控件的相关属性。控件位置等属性在层 的样式 CSS 中设置，这里通过调用 myTextCss()函数实现。控件的高度在 textbox({height:?})设置。

```
$("#"+id+'_div').css(myTextCss(top,1*(left+labelwidth),height,width));
$("#"+id).css({height:"100%", width:"100%"});
$("#"+id).textbox({height: height});
```

⑤设置文本框的其他自定义属性（如 xparent、xid、xlabel 等），这些属性不是 EasyUI 本身拥有的，而是为了程序设计方便，程序员自己添加的。

```
$("#"+id).attr('xparent',parent);  //自定义属性，父类容器的标识符
$("#"+id).attr('xlabel',label);  //自定义属性，输入提示符
$("#"+id).attr('xid',id);  //自定义属性，控件本身的标识符
```

⑥根据 style 参数值设置控件的其他属性，例如：控件是否只读，是否按 email 格式、url 格式验证数据。如果 style 中包含 password 字符，则文本框按密码形式输入。这项功能通过调用 myFieldStyle()函数实现，即：myFieldStyle(id,style);

myTextField()函数应用举例：

```
myDefineTextField('email','myFieldset1','Email：', 70, 20, 12, 0, 180, 'zxywolf@163.com', 'email');
myDefineTextField('psw','myFieldset1','密码：', 70, 55, 12, 0, 180, '888, 'password');
myDefineTextField('pycode','myFieldset1','姓名拼音：', 70, 55, 12, 0, 180, ' ', 'readonly');
```

下面给出 myTextField()、myFieldLabel()、myTextCss()、myFieldStyle()这四个函数的完整程序代码：

程序 2-5-1：文本框定义函数 myTextField()

```
function myTextField(id,parent,label,labelwidth,top,left,height,width,value,style){
    var str='';
    if (parent=='') parent='main';
    if (height==0) height=systext.height;
    if (labelwidth==0 && label!=''){
        myFieldLabel(id,parent,label,labelwidth,top,left+2);
        top=1*(top+syslabel.fontsize+syslabel.topmargin+2); //换行显示文本
        labelwidth=0;
    }else{
        myFieldLabel(id,parent,label,labelwidth,top,left);
    }
    str+='<div id='"+id+'_div"><input ';
    if (value!=undefined && value!=''){   //设置初值
        str+=' value='"+value+'"';
    }
    str+=' class="easyui-validatebox easyui-textbox" type="text" id='"+id+'" style="padding:0px 6px
0px 6px;" ></div>';
    $("#"+parent).append($(str));
    $("#"+id+'_div').css(myTextCss(top,1*(left+labelwidth),height,width));
    $("#"+id).css({height:"100%", width:"100%"});
    $("#"+id).textbox({height: height});
    $("#"+id).attr('xparent',parent);  //自定义属性，父类容器的标识符
    $("#"+id).attr('xlabel',label);  //自定义属性，输入提示符
    $("#"+id).attr('xid',id);  //自定义属性，控件本身的标识符
    myFieldStyle(id,style);
}
```

程序2-5-2：控件中的文字标签label子定义函数myFieldLabel()

```
function myFieldLabel(id,parent,label,labelwidth,top,left){
    //定义控件中的label
    if (label!=undefined && label!=''){
        $("#"+parent).append("<label id='label"+id+'" align='"+syslabel.align+'">"+label+"</label>");
        if (labelwidth>0){
            $("#label"+id).css(myLabelCss(top+syslabel.topmargin,left,0,labelwidth));
        }else{
            $("#label"+id).css(myLabelCss(top,1*left+2,0,label.length*syslabel.fontsize));
            top=1*(top+syslabel.fontsize+syslabel.topmargin); //换行显示文本
        }
    }else labelwidth=0;
}
```

程序 2-5-3：文本框控件样式 CSS 子定义函数 myTextCss()

```
function myTextCss(top,left,height,width){
    var str='';
    var css='';
    if (top!=undefined && left!=undefined){
        str='var css={padding:"0px 2px 0px 4px", position: "absolute", top:'"+top+'px", left:'"+left+'px"';
        if (width!=undefined && width>0){
            str+=', width: '+width;
        }
        if (height!=undefined && height>0){
            str+=', height: '+height;
        }
        str+=', "z-index":2};\n';
    }else{
        str+="\n";
    }
    eval(str);
    //console.log(str);
    return css;
}
```

程序 2-5-4：文本框其他属性特征自定义函数 myFieldStyle

```
function myFieldStyle(id,style){   //设置style只读、图标
    if (style!=undefined && style!=''){
        var tmp=(style+';').split(';');
        for (var i=0;i<tmp.length;i++){
            //console.log(tmp[i]);
            if (tmp[i]=='readonly') $("#"+id).textbox({readonly: true});
            if (tmp[i]=='email') $("#"+id).textbox({validType:'email'});
            if (tmp[i]=='url') $("#"+id).textbox({validType:'url'});
            if (tmp[i]=='password') $("#"+id).textbox({type:'password'});
            if (tmp[i].indexOf('icon:')>=0){
                var x1=tmp[i].indexOf('icon:');
                var x2=tmp[i].indexOf(';',x1+1);
                if (x1>=0 && x2>0) var icon=tmp[i].substring(x1+5,x2-1);
                else var icon=tmp[i].substr(x1+5,255);
                //$("#"+id).textbox({iconCls: icon});   //
                $("#"+id).textbox({buttonIcon: icon});
            }
        }
    }
}
```

相关知识点：

①string.split函数：将一个字符串按某个分隔符进行分割后，将分隔的各个值存储到一个数组变量中。例如:

```
var xfields="stuid;stuname;gender;birthdate;address";
var tmp=xfields.split(';');
```

这是将xfields中的字符按分号（;）去分割，之后得到5个字符串，存放到数组tmp中。

②eval()函数：可计算某个字符串，并执行其中的 JavaScript 代码。例如：

eval("var x=100; var y=x+10;");

命令执行后，得到 x 和 y 两个变量，其值分别为 100 和 110。又如：

var data='{stuid:"2014540101", name:"贾宝玉"}';

eval("var source="+data);

alert(source.name);

命令执行后，在屏幕上显示“贾宝玉”这个字符串。

实例 6. 控件定义函数的综合应用。

本例利用控件自定义函数，创建一个表单和四个 fieldset 控件，在每个 fieldset 中添加若干个 textbox、datebox、numberbox、checkbox、combobox、filebox 等控件。与实例 3 相比，采用这种方式定义控件，可以大大减少程序代码量，增强了程序的可读性。对于一些在函数中没有涉及到的控件属性和控件事件，可以在引用自定义函数定义之后单独编写代码实现。例如本例为学生文本框添加了两个属性和一个 onChange 事件。

在后续实例中，为了节省篇幅，一般控件的定义将主要采用这种函数的形式。本实例程序运行界面如图 2-6 所示，完整程序代码见程序 2-6。

图 2-5　利用控件自定义函数的程序运行界面

程序 2-6：example06_functionapp.jsp

```
<%@ page language="java" import="java.util.*" pageEncoding="utf-8"%>
<html>
<style type="text/css">
</style>
<head>
 <meta charset="utf-8">
 <link rel="stylesheet" type="text/css" href="jqEasyUI/themes/default/EasyUI.me.css">
 <link rel="stylesheet" type="text/css" href="jqEasyUI/themes/icon.css">
 <link rel="stylesheet" type="text/css" href="system/css/icon.css">
     <script type="text/javascript" src="jqEasyUI/jQuery.min.js"></script>
```

```
    <script type="text/javascript" src="jqEasyUI/jQuery.EasyUI.min.js"></script>
    <script type="text/javascript" src="jqEasyUI/EasyUI-lang-zh_CN.js"></script>
 <script type="text/javascript" src="system/Easyui_functions.js"></script>
</head>
<body style="margin: 2px 2px 2px 2px;">
<div id="main" style="margin:2px 2px 2px 2px;">
</div>
<script>
 $(document).ready(function(){
        //定义 EasyUI-panel 容器控件
        myForm('myForm1','main','学生信息编辑',0,0,490,695,'close;collapse;min;max');
        myFieldset('myFieldset1','myForm1','基本信息',010,10,230,290);
        myFieldset('myFieldset2','myForm1','通信信息',010,320,230,350);
        myFieldset('myFieldset3','myForm1','个人特长',250,10,200,290);
        myFieldset('myFieldset4','myForm1','上传照片',250,320,200,350);
        myTextField('stuid','myFieldset1','学号：',70,33*0+20,12,0,120,'D20101');
        myTextField('stuname','myFieldset1','姓名：',70,33*1+20,12,0,160,'贾宝玉');
        myTextField('pycode','myFieldset1','汉语拼音：',70,33*2+20,12,0,160,'');
        myDateField('birthdate','myFieldset1','出生日期：',70,33*3+20,12,0,120,'1997-2-17');
        myComboField('gender','myFieldset1','性别：',70,33*4+20,12,0,120,'男;女','');
        myNumericField('weight','myFieldset1','体重：',70,33*5+20,12,0,120,8,2,'60.25','10','100');
        myLabelField('label1', 'myFieldset1','KG',33*5+18+7,220,0,0);
        myTextField('address','myFieldset2','家庭地址：',70,33*0+20,12,0,255,'浙江省杭州市西湖区');
        myTextField('mobile','myFieldset2','手机号码：',70,33*1+20,12,0,180,'');
        myTextField('homephone','myFieldset2','家庭电话：',70,33*2+20,12,0,180,'');
        myTextField('email','myFieldset2','Email：',70,33*3+20,12,0,180,'zxywolf@163.com');
        myTextField('qq','myFieldset2','QQ 号：',70,33*4+20,12,0,180,'857199052');
        myTextField('weixin','myFieldset2','微信号：',70,33*5+20,12,0,180,'zxywolf888');
        myCheckBoxField('research','myFieldset3','个人特长：',70,33*0+20,12,24,3,
        '围棋;国际象棋;足球;长跑;数学;信息技术');
        myMemoField('notes','myFieldset3','个人简介：',0,33*2+5,10,100,273,'');
        myFileField('photo','myFieldset4','',0,26,6,0,240,' ');
        //定义事件
        $("#photoupload").bind('click',function(e){
                var targetname=$("#stuid").textbox('getValue');
                var result=myFileupLoad('photo','mybase/students/',targetname);
                console.log(result);
        });
        //为“学生”文本框添加在自定义函数中没有涉及的属性和事件
        $("#stuname").textbox({
                prompt:"姓名有效长度为 20 个字符",
                required: true,
                onChange: function(newv,oldv){
                        alert("姓名已发生变化，请重新输入拼音");
                }
        });
    });
</script>
</body>
</html>
```

思考题：

深入分析研究 myCheckBoxField()自定义函数的构造及其应用。

实例 7. 使用 tabs 标签页控件分页显示多个表单控件。

EasyUI 的标签页/选项卡（tabs）控件可以将多个表单组合在一起，采用类似于分页的形式显示信息，其主要属性包括：title、closable、tabPosition、tabWidth、tabHeight 等。标签页通常作为一种容器加以使用，一般情况可以有多个分页标签同时存在，但是同时只有一个处于激活状态。

本例将实例 6 中四个栏目的学生信息按标签页进行分页显示，以三种方法创建标签页 myTab。

①方法 1：在网页的<body>…</body>之间添加下列 HTML 语句，以 层嵌套形式实现标签页，例如下列语句可定义 4 个标签页。

```
<div id="myTab" class="EasyUI-tabs" style="overflow: auto; margin: 0px 0px -19px 0px; width:390px; height:285px;">
    <div id="myTab1" title="基本信息" style="position:relative; overflow:auto;"></div>
    <div id="myTab2" title="通信信息" style="position:relative; overflow:auto;"></div>
    <div id="myTab3" title="个人简历" style="position:relative; overflow:auto;"></div>
    <div id="myTab4" title="上传照片" style="position:relative; overflow:auto;"></div>
</div>
```

②方法 2：在<script>…</script>之间编写 jQuery 程序，例如：

```
var str='<div id="myTab" class="EasyUI-tabs" style="overflow: auto; margin: 0px 0px -19px 0px; padding 0px 0px 0px 0px;">';
str+='<div id="myTab1" title="基本信息" style="position:relative; overflow:auto;"></div>';
str+='<div id="myTab2" title="通信信息" style="position:relative; overflow:auto;"></div>';
str+='<div id="myTab3" title="个人简历" style="position:relative; overflow:auto;"></div>';
str+='<div id="myTab4" title="上传照片" style="position:relative; overflow:auto;"></div>';
str+='</div>';
$("#main").append($(str));
$("#myTab").tabs({height:285, width:385});
```

③方法 3：直接调用控件自定义函数 myTabs()。例如调用下列函数可以产生 4 个选项卡的标签页，其高度为 285 像素，宽度为 385 像素。

```
myTabs('myTab', 'main', '基本信息;通信信息;个人简历;上传照片', 0, 0, 285, 385, ' ');
```

在定义 myTab 标签页之后，可以使用$("#myTab").tabs('select', index)方法聚焦（选中）某个标签页中的选项卡分页，使用 $("#myTab").tabs("tabs").length 获取标签页的选项卡总数，使用 $("#myTab").tabs('update', { tab: $("#myTab").tabs('getTab', 0), options:{} })更新某个选项卡分页。本实例程序运行界面如图 2-6 所示。

图2-6 标签页Tabs控件程序运行界面

作业题：

编写 JS 程序，实现 tabs 标签页控件中选项卡分页的动态增加与删除。

实例 8. 静态组合框 combobox 控件及其事件和初值的设置。

在EasyUI中，组合框控件（combo或combobox，也称下拉框）需要绑定一个数据源（source），以指定其选项的来源。通常，这个数据源可以是本地数据集（静态），也可以是远程服务器端符合规定格式数据集（动态）。

在定义本地数据源时一般采用JSON格式的数组。例如，有关性别的数据源可以定义如下：

var gendersource =[{"gender":"男"},{"gender":"女"}];

在这个JSON格式数据中，“性别”对应的列名为gender，选项为“男”和“女”两个。也可以采用多维数组定义组合框数据源。例如，有关城市组合框的数据源可定义如下：

var citysource=[{"cityid":"330100", "city":"杭州市"}, {"cityid":"330200", "city":"宁波市"}]

这里，每个城市选项中有两个列：cityid和city。

在定义数据源之后，需要设置组合框控件的相关属性。数据源由data属性指定，组合框显示的值和组合框返回的值分别由textField和valueField这两个关键属性指定。当valueField缺省时，由combobox("getValue")返回的值总是为空。

本例在一个表单中定义5个数据源及其相应的组合框控件，并为“学历”组合框和“城市”组合框分别定义一个onChange事件和onSelect事件。这两个事件在用户改变组合框的选项时都会被触发，其中onSelect事件可以提取当前选项对应的各个数据源值。

组合框的初值通过combobox('select')或combobox('setValue')两种方法进行设置。例如，下列语句将“学历（degree）”组合框的初值设置为第3个选项值（即“本科”）。

$('#degree').combobox('select',degreesource[2].degree);

组合框选项初值的赋值语句应当放在控件属性和事件定义之后。下面给出“城市”组合框的完整定义过程和方法：

①使用JSON格式定义数据源，存放在一个数组变量citysource中。

```
var citysource=[{"cityid":"330100","city":"杭州市"},{"cityid":"330200","city":"宁波市
"},{"cityid":"330300","city":"温州市"},{"cityid":"330400","city":"嘉兴市"},{"cityid":"330500","city":"湖州市
"},{"cityid":"330600","city":"绍兴市"},{"cityid":"330700","city":"金华市"},{"cityid":"330800","city":"衢州市
"},{"cityid":"330900","city":"舟山市"},{"cityid":"331000","city":"台州市"},{"cityid":"331100","city":"丽水市"}];
```

②使用JS代码在$("#myFieldset1")中添加一个组合框控件，并在其style中指定位置和尺寸

```
$("#myFieldset1").append('<div style="position: absolute; top:300px; left:87px; width: 100px;
padding-left: 4px;" id="city_div"><input class="EasyUI-combobox" id="city" style="padding:0px 6px 0px
6px" /></div>');
```

③设置组合框的valueField和textField属性属性，使用data属性绑定选项数据源citysource。panelHeight为组合框下拉时显示的高度（按像数计算），选项较少时，可以为auto值。

```
$("#city").combobox({
    width:160,
    panelHeight: 120,
    data: citysource,
    valueField: 'cityid',   //getValue组合框中提取出来的值
    textField: 'city'   //组合框显示的值
});
```

④编写组合框onSelect事件。当用户选中组合框的某个选项时，onSelect事件被触发。

```
$("#city").combobox({
    onSelect: function(record) {   //选中事件
        if (record) {
```

```
                alert(record.city+'---'+record.cityid+'---'+$("#city").combobox('getValue'));
            }
        }
    });
```

⑤使用select方法设置组合框的初值，也可以用setValue方法设置初值。两者不同之处在于select方法可以触发onSelect事件，而setValue不会触发事件。

```
$('#city').combobox('select', citysource[0].city);
$('#city').combobox('setValue', citysource[0].city);
```

从上面代码可以发现，静态组合框的定义也是比较繁琐的。其实，组合框的定义过程也可以用一个函数实现，下面给出 myComboField()自定义函数的参数及应用示例。

```
myComboField(id,parent,label,labelwidth,top,left,height,width,items,value,style)
```

这里，items 为静态组合框的选项，各个选项之间用分号分隔。例如定义一个“职称”组合框：

```
myComboField('title', 'myFieldset1', '职称：', 70, 300, 16, 0, 120, '教授;副教授;讲师;助教;其它',
'副教授', 'autodrop');
```

这里，“职称”组合框共 5 个选项，其初值为“副教授”。style 参数值 autodrop 表示组合框为非编辑状态，即点击组合框时就自动弹出其选项下拉框。myComboField()函数的具体代码见 Easyui_functions.js 文件。

相对于静态组合框，基于数据库的动态组合框的定义更为复杂些，有关内容将在第 3 章相关实例中介绍。

实例 9. 利用滑块（slider）控件实现图片的等比例缩放。

滑块（slider）允许用户从一个有限的范围内选择一个数值，当沿着轨道移动滑块控件时，该控件将显示一个表示当前值的提示框，用户可通过设置它的属性来自定义滑块。可以使用<input>创建滑块控件，例如：

```
<input class="EasyUI-slider" value="12" style="width:300px" data-options="showTip:true, rule:[0,
'|', 25, '|', 50, '|', 75, '|', 100]" >
```

上面使用<input>方法在CSS中设置滑块的绝对位置是无效的。也可以通过 创建滑块，但是这时value属性是无效的。这里我们通过<input>创建滑块，在滑块外添加一个层 ，用来设置滑块的绝对坐标位置。

图2-7. 利用滑块控件实现图片等比例缩放程序运行界面

本例在表单中定义一个HTML图片控件（image1），学生学号为一个组合框（combobox），每个学生对应一个图片文件，其文件名为姓名拼音与png文件的组合（如jiabaoyu.jpg）。当用户在组合框中选定某个学生时，触发组合框的onSelect事件，这时image1控件中即时显示该学生对应的图片文件。滑块用于调整图片的大小，图片可以最大放大至100%。resizeImage()函数根据图片原始大小，等比例计算图片缩放后的宽度和高度，保证图片大小变化过程中不被扭曲。本实例程序运行界面图2-7所示，其主要程序实现方法与过程如下。

①定义表单myForm1，在表单中添加多个控件，其中包括学号对应的组合框数据源数组studentsource及组合框控件stuid。

```
var studentsource=[
   {"stuid":"D2014540101","stuname":"贾宝玉","pycode":"jiabaoyu","gender":"男",
    "birthdate":"1996-02-10", "place":"浙江省杭州市"},
   {"stuid":"D2014540102","stuname":"林黛玉","pycode":"lindaiyu","gender":"女",
    "birthdate":"1996-07-15", "place":"浙江省温州市"},
   {"stuid":"D2014540103","stuname":"薛宝钗","pycode":"xuebaocha","gender":"女",
    "birthdate":"1995-12-20", "place":"浙江省绍兴市"}
];
myForm('myForm1','main','学生信息',0,0,315,520,'');
myFieldset('myFieldset1','myForm1','基本信息',10,10,240,255);
myLabelField('stuid','myFieldset1','学号：',36*0+25,12,0,0);
myTextField('stuname','myFieldset1','姓名：',70,36*1+20,12,0,160);
myTextField('pycode','myFieldset1','汉语拼音：',70,36*2+20,12,0,160,'');
myDateField('birthdate','myFieldset1','出生日期：',70,36*3+20,12,0,120,'1997-2-17');
myComboField('gender','myFieldset1','性别：',70,36*4+20,12,0,120,'男;女','');
myTextField('place','myFieldset1','出生地：',70,36*5+20,12,0,120,8,2,'60.25','10','100');
//定义学生编码对应的组合框
$("#myFieldset1").append('<div style="position: absolute; top:22px; left:82px; width: 100px;
 padding-left: 4px;" id="stuid_div"><input class="EasyUI-combobox" id="stuid"
 style="padding:0px 6px 0px 6px" /></div>');
 $("#stuid").combobox({
      width:160,
      panelHeight: 'auto',
      editable: false,
      data: studentsource,
      valueField: 'stuid',
      textField: 'stuid'
 });
```

②在表单中定义学生照片图片控件image1和滑块控件slider1，并设置图片的src（文件路径和名称）值为空，即先暂时不显示图片。

```
var str='<img id="image1" src=" " style="position: absolute; top:18px; left:285px;"></img>';
str+='<div style="position: absolute; top:230px; left:290px; width:200px;">';
str+='<input id="slider1" class="EasyUI-slider" value="0" ></div>';
$("#myForm1").append($(str));
//定义slider控件
$("#slider1").slider({
      showTip: true,
      min:1,    max:100,
      rule: [0,'|',25,'|',50,'|',75,'|',100],
      tipFormatter: function(value){
            return value;
      },
});
```

③编写滑块控件的onChange事件，即当滑块在轨道上移动时，触发该事件，调用函数resizeImage()，将滑块当前值value传给函数，以改变学生照片的大小，对图片进行等比例缩放。

```
$("#slider1").slider({
    onChange: function(value){
        var src=$("#image1").attr("src");
        resizeImage('image1',src,value);
    }
});
```

④编写学生组合框的onSelect事件，即当组合框中选定某个学生时，从数组studentsource中提取全部数据，将该学生的姓名、拼音、性别、出生日期等信息赋值到表单对应的控件中去；同时将图形控件中的src设置为该学生学号对应的图形文件名，并调用函数重新显示图片。

```
$("#stuid").combobox({
    onSelect: function(record) {   //定义选中事件
        if (record){
            $("#stuname").textbox("setValue",record.stuname);
            $("#pycode").textbox("setValue",record.pycode);
            $("#gender").textbox("setValue",record.gender);
            $("#birthdate").textbox("setValue",record.birthdate);
            $("#place").textbox("setValue",record.place);
            $("#slider1").slider('setValue',30);
            var src="system/images/"+record.stuid+".jpg";
            $("#image1").attr('src',src);
            resizeImage('image1',src);
        }
    }
});
```

⑤设置组合框的初值为第一个学生，滑块的初值为30。这时由于滑块值发生变化，其onChange事件被触发；同时由于组合框选择了一个学生，其onSelect事件也被触发。

```
$("#slider1").slider('setValue',30);
$('#stuid').combobox('select',studentsource[0].stuid);
```

⑥编写图片缩放程序。在图片的onload事件中编写程序，先计算出图片的大小尺寸，即图片的高度和宽度；然后根据滑块值，按百分比等比例缩放图片的高度和宽度。

```
function resizeImage(img,src){
    var image=new Image();
    image.src=src;
    ratio=$("#slider1").val()/100.0;
    image.onload=function() {   //必须放在onload事件中
        var aheight=image.height;
        var awidth=image.width;
        if (ratio!=0){ //调整图片大小,按比例缩放
            awidth = awidth * ratio;
            aheight = aheight * ratio;
            $("#image1").css({width: awidth, height: aheight});
        }
    }
}
```

实例 10. 工具栏、菜单和消息框控件的使用。

EasyUI 菜单由 EasyUI-menubutton 类指定。菜单是树型结构，即每个菜单项（menu item）之下还可以有子菜单项。主菜单和子菜单可以通过 的嵌套来实现，也可以通过主菜单的 data-options 选项中定义 menu 属性来指定子菜单。菜单项的图标可由 iconCls 属性指定，菜单项之间的分隔符由 menu-sep 类表示。简单的菜单定义举例如下：

```
<div id="mm" class="EasyUI-menu" style="width:120px;">
    <div>New</div>
    <div>
     <span>Open</span>
     <div style="width:150px;">
          <div><b>Word</b></div>
          <div>Excel</div>
          <div>PowerPoint</div>
     </div>
    </div>
    <div data-options="iconCls:'icon-save'">Save</div>
    <div class="menu-sep"></div>
    <div>Exit</div>
</div>
```

菜单项对应的事件可以在主菜单（即 class="EasyUI-menu"的菜单）的 data-options 中指定，也可以按下列方法定义：

```
$('#mm').menu({
     onClick:function(item){
          alert(item.id);
     }
});
```

消息框（messager）提供不同样式的信息提示框，包括警示（alert）、确认（confirm）、提示（prompt）、进展（progress）等。所有的消息框都是异步的。用户可以在与消息框交互后使用回调函数来完成一些动作。

本例在页面中定义一个工具栏、5 个 linkbutton 按钮、一个菜单按钮及其下拉菜单条、一个右键菜单。在“新增”、“删除”和“保存”按钮事件中使用$.messager 消息框进行信息提示和操作确认。本例程序运行界面如图 2-8 所示，其主要程序实现方法与过程如下：

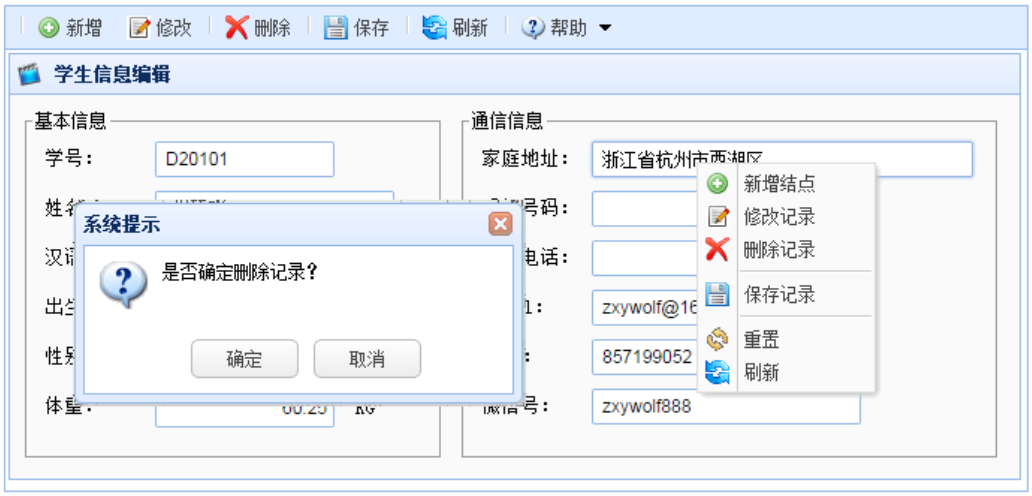


图2-8　工具栏、菜单和消息框程序运行界面

①首先使用layout控件将屏幕分成上下两部分，在屏幕上方定义一个工具栏；在工具栏中定义若干个EasyUI-linkbutton按钮和一个EasyUI-menubutton菜单（“帮助”），每个按钮可定义一个事件。

```
<div id='layout' class="EasyUI-layout" style="height:500px">
    <div id='toolbar' class='EasyUI-panel' data-options="region:'north'"
      style="overflow:hidden; background-color: #E0ECFF; height:30px; padding: 1px 1px 1px 10px;">
        <a href="#" class="btn-separator"></a>
        <a href="#" id="btnadd" xtype="button" class="EasyUI-linkbutton" data-options="iconCls:'addIcon',
          plain:true, onClick:fn_add" style="">新增</a>
        <a href="#" id="btnedit" xtype="button" class="EasyUI-linkbutton" data-options="iconCls:'editIcon',
          plain:true" style="">修改</a>
        <a href="#" class="btn-separator"></a>
        <a href="#" id="btndelete" xtype="button" class="EasyUI-linkbutton" data-options="iconCls:'deleteIcon',
          plain:true, onClick:fn_delete" style="">删除</a>
        <a href="#" class="btn-separator"></a>
        <a href="#" id="btnsave" xtype="button" class="EasyUI-linkbutton" data-options="iconCls:'saveIcon',
          plain:true, onClick:fn_save" style="">保存</a>
        <a href="#" class="btn-separator"></a>
        <a href="#" id="btnrefresh" xtype="button" class="EasyUI-linkbutton"
          data-options="iconCls:'refreshIcon',
          plain:true" style="">刷新</a>
        <a href="#" class="btn-separator"></a>
        <a href="#" xtype="button" class="EasyUI-menubutton" data-options="menu:'#btnhelp_menuitem',
          iconCls:'helpIcon'">帮助</a>
    </div>
    <div id='bottom' data-options="region:'center'" style="padding: 2px 0px 0px 2px; "></div>
</div>
```

②使用层嵌套方法，定义上面“帮助”菜单（btnhelp_menuitem）的两个子菜单项。

```
<div id="btnhelp_menuitem" style="width:150px;">
    <div id="btnbookonline" data-options="iconCls:'pdfIcon'">操作说明</div>
    <div id="btndemo" data-options="iconCls:'wordIcon'">实例演示</div>
</div>
```

③定义一个右键菜单myMenu1及其它的点击事件fn_myMenu1Click。该右键菜单的子菜单项通过appendItem()方法在javascript语句中动态添加实现。

```
<div id="myMenu1" class="EasyUI-menu" data-options="onClick: fn_myMenu1Click" style="width: auto;"></div>
```

④利用appendItem()方法，逐个添加右键菜单myMenu1的六个子菜单项和两个菜单的分隔线。

```
$("#myMenu1").menu('appendItem', {text: '新增结点',id: 'mnadd', iconCls:'addIcon'} );
$("#myMenu1").menu('appendItem', {text: '修改记录',id: 'mnedit', iconCls:'editIcon'} );
$("#myMenu1").menu('appendItem',
    {text: '删除记录',id: 'mndelete', iconCls:'deleteIcon', onclick: function(){ fn_delete(); }});
$("#myMenu1").menu('appendItem', {separator:true});
$("#myMenu1").menu('appendItem',
    {text: '保存记录',id: 'mnsave', iconCls:'saveIcon', onclick: function(){ fn_save(); }});
$("#myMenu1").menu('appendItem',{separator:true});
$("#myMenu1").menu('appendItem',{text: '重置',id: 'mnreset', iconCls:'resetIcon'}  );
$("#myMenu1").menu('appendItem',{text: '刷新',id: 'mnrefresh', iconCls:'refreshIcon'});
```

⑤绑定（bind）“刷新”按钮和菜单的click事件，这是另一种定义按钮和菜单事件的方法。

```
$("#btnrefresh,#mnrefresh").bind('click', function(e){ fn_refresh(e); });
```

⑥定义表单的右键点击事件。在表单myForm1中通过绑定其contextmenu事件，采用菜单的show()方法，实现右键菜单的弹出和触发。这里，e.preventDefault()用于禁止系统原有菜单的显示。

```
$('#myForm1').bind('contextmenu',function(e){
    e.preventDefault();  //关闭系统原有的右键菜单
    $('#myMenu1').menu('show', {
        left: e.pageX,
        top: e.pageY
    });
});
```

⑦编写各按钮和菜单的click点击事件。点击“新增”按钮，使用$.messager.show()缓缓弹出一个消息框；点击“保存”按钮，使用$.messager.alert()弹出一个信息提示框；点击“删除”按钮，使用$.messager.confirm()弹出一个信息确认框；点击“重置”菜单，在重置表单各个控件值之后，使用$.messagershow()缓缓弹出一个信息提示框，2秒钟后消息框自动消失。

```
function fn_add(){
    $.messager.show({
        title:'系统提示',
        msg:'系统已切换至新增记录模式。<br>表单已经清空。',
        showType:'show',
        width:200,
        timeout:0,
        style:{
            top: 100
        }
    });
}
function fn_save(){
    $.messager.alert('系统提示','数据已经保存成功！','info');  //icon可选项：error、question、warning
}
function fn_delete(){
    $.messager.confirm('系统提示','是否确定删除记录？',function(r){
        if (r){
            alert('确定删除');
        }
    });
}
function fn_refresh(){
    alert('refresh');
}
function fn_myMenu1Click(item){
    if (item.id=='mnreset'){
        $("#main")[0].reset();
        $.messager.show({
            title:'系统提示',
            width:200,
            msg:'表单数据已经被重置。',
            timeout:2000,
            showType:'slide'
        });
    }
}
```

实例 11. 窗口控件 window 的应用及其函数构造。

在EasyUI中，窗口（window）是一个浮动的、可拖拽的面板，可以当作应用程序窗口使用。默认情况下，窗口可拖动、可调整尺寸、可关闭。窗口既可以通过静态HTML定义，也可以通过JS语句动态创建。

本例使用layout布局，将屏幕分成上、下两部分，其中屏幕上方为一个工具栏。当用户点击工具栏中的“修改”按钮时，利用window("open")方法打开一个子窗口（myWin1），显示学生的详细信息。本例程序设计的主要过程和方法如下。

①利用HTML层 创建窗口myWin1控件。这里style属性中已经定义了窗口的尺寸大小。

```
<div id="myWin1" class="EasyUI-window" title="学生信息" data-options="iconCls:'panelIcon'"
style="position:relative; width:340px; height:560px; padding:0px;"></div>
```

②在javascript语句中，改变窗口的一些常用属性，其中{modal: true}表示该窗口为模态窗口，也就是说，在打开子窗口时，用户再也无法点击操作主窗口中的任何控件。

```
$("#myWin1").window({
    closable:true, collapsible:false, resizable:false,
    draggable:false, modal:true, maximizable:false
});
```

③在窗口中定义两个按钮，编写其中的“关闭”按钮的click事件，点击该按钮时关闭窗体myWin1。

```
$("#myWin1").append("<input type='button' value='确定' id='btnok' />");
$("#myWin1").append("<input type='button' value='关闭' id='btnclose' />");
$("#btnok").css({position:'absolute', top:485, left:170, width:65});
$("#btnclose").css({position:'absolute', top:485, left:236, width:65});
$("#btnclose").on('click', function () {
    $("#myWin1").window('close');
});
```

思考题：

如何构造一个窗口定义函数？需要包括哪些参数？参考Easyui_function.js文件中的myWindow()函数。

实例 12. 利用布局（layout）控件实现页面布局。

在 EasyUI 中，布局控件（layout）将屏幕分成五个区域（北区 north、南区 south、东区 east、西区 west 和中区 center），每个区域都是一个容器，用于包含其他控件。中间区域面板（region: center）是必需的，边缘区域面板是可选的。每个边缘区域面板可通过拖拽边框调整尺寸，也可以通过点击折叠触发器来折叠面板。布局可以嵌套，嵌套在里层的布局也分为 5 个区域，因此利用布局可建立复杂的页面排版。当布局的选项 fit 设置为 true 时，该布局的尺寸将与它的父容器相适应。因此，当在<body>标签上创建布局时，它将自动最大化到整个页面的全部尺寸。

本例程序运行界面如图 2-9 所示，具体布局过程与方法如下。

①在<body>标签中创建一个布局，在该布局中创建 4 个区域面板（其中东区缺省）。

```
<body id='main' class="EasyUI-layout" data-options="fit:true" style="margin: 1px 1px 1px 1px;">
    <div id='top' class='EasyUI-panel' data-options="region:'north'" style="overflow:hidden;
      background-color: #E0ECFF; height:30px; padding: 1px 1px 1px 10px;"></div>
    <div id='bottom' class='EasyUI-panel' data-options="region:'south'" style="height:60px;
      overflow:auto;"></div>
    <div id='left' class='EasyUI-panel' data-options="region:'west', split:true" style="overflow:auto;
      width:250px;"></div>
    <div id='middle' class='EasyUI-panel' data-options="region:'center', split:true"
      style="overflow:auto;"></div>
</div>
```

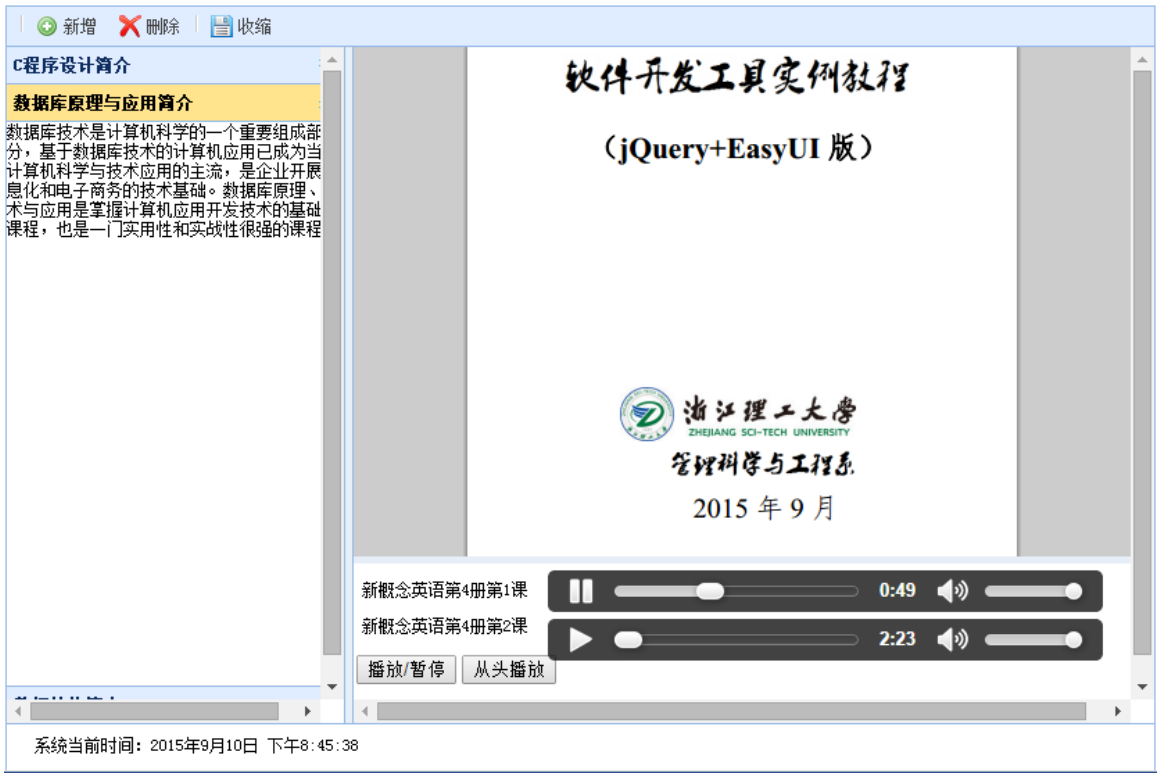


图 2-9　利用 layout 实现屏幕区域分割与布局

②在中间区域（id='middle'）中再创建另一个布局 middlelayout，并将该区域分成左右两个部分：middletop 和 middlebottom。

```
<div id='middle' class='EasyUI-panel' data-options="region:'center', split:true" style="overflow:auto;">
    <div id="middlelayout" class="EasyUI-layout" data-options="fit:true">
        <div id='middletop' class='EasyUI-panel' data-options="region:'north',split:true,
          border:false" style="overflow:auto;height:300px;"></div>
        <div id='middlebottom' class='EasyUI-panel' data-options="region:'center', split:true,
           border:false" style="overflow:auto;"></div>
    </div>
</div>
```

③在左侧区域（id='left'）中添加一个 EasyUI-accordion 类型的（手风琴）可折叠控件，在该可折叠容器中添加 3 个栏目。

```
<div id='left' class='EasyUI-panel' data-options="region:'west', split:true"
  tyle="overflow:auto; width:250px;">
    <div class="EasyUI-accordion" data-options="fit:true, border:false">
        <div id='content1' title="C 程序设计简介"></div>
        <div id='content2' title="数据库原理与应用简介" data-options="selected:true"></div>
        <div id='content3' title="数据结构简介"></div>
    </div>
</div>
```

④在 javascript 中编写程序，在 bottom 区域实时显示当前系统时间，第一个手风琴可折叠面板中加载一个服务器端的文本文件，在第 2 和第 3 个可折叠面板中添加 HTML 文字描述。

```
$(document).ready(function() {
    setInterval(function() {    //实时显示系统当前时间
        var now = (new Date()).toLocaleString();
```

```
            $('#bottom').text("系统当前时间："+now);
        },1000);
    //提取服务器的文本文件
    $('#content1').load('mybase/c-programming.txt');
    //在手风琴控件中添加文字
    $('#content2').append("数据库技术是计算机科学的一个重要组成部分，基于数据库技术的计算机应用已成为当今计算机科学与技术应用的主流，是企业开展信息化和电子商务的技术基础。数据库原理、技术与应用是掌握计算机应用开发技术的基础性课程，也是一门实用性和实战性很强的课程。");
    $('#content3').html("《数据结构》的前半部分从抽象数据类型的角度讨论各种基本类型的数据结构及其应用；后半部分主要讨论查找和排序的各种实现方法及其综合分析比较。");
```

⑤编写程序，在右上角区域（middletop）使用<object>标签显示一个 PDF 文件（test1.pdf），设置其类型为"application/pdf"。

```
var str='<object data="mybase/test1.pdf" type="application/pdf" width="100%" height="100%" >\n';
str+='</object>';                    //swf ,,application/x-shockwave-flash, x-mplayer2
$('#middletop').append($(str));
```

⑥编写程序，在右下角区域（middlebottom）使用<audio>标签添加一个 mp3 文件播放器，可以同时选择播放两个 mp3 文件。只有当出现 controls 属性时，audio 控件及其进度控制条才会在屏幕上显示出现。也可以添加按钮，用程序代码控制 mp3 的播放过程。

```
str='<p> 新概念英语第 4 册第 1 课</p>'+
'<audio id="main_audio1" src="mybase/lesson4-01.mp3" controls="controls" autoplay="autoplay" preload="auto" loop="loop" '+
'style="position:absolute; top:5px;left:140px; width:400px;"></audio>'+
'<p> 新概念英语第 4 册第 2 课</p>'+
'<audio id="main_audio2" src="mybase/lesson4-02.mp3" controls="controls" '+
'style="position:absolute; top:40px;left:140px; width:400px;"></audio>'+
'<p><button onclick="audioPlay()">播放/暂停</button>'+
'<button onclick="goToFirst()">从头播放</button></p>';
$('#middlebottom').append($(str));
```

相关知识：

1. <audio> 标签是 HTML 5 的新标签。autoplay 属性表示音频在就绪后马上播放，controls 属性向用户显示控件（比如播放按钮）。Loop 属性表示音频结束时重新开始播放。Src 指定要播放的音频的 URL。

2. 可通过 append 和 html 方法添加 HTML 语句。

3. 在使用<object>标签时，flash 文件（.swf）的类型设置为“application/x-shockwave-flash”。

实例 13. 获取表单中可编辑控件的名称、类型及其值。

在jQquery中，可以通过多种方法获取页面中的所有可编辑控件的名称、类型和值。不同类型的控件其取值方法不同。一般控件可直接使用jQuery语句$(#id).val()取值，而textbox和combobox控件需要使用getValue方法取值。checkbox控件的取值比较复杂，可以在判断其选中状态的基础上，使用attr()方法从属性中取值。

本例使用$('input, select, textarea').each(function(index)方法遍历一个页面所有控件，利用循环，先使用this和attr(‘type’)分别提取控件的标识符和类别，然后根据控件类别采用不同的取值方式获取控件内容值，并将结果按照JSON格式存放到一个data变量中，在一个textarea框中显示出来。具体方法和代码如下：

```
$('input, select, textarea').each(function(index){
    var input = $(this);
    var id=input.attr('id');
    var value=undefined;
    var type=input.attr('type');
    var hidden=input.attr('hidden');
    if (id!=undefined){
        if (type=='text' && hidden!='hidden'){
            value=input.textbox('getValue');
        }else if (type=='combobox'){
            value=input.combobox('getValue');
        }else if (type=='checkbox'){
            if (input.is(':checked')) value=input.attr('xtext');
        }else if (type!='button'){
            value=input.val();
        }
        if (value!=undefined){
            if (data!='') data+=',';
            data+='"'+id+'":"'+value+'"';
            console.log(id+'----'+type+'----'+value);
        }
    }
});
data='{'+data+'}';
$("#xformvalues").val(data);
```

由于获取一个控件值的程序比较繁琐，因此可以专门构造一个自定义函数，在给定一个控件ID值的基础上，返回其内容值。具体方法可参考Easyui_function.js文件中的myGetInputValue()函数。

本例所属的控件遍历方法也可应用于其他事务处理中。例如，将一个表单中的所有可编辑控件内容清空，或设置为只读状态等。

作业题：

1．根据页面中各个控件名称及其内容值，生成一条数据库Insert语句和update语句。

2．在利用$('input, select, textarea').each(function(index)获取页面控件时，如何避免那些不是表单中的控件，如本例中的xformvalues，它是window窗口中的控件。

实例 14. 表单的键盘控制与控件聚焦。

在Web应用开发中，键盘操作控制可以通过绑定控件的keydown事件实现。本例创建的表单中包含文本框、日期框、组合框、复选框等多种控件。键盘控制要求在页面初始化时将光标聚焦在第一个控件（学号）上，当用户按下回车键、向下键（↓）、向上键（↑）时，光标自动将聚焦到相应的下一个或上一个控件上。本例键盘控制的具体实现过程和程序如下：

①通过$('input, select, textarea').each(function(index)遍历方法，将上述不同类型的控件与keydown事件绑定，具体控制键盘操作的程序在函数fnKeyDownEvent()中实现。

```
$('input, textarea').each(function(index){
    var input = $(this);
    var id=input.attr('id');
    var value=undefined;
    var type=input.attr('type');
    var hidden=input.attr('hidden');
    if (hidden!='hidden' && id!=undefined && type!='checkbox'){
        if (type=='textarea'){
```

```
                $('#'+id).on('keydown', 'input', function(e){
                    fnKeyDownEvent(e,id);
                });
            }else{
                $('#'+id).textbox('textbox').bind('keydown',function(e){
                    fnKeyDownEvent(e,id);
                });
            }
        }
    });
```

②在fnKeyDownEvent函数中，使用e.which获取当前键盘值（即用户按下的键盘值）；通过页面遍历，将各个控件的标识符和类别存放到数组xcmp和xtype中，并求出当前控件的数组下标，存放到变量xno中；根据键盘值找到新聚焦控件的标识符。例如，当键盘值为回车键或向下键时，新聚焦控件为xcmp[xno+1]；当键盘值为向上键时，新聚焦控件为xcmp[xno-1]。

在 jQuery 和 EasyUI 中，textarea 控件的遍历与聚焦方式与其它控件不同，它可直接使用 jQuery 语句实现，例如：$("#"+id).focus()；而文本框等其它控件则使用不同的聚焦方式，例如：$("#"+id).next("span").find("input").focus();

```
    function fnKeyDownEvent(e,id){
        var key=e.which;
        var xcmp=[];
        var xtype=[];
        var i=0;
        if (key==13 || key==40 || key==38){    //38--up    40--down
            var xno=-1;
            $('input, textarea').each(function(index){
                var input = $(this);
                field=input.attr('id');
                type=input.attr('type');
                hidden=input.attr('hidden');
                if (field!=undefined && hidden!='hidden' && type!='checkbox'){
                    if (id==field) xno=i;
                    xcmp[i]=field;
                    xtype[i]=type;
                    i++;
                }
            });
            if (xno<xcmp.length && xno>=0){
                var n=0;
                if (key==13 || key==40 ){    //向下
                    if (xno<=i) n=xno+1;
                    else n=i;
                }else if (key==38 ){     //向上
                    if (xno>0) n=xno-1;
                    else n=0;
                }
                var xnewcmp=xcmp[n];
                var xtype=xtype[n];
                if (xtype=='textarea') $("#"+xnewcmp).focus();
                else $("#"+xnewcmp).next("span").find("input").focus();
            }
        }
    }
```

③当光标聚焦到某个控件时，实现全选该文本框中的内容，类似于有些软件开发工具中的selectAll 的效果。

```
function fnSelectOnFocus(){
$('input').on('focus', function() {
    var $this = $(this)
    .one('mouseup.mouseupSelect', function() {
        $this.select();
        return false;
    })
    .one('mousedown', function() {
        $this.off('mouseup.mouseupSelect');
    })
    .select();
});
```

本例键盘控制程序仅适合于 textbox、datefield、combobox、textarea 等类型控件，对 checkbox 控件不作处理。

思考题：

如何判断 textarea 控件中光标在输入框中的第一行第一个字符的位置，这时如果按向上键，则将光标聚焦到上一个控件上（set cursor position in a text field）。

实例 15. 利用文件框控件 file 实现文件的上传。

文件上传的方法有很多，jQuery也有许多文件上传控件。本例采用最基本的HTML文件框（<input type='file'）实现文件上传，不限制上传文件的类型和大小。具体地，在表单myForm1中添加一个学生组合框，每个学生可以上传一个照片的图形文件到服务器端，图形文件以学号命名。点击“上传”按钮后，客户端调用服务器端程序Easyui_fileUpload.jsp文件。当文件上传成功后，通过调用 myresizeImage()函数将图形等比例缩放后在表单规定位置中显示出来。客户端程序实现的主要方法如下：

①使用jsp语句获取工程文件路径，在javascript中获取此值，以便设置照片上传后在服务器端存放的路径。

```
<%
    String path = request.getContextPath();
    String basePath = request.getScheme()+"://"+request.getServerName()+":"+
    request.getServerPort()+path+"/";
%>
<script>
    var photopath='<%=basePath %>'+'mybase/students/';
    …
</script>
```

②定义表单、学号组合框及图片等控件。

③在表单myForm1中添加一个文件选择框控件file1（input type="file"），同时添加一个“上传”按钮。绑定file1控件的change事件，只有当选择的文件为非空时，“上传”按钮才处理激活状态。编写“上传”按钮的点击事件，点击调用fnUpload()函数开始上传文件。

```
$("#myForm1").append('<input type="file" id="file1" style="position: absolute; top:260px; left:16px;
  width: 200px; padding-left: 4px;" />');
myButton('btnupload','myForm1','上传',260,440,25,65,'uploadIcon','');
$('#file1').bind('change',function(v){
    var filename = $("#file1").val();
    if (filename=='')    $("#btnupload").linkbutton('disable');
    else $("#btnupload").linkbutton('enable');
});
```

```
$('#btnupload').bind('click',function(e){   //绑定“上传”按钮的点击事件
    fnUpload();
});
```

④编写fnUpload()程序。先获取文件扩展名并作判断，然后使用FormData()方法和XMLHttpRequest对象，调用后台服务器端程序system//Easyui_fileUpLoad.jsp，将目标文件名和照片文件存放路径两个参数传递给该程序。服务器端上传文件成功后，返回文件的有关信息（如文件大小、文件扩展名等），并将文件选择控件置空、“上传”按钮为未激活状态，结束文件上传过程。由于游览器缓存图片，新上传的图片不能马上在客户端显示出来，需要在图片地址src中添加一个时间戳（timestemp）。

```
var fileext=filename.substring(filename.lastIndexOf(".")+1,255).toLowerCase();//文件扩展名
var fileObj = $("#file1")[0].files[0]; //  获取文件对象
var form = new FormData();   // FormData  对象
form.append("file", fileObj);//  文件对象
var xhr = new XMLHttpRequest(); //XMLHttpRequest  对象
xhr.open("post", "system//Easyui_fileUpLoad.jsp?targetname="+targetname+"&targetpath=mybase/students", true);
xhr.onload = function () {
    if(xhr.status == 200){
        var data = JSON.parse(xhr.responseText);
        if (data.error == 0) {       //上传成功
            var src=photopath+$("#stuid").combobox("getValue")+'.jpg?timestemp='+
            new Date().getTime();
            $("#image1").attr('src',src);
              resizeImage('image1',src,231,224);
             $.messager.show({
                title:'系统提示',
                width:200,
                msg:'文件已经上传成功!',
                timeout:2000,
                showType:'slide'
            });
              console.log('文件上传成功!'+src+'，文件大小：'+data.filesize);
        }else{
            $.messager.alert('系统提示','文件已上传失败！','error');
            console.log(data.message);
        }
        $("#file1").val(' ');
    }
};
xhr.send(form);
```

程序2-15. 文件上传服务器端程序Easyui_fileUpLoad.jsp

```
<%@ page language="java" import="java.util.*" pageEncoding="UTF-8"%>
<%@ page import="java.io.*"%>
<%@ page import="net.sf.JSON.JSONObject"%>
<%@ page import="org.apache.commons.fileupload.FileItem"%>
<%@ page import="org.apache.commons.fileupload.disk.DiskFileItemFactory"%>
<%@ page import="org.apache.commons.fileupload.servlet.ServletFileUpload"%>
<%
    String path = request.getContextPath();
    String basePath = request.getScheme()+"://"+request.getServerName()+":"+request.getServerPort()+path+"/";
    request.setCharacterEncoding("UTF-8");
    response.setCharacterEncoding("UTF-8");
```

```
String targetpath = request.getParameter("targetpath");   //目标路径
String targetname = request.getParameter("targetname");   //目标文件名称
System.out.println("target="+targetpath);
String savePath = this.getServletContext().getRealPath(targetpath);
File file = new File(savePath);
//判断上传文件的保存目录是否存在
if (!file.exists() && !file.isDirectory()) {
        System.out.println(savePath+"目录不存在，需要创建");
        file.mkdir(); //创建目录
}
String message = ""; //消息提示
int error = 0;
String targetfilename="";
String fileext="";
long xsize=0;
try{
        //使用Apache文件上传组件处理文件上传步骤：
        //1、创建一个DiskFileItemFactory工厂
        DiskFileItemFactory factory = new DiskFileItemFactory();
        //2、创建一个文件上传解析器
        ServletFileUpload upload = new ServletFileUpload(factory);
        //解决上传文件名的中文乱码
        upload.setHeaderEncoding("UTF-8");
        //3、判断提交上来的数据是否是上传表单的数据
        if(!ServletFileUpload.isMultipartContent(request)){
                //按照传统方式获取数据
                return;
        }
        //4、使用ServletFileUpload解析器解析上传数据，解析结果返回的是一个List<FileItem>集合，
        //每一个FileItem对应一个Form表单的输入项
        List<FileItem> list = upload.parseRequest(request);   //compiler5.0级以上支持
        for(FileItem item : list){
                //如果fileitem中封装的是普通输入项的数据
                if(item.isFormField()){
                        String name = item.getFieldName();
                        //解决普通输入项的数据的中文乱码问题
                        String value = item.getString("UTF-8");
                        //value = new String(value.getBytes("iso8859-1"),"UTF-8");
                        System.out.println(name + "=" + value);
                }else{//如果fileitem中封装的是上传文件
                        //得到上传的文件名称和扩展名
                        String filename = item.getName();
                        fileext=filename.substring(filename.lastIndexOf(".")+1,filename.length()).toLowerCase();
                        targetfilename=filename;
                        if (!targetname.trim().equals("")) targetfilename=targetname+"."+fileext;   //目标文件名
                        if(filename==null || filename.trim().equals("")){
                                continue;
                        }
//注意：不同的浏览器提交的文件名是不一样的，有些浏览器提交上来的文件名是带有路径的，如：c:\a\b\1.txt，而有些只是单纯的文件名，如：1.txt
                        //处理获取到的上传文件的文件名的路径部分，只保留文件名部分
                        filename = filename.substring(filename.lastIndexOf("\\")+1);
                        //获取item中的上传文件的输入流
                        InputStream in = item.getInputStream();
```

```
                    //创建一个文件输出流
                    //FileOutputStream out1 = new FileOutputStream(savePath + "\\" + filename);
                    FileOutputStream out1 = new FileOutputStream(savePath + "\\" + targetfilename);
                    //创建一个缓冲区
                    byte buffer[] = new byte[1024];
                    //判断输入流中的数据是否已经读完的标识
                    int len = 0;
                    //循环将输入流读入到缓冲区当中，(len=in.read(buffer))>0就表示in里面还有数据
                    while((len=in.read(buffer))>0){
                      //使用FileOutputStream输出流
                      //将缓冲区的数据写入到指定的目录(savePath + "\\" + filename)当中
                        out1.write(buffer, 0, len);
                    }
                    in.close();   //关闭输入流
                    out1.close();   //关闭输出流
                    item.delete(); //删除处理文件上传时生成的临时文件
                    message = "文件上传成功！";
                    File xfile = new File(savePath + "\\" + targetfilename);
                    xsize=xfile.length();
                }
            }
        }catch (Exception e) {
            message= "文件上传失败！";
            error = 1;
            e.printStackTrace();
        }
        //确定返回内容JSON格式
        Map map = new HashMap();
        map.put("error", error);
        map.put("message", message);
        map.put("targetfilename", targetfilename);
        map.put("fileext", fileext);
        if (xsize<=1024) map.put("filesize", xsize+"B");
        else if (xsize<=1024*1000) map.put("filesize", xsize/1024.00+"KB");
        else map.put("filesize", xsize/1024/1024.00+"MB");
        JSONObject JSON = JSONObject.fromObject(map);
        response.setContentType("text/html; charset=utf-8");
        PrintWriter pw = response.getWriter();
        System.out.print(JSON.toString());
        pw.write(JSON.toString());
    %>
```

实例 16. 服务器端文件下载的实现。

从服务器端下载文件到客户端的实现方式有很多。本例从服务器端下载图片文件和wod文档至客户端。首先在表单中添加一个图形控件image1和学生组合框，每个学生的照片对应于一个图片文件，其文件名为学号与jpg的组合（如D2014540101.jpg）。图片文件下方添加一个学生简介的超链接，其链接文件名为学号与doc文件的组合（如D2014540101.doc）。下面给出三种不同的下载方式。

①直接使用location.href赋值方式，例如：

```
window.document.location.href='system/images/'+$("#stuid").combobox("getValue")+".jpg";
```

在有些游览器中，这种方式会在当前网页中打开下载文件，从而改变当前打开的网页地址。

②使用超链接<a href>方式将下载文件包含在链接地址中，例如：

```
<a href="mybase//D2014540101.doc">贾宝玉</a>
```

这种方式将服务器端文件打开后直接下载到本地游览器，而无需编写服务器端程序，但有些游览器会打开一个新的页面窗口。

③利用服务器端程序（Easyui_fileDownLoad.jsp）实现文件下载，可实现多个文件下载并命名下载文件。具体方法和程序如下：

客户端程序：

```
//指定源文件
var xsourcefilename=$("#image1").attr("src");
//指定目标文件
var xtargetfilename=$("#stuname").textbox("getValue")+".jpg";
//调用服务器端程序Easyui_fileDownLoad.jsp
var url='system//Easyui_fileDownLoad.jsp?path='+xsourcefilename+'&name='+xtargetfilename;
window.location.href=url;
```

服务器程序：Easyui_fileDownLoad.jsp代码：

```
<%@page contentType="application/x-msdownload" %>
<%@page language="java" import="java.util.*" pageEncoding="utf-8" %>
<%@page import="com.StringUtil" import="java.io.*" %>
<%@page import="java.io.BufferedInputStream" %>
<%@page import="java.io.BufferedOutputStream" %>
<%@page import="java.net.URLEncoder" %>
<%
//输入两个参数：一个源文件名称sourcefile，一个目标文件名targetfile
String root = application.getRealPath("/");
String sourcefile = StringUtil.getToUtf8(request.getParameter("source"));
String targetfile = StringUtil.getToUtf8(request.getParameter("target"));
targetfile=URLEncoder.encode(targetfile,"UTF8");  //必须转换，否则中文乱码
String result="";
String s=root+sourcefile;
String s1=s.substring(0,s.lastIndexOf("\\"));  //所在文件夹
String s2=s.substring(s.lastIndexOf("\\")+1,s.length());
File f=new File(s1,s2);
if (!f.exists()) {
    result="源文件没有找到，下载失败！";
}
response.setHeader("Content-Type","application/x-msdownload;");
response.setHeader("Content-disposition","attachment; filename="+targetfile+"");
BufferedInputStream bis = null;
BufferedOutputStream bos = null;
try {
    bis = new BufferedInputStream(new FileInputStream(root+sourcefile));
    bos = new BufferedOutputStream(response.getOutputStream());
    byte[] buff = new byte[10 * 1024];
    int bytesRead;
    while ( -1 != (bytesRead = bis.read(buff, 0, buff.length))) {
        bos.write(buff, 0, bytesRead);
    }
    bos.flush();
} catch (IOException ioe) {
    System.out.println("下载错误：" + ioe);
    result="文件下载错误：<BR>"+ioe;
```

```
} finally {
        if (bis != null) bis.close();
        if (bos != null) bos.close();
}
if (bos !=null ){
        bos.flush();
        bos.close();
}
bos = null;
response.flushBuffer();
out.clear();
out = pageContext.pushBody();
%>
```

需要指出，对于不同游览器和操作系统，文件下载、保存以及打开方式有所差异。

相关知识点：

①使用tool属性，在表单panel中添加自定义按钮（本例添加了3个按钮）

②图片等比例缩放函数resizeImage()。

③组合框的onSeelct事件。

④ 中元素的去除（remove()方法）。

作业题：

1. 数据查询窗体的设计。
2. 设计一个简单的计算器

第 3 章、EasyUI 数据库控件

实例 17. 构造连接数据库 Java 类，实现数据查询操作。

与服务器数据库进行连接操作是动态网页设计的基础。J2EE连接数据库的一种常用方法是构造一个java类，在各类数据库操作时调用这个类，统一进行数据库连接处理，。

本例创建一个连接SQL Server数据库的类DBConn，建立一个带参数的数据库连接对象程序setConnection，当用户指定数据库服务器名称（或地址）、数据库用户、登录密码和连接的数据库名时，该程序通过JDBC/jtds驱动程序连接SQL Server数据库。数据库端口号为1433，默认数据库服务器名称为localhost，数据库用户为sa，数据库名为emlab，登录密码为sql2008。

在连接数据库时，需要在SQL Server中设置下列选项（具体设置方法见第一章）。

①在SQL Server配置管理器中，设置网络配置，将MSSQLServer协议中的TCP/IP启用，同时将IP地址中的IPALL栏目上的TCP端口设置为1433；

②重新启动SQL Server服务。

创建DBConn的步骤如下：

①将jtds1.2.jar驱动程序包添加到当前工程的referenced libarary中去；

②在src下新建一个package，取名com；

③在com下新建一个class，取名setConnection；

④在public Connection setConnection中编写程序。

程序3-1-1. 数据库连接程序DBConn类代码

```
package com;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
public class DBConn {
    public DBConn() {
    }
    public Connection setConnection(String host, String password, String dbname, String username){
        Connection conn = null;
        String result="";
        if (host.equals("")) host="localhost";
        if (password.equals("")) password="sql2008";
        if (dbname.equals("")) dbname="examples";
        if (username.equals("")) username="sa";
        String url = "jdbc:jtds:sqlserver://"+host+":1433;DatabaseName="+dbname;
        try{
            Class.forName("net.sourceforge.jtds.jdbc.Driver");
            conn = DriverManager.getConnection(url, username, password);
            result="success";
        }catch (ClassNotFoundException e){
            e.printStackTrace();
            result=e.getMessage();
        }catch (SQLException e){
            e.printStackTrace();
            result=e.getMessage();
        }
        return conn;
    }
}
```

借助DBConn类，可以在后台JSP程序中连接数据库，实现各类数据库操作。其中数据查询操作（select语句）的主要步骤如下：

①在JSP页面中创建一个Connection对象，调用DBConn类，指定数据库服务器、sa用户、登录密码和数据库名等参数（database默认值为：localhost sa sql2008 emlab），连接数据库；具体语句如下：

```
DBConn con=new DBConn();
Connection connection=con.getConnection(database);  //database为数据库连接字符串
```

②创建一个Statement对象，生成具有给定类型和并发性的ResultSet对象；具体语句如下：

```
Statement stmt=connection.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE,
ResultSet.CONCUR_READ_ONLY);
```

③使用executeQuery()和getResultSet()方法执行查询语句，生成一个ResultSet对象rs（结果集）；具体语句如下：

```
stmt.executeQuery(sqlString);
ResultSet rs=stmt.getResultSet();
```

④利用循环对rs逐行进行输出处理，将查询结果保存到缓存变量中，具体语句如下：

```
ResultSetMetaData rsmd=rs.getMetaData();  //获取列信息
rs.beforeFirst();
while(rs.next()) {  //循环提取各行数据
        ……
  }
```

对结果集进行处理时，rs.last()指向查询结果集的最后一行，此时的行号rs.getRow()就是查询结果集的总行数，rs.getString()获取rs当前行中某一列的值。输出数据符合JSON格式，该格式数据举例如下：

```
[
   {"areaid":"110000","areaname":"北京市"},
   {"areaid":"120000","areaname":"天津市"},
   {"areaid":"130000","areaname":"河北省"},
   {"areaid":"140000","areaname":"山西省"},
   ……
   {"areaid":"650000","areaname":"新疆维吾尔自治区"}
]
```

这里，每行内容由大括号{ }包含，大括号内为某一行各个列的名称及其值，列名和列值用双引号包含，各列之间用逗号分隔。

⑤将缓存变量中的查询结果返回给客户端。具体语句如下：

```
response.getWriter().write(record.toString());
```

本例包含客户端和服务器端两个程序。服务器端程序调用DBConn类连接数据库，在数据库emlab中执行一条客户端发送的SELECT查询语句，从areas表中提取省份编码和名称。客户端程序通过$.ajax调用服务器端程序Easyui_getComboxData.jsp，使用success方法接受服务器端返回的结果，并将查询结果显示在一个textarea控件中，具体语句如下：

```
$.ajax({
        url: "system\\Easyui_getComboxData.jsp",
        data: { database: sysdatabasestring, sqlString: xsql },
        async: false,
        success: function(data) {
                $("#result").val(data);
        }
});
```

这里，客户端向服务器端发送两个变量：①数据库连接字符串（localhost sa　sql2008 emlab），该字符串将数据库服务器名、数据库用户名、登录密码、数据库名等四个参数组合在一起，中间用tab键分隔。传至服务器端后，后台JSP程序用split语句再把这个字符串分解为四个参数，这样处理的目的是为了减少传送变量的数量；②SELECT查询语句，可以包含合法的CTE、衍生表等。本实例程序运行界面如图3-1所示。

图3-1　JSP数据库查询操作以及JSON格式数据返回界面

实例 18. 服务器端数据更新语句的执行。

数据更新语句主要是指除查询语句之外的其他SQL语句，如INSERT、UPDATE、DELETE、CREATE等语句。由于这些语句运行之后不直接返回结果集，因此在服务器端的程序与前面所述的查询语句不同，通常是通过executeUpdate()方法实现。客户端可以将多条数据更新语句组合在一起，一次性地传递给服务器端运行，但不同语句之间不能出现GO语句。

本例利用服务器端程序实现数据库更新操作，包括新建数据表、插入记录、删除记录等。点击“创建”按钮，在数据库emlab中新建一个省份表myProvinces；点击“数据”按钮，使用insert…select语句将areas表中的省份编码和名称等列批量插入到新表中；点击“删除”按钮，删除省份表中的前20行。每次数据更新后，查询语句将myProvinces表中最新数据以JSON格式显示在一个textarea控件中。本例程序运行界面如图3-2所示，服务器端实现数据库更新操作的主要步骤如下。

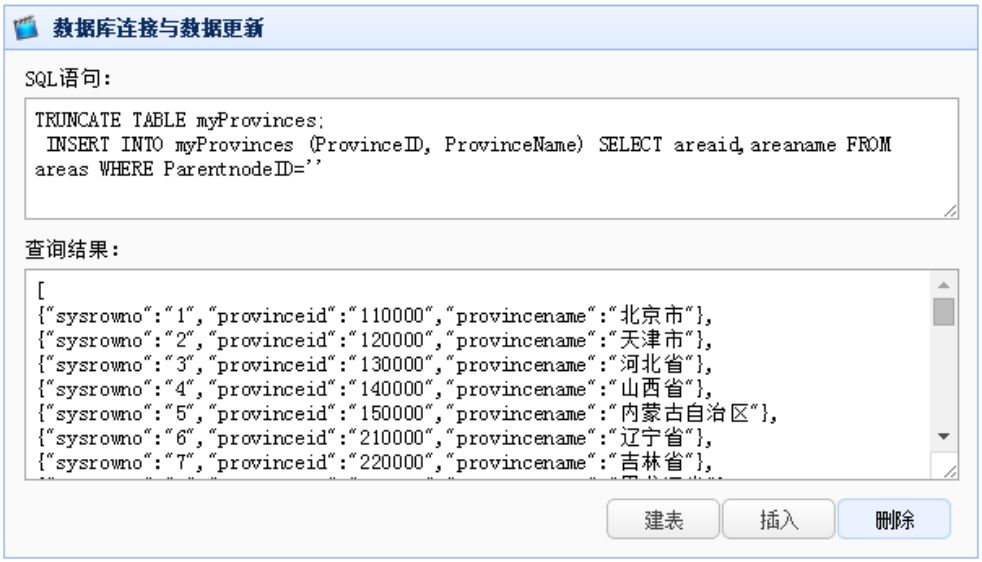


图3-2　JSP数据库更新操作程序运行界面

①创建Connection对象，调用DBConn类，连接数据库，具体方法与数据查询相同。

②创建一个Statement对象，具体方法与数据查询相同。

③使用executeUpdate()方法执行数据更新语句，并利用try判断数据更新是否成功。如果失败，则通过e.getMessage()返回错误信息。

④向客户端返回SQL命令执行结果。

程序3-18：服务器端数据库更新程序代码。

```
<%@ page language="java" import="java.util.*"    import="java.sql.*,com.DBConn"
pageEncoding="utf-8"%>
<%@page import="com.StringUtil"%>
<%
   Statement stmt=connection.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE,ResultSet.
   CONCUR_READ_ONLY);
   if (!updatesql.equals("")){
      try{
            stmt.executeUpdate(updatesql);
      }catch (SQLException e){
            errmsg=e.getMessage();
      }
   }
   stmt.close();
   connection.close();
   System.out.println("errmsg="+errmsg);
   if (errmsg.equals("")){
      response.getWriter().write("{'error':''}");
   }else{
      response.getWriter().write("{'error':\""+errmsg+"\"}");
   }
%>
```

其他知识点：

①JS中全局变量与局部变量的定义

sql=" "; //（全局变量，不带var）

var sql=" "; //（局部变量，带var）

②客户端解析服务器端返回的JSON数据方法。

```
success: function(data) {
      eval("var result="+data);   //服务器端返回的值赋值到result变量中
}
```

③linkbutton按钮状态设置。

```
$("#cmdinsert").linkbutton("enable");   //激活
$("#cmdinsert").linkbutton("disable");   //未激活
```

思考题：

1．本例在使用ajax在服务器端执行数据更新语句之后还需要执行数据查询语句，而查询语句的执行也要使用ajax。怎样在服务器端编写一个JSP程序，既可以执行数据更新语句有可以执行查询语句？

2．在客户端编写判断程序，当myProvinces表中没有记录时，将cmddelete按钮变成未激活（灰色）状态。

实例 19. 服务器端存储过程和用户定义函数的调用。

SQL Server存储过程和用户定义函数都可以包含输入/输出参数。标量用户定义函数可作为一个列进行输出（通常需要为其指定一个别名），因此可使用查询语句返回其计算结果。在JSP中，存储过程的调用相对比较复杂些，其核心是参数的定义、赋值及返回取值，主要方法和步骤如下：

①连接数据库emlab，与执行查询语句和更新语句相同；

②通过CallableStatement的prepareCall()方法，用问号（？）指定存储过程的输入输出参数个数；

CallableStatement stmt=connection.prepareCall("{ call sys_getpycode(?, ?) }");

③根据输入参数的类型，使用setString("@输入参数", 参数值)方法为输入参数赋值。日期型、整型和浮点型输入参数分别使用setDate()、setFloat()、setInt()方法赋值；

stmt.setString("@str",chnstr);

④使用registerOutParameter(("@参数名称", 参数类型)方法，指定输出参数的类型。字符型参数使用java.sql.Types.VARCHAR，日期型、整型和浮点型数据分别使用Types.INTEGER和Types.FLOAT。

stmt.registerOutParameter("@pycode",java.sql.Types.VARCHAR);

⑤使用execute()方法执行存储过程，计算出结果；

stmt.execute();

⑥使用getString("@输出参数")等方法获取来自存储过程输出参数的返回值。

String str=stmt.getString("@pycode");

⑦使用JSON格式将服务器端各个输出参数的结果只返回至客户端。

response.getWriter().write("{pycode:'"+str+"'}");

本例调用服务器端的一个SQL Server存储过程sys_getPycode（具体代码见sys_getpycode.sql），其功能是输入一个中文姓名，返回该姓名对应的汉语拼音。这里，将stuname和pycode这两个文本框控件的buttonIcon属性分别设置为locateIcon和helpIcon，这时文本框的右侧各出现一个小按钮。点击小按钮，将触发文本框的onClickButton事件，客户端程序将通过ajax调用服务器端的存储过程或用户定义函数，服务器计算出汉语拼音后返回至客户端，由客户端程序将拼音值赋值到pycode控件中。可以发现，用户定义函数的调用比存储过程要简便很多，在服务器端只需要调用通用查询程序即可。本例程序运行界面如图4-3所示，服务器端完整程序见example19_getpycode_server.jsp，客户端程序的设计过程和方法如下。

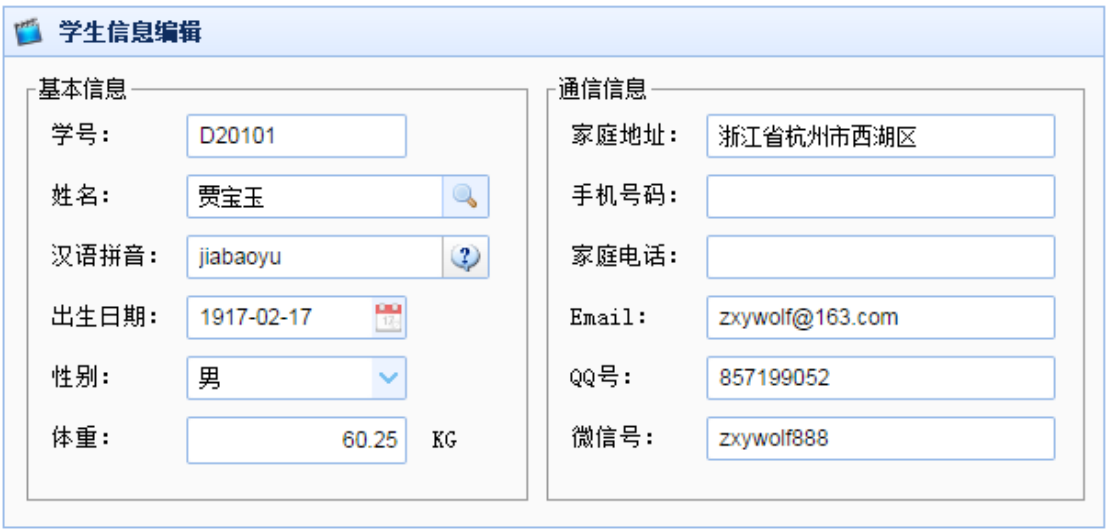


图4-3　客户端调用服务器端存储过程和用户定义函数

①在表单中定义学生相关信息的控件。

②在学生姓名对应的文本框(stuname)的右侧添加一个小图标按钮，在其点击事件中编写代码，通过example19_getpycode_server.jsp程序调用服务器端的存储过程，然后将存储过程计算结果返回，赋值给拼音对应的文本框中去。

```
$('#stuname').textbox({
    buttonIcon:'locateIcon',
      onClickButton: function(e){
        $.ajax({
            url: "example19_getpycode_server.jsp",
            data: { database: sysdatabasestring, chnstr: $("#stuname").textbox("getValue") },
            async: false,
            success: function(data) {
                eval("var result="+data);   //返回存储过程输出参数的结果
                $("#pycode").textbox("setValue",result.pycode);
            }
        });
    }
});
```

③采用类似的方法，在汉语拼音对应的文本框（pycode）的右侧添加一个小图标按钮，在其点击事件中编写代码，将用户定义函数dbo.sys_GenPycode()作为查询语句的一个列进行输出（别名为pycode），调用服务器端程序easyui_execSelect.jsp后将用户定义函数的结果返回客户端。

```
$("#pycode").textbox({
    buttonIcon:'helpIcon',
    onClickButton: function(e){
    var xsql="select dbo.sys_GenPycode('"+$("#stuname").textbox("getValue")+"') as pycode";
    $.ajax({
        url: "system\\easyui_execSelect.jsp",
        data: { database: sysdatabasestring, selectsql:xsql },
        async: false,
        success: function(data) {
            eval("var result="+data);
            $("#pycode").textbox("setValue",result[0].pycode);
        }
    });
}
```

思考题：

是否可以在stuname控件的onChange事件中编写程序实现同步显示学生姓名的拼音？

实例 20. 动态 combobox 组合框控件的定义与初值设置。

组合框的数据源也可以是基于远程服务器端符合规定格式的数据，有时把这种从数据库中取数据的组合框称为动态组合框。在EasyUI中，动态与静态组合框的定义相同，不同的只是数据来源。可以有不同的方法实现动态组合框。本例采用ajax将数据库查询语句从客户端传送给服务器端，服务器端将查询结果以数组形式返回客户端，然后与组合框的data属性相绑定，这样组合框中就出现数据库中数据选项。下面以“党派（party）”组合框为例，具体阐述其中的定义步骤：

①使用JS语句在myFieldset1中定义party组合框控件，在组合框之外添加一个层，用于确定控件的显示位置。

```
var   str='<div id="party_div"><input class="EasyUI-combobox" id="party"></div>';
$("#myFieldset1").append($(str));
```

②设置层的样式，即层的绝对坐标位置（可以调用myTextCss()函数实现）。

```
$("#party_div").css({position:'absolute',  padding:'0px  2px  0px  4px',  top:'185px',  left:'86px',
'z-index':2});
```

③定义党派组合框属性值，包括组合框宽度、下拉选项的行数（按像数计算）、内容列与显示列（valueField、textField）等。

```
$("#party").combobox({
    width:160,
    panelHeight: 'auto',   //自动计算行数，选项比较少时适用
    valueField: 'party',
    textField: 'party'
});
```

④采用ajax调用服务器端程序Easyui_getComboxData.jsp，客户端发送查询语句sql.party给服务器端；服务器端将查询结果按JSON格式返回给客户端的data变量，data变量经过eval计算后赋值给source数组变量；将source数组与组合框的data属性绑定，并选定第一个选项为组合框的初值。

```
$.ajax({
    type: "Post",
    url: "system/Easyui_getComboxData.jsp",
    data: {database: sysdatabasestring, selectsql: jqsql.party},
    async: false,
    success: function(data) {
        //返回的数据用data获取内容,直接复制到客户端数组source
        var source=eval(data);
        $("#party").combobox({data: source}); //绑定data属性，从此产生选项数据
          //设置组合框初值为第一个选项
        if (source.length > 0) {
            $("#"+id).combobox('select', source[0].party);
        }
    },
    error: function(err) {
        console.log(err);
    }
});
```

这里，Easyui_getComboxData.jsp是一个服务器端后台程序，该程序对于给定的一条select查询语句，可以按JSON格式返回查询结果（包括所有行和列），存放在一个数组中。

从上述过程可以看出，动态组合框的定义是比较繁琐的。本例在定义其他4个组合框时，从数据库中提取选项的这部分程序由一个函数myGetComboxData(id,sql)实现。该函数包含两个变量id和sql，分别代表控件的标识符和查询数据的SELECT语句。

思考题：

1．能否通过调用myComboField()函数（静态组合框）和myGetComboxData()实现动态组合框的快速定义？

2．实例中城市组合框选项有2000多条，如何与省份选项相关联进行过滤。

实例 21. 动态 combobox 组合框控件之间联动效果的实现。

在实际应用中，不同combobox组合框之间可能存在数据上的相互关联，即一个组合框中的选项数据源与另一个组合框的取值有关。例如在实例20中，教师出生地所在的省份与城市都是组合框，但城市选项依赖于省份选项，即当某个省份被选定时，所在城市的选项应当限制在这个省份所属的那些城市内，这就是说两个组合框之间存在联动效应。本例实现省份与城市组合框之间的联动效应，程序运行界面如图3-3所示，主要实现过程和方法如下。

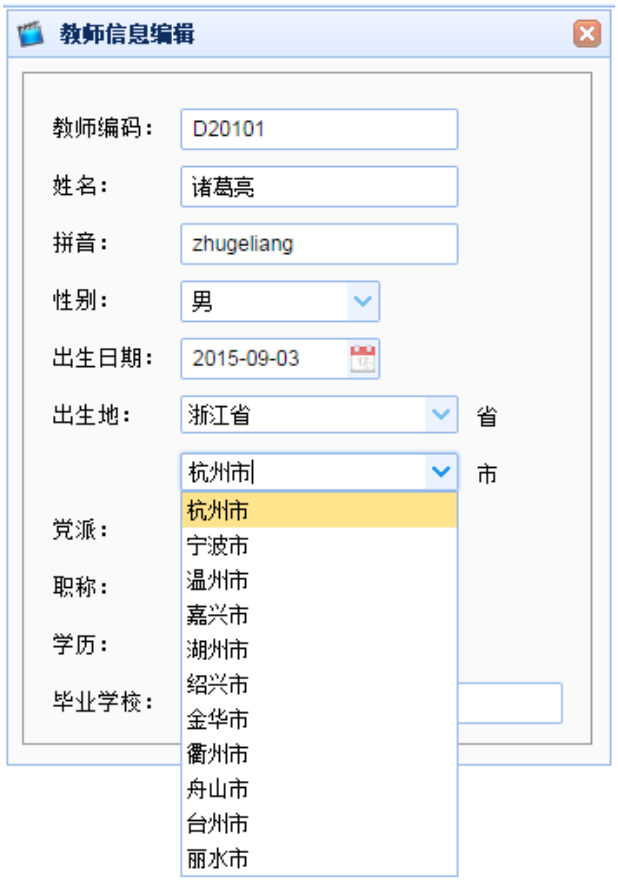


图3-3. 动态组合框combobox之间联动的实现

①定义省份和城市两个动态组合框，城市组合框先不从数据库中取数，即为空选项，而省份组合框调用myGetComboxData()函数从数据库中提取选项。

```
$("#provinceid").combobox({
    width:160,
    panelHeight: 'auto',
    valueField: 'provinceid',
    textField: 'province'
});
$("#cityid").combobox({
    width:160,
    panelHeight: 'auto',
    valueField: 'cityid',
    textField: 'city'
});
myGetComboxData('provinceid',jqsql.province);   //体用函数从数据库中提取省份选项的数据
```

②在省份组合框的onSelect事件中编写程序，当用户选定某个省份时，编写城市组合框的查询语句，即时从数据库中提取城市组合框的选项值，具体程序如下：

```
$("#provinceid").combobox({
    onSelect: function(record) {   //定义选项被选中的事件
        if (record) {   //判断选项是否为空
            var xvalue = $("#provinceid").combobox('getValue');   //提取省份选项的值
            //确定城市选项取数的查询语句
            var sql="select * from ("+jqsql.city+") as p where provinceid='"+xvalue+"'";
            $("#city").combobox('clear');   //清空城市组合框中原来的选项
            myGetComboxData('cityid',sql);   //从数据库中提取新的城市选项
        }
    }
});
```

有关combobox的定义可用控件自定义函数myDBComboField()快速实现：

myDBComboField('provinceid','myFieldset1','出生地：', 70, 33*7+20, 16, 0, 135, jqsql.province, 'province',"");

思考题：

1．实例中省份选项的初值为空，城市选项也为空。为省份选项的设置一个初值，这时城市组合框是否有选项，选项是否正确？

2．组合框过滤是指当用户输入一个字符串时，组合框中只显示与其关键字（通常为拼音助记码）中相匹配的那些选项，而将其他选项剔除或过滤掉，以快速定位和选择。EasyUI 组合框是否具有选项过滤的功能？

实例 22. 服务器端取数据库中数据赋值到表单。

本例创建一个教师信息表单，要求输入教师编码，点击“刷新”按钮后，客户端程序从服务器提取该教师的全部信息，并以JSON格式返回数据，然后填充到表单的各个控件中去。如果教师编码对应的记录不存在，则予以警告提示。本实例实现步骤和方法如下：

①将页面中除教师编码之外的所有控件设置为只读状态。不同类型的控件其只读状态的设置方法不同。例如：checkbox和combobox可以使用disable方法禁用。

```
$('input, select, textarea').each(function(index){
    var input = $(this);
    id=input.attr('id');
    type=input.attr('type');
    hidden=input.attr('hidden');
    if (id!=undefined && hidden!='hidden'){
        if (type=='text') $("#"+id).textbox('readonly',true);
        else if (type=='textarea') $("#"+id).attr('readonly',true);
        else if (type=='combobox') $("#"+id).combobox('disable');
        else if (type=='checkbox') $("#"+id).attr('disabled',true);
        else $("#"+id).attr('readonly',true);
    }
});
```

上述控件只读状态的设置也可以直接调用自定义函数mySetFormReadonly()实现：

mySetFormReadonly(true);

②根据教师编码值，创建SELECT查询语句；调用ajax和服务器端公共查询程序Easyui_execSelect.jsp获取查询结果；将查询结果赋值到result数组变量中。

```
var sql="select * from teachers where teacherid='"+$("#teacherid").textbox('getValue')+"'";
var result=[];
```

```
$.ajax({
    type: "Post",
    url: "system/Easyui_execSelect.jsp",
    data: {database: sysdatabasestring, selectsql: sql},
    async: false,
    success: function(data) {
        //返回的数据用data获取内容,直接复制到客户端数组source
        eval("result="+data+";");
    }
});
```

上述服务器端执行查询语句的功能也可以调用自定义函数实现：

```
var data=myRunSelectSql(sysdatabasestring, sql);
eval("result="+data+";");
```

③由于result是数组变量，可以根据result.length的值来判断教师编码是否存在。如果教师编码在数据库中不存在，则用$.messager.alert()方法予以警告提示。

```
if (result.length==0){
    $.messager.alert('系统提示', '<br> 教师编码找不到！', 'error');
}
```

④如果数据库中找到教师编码，则将result中所有教师信息赋值到控件中（由于result是数组变量，而且查询结果只有一行，故使用result[0]表示这一行）。这个过程由$.each(result[0], function(id, value) { }方法实现，jQuery可以循环式地将JSON数据中每一个列的标识符id和值value取出来，如果该标识符id对应的控件存在，则把value值赋值给这个控件。需要注意的是，不同类型的控件其赋值方式也不同，这个与控件取值方法类似。

```
$.each(result[0], function(id, value) {   //each循环取值
    var input = $("#"+id);
    var type=input.attr('type');
    if (input!=undefined){
        if (type=='text'){
            input.textbox('setValue',value);
        }else if (type=='combobox'){
            input.combobox('setValue',value);
        }else if (type=='checkbox'){
            if (input.attr('masterid')==''){
                if (input.is(':checked')) input.attr('value',value);
            }
        }else{
            input.val(value);
        }
    }
});
```

相关知识点：

<BackSpace>键的禁用。当页面中有控件是只读状态时，点击键盘上的<BackSpace>键，一些游览器会退出当前页面返回到上一个地址的页面上去。本例调用函数myBanBackSpace()直接实现。

作业题：

编写一个函数mySetCmpReadOnly(fields)，要求输入一个控件名称集，设置该控件集中所有控件的只读状态，各个控件之间用分号分隔。

实例 23. 客户端和服务器端表单数据验证。

在 EasyUI 中，数据验证可分为客户端（本地）和服务器端（远程）两种层次。客户端又可以按照数据的基本规则实施控件级和表单级验证。控件级验证可以在控件定义时采用 validatebox 设置规则（例如是否为有效的 email、url 等），也可以利用扩展的 validatebox 正则表达式；表单级验证则在表单提交时根据业务规则进行判断，以确保数据在保存到数据库之前是完整正确的。

除此之外，有些数据的验证必须借助服务器端才能完成，例如新增一个客户，其客户编码是否存在重复；输入一个用户账号，判断该账号是否存在，等等。

本例创建一个表单，采用客户端和服务器端相结合对表单数据进行正确性验证。数据验证规则及其实现方法具体描述如下：

①控件级验证。email 地址和 homepage 主页必须符合规定格式。

直接利用控件的 validType 属性进行验证，代码如下：

```
$("#email").textbox({validType:'email'});
$("#homepage").textbox({validType:'url'});
```

②控件级正则表达式验证。教师姓名必须为有效的汉字；姓名拼音只能是英文字符；出生日期必须大于 1949-10-01 小于系统当前日期；教师编码由 7 位数字组成。

这项验证可以利用扩展的validatebox，具体方法是先定义一个规则名称（如integer、date、CHS等），然后在$.extend($.fn.validatebox.defaults.rules, {})中编写程序，具体代码如下：

```
$('#teacherid').textbox({
    validType:"integer"     //integer为自己定的规则名称，下同。
});
$('#birthdate').datebox({
    validType:"date"
});
$('#name').textbox({
    validType:"CHS"
});
$('#pycode').textbox({
    validType:"english"
});
//验证程序
$.extend($.fn.validatebox.defaults.rules, {
    date: {   //自定义的规则名称，下同
        validator: function(value, param){
            var now = new Date();
            var d1 = new Date('1949-10-01');
            var d2 = new Date(now.getFullYear(), now.getMonth(), now.getDate());
            var d3 = now.getFullYear()+'-'+now.getMonth()+'-'+now.getDate();
            return d1<=new Date(value) && new Date(value)<=d2;
          },
          message: '日期必须在1949-10-01与'+today+'之间！'
      },
    CHS:{   //验证汉字
        validator:function(value){
            return /^[\u0391-\uFFE5]+$/.test(value);
        },
        message:"教师姓名只能输入汉字！"
    },
    english : {// 验证英文字母
        validator : function(value) {
            return /^[A-Za-z]+$/i.test(value);
```

```
            },
            message : '姓名拼音只能输入英文字符！'
        },
        integer : {// 验证整数
            validator : function(value) {
                return /^[+]?[1-9]+\d*$/i.test(value);
            },
            message : '教师编码只能输入数字！'
        },
});
```

③表单级验证。教师编码不能为空，并由 8 数字组成，前 4 位为 1970~今年之间有效的年份；教师姓名不能为空；出生日期不能为空。

```
var errormsg=[];   //存放数据验证发现的错误信息
//先判断各个控件是否符合格式要求
if (!$("#teacherid").textbox('isValid')) errormsg.push('教师编码输入错误！');
if (!$("#name").textbox('isValid')) errormsg.push('教师姓名输入错误！');
if (!$("#pycode").textbox('isValid')) errormsg.push('姓名拼音输入错误！');
if (!$("#birthdate").datebox('isValid')) errormsg.push('出生日期输入错误！');
if (!$("#email").textbox('isValid')) errormsg.push('Email 地址格式错误！');
if (!$("#homepage").textbox('isValid')) errormsg.push('个人主页格式错误！');
//判断其他逻辑
var s1=$("#teacherid").textbox('getValue');
var s2=$("#name").textbox('getValue');
var s3=$("#province").combobox('getText');
var s4=$("#city").combobox('getText');
if (s1.length==0) errormsg.push('教师编码不能为空！');
if (s1.length!=8) errormsg.push('教师编码必须是 8 位数字！');
else if (s1.substring(0,4)<'1970' || s1.substring(0,4)>now.getFullYear()){
    errormsg.push('教师编码前 4 位年份超出范围！');
}
if (s2.length==0) errormsg.push('教师姓名不能为空！');
```

④服务器端验证。教师编码必须是独一无二的；由于教师籍贯所对应的省份和所在城市是可以编辑的，因此必须验证这两个值在地区表（areas）中是否合法存在的。

◆ 教师编码的唯一性验证：通过查询语句在 teachers 表中检索教师编码是否存在，如果存在重复，则查询结果返回值的数组长度大于 0。

```
var sql="select * from teachers where teacherid='"+s1+"'";
var result=myRunSelectSql(sysdatabasestring,sql);   //result为数组变量
if (result.length>0){
    errormsg.push('教师编码重复！');
}
```

◆ 省份及城市的存在性验证：通过查询语句按省份名称检索地区表 areas，如果省份名称不存在，则返回值的数组长度为 0。

```
var sql1="select areaid as provinceid from areas where areaname='"+s3+"'";
var result1=myRunSelectSql(sysdatabasestring,sql1);
if (result1.length==0){   //数组长度为0，表示无返回结果
    errormsg.push('省份名称输入错误！');
}
```

◆ 在存在的省份中去查找输入的城市是否存在，如果不存在或不属于所属的省份，则返回输入错误信息。

```
var sql2="select 1 as n from areas where areaname='"+s4+"' and
parentnodeid='"+result1[0].provinceid+"'";
```

```
var result2=myRunSelectSql(sysdatabasestring,sql2);
if (result2.length==0){
    errormsg.push('城市名称输入错误！');
}
```

⑤输出验证出错信息。前面各种验证发现的错误信息被存放到一个数组变量 errormsg 中，将该数组中的各条验证错误信息分行显示在一个消息警告框中。

```
if (errormsg.length>0){
    var str='';
    for (var i=0;i<errormsg.length;i++){
        str+='<br>';
        if (i>0) str+='<span style="padding:0px 0px 0px 42px;">'+errormsg[i]+'</span>';
        else str+=errormsg[i];
    }
    $.messager.alert('系统提示','数据验证发现下列错误，提交失败！<br>'+str,'error');
}
```

综上所述，由于控件级验证只能提示数据错误信息，而不能有效防止错误数据保存到数据库中去，因此完整有效的数据验证应当在表单提交前通过必要的程序设计才能实现。

实例 24. 服务器端数据库记录的增删改操作。

在Web开发技术中，许多应用程序是由数据库记录增加、修改、删除和查询等操作组合而成的，因此数据库记录的增删改操作是构成一个应用系统的最基本单元。

本例将前面所述的数据库连接、数据查询、数据库更新、数据验证等知识进行综合应用，构造一个教师信息编辑程序。页面的左侧使用一个表单（myForm1）将教师编码和名称通过HTML超链接显示出来，每个超链接记录一个序号，同时包含一个复选框checkbox控件。点击超链接或复选框，调用fnSetRecord ()函数，根据序号从教师数据源数组中提取数据，然后显示在页面右侧的表单（myForm2）控件中。

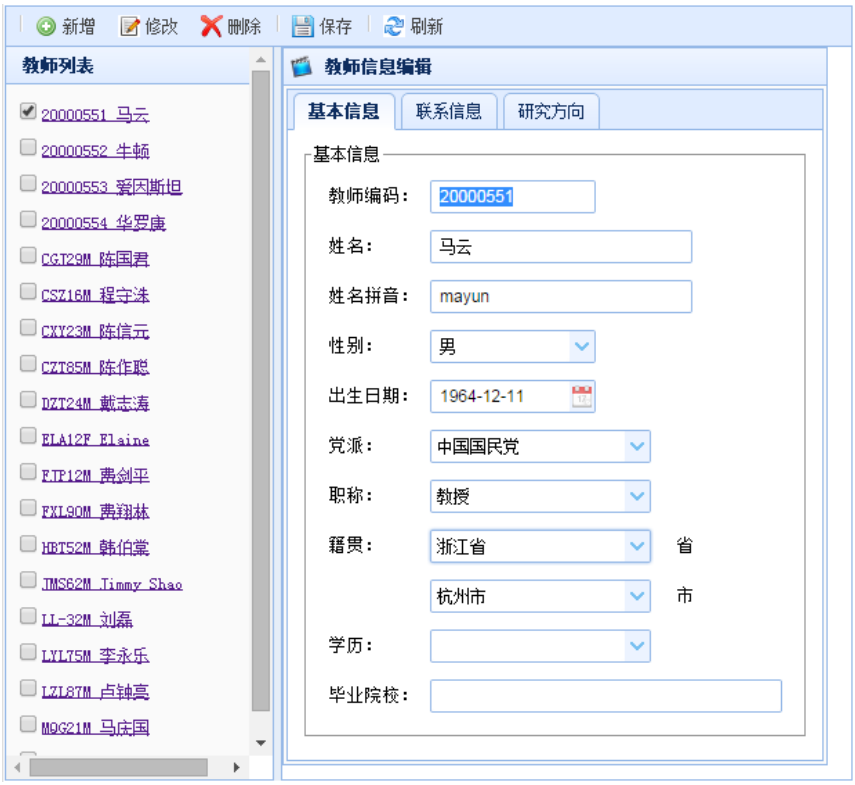


图3-4 数据库记录增删改操作实现

工具栏提供教师记录的新增、修改、删除、保存和刷新等按钮。新增教师时表单各控件值被重置；修改教师信息时教师编码被设置为只读，其他控件为可编辑状态；删除教师时提醒用户确认，使用DELETE语句直接删除记录；记录保存前从客户端和服务器端对数据进行验证；保存记录时对新增或修改数据这两种情况进行区分，这里使用一个隐藏控件变量（addoredit）进行标记。数据保存或删除后，调用fnRefresh()函数重新从数据库中提取教师信息显示在页面的左侧超链接中。本例程序运行界面如图3-4所示，具体实现过程和方法如下：

①构造fnRefresh()数据刷新函数，建立教师超链接。

◆ 从服务器端教师表（teachers）库中提取教师信息，保存到一个数组变量teacherdata中（JSON格式）。

```
var sql="select rtrim(teacherid)+' '+rtrim(name) as teacher,* from teachers";
var result=myRunSelectSql(sysdatabasestring,sql);
teacherdata=result;
```

◆ 构造教师信息的超链接，每个超链接包括一个层（用于坐标定位）和复选框checkbox，超链接内容为教师编码与教师姓名的组合。每个超链接的click事件调用一个fnSetRecord(*n*)函数，其功能是将当前超链接对应的教师信息从数组teacherdata中赋值到右侧表单控件中，因此在函数中需要一个参数记录教师序号*n*。

```
var str='';
for (var i=1;i<=result.length;i++){
    //定义每个教师的超链接
    str+='<div id="itemrowno'+i+'" style="position:absolute; left:6px; top: '+
    (26*(i-1)+10)+'px;" >';
    str+='<input type="checkbox" id="itemcheckbox'+i+'" onClick="return
    fnSetRecord('+( i)+');" >';
    str+='<a href="#" onClick="return fnSetRecord('+(i)+');" >' + result[i-1].teacher +
    '</a></div>';
}
```

◆ 将各个超链接添加到左侧的panel控件（myForm1）中，添加之前需先清空原有超链接。

```
$("#myForm1").empty();   //清除原来的超链接
$("#myForm1").append($(str));
```

◆ 超链接构造完成之后，调用fnSetRecord(1)函数，将光标定位在第一个教师超链接上。这时右侧屏幕显示第一位教师的基本信息。

```
fnSetRecord(1);
```

②在函数fnSetRecord(n)中，从数组teacherdata[]中取值，将超链接对应的第*n*个教师的数据赋值到表单myForm2的控件中。

◆ 直接调用自定义函数mySetFormValuesbyJSON加以实现。具体代码见Easyui_function.js文件中。

```
mySetFormValuesbyJSON(teacherdata[n-1]);
```

◆ 每个超链接都有一个checkbox复选框，但同时只能打钩选中一个（即第*n*个）。实现的方法是：利用循环和prop("checked", false)方法，打钩选中当前超链接上的checkbox，去掉其他超链接上的checkbox。

```
for (var i=1;i<=teacherdata.length;i++){
    if (i==n) $("#itemcheckbox"+i).prop("checked", true);
    else $("#itemcheckbox"+i).prop("checked", false);
    //$("#itemcheckbox"+i).prop("checked", (i==n)? true: false)
}
```

◆ 使用一个隐藏控件addoredit，记录当前操作为记录修改状态（即非记录新增状态）

```
$("#addoredit").val('edit');
```

将当前教师信息显示在右侧的myForm2上之后，将其控件设置成只读状态（readonly=true）。可以调用用户自定义函数mySetReadonly(' ', true)实现。具体代码如下。

```
function mySetReadonly(c, flag){   //对函数定义的控件（包括xid自定义属性的）只读
    //c为空时，对所有xid属性非空的控件设置只读
    var xfields=[];
    if (c!='') xfields=c.split(';');
    else{
        var k=0;
        $('input, select, textarea').each(function(index){
            var input = $(this);
            if (input.attr('id')!=undefined && input.attr('xid')!=undefined && input.attr('xid')!=''){
                xfields[k]=input.attr('id');
                console.log(xfields[k]+'----'+input.attr('xid'));
                k++;
            }
        });
    }
    var type='';
    var id='';
    var value='';
    var hidden='';
    var pid='';   //checkbox的大组名称
    for (k=0;k<xfields.length;k++){
        //var input = $(this);
        var input=$("#"+xfields[k]);
        id=input.attr('id');
        type=input.attr('type');
        hidden=input.attr('hidden');
        if (id!=undefined && hidden!='hidden'){
            if (type=='text') $("#"+id).textbox('readonly',flag);
            else if (type=='textarea') $("#"+id).attr('readonly',flag);
            else if (type=='combobox') $("#"+id).combobox('readonly',flag);
            else if (type=='checkbox') $("#"+id).attr('disabled',flag);
            else $("#"+id).attr('readonly',flag);
        }
    }
}
```

③在函数fnAdd()编写程序。在新增记录时完成下列各个操作：

◆ 将表单中原有教师信息清空（本例采用重置（reset）的方式），其中combobox和checkbox控件的值不重置。此项功能通过调用myreSetForm()自定义函数实现，其具体代码如下：

```
function myreSetForm(){   //重置表单
    var type='';
    var id='';
    var value='';
    var hidden='';
    $('input, select, textarea').each(function(index){
        var input = $(this);
        id=input.attr('id');
```

```
            type=input.attr('type');
            hidden=input.attr('hidden');
            if (id!=undefined){
                if (type=='text' && hidden!='hidden') $("#"+id).textbox('reset');
                else if (type=='textarea') $("#"+id).val('');
                else if (type=='combobox'){
                    //值不变
                }
                else if (type=='checkbox' || type=='checkboxgroup'){
                    //值不变
                }else{
                    $("#"+id).val('');
                }
            }
        });
    }
```

◆ 将表单中的各个控件设置为非只读（readonly=false），即可编辑状态。这项功能通过调用函数实现。

```
mySetFormReadonly(false);
```

◆ 由于右侧myForm2表单使用标签页（Tabs），新增记录时将光标锁定在第一个页面上，并聚焦教师编码这个控件上。

```
$("#myTab").tabs('select',0);
$("#teacherid").next("span").find("input").focus();
$("#teacherid").select();   //IE不支持
```

◆ 最后将addoredit值设置为新增add，因为在记录保存时需要区分是新增记录还是修改记录。

```
$("#addoredit").val('add');   //新增记录状态值
```

④在函数fnEdit()编写程序。在修改记录时完成下列各个操作，具体描述参考新增记录的方法。不同的是，修改记录时，主键（教师编码）不能修改，还是只读状态。

```
mySetFormReadonly(false);
$("#teacherid").textbox('readonly',true);   //新增记录
$("#myTab").tabs('select',0);
$("#teacherid").next("span").find("input").focus();
$("#teacherid").select();   //IE不支持
$("#addoredit").val('edit');   //修改记录
```

⑤在函数fnDelete编写程序。在删除记录时，使用$.messager.confirm()先由用户确定是否真的需要删除记录。待确定删除后，编写DELETE语句，调用函数myRunUpdateSql(sysdatabasestring,sql)执行数据库中记录删除操作。删除记录后需要刷新数据，即从数据库中重新提取数据，生成屏幕左侧的各个超链接，这项功能可直接调用前面所述的fnRefresh()函数实现。

```
var keyvalue=$("#teacherid").textbox("getValue");
$.messager.confirm('系统提示','删除教师编码'+keyvalue+"<br>是否确定？",function(r){
    if (r){
        var sql="delete teachers where teacherid='"+keyvalue+"'";
        var result=myRunUpdateSql(sysdatabasestring,sql);
        if (result.error==''){
            fnRefresh();
        }else{
            console.log(result.error);
        }
```

```
    }
});
function myRunUpdateSql(databasestring, sql){   //单行结果集
    result={};
    $.ajax({
        type: "Post",
        url: "system/Easyui_execUpdate.jsp",
        data: {database: sysdatabasestring, updatesql: sql},
        async: false,
        success: function(data) {
            //返回的数据用data获取内容,直接复制到客户端数组source
            eval("result="+data);
        },
        error: function(err) {
            console.log(err);
        }
    });
    return (result);
}
```

⑥在函数fnSave编写程序保存记录。与前面操作相比，记录保存往往是比较繁琐的。首先需要判断和验证数据是否正确，然后判断是新增记录之后还是修改记录之后进行保存，最后发送SQL数据更新语句由服务器端执行。具体过程和方法如下：

◆ 验证进行表单级和服务器端数据验证，本例通过调用数据验证函数fnValidation()实现，并将数据验证结果进行返回。这个函数有一个参数addoredit，用来区分是新增或修改j记录状态。数据验证如果没有发现错误，返回值1，否则返回0。

```
var addoredit=$("#addoredit").val();
var flag=fnValidation(addoredit);
if (flag==1){   //数据验证通过   }
```

◆ 数据验证正确后，如果是新增记录后保存那么编写一条insert语句，否则编写一条update语句。

```
if (addoredit=='edit'){   //update record
    var sql="update teachers set ";
    sql+="name='"+$("#name").textbox('getValue')+"',";
    sql+="pycode='"+$("#pycode").textbox('getValue')+"',";
    sql+="gender='"+$("#gender").combobox('getValue')+"',";
    …
    sql+="notes='"+$("#notes").val()+"'";
    sql+=" where teacherid='"+$("#teacherid").textbox('getValue')+"'";
    var result=myRunUpdateSql(sysdatabasestring,sql);
}else{   //add new record
    var sql="insert into teachers (teacherid,name,pycode,gender,birthdate,party,";
    sql+="title,province,city,degree,graduate,address,homephone,mobile,email,";
    sql+="weixin,homepage,qq,research,notes) values(";
    sql+="'"+$("#teacherid").textbox('getValue')+"',";
    sql+="'"+$("#name").textbox('getValue')+"',";
    …
    sql+="'"+$("#research").val()+"',";
    sql+="'"+$("#notes").val()+"'";
    sql+=")";
}
```

◆ 调用myRunUpdateSql(sysdatabasestring,sql)函数在服务器端执行SQL语句，之后刷新超链接数据，并将addoredit值设置为edit修改状态。

```
var result=myRunUpdateSql(sysdatabasestring,sql);
if (result.error==''){
    fnRefresh();
    $("#addoredit").val('edit');
}else{
    console.log(result.error);
}
```

作业题：

1．在fnRefresh()函数中补充程序，在保存数据后，将超链接定位到当前新增或修改的那条记录上，而不是目前的总是在第一条记录上。

2．将实例左侧的超链接改为一个DataList控件，实现教师记录增删改操作。

相关知识点：

①判断一个控件是否存在：if（$("#id").length>0)

②清空容器中的子项控件：$("#myForm1").empty();

③超链接中onclick事件的定义及触发：onClick="return fn(parm1,parm2..); "

④teacherdata变量需要定义在jQuery之外，即$(document).ready(function()之前。

实例 25. 服务器端 SQL 脚本文件.sql 的运行。

SQL Server脚本文件是一个以文本文件格式存储的SQL命令集。通常，脚本文件在SQL Server系统中运行时语句之间可以包含GO语句，但在服务器端运行时遇到GO语句时需要分段运行脚本。

本例在表单中创建一个可复选的组合框，即用户可以从下拉列表中选择一个或多个SQL脚本文件。在点击“运行脚本”按钮后，客户端将选中的脚本文件组合为一个字符串（文件之间用tab键分隔），通过调用服务器端程序Easyui_runSqlScriptFile.jsp，将选中的SQL脚本文件逐个在服务器端数据库系统中运行。如果发现脚本文件错误，服务器端返回出错信息，显示在客户端的一个文本框中。本例程序运行界面如图3-5所示，客户端程序的主要过程和代码如下。

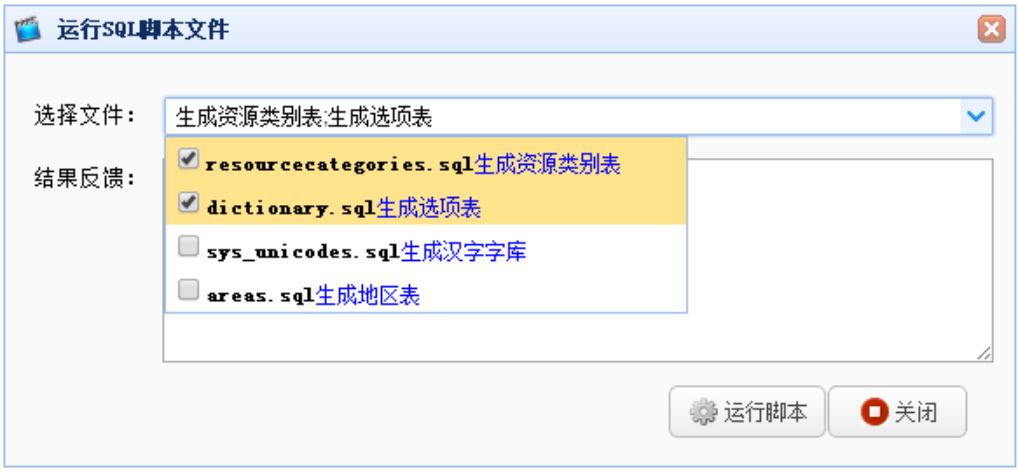


图3-5. SQL脚本文件的运行程序界面

①定义一个组合框数组，包含filename和filedesc两个列；在表单中定义一个组合框，设置该组合框的属性，将其multiple设置为true。EasyUI组合框具有多选的功能，但本身不带复选按钮。为此需要在组合框的formatter属性中指定一个formatItem函数，在每个选项条目中增加一个checkbox。

```
var filesource=[
```

```
{filename:"resourcecategories.sql", filedesc:"生成资源类别表"},
{filename:"dictionary.sql", filedesc:"生成选项表"},
{filename:"sys_unicodes.sql", filedesc:"生成汉字字库"},
{filename:"areas.sql", filedesc:"生成地区表"},
];
$("#sqlfile").combobox({
    width:575,
    panelHeight: 'auto',
    panelWidth: 300,
    editable: false,
    multiple:true,
    separator: ';',
    data: filesource,
    valueField: 'filename',
    textField: 'filedesc',
    formatter: formatItem
});
function formatItem(row){
    var s ='<input id="" type="checkbox" class="combobox-checkbox">'+
    '<span style="font-weight:bold">' + row.filename + '</span>'+
    '<span style="color:blue">' + row.filedesc + '</span>';
    return s;
}
```

②为复选框编写loadSuccess、select和unselect事件，使组合框中复选框状态与多选内容一致。

```
$("#sqlfile").combobox({
    onLoadSuccess:function(){
        var opts = $(this).combobox('options');
        var target = this;
        var values = $(target).combobox('getValues');
        $.map(values, function(value){
            var el = opts.finder.getEl(target, value);
            el.find('input.combobox-checkbox')._propAttr('checked', true);
        })
    },
    onSelect:function(row){
        var opts = $(this).combobox('options');
        var el = opts.finder.getEl(this, row[opts.valueField]);
        el.find('input.combobox-checkbox')._propAttr('checked', true);
    },
    onUnselect:function(row){
        var opts = $(this).combobox('options');
        var el = opts.finder.getEl(this, row[opts.valueField]);
        el.find('input.combobox-checkbox')._propAttr('checked', false);
    }
});
```

③在“运行脚本”按钮中编写点击事件代码，先将组合框中选中的脚本文件逐个提取出来，连接为一个字符串变量files中（文件名之间以tab键分割），然后调用服务器端Easyui_runSqlScriptFile.jsp，将SQL脚本文件、文件所在路径传递给服务器。服务器运行脚本后，如果发生错误，则在一个多行文本框textarea中显示错误信息。

```
$('#btnrun').bind('click',function(e){
    var files='';
    var records=$("#sqlfile").combobox("getValues");
    for (var i=0;i<records.length;i++){
     if (i>0) files+='';   //文件名之间以tab键分割
     files+=records[i];
    }
     $("#results").val('');
    $.ajax({
        url: "system\\Easyui_runSqlScriptFile.jsp",
        data: { database: sysdatabasestring, filename:files,filepath:'sqlscript' },
        async: false,
        success: function(data) {
            var message=data.trim()+'';
            if (message=='){
                $.messager.show({
                    title:'系统提示',
                    width:200,
                    msg:'脚本文件运行结束！',
                    timeout:2000,
                    showType:'slide'
                });
            }else{
                $("#results").val('错误信息返回：\n'+message);
            }
        },
        error: function(err) {
            console.log(err);
        }
    });
});
```

在服务器端JSP中运行SQL脚本文件的主要方法和步骤如下：

①服务器接受参数值；连接数据库。

②使用split分割多个SQL脚本文件。

```
String tmp[];
tmp=sqlfile.split("    ");   //tab
```

③利用循环逐个处理脚本文件。先判断脚本文件是否存在，如果存在则使用FileReader打开脚本文件。清空批语句执行缓存。

```
String realfile= realpath+"\\"+filepath+"\\"+tmp[i];
File f=new File(realfile);
if (f.exists()){
    FileReader fileReader = new FileReader(f);
    BufferedReader br = new BufferedReader(fileReader);
    …
```

④清空SQL批处理语句缓存；利用循环和readLine()方法逐行读取脚本文件，使用addBatch()将读取的SQL语句添加到缓存中；判断每一条读入的语句，如果是GO语句，则执行之前保存在缓存中的脚本语句。这样可以节省缓存，防止缓存和内存溢出。使用try语句，可以及时发现脚本文件中的可能的错误。一旦发现错误，则终止所有脚本文件运行。

```
stmt.clearBatch();
while((str = br.readLine() ) != null){
    str=str.trim();
    if (!str.toLowerCase().equals("go")){ //跳过脚本中的go语句
        stmt.addBatch("\n"+str);
    }else{
        //执行go之前的sql语句
        try{
            tmt.executeBatch();
        }catch (Exception e){
            message+=e.getMessage()+"\n";
        }
        stmt.clearBatch();
        }
    }
    try{
        stmt.executeBatch();
    }catch (Exception e){
        message+=e.getMessage()+"\n";
    }
    if (!message.equals("")) break;
}
```

第 4 章、树与数据网格及其应用

实例 26. 静态树 Tree 控件及其基本操作。

树（Tree）是一种常见的数据结构，在数据库应用系统中占有重要地位。树在网页中以树形结构显示分层数据。它向用户提供展开、折叠、拖拽、编辑和异步加载功能。在HTML中，树定义在<ul>元素中。该标记可定义叶节点和子节点。节点将是ul列表内的<li>元素。树也可以在一个空的<ul>元素中定义，使用javascript加载数据。

在EasyUI中，树可以有多个根节点（第一层节点都成为根节点），树中每个节点必须有Text和id两个列，分别为节点显示内容和节点关键字值。本例先利用javascript语句创建一个空的树，使用append方法在树中添加一个根节点（全部地区），然后在根节点之下添加若干个子节点（各个省份），并给出树中节点查找和选中的方法。本实例程序运行界面如图4-1所示，具体实现过程和代码如下。

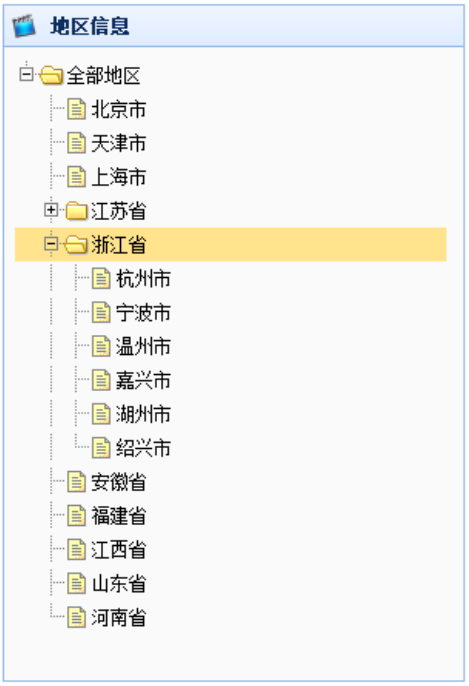


图4-1　树Tree控件及其基本操作

①使用jQuery定义树控件myTree1，设置树的lines属性为true（默认值为false）。使用append方法，在树中添加一个根节点（全部地区），设置其parent属性为null。

```
var str='<div id="myTree1" class="EasyUI-tree" style="width:380px; height:500px;
padding:5px;"></div>';
$("#myForm1").append($(str));
$("#myTree1").tree({ checkbox: false, lines:true});
//添加一个根节点
$("#myTree1").tree('append',{
    parent: null,
    data:{text:"全部地区", id:"*"}
});
```

②使用getRoot提取第一个根节点（与getRoots方法不同），先在该根节点之下添加一个子节点（北京市），再一次性添加8个子节点（8个省份）。

```
var root = $("#myTree1").tree("getRoot");
```

```
if (root==null) var pnode=null;
else var pnode=root.target;
//在根节点之下增加一个子节点
$("#myTree1").tree('append',{
    parent: pnode,
    data:{text:"北京市", id:"110000"}
});
//在根节点之下一次性增加多个子节点
$("#myTree1").tree('append',{
    parent: pnode,
    data:[
        {text:"天津市",id:"120000"},
        {text:"上海市",id:"310000"},
        {text:"江苏省",id:"320000"},
        {text:"浙江省",id:"330000"},
        {text:"安徽省",id:"340000"},
        {text:"福建省",id:"350000"},
        {text:"江西省",id:"360000"},
        {text:"山东省",id:"370000"},
        {text:"河南省",id:"410000"}
    ]
});
```

③使用两种方法在树中查找节点。第一种采用find方法，根据节点ID值在树中查找某个节点；第二种通过getChildren方法获取父节点下的所有子节点，利用循环在这些子节点中找到该节点。找到节点后，采用append方法在该节点之下添加若干个子节点。至此，myTree1树中共有三层节点。

```
//方法1：根据id值和find方法查找节点，找到节点后增加其子节点
var node1=$("#myTree1").tree('find','320000');
if (node1!=null){
    $("#myTree1").tree('append',{
        parent: node1.target,
        data:[
            {text:"南京市",id:"320100"},
            {text:"无锡市",id:"320200"},
            {text:"徐州市",id:"320300"},
            {text:"常州市",id:"320400"},
            {text:"苏州市",id:"320500"},
            {text:"南通市",id:"320600"},
        ]
    });
}
//方法2：取根节点的所有子节点，在子节点中根据id值查找某个节点。
var node2=null;
var children = $('#myTree1').tree('getChildren', root.target);
for (var i=0;i<children.length;i++){
    if (children[i].id=='330000'){   //找到子节点后终止循环
        node2=children[i];
        break;
    }
}
```

```
if (node2!=null){
    $("#myTree1").tree('append',{   //添加第3层节点
        parent: node2.target,
        data:[
            {text:"杭州市",id:"330100"},
            {text:"宁波市",id:"330200"},
            {text:"温州市",id:"330300"},
            {text:"嘉兴市",id:"330400"},
            {text:"湖州市",id:"330500"},
            {text:"绍兴市",id:"330600"},
        ]
    });
}
```

④默认情况下，添加子节点之后，父节点处于自动展开状态，这里使用collapse方法收缩该node1节点（即将其子节点收起）。使用select方法选中树中或聚焦某个节点。

```
$("#myTree1").tree('collapse', node1.target);    //收缩父节点，与expand相反
$("#myTree1").tree('select',node2.target);       //选中节点
```

主要知识点：

①append方法添加根节点和子节点

②利用find方法查找树节点

③树节点的收缩、展开

④选中某个树节点

实例 27. 基于 JSON 数据的静态树加载与操作。

与静态组合框（combobox）的概念一样，静态树是指树中节点数据来自本地客户端，可以是符合JSON格式的数据。由于树具有层次结构，因此其JSON数据必须体现父节点与子节点之间上的下层关系。在EasyUI中，这层关系通过children这个属性及其嵌套来定义。例如，描述省份、城市及行政区之间树型结构的JSON数据如下：

```
var source=[
    {text:"江苏省", id:"320000"},
    {text:"浙江省",id:"330000", children:[
        {text:"杭州市", id:"330100", children:[
            {text:"西湖区",id:"330101"},
            {text:"上城区",id:"330202"},
            {text:"下城区",id:"330303"},
            {text:"江干区",id:"330404"}
        ]},
        {text:"宁波市", id:"330200"},
        {text:"温州市", id:"330300"},
        {text:"嘉兴市", id:"330400"},
    ]},
    {text:"安徽省", id:"340000"}
];
```

本例定义一个树形结构的JSON地区数组，将数据与树的data属性绑定后构造树中节点。为了演示树中节点的新增、修改和删除等操作，本例构造一个树的右键菜单myMenu1和一个节点编辑窗口myWin1。本例程序运行界面如图4-2所示，具体实现过程和程序代码如下：

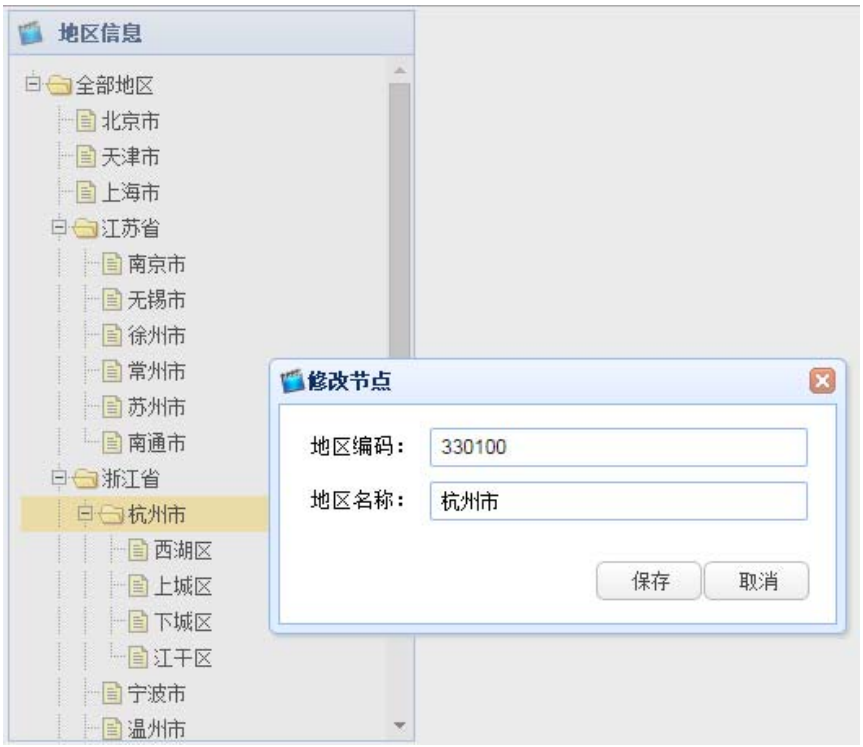


图4-2　静态树节点增删改操作

①定义JSON格式数据，存放在一个数组变量areasource中。每个记录对应一个节点，每个节点包含text和id两个列值。

```
var areasource=[{text:"全部地区", id:"*", children:[
    {text:"北京市", id:"110000"},
    {text:"天津市", id:"120000"},
    {text:"上海市", id:"310000"},
    {text:"江苏省", id:"320000", children:[
        {text:"南京市", id:"320100"},
        {text:"无锡市", id:"320200"},
        {text:"徐州市", id:"320300"},
        {text:"常州市", id:"320400"},
        {text:"苏州市", id:"320500"},
        {text:"南通市", id:"320600"}
    ]},
    {text:"浙江省",id:"330000", children:[
        {text:"杭州市", id:"330100"},
        {text:"宁波市", id:"330200"},
        {text:"温州市", id:"330300"},
        {text:"嘉兴市", id:"330400"},
        {text:"湖州市", id:"330500"},
        {text:"绍兴市", id:"330600"}
    ]},
    {text:"安徽省", id:"340000"},
    {text:"福建省", id:"350000"},
    {text:"江西省", id:"360000"},
    {text:"山东省", id:"370000"},
    {text:"河南省", id:"410000"}
]}];
```

②定义一个表单myForm1、一个窗口myWin1和两个文本编辑框，在表单中定义树控件myTree1。

关键之一是将树的data属性与数组areasource绑定，这时数中相应节点已经自动产生。

```
myForm('myForm1','main','地区信息',0,0,450,250,'');
myWindow('myWin1','编辑节点',0,0,170,355,'save;cancel','close;modal');
myTextField('areaid','myWin1','地区编码：',70,33*0+14,18,0,232,'','');
myTextField('areaname','myWin1','地区名称：',70,33*1+14,18,0,232,'','');
var str='<div id="myTree1" class="EasyUI-tree" style="width:215px; height:380px;
padding:5px;"></div>';
$("#myForm1").append($(str));
$("#myTree1").tree({
    checkbox: false,
    lines:true,
    data: areasource   //绑定数组
});
```

③调用自定义函数myMenu定义树myTree1的一个右键菜单myMenu1，该菜单共有4个子菜单项，分别为新增节点、新增子节点、修改节点和删除节点。

```
myMenu('myMenu1','新增结点/mnaddnode/addIcon;新增子结点/mnaddsubnode/addIcon;修改结点
/mneditnode/editIcon;-;删除结点/mndeletenode/deleteIcon',' ');
$("#myTree1").tree({
    onContextMenu: function (e, title) {
        e.preventDefault();
        $("#myMenu1").menu('show', {
            left: e.pageX,
            top: e.pageY
        });
    }
});
```

④为更好地演示树节点的各种操作，使用append方法，在找到“杭州市”这个节点的基础上，添加5个行政区的子节点，并将光标聚焦在“杭州市”这个节点上。

```
var root = $("#myTree1").tree("getRoot"); //提取根节点
if (root==null) var pnode=null;
else var pnode=root.target;
var node1=$("#myTree1").tree('find','330100');   //查找杭州市这个节点
if (node1!=null){
$("#myTree1").tree('append',{
    parent: node1.target,
    data:[
            {text:"西湖区",id:"330101"},
            {text:"上城区",id:"330202"},
            {text:"下城区",id:"330303"},
            {text:"江干区",id:"330404"}
        ]
    });
$("#myTree1").tree('select',node1.target);
}
```

⑤编写节点新增节点、新增子节点和修改节点三个操作的程序。在新增节点和子节点时需要将两个文本框内容清空；在修改节点时需要将当前节点值赋值到两个对应的文本框中。随后根据3个不同操作类型，设置窗口myWin1的显示标题，打开窗口开始编辑数据。

```
$("#mnaddnode").bind('click', function(e){ //新增兄弟节点;
```

```
        $("#areaid").textbox('setValue','');
        $("#areaname").textbox('setValue','');
        $("#myWin1").window({title: '新增节点'});
        $("#myWin1").window('open');
    });
    $("#mnaddsubnode").bind('click', function(e){ //新增子节点;
        $("#areaid").textbox('setValue','');
        $("#areaname").textbox('setValue','');
        $("#myWin1").window({title: '新增子节点'});
        $("#myWin1").window('open');
    });
    $("#mneditnode").bind('click', function(e){ //修改节点;
        var node=$("#myTree1").tree('getSelected');   //获取树中当前节点
        if (node!=null){
            $("#areaid").textbox('setValue',node.id);
            $("#areaname").textbox('setValue',node.text);
            $("#myWin1").window({title: '修改节点'});
            $("#myWin1").window('open');
        }
    });
```

⑥编写程序，保存节点数据。在保存数据前，先判断节点操作的类型。修改节点后保存时，只要将文本框的值赋值到当前节点，使用使用update方法即可更新树节点。对于新增节点（即新增兄弟节点），需要求出当前节点的父节点；新增子节点时，当前节点即为新节点的父节点。在确定父节点的基础上，使用append方法增加节点，这时树中即时显示新节点。节点编辑保存后，关闭窗口myWin1。

```
    $("#myWin1SaveBtn").bind('click', function(e){ //点击窗体中的保存按钮
        var options=$("#myWin1").window('options');
        var action=options.title;
        var xid=$("#areaid").textbox('getValue');
        var xname=$("#areaname").textbox('getValue');
        var node=$("#myTree1").tree('getSelected');
        if (action=='修改节点'){
            node.id=xid;
            node.text=xname;
            $("#myTree1").tree('update',node);   //刷新树结点
        }else if (action=='新增节点'){
            var node=$("#myTree1").tree('getSelected'); //获取树中当前节点
            var pnode=$("#myTree1").tree('getParent', node.target); //求节点的父节点
            $("#myTree1").tree('append',{
                parent:pnode.target,
                data:{ id:xid, text:xname }
            });
        }else if (action=='新增子节点'){
            var pnode=$("#myTree1").tree('getSelected'); //获取树中当前节点
            $("#myTree1").tree('append',{
                parent:pnode.target,
                data:{ id:xid, text:xname }
            });
        }
        $("#myWin1").window('close');
```

```
});
```

⑦使用getSelected方法找到树中当前节点，采用remove方法删除该节点。

```
$("#mndeletenode").bind('click', function(e){ //删除节点;
    var node=$("#myTree1").tree('getSelected'); //获取树中当前节点
    if (node!=null){
        var pnode=$("#myTree1").tree('getParent');
        $("#myTree1").tree('remove', node.target);   //删除节点
    }
});
```

相关知识点：

①每个树节点中必须有一个id值。

②使用options获取控件的属性值，如获取窗口的标题的方法：

```
var options=$("#myWin1").window('options');
var action=options.title;
```

作业题：

1．编写程序，在新增节点保存后，将光标聚焦到新增的节点上；在伤处节点后，将光标聚焦到树最近的兄弟节点上，兄弟节点不能存在时，则聚焦到父节点上。

2．修改本例程序，在点击新增节点和子节点菜单时，先在树中产生一个空节点，然后保存后将文本框的值填充到新节点上去。

实例 28. 从数据库一次性加载数据到树节点。

在EasyUI中，从数据库加载数据到树节点总体是比较简单的。由于树节点可以从JSON数据中取值，因此，只要将数据库中记录转换成JSON格式数据，就可以实现从数据库加载数据到树节点上。可以一次性将数据库记录全部加载到树节点中，也可以通过分层逐级加载到树节点中。

本例在客户端定义一个地区分类树，其方法与实例27基本相同，只是在属性描述之后，编写一段程序从数据库中一次性（完整）提取JSON数据返回到客户端，通过data属性加载（填充）到树中。服务器端程序Easyui_getAllTreeNodes.jsp利用一次循环，自顶向下（而非函数递归方式）实现树中父节点与子节点之间的嵌套，其运行效率较高。经模拟测算，服务器端生成3000个节点填充到树中耗时不足2秒，比采用函数递归方式快3倍多。

必须指出，按照服务器端程序，存放树节点的数据库表结构有一定的规定。例如地区分类表（areas）结构如下，记录举例如图4-3所示。

```
CREATE TABLE Areas(
    AreaID nchar(10) primary key default '',
    AreaName nvarchar(30) default '',
    PYCode nvarchar(15) default '',
    Zip nchar(10) default '',
    PhoneCode nchar(10) default '',
    ParentNodeID nchar(10) default '',   --父节点标识符
    IsparentFlag tinyint default '',      --父节点或叶子节点标记值
    Ancester varchar(255),          --祖先节点
    level tinyint default 0         --节点层次
)
```

	AreaID	AreaName	PYCode	Zip	ParentNodeID	IsparentFlag	Ancester	level
1	320000	江苏省	JSS			1		1
2	320100	南京市	NJS		320000	1	320000#	2
3	320124	溧水县	LSX		320100	0	320000#320100#	3
4	320125	高淳县	GCX		320100	0	320000#320100#	3
5	320200	无锡市	WXS		320000	1	320000#	2
6	320281	江阴市	JYS		320200	0	320000#320200#	3
7	320282	宜兴市	YXS		320200	0	320000#320200#	3
8	320300	徐州市	XZS		320000	1	320000#	2
9	320321	丰　县	F　X		320300	0	320000#320300#	3
10	320322	沛　县	P　X		320300	0	320000#320300#	3
11	320323	铜山县	TSX		320300	0	320000#320300#	3
12	320324	睢宁县	SNX		320300	0	320000#320300#	3
13	320381	新沂市	XYS		320300	0	320000#320300#	3
14	320382	邳州市	PZS		320300	0	320000#320300#	3
15	320400	常州市	CZS		320000	1	320000#	2
16	320481	溧阳市	LYS		320400	0	320000#320400#	3
17	320482	金坛市	JTS		320400	0	320000#320400#	3
18	320500	苏州市	SZS		320000	1	320000#	2
19	320581	常熟市	CSS		320500	0	320000#320500#	3
20	320582	张家港市	ZJGS		320500	0	320000#320500#	3
21	320583	昆山市	KSS		320500	0	320000#320500#	3
22	320584	吴江市	WJS		320500	0	320000#320500#	3
23	320585	太仓市	TCS		320500	0	320000#320500#	3
24	320600	南通市	NTS		320000	1	320000#	2
25	320621	海安县	HAX		320600	0	320000#320600#	3
26	320623	如东县	RDX		320600	0	320000#320600#	3
27	320681	启东市	QDS		320600	0	320000#320600#	3

图4-3　关系数据库中的树形结构数据示例

这里，ParentNodeID、IsparentFlag、Ancester和level这4个列是必须的，其含义分别表示每个节点的父节点ID值、叶子节点或父节点的标记值（1-为父节点，0-为叶子节点）、祖先节点值（即各级父节点的组合，包括父节点的父节点）和树节点所在层次。

本例客户端程序的实现过程和方法如下：

①使用javascript语句定义基本树控件myTree1，设置其lines属性为true。

```
var str='<div id="myTree1" class="EasyUI-tree" style="fit:auto; border: false; padding:5px;"></div>';
$("#myForm1").append($(str));
$("#myTree1").tree({
    checkbox: false,
    lines:true
});
```

②使用ajax调用服务器端程序Easyui_getAllTreeNodes.jsp，一次性获取数据表areas中的所有记录，通过data属性加载到树myTree1中。客户端传递给服务器端主要参数为查询语句jqsql.area和关键字areaid。

```
$.ajax({
    url: "system\\Easyui_getAllTreeNodes.jsp",
    data: { database: sysdatabasestring, selectsql: jqsql.area, keyfield:'areaid', sortfield:'' },
    async: false,
    success: function(data) {
        var source=eval(data);
        $('#myTree1').tree({ data: source });
    }
});
```

③由于在EasyUI中双击父节点时，父节点不会自动展开子节点，带来操作不便。为此在树控件中添加一个onDblClick事件，实现双击展开父节点。

```
$('#myTree1').tree({   //双击展开或收缩结点
    onDblClick: function(node){
        if (node.state=='closed') $(this).tree('expand', node.target);
        else $(this).tree('collapse', node.target);
    }
});
```

④初始值和状态设置。由于地区节点较多，在加载完树之后，采用collapse方法，收缩树中第一层节点（即根节点），仅展开第三个节点（河北省），并选中这个节点。

```
$("#myTree1").tree('collapseAll');
var roots=$("#myTree1").tree("getRoots");
$("#myTree1").tree('expand', roots[2].target);
$("#myTree1").tree('select', roots[2].target);
```

本例服务器端程序的实现过程和方法如下：

①连接数据库，执行查询语句，将指针指向结果集中第一条记录。

```
querySQL=new StringBuffer();
StringBuffer returnResult=new StringBuffer();//创建一个stringbuffer对象用于存放返回结果
String querystring="";
ResultSet node_rs=stmt.getResultSet();
if (sortfield.equals("")){
    querystring="select * from ("+selectsql+") as p order by ancester+"+keyfield;
    }else{
    querystring="select * from ("+selectsql+") as p order by ancester+"+sortfield;
    }
querySQL.append(querystring);
stmt.executeQuery(querySQL.toString());
node_rs=stmt.getResultSet();
ResultSetMetaData rsmd=node_rs.getMetaData();
int xcolcount=rsmd.getColumnCount();
node_rs.first();
node_rs.beforeFirst();
```

②循环处理每条记录。记录当前节点的标志值和层次值。

```
int xlevel=1;
int flag=0;
returnResult.append("[");
while(node_rs.next()){   //循环输出结果集
```

◆ 记录当前节点的标志值和层次值。

```
    isParentFlag=node_rs.getInt("IsParentFlag");
    level=node_rs.getInt("level");
    str="";
```

◆ 将当前节点的层次值level与之前节点的层次值xlevel进行比较：若两者相同（即层次没有发生变化），则继续（在原来父节点上）添加兄弟节点，节点之间用逗号分隔；如果前者小于后者（即level小于xlevel），则标明之前节点已经没有兄弟节点，在JSON数据中添加若干个“]}”以封闭父节点的children属性集合。

```
    if (level==xlevel){
        if (flag==1) str+=",";
    }else{
```

```
            for (i=level;i<xlevel;i++) returnResult.append("]}\n");
            xlevel=level;
            if (flag!=0) str+=",";
        }
        str+="{\"id\":\""+StringUtil.filterNull(node_rs.getString(keyfield)).trim().
        replace("\"","").replace("","\\")+"\"";   //替换输出值中的单引号和双引号
        for (j=1;j<=xcolcount;j++) {   //生成一条记录的JSON数据
            field=rsmd.getColumnName(j).toLowerCase();
            if (!field.equals("id")){
                str=str+",\""+field+"\":\""+StringUtil.filterNull(node_rs.getString(field)).trim().
                replace("\"","").replace("","\\")+"\"";
            }
        }
        returnResult.append(str);   //添加到结果集
```

◆ 判断如果当前节点是父节点，则在JSON数据中添加children子节点属性；否则作为叶子节点，结束本条记录JSON数据标识。

```
        flag=1;
        if (isParentFlag==1){
            returnResult.append(",\n\"children\":[");
            flag=0;
        }else{
            returnResult.append("}\n");
        }
        xlevel=level;
        i++;
    } //while
```

◆ 循环结束后对剩余的节点进行处理。

```
    for (i=1;i<xlevel;i++) returnResult.append("]}\n");
    returnResult.append("]\n");
```

相关知识点：

①直接调用树控件自定义函数构造树（见Easyui_functions.js文件）。

function myDBTree(id,parent,title,top,left,height,width,sql,keyfield,sortfield,style)

其中style可以是下列选项的组合：checkbox;animate;line;edit;full;menu:。举例：

myDBTree('myTree1', 'myForm1', '地区分类', 0, 0, 0, 0, jqsql.area, 'areaid', ' ' ,' full');

②myTree1树种第3个节点的获取方法。

③判断节点的展开或收缩状态，根据状态再展开或收缩父节点。

④把main层的fit设置为true，在定义myForm1时不指定表单的高度和宽度。

实例 29. 动态树节点的分层逐级加载。

当数据库记录较多时，采用一次性加载全部数据到树节点会花费较多网络资源和影响网页加载速度。通常当树节点数超过 500 个时，需要通过分层将后台数据逐级加载到客户端树节点中。

本例定义一个地区分类树控件，先从数据库中加载第一层节点（即省份）到树中，每次只有当用户点击展开父节点时，才将该父节点的下一层子节点从数据库中提取出来再加载到树中。这样可以大大减少每次从数据库中提取节点的数量，加快网络访问速度。

本例程序运行界面如图 4-4 所示，分层逐级加载树节点的主要过程和方法如下：

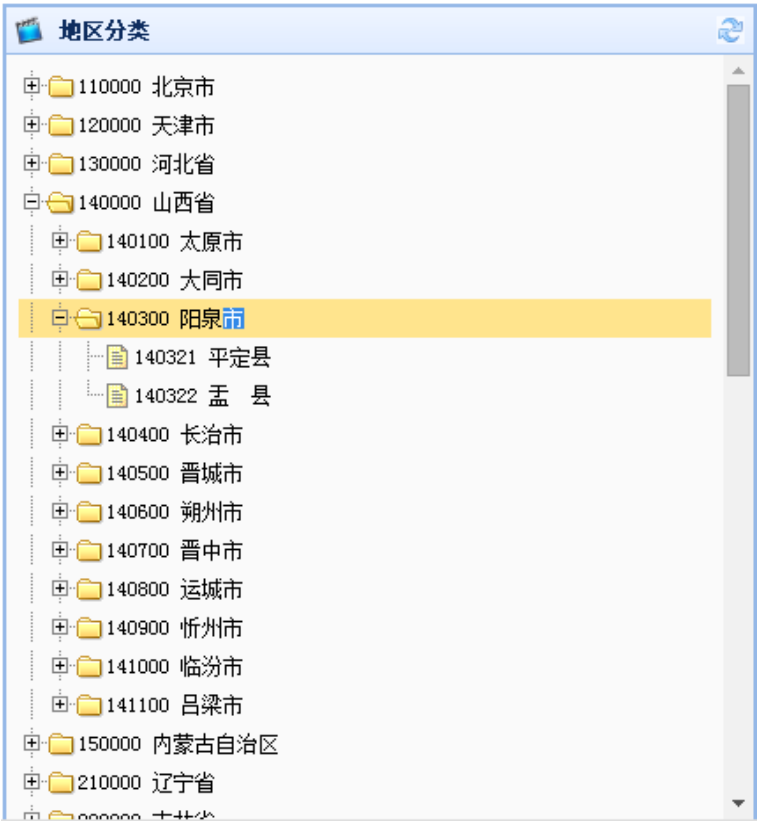


图 4-4 树节点分层逐级展开程序运行界面

①定义树控件 myTree1，将其 sql 语句作为一个自定义属性保存起来，以备后用。

```
var str='<div id="myTree1" class="EasyUI-tree" style="fit:auto; border: false; padding:5px;"></div>';
$("#myForm1").append($(str));
$("#myTree1").tree({
    checkbox: false,
    lines:true,
});
$("#myTree1").attr('xsql', jqsql.area);   //自定义属性
```

②利用 ajax 调用服务器端程序 Easyui_getChildNodes.jsp，从数据库中提取第一层节点(省份)加载到树中（即 ParentNodeID 值为空的节点）。服务器端程序在生成第一层节点时，必须判断是叶子节点还是父节点。如果是父节点则为它添加一个空的虚拟子节点，这样父节点上就会出现一个可以展开的小图标。树控件通过 data 属性加载节点后，必须使用 collapseAll 将所有节点收缩起来不展开。

```
var sqlx="select * from ("+jqsql.area+") as p where parentnodeid="";   //第一层
$.ajax({
    url: "system\\Easyui_getChildNodes.jsp",
    data: { database: sysdatabasestring, selectsql: sqlx, keyfield:'areaid', sortfield:''},
    async: false,
    success: function(data) {
        var source=eval(data);
        $("#myTree1").tree({ data: source });   //加载 JSON 数据到树
    }
});
$("#myTree1").tree('collapseAll');   //不能展开节点
```

③在树 myTree1 的点击展开事件（onBeforeExpand）中编写程序，当用户第一次点击展开某个父节点时，先取出该父节点的关键字值，然后根据该关键字值产生一条查询语句，执行 Easyui_getChildNodes.jsp 程序返回子节点到 source 变量中。注意：第二次点击展开父节点时不需要再从数据库提取数据。

```
$("#myTree1").tree({
    onBeforeExpand: function (node){   //点击展开事件
        var pid=node.id;
        var sql=$("#myTree1").attr('xsql');
        var child_node = $("#myTree1").tree('getChildren', node.target);
        if (child_node.length==1 && child_node[0].id=='_ '+pid){ //判断是否是第一次点击父节点
            if (sql!='') sqlx="select * from ("+sql+") as p where parentnodeid='"+pid+"'";
            else sqlx='';
            $.ajax({
                url: "system\\Easyui_getChildNodes.jsp",
                data: { database: sysdatabasestring, selectsql: sqlx, keyfield:'areaid', sortfield:'' },
                async: false,
                success: function(data) {
                    var source=eval(data);
```

④删除原来的虚拟空节点，在父节点之下使用 append 方法和 data 属性添加子节点。

```
                    $("#myTree1").tree('remove', child_node[0].target); //删除子节点
                    $("#myTree1").tree('append', {   //增加数据作为子节点
                        parent: node.target,
                        data: source
                    });
                }
            });
        };
    }
});
```

程序 2-29. 服务器端提取子节点 Easyui_getChildNodes.jsp

```
<%@ page language="java" import="java.util.*"    import="java.sql.*,com.DBConn" pageEncoding="utf-8"%>
<%@page import="com.StringUtil"%>
<%
 request.setCharacterEncoding("ISO-8859-1");
 String database=StringUtil.getUrlCHN(request.getParameter("database"));
 String keyfield=StringUtil.getUrlCHN(request.getParameter("keyfield"));     //获取主键字段名
 String sortfield=StringUtil.getUrlCHN(request.getParameter("sortfield"));
 String selectsql=StringUtil.getUrlCHN(request.getParameter("selectsql")); //获取要显示的所有列名
 String idfield="";
 StringBuffer result=new StringBuffer();//创建一个 stringbuffer 对象 用于存放最终返回结果
 //System.out.println("childsql="+selectsql);
 //连接数据库
 if (!selectsql.equals("")){
        DBConn con=new DBConn();
        Connection connection=con.getConnection(database);
        StringBuffer querySQL=new StringBuffer();//创建一个 stringbuffer 对象用来存放查询语句
        Statement stmt=connection.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE,ResultSet.
         CONCUR_READ_ONLY);
        if (!sortfield.equals("")){
                querySQL.append("select * from ("+selectsql+") as p order by "+sortfield);
       }else{
                querySQL.append(selectsql);
       }
        stmt.executeQuery(querySQL.toString());
```

```
        ResultSet node_rs=stmt.getResultSet(); //有多个结果集时，需要使用 getResultSet()方法
        //System.out.println("querySQL="+querySQL.toString());
        int isparentflag= -1;
        node_rs.last(); //移到最后一行
        int rowCount =node_rs.getRow(); //得到当前行号，也就是记录数
        node_rs.beforeFirst(); //如果还要用结果集，就把指针再移到初始化的位置
        ResultSetMetaData rsmd=node_rs.getMetaData();
        int xcolcount=rsmd.getColumnCount();
        String str="";
        String field="";
        int i=0;
        int j=0;
        for (j=1;j<=xcolcount;j++) {
            field=rsmd.getColumnName(j).toLowerCase();
            if (field.equals("id")){
                idfield=field;
                break;
            }
        }
        if (idfield.equals("")) { idfield=keyfield; }
        //从数据库取记录，转换成 JSON 格式数据
        while(node_rs.next()){
            str="{\"id\":\""+StringUtil.filterNull(node_rs.getString(idfield)).trim().
            replace("\"","").replace("","\\")+"\"";    //这里\之后的"为普通字符
            for (j=1;j<=xcolcount;j++) {
                field=rsmd.getColumnName(j).toLowerCase();
                if (!field.equals("id")){
                    str=str+",\""+field+"\":\""+StringUtil.filterNull(node_rs.getString(field)).trim().
                     replace("\"","").replace("","\\")+"\"";
                }
            }
            isparentflag=node_rs.getInt("IsParentFlag");
            if (isparentflag==1){ //下面的 closed 很重要
                //子节点的值也要各不相同  =下划线+主键值
                String xid="_"+StringUtil.filterNull(node_rs.getString(keyfield)).trim().
                replace("\"","").replace("","\\")+"\"";
                //添加一个虚拟空节点其值为'_'+父节点值
                result.append(str+",state:\"closed\",\"children\":[{\"id\":\""+xid+",\"cid\":\"\", \"text\":\" \"");
                if (!keyfield.equals("id")) result.append(",\""+keyfield+"\":\"\"");
                    result.append("}]}");  //添加一个虚拟空节点
                    result.append((i==rowCount-1?"":",")+"");
                }else{
                    result.append(str+"}");
                    result.append((i==rowCount-1?"":",")+"");
                }
                i++;
            }
        node_rs.close();
        stmt.close();
        connection.close();
    }else{
        result.append("{}");
    }
    response.getWriter().write("["+result.toString()+"]");
   %>
```

树节点的分层逐级展开，可以大幅减少和分散从数据库中提取节点的时间，但同时增加了树节点定位、过滤和查找的难度。

主要知识点：

①直接调用树控件自定义函数构造树（见Easyui_functions.js文件）。

myDBTree('myTree1', 'myForm1', '地区分类', 0, 0, 0, 0, jqsql.area, 'areaid', ' ' ,' ');

②虚拟空子节点的作用及其值设定。其值不能为空，因为 treegrid 中以 id 值找节点、删节点。

③关闭某个节点 state: 'closed'

实例 30. 动态组合树 ComboTree 控件及其应用。

组合树（Combotree）是组合框（Combo）和树（tree）控件两者的结合，用户点击该控件时它以树状结构（而非列表结构）显示数据，其树节点可以与树控件一样从数据库提取。同样地，可以一次性从数据库中提取全部选项节点，也可以分层逐级展开节点。本例在输入教师的“出生地”时使用一个组合树控件，使用分层逐级展开方法从地区分类树中选择一个城市选项。

本例程序运行界面如图 4-5 所示，主要程序代码见程序 4-30。

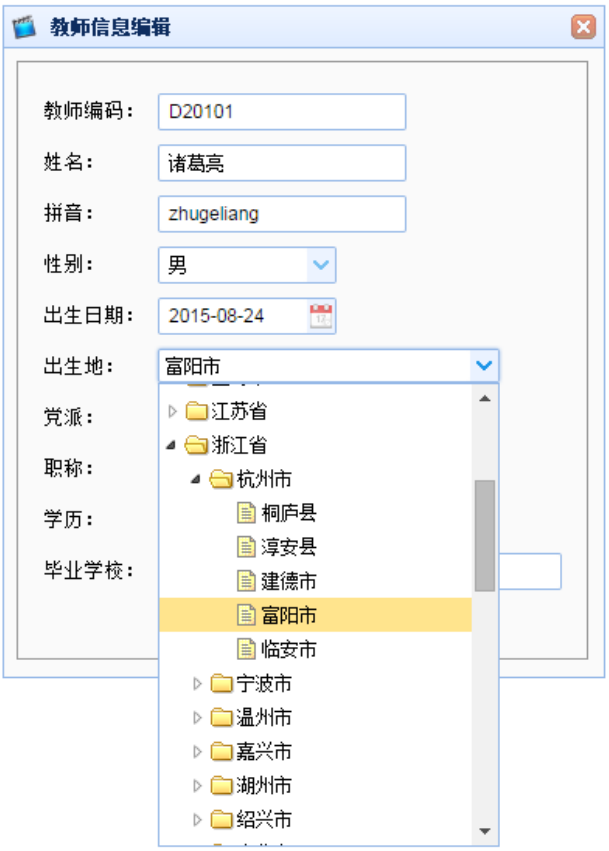


图 4-5 动态组合树 ComboTree 控件程序运行界面

程序 4-30：动态组合树应用程序 example30_dbcombotree.jsp

```
<%@ page language="java" import="java.util.*" pageEncoding="utf-8"%>
<!doctype html>
<html lang="en">
<style type="text/css">
</style>
<head>
<meta charset="utf-8">
<link rel="stylesheet" type="text/css" href="jqEasyUI/themes/default/EasyUI.me.css">
<link rel="stylesheet" type="text/css" href="jqEasyUI/themes/icon.css">
<link rel="stylesheet" type="text/css" href="system/css/icon.css">
    <script type="text/javascript" src="jqEasyUI/jQuery.min.js"></script>
```

```
    <script type="text/javascript" src="jqEasyUI/jQuery.EasyUI.min.js"></script>
    <script type="text/javascript" src="jqEasyUI/EasyUI-lang-zh_CN.js"></script>
<script type="text/javascript" src="system/Easyui_functions.js"></script>
</head>
<body style="margin: 2px 2px 2px 2px;">
<div id='main' style="margin:0px 0px 0px 0px;">
</div>
<script>
$(document).ready(function() {
    var jqsql={};   //sql 语句中不能有双引号
    jqsql.party="select Pycode as id,description as party from dictionary where Type='党派'";
    jqsql.title="select Pycode as id,description as title from dictionary where Type='学历'";
    jqsql.area="select AreaName as text, * from Areas ";
    jqsql.degree="select Pycode as id,description as degree from dictionary where Type='学历'";
    myForm('myForm1','main','教师信息编辑',0,0,435,388,'close;drag');
    myFieldset('myFieldset1','myForm1','',8,6,385,360);
    myTextField('teacherid','myFieldset1','教师编码：',70,33*0+20,16,0,160,'D20101');
    myTextField('name','myFieldset1','姓名：',70,33*1+20,16,0,160,'诸葛亮');
    myTextField('pycode','myFieldset1','拼音：',70,33*2+20,16,0,160,'zhugeliang');
    myComboField('gender','myFieldset1','性别：',70,33*3+20,16,0,115,'男;女;take;task;book;buring','');
    myDateField('birthdate','myFieldset1','出生日期：',70,33*4+20,16,0,115,'5/12/1997');
    myLabelField('areaidx','myFieldset1','出生地：',33*5+20+4,16,0,0);
    myDBComboField('party','myFieldset1','党派：',70,33*6+20,16,0,160,jqsql.party,'','');
    myDBComboField('title','myFieldset1','职称：',70,33*7+20,16,0,160,jqsql.title,'','');
    myDBComboField('degree','myFieldset1','学历：',70,33*8+20,16,0,160,jqsql.degree,'','');
    myTextField('school','myFieldset1','毕业学校：',70,33*9+20,16,0,260,'');
    //定义 combotree 控件
    var str='<div id="areaid_div"><input class="EasyUI-combotree" id="areaid"></div>';
    $("#myFieldset1").append($(str));
    $("#areaid_div").css(myTextCss(33*5+20,86,0,260));
    $("#areaid").combotree({
        width:220,
        panelHeight: 300,
        valueField: 'areaid',
        textField: 'areaname'
    });
    $("#areaid").attr('xsql',jqsql.area);   //自定义属性
    $("#areaid").attr('xkeyfield','areaid'); //自定义属性
    //提取第一层节点
    var xsql="select * from ("+jqsql.area+") as p where parentnodeid='';
    $.ajax({
        type: "Post",
        url: "system\\Easyui_getChildNodes.jsp",
        data: { database: sysdatabasestring, selectsql: xsql, keyfield:'areaid', sortfield:'' },
        async: false,
        success: function(data) {
            var source=eval(data);
            $('#areaid').combotree({ data: source });
        }
    });
    //点击展开事件，提取子节点
    $("#areaid").combotree({
        onBeforeExpand: function (node){   //点击展开事件
```

```
                var sql=$('#areaid').attr('xsql');
                var keyfield=$('#areaid').attr('xkeyfield');
                var pid=eval("node."+keyfield);
                var xcbtree=$('#areaid').combotree('tree');
                var child_node = xcbtree.tree('getChildren', node.target);
                if (child_node.length==1 && child_node[0].id=='_'+pid){ //生成子节点
                        var xsql="select * from ("+sql+") as p where parentnodeid='"+pid+"'";
                        $.ajax({
                                url: "system\\Easyui_getChildNodes.jsp",
                                data: { database: sysdatabasestring, selectsql: xsql, keyfield:keyfield, sortfield:'' },
                                async: false,
                                success: function(data) {
                                        var source=eval(data);
                                        xcbtree.tree('remove', child_node[0].target); //删除子节点
                                        xcbtree.tree('append', {   //增加数据作为子节点
                                                parent: node.target,
                                                data: source
                                        });
                                }
                        });
                };
        }
    });
    //选定树初始聚焦位置
    var cbtree=$('#areaid').combotree('tree');
    var node=cbtree.tree('find','330000');
    cbtree.tree('select',node.target);
});   //endofjQuery
</script>
</body>
</html>
```

组合树 combotree 可以直接引用绝大部分 combo 控件的属性，但在引用树控件属性时需要先使用 tree 方法，例如：

```
var xcbtree=$('#areaid').combotree('tree');
var child_node = xcbtree.tree('getChildren', node.target);
```

本例组合树的定义及其事件已经封装在一个函数 function myDBComboTreeField(id, parent, label, labelwidth, top, left, height, width, sql, keyfield, sortfield, style)中，其中主要参数 sql 为数据库查询语句、keyfield 为查询语句对应的主键列、style 值为 full 时一次性从数据库提取全部节点，否则分层逐级展开提取子节点。该函数的具体调用方法如下：

```
myDBComboTreeField('areaid', 'myFieldset1', '出生地：',70,33*5+20, 16,0,220, jqsql.area, 'areaid', ' ','full');
```

实例 31. 静态 JSON 数据网格控件基础。

数据网格（datagrid）是 EasyUI 的核心组件之一，它以一种表格的形式显示数据。与树控件一样，网格也可以从本地获取数据或从服务器数据库中获取数据。定义数据网格时最主要的属性是其列集合（columns），包括列标题、列字段名、列宽度以及对齐方式等。

本例定义一个 JSON 格式的数组变量 griddata 和一个只读数据网格控件 myGrid1。griddata 为网格提供数据源。myGrid1 网格的列集共包含 8 个列，外加一个行号列和一个选择框列（checkbox），每列标题居中，网格一次性显示整个数组中的全部行（即不分页）。

本例程序运行界面如图 4-6 所示，主要程序设计过程和方法如下。

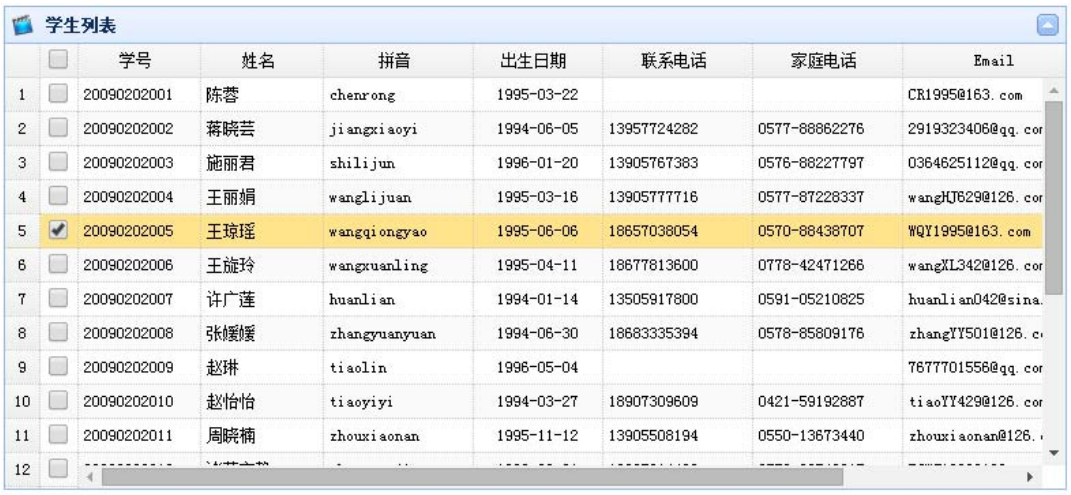


图 4-6 静态数据网格程序运行界面

①在 HTML 的 中定义数据网格控件 myGrid1。

```
<div id='main' style="margin:0px 0px 0px 0px;">
    <div id="myGrid1" class="EasyUI-datagrid"></div>
</div>
```

②定义网格的 JSON 数据源(共 20 行)，存放在一个 griddata 数组中。JSON 数据格式中的 total 表示总的行数，rows 指定各行的值。

```
var griddata={"total":"200","rows":[
    {"rownumber":"1","studentid":"20090202001","name":"陈蓉","pycode":"chenrong","gender":"女",
     "birthdate":"1995-03-22", "province":"贵州省","city":"德江县","mobile":"","homephone":"",
      "email":"CR1995@163.com","weixin":"chenrong766","qq":""}
   ,{"rownumber":"2","studentid":"20090202002","name":"蒋晓芸","pycode":"jiangxiaoyi","gender":"女",
     "birthdate":"1994-06-05","province":"浙江省","city":"泰顺县","mobile":"13957724282","homephone":
     "0577-88862276","email":"2919323406@qq.com", "weixin":"2919323406@qq.com","qq":"2919323406"}
      ……
   ,{"rownumber":"20","studentid":"20090202020","name":"唐胜利","pycode":"tangshengli","gender":"男",
      "birthdate":"1995-05-01","province":"浙江省","city":"慈溪市","mobile":"13905748349",
      "homephone":"0574-88495772","email":"tangshengli@hotmail.com","weixin":"13905748349","qq":""}
]};
```

③定义网格每列的标题（title)、列字段名（field)、列宽度等，存放在一个变量 xcolumns 中。各个列标题水平居中(halign:'center')，表体内各行内容中第 4 列(“出生日期”)居中(align: 'center')，其他默认为左对齐（align: 'left')。此外还定义一个固定列（frozenColumns)，内容为一个复选框（checkbox)，存放在一个变量 xfixedcolumns 中。

```
var xcolumns=[[
        { title: '学号', field: 'studentid', width: 90, sortable: true, halign:'center', align: 'left'  },
        { title: '姓名', field:'name', width: 90, halign:'center', align: 'left' },
        { title: '拼音', field:'pycode', width: 110, halign:'center', align: 'left' },
        { title: '出生日期', field: 'birthdate', width: 95, halign:'center', align: 'center'},
        { title: '联系电话', field: 'mobile', width: 110, halign:'center', align: 'left' },
        { title: '家庭电话', field: 'homephone', width: 110, halign:'center', align: 'left' },
        { title: 'Email', field: 'email', width: 140, halign:'center', align: 'left' },
        { title: '微信号', field: 'weixin', width: 130, halign:'center', align: 'left' }
]];
var xfixedcolumns=[[   //定义固定列
        { title: 'id', field: 'id', width: 20, checkbox: true,   align: 'center'}
]];
```

④定义网格 myGrid1 的属性：使用 data 属性绑定 JSON 数组数据 griddata；使用 columns 属性绑定列集 xcolumns；使用 frozenColumns 绑定固定列 xfixedcolumns。

```
$('#myGrid1').datagrid({
    title: ' 学生列表',
    iconCls: "panelIcon",
    width:780,
    height:355,
    data: griddata,
    nowrap: true,
    rownumbers: true,
    striped: true,
    collapsible: true,
    singleSelect: true, //false,
    idField: 'studentid',
    columns: xcolumns,
    frozenColumns: xfixedcolumns
});
```

相关知识点：

①网格数据的绑定可以用 data 属性绑定，也可以用 loadData 方法，例如：

```
$('#myGrid1').datagrid('loadData', griddata);
```

②网格中的行号是自动产生的，与 griddata 数组中的 rownumber 无关（该列用其他用途）。

实例 32. 动态数据库网格控件及其分页处理。

在 EasyUI 中，基于服务器端数据库的动态数据网格在定义方式上与实例 31 的静态网格十分相似，不同只是从服务器数据库中提取据数据与分页的技术与方法。其中分页模式的设置主要包括：①在数据网格定义时，设置 pagination 值为 true、设置分页栏的显示位置、设置 pageSize 和 pageNumber 的值（一般起始值为 1）；②设置分页栏的显示格式，包括：showPageList、beforePageText、afterPageText、displayMsg 等属性；③定义分页点击按钮的触发事件，执行从数据库中提取某一页记录的查询语句。

本例首先使用 javascript 语句中定义一个数据网格 myGrid1，分页显示学生基本信息，每页显示 10 行。在分页栏的 onRefresh、onSelectPage 和 onChangePageSize 事件中编写程序，调用服务器端程序 Easyui_getGridData.jsp，从数据库中分页提取记录，并以 JSON 格式返回。

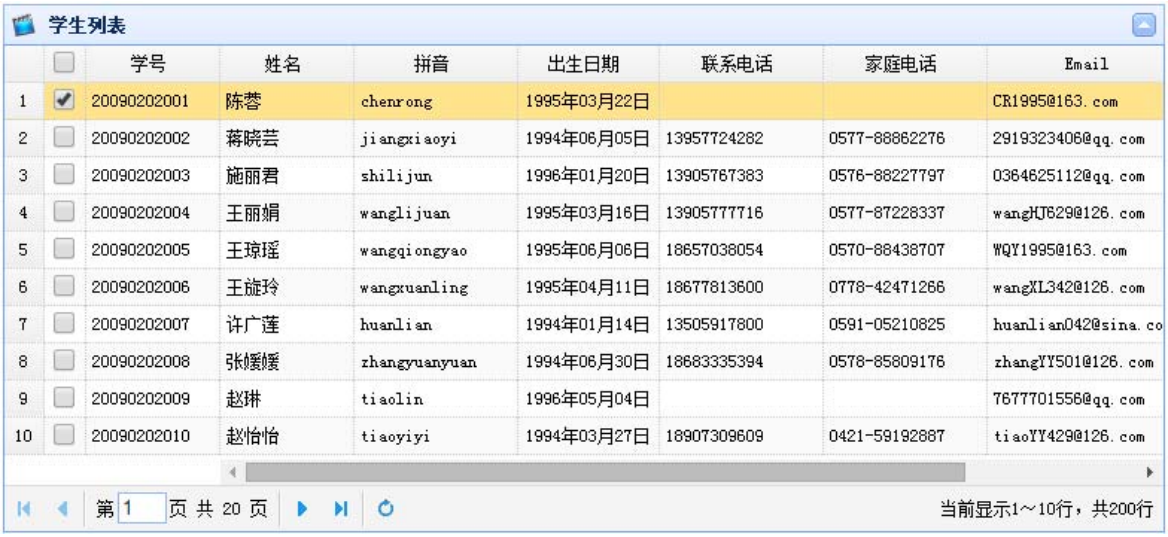


图 4-7. 动态数据网格分页程序运行界面

本例程序运行界面如图 4-7 所示，主要程序实现过程和方法如下：

①使用 append 语句在 main 层之下定义一个数据网格控件 myGrid1，将 main 设置为一个 layout 控件，其 fit 为 true。之后，定义网格各列的标题、宽度、对其方式及列名，并定义复选框和学号两个列为固定列。

```
var str='<div id="myGrid1" class="EasyUI-datagrid"></div>';
$("#main").append($(str));
var xcolumns=[[
        { title: '姓名', field:'name', width: 90, halign:'center', align: 'left' },
        { title: '拼音', field:'pycode', width: 110, halign:'center', align: 'left' },
        { title: '出生日期', field: 'birthdate', width: 95, halign:'center', align: 'center',
         formatter: function(value){
                    value=value+";   //类型转换
                    return '<div align="center">' + value.substr(0,4)+'年'+value.substr(5,2)+'月'
                    +value.substr(8,2)+'日'+'</div>';
              }
        },
        { title: '联系电话', field: 'mobile', width: 110, halign:'center', align: 'left' },
        { title: '家庭电话', field: 'homephone', width: 110, halign:'center', align: 'left' },
        { title: 'Email', field: 'email', width: 140, halign:'center', align: 'left' },
        { title: '微信号', field: 'weixin', width: 130, halign:'center', align: 'left' }
]];
var xfixedcolumns=[[
        { title: 'id', field: 'id', width: 20, checkbox: true, sortable: true, align: 'center'},
        { title: '学号', field: 'studentid', width: 90, checkbox: false, sortable: true, halign:'center', align: 'left' }
]];
```

②设置网格属性，设置其 pagination 值为 true，pagePosition 为'bottom'，每页显示 10 行（easyui 数据网格每页显示的行数必须是 10 的倍数？）、起始页为第 1 页。

```
$("#myGrid1").datagrid({
      title: ' 学生列表',
      iconCls: "panelIcon",
      width:780,
      nowrap: true,
      pagePosition: 'bottom',   //top,both
      autoRowHeight: false,
      rownumbers: true,
      pagination: true,
      pageSize: 10,
      pageNumber:1,
      striped: true,
      collapsible: true,
      singleSelect: true, //false,
      idField: 'studentid',
      columns: xcolumns,
      frozenColumns: xfixedcolumns
});
```

③定义分页模式。设置分页栏显示信息和格式 showPageList（是否可以设置每页显示行数的组合框，一般不需要）、beforePageText（页数文本框之前显示的文字，一般设置为'第'）、afterPageText（页数文本框之后的文字前，一般设置为：'页　　共 {pages} 页'）、displayMsg（分页栏右侧显示当前页记录行区间，一般设置为：'当前显示{from}～{to}行，共{total}行'），其中{pages}、{from}、{to}、{total}为系统变量，自动根据分页记录显示其值。

```
var pg = $("#myGrid1").datagrid("getPager");
(pg).pagination({
```

```
        showPageList: false,   //是否显示设置每页显示行数的组合框
        beforePageText: '第',  //页数文本框前显示的汉字
        afterPageText: '页      共 {pages} 页',
        displayMsg: '当前显示{from}～{to}行，共{total}行'
    });
```

④在分页栏的 onRefresh、onSelectPage 和 onChangePageSize 中编写程序，调用 fnLoadGridData()函数，从数据库中提取指定页的记录。传给该函数的参数主要为当前页码和每页显示行数。

```
    (pg).pagination({
        onRefresh:function(pageNumber,pageSize){
            fnLoadGridData(pageNumber,pageSize);
        },
        onChangePageSize:function(pageSize){
            var opts = $('#myGrid1').datagrid('options');
            opts.pageSize=pageSize;
            fnLoadGridData(1,pageSize);
        },
        onSelectPage:function(pageNumber,pageSize){
            var opts = $('#myGrid1').datagrid('options');
            opts.pageNumber=pageNumber;
            opts.pageSize=pageSize;
            fnLoadGridData(pageNumber,pageSize);
        }
    });
```

⑤构造 fnLoadGridData()函数，实现从数据库中提取某页的记录。客户端传给服务器端 6 个变量，其中包括 select 语句和主键名。limit 为每页显示行数，start 为起始行序号。

```
    function fnLoadGridData(pageNumber,pageSize){
        $.ajax({
            type: "Post",
            url: "system/Easyui_getGridData.jsp",
            //contentType: "application/JSON; charset=utf-8",
            //dataType: "JSON",
            data: {
                database: sysdatabasestring,
                selectsql: jqsql.students,
                keyfield: 'studentid',
                sortfield: '',
                limit: pageSize,
                start: (pageNumber-1)*pageSize
            },
            async: false,
            success: function(data) {
                eval("var source="+data+";");
                $('#myGrid1').datagrid( "loadData", source );   //必须用 loaddata
                //根据总行数改变行号的列宽度，改 css
                var rowcount=$("#myGrid1").datagrid('getData').total+";   //转换为字符型
                var width=(rowcount.length*6+8);
                if (width<25) width=25;
                var px=width+'px';
                //console.log(px);
                $('.datagrid-header-rownumber,.datagrid-cell-rownumber').css({"width": px});
```

```
                $('#myGrid1').datagrid('resize');   //必须写
                $('#myGrid1').datagrid('selectRow',0); //选中某行
            }
        });
    }
```

EasyUI 在实现网格分页显示总体比较简便，核心依然是服务器从数据库中提取数据的程序 Easyui_getGridData.jsp，关键技术是如何提取一个查询结果集中间一部分的记录，即从第 start 行开始取 limit 行。例如，取学生表中第 31~40 行，可参考下列查询语句：

```
select top 10 * from Students where studentid not in
(select top 30 studentid from Students order by studentid )
order by studentid
```

换成 start 和 limit 变量之后，查询语句可以描述为：

```
xquerySql="select top "+xlimit+" * from query_tmp where "+xkeyfield+" not in \n";
xquerySql+="(select top "+(xstart)+" "+xkeyfield+" from query_tmp order by "+xsortfield+")\n";
xquerySql+=" order by "+xsortfield;
```

作业题：

①构造一个数据网格列自定义函数，输入一个字符串，从字符串中提取各个列，生成一个符合 EasyUI 格式的网格列集。字符串的格式为[@alignment%datatype#length,dec]，其中：各个列之间用分号隔开，每个列包括 3 部分，方括号内为分为列数据对其方式、类型和长度（包括小数位数），斜杠前为列标题，斜杠后为列字段名称，使用默认值时可以省略对其方式和数据类型。举例：

[@l%c#90] 姓名/name;[110] 拼音/pycode;[@c#90] 出生日期/birthdate;[100] 所属城市/hometown;[110] 联系电话/mobile;[110] 家庭电话/homephone;[140]Email/email;[130] 微信号/weixin;[100]QQ 号/qq

②构造一个自定义函数，设置网格分页模式，并实现从后台提取数据。

实例 33. 数据库网格中数据的过滤与定位。

从上面两个实例中可以看出，在定义数据网格时，列标题、属性及分页模式等信息的描述比较繁琐，对于重复性较多的代码可使用函数进行定义。本例构造一个数据网格列处理函数 myGridColumns()，对于给定的一个列字符串，该函数将各个列进行分割处理后返回一个符合网格定义要求的列集数组，这样可以大大简化网格列的定义。

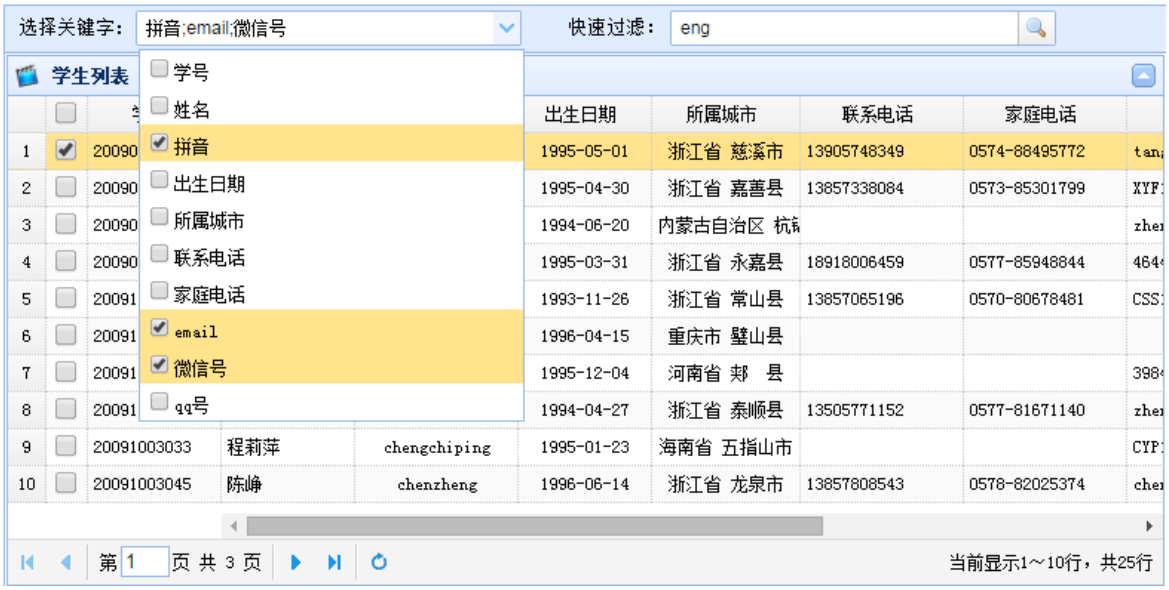


图 4-8 网格数据过滤和定位程序运行界面

为实现网格中数据查询，在工具栏中定义一个关键字组合框 searchfield 和一个文本输入框 searchtext。组合框可以选择网格中的各个列作为查询关键字，文本框为查询的内容。当文本框和查询关键字都为非空时，可以按模糊匹配方式（like %）过滤出满足条件的所有学生记录。

数据定位是在所有学生中找到第一条满足条件的记录，并将光标聚焦到这一条记录上。当有多条记录满足查询条件时，可在不同记录之间进行上下移动定位。相对而言，数据定位比过滤更复杂些，具体可参考 example33_dbgridsearch.jsp 程序。

本实例程序运行界面如图 4-8 所示，具体实现过程及方法如下：

①由于数据网格相关参数较多，这里定义一个对象变量 pmyGrid1 用来存储网格参数，其主要参数及其含义如下：

```
var pmyGrid1={};  //定义一个对象变量
pmyGrid1.id='myGrid1';  //网格名称
pmyGrid1.parent='main';  //父类容器标识符
pmyGrid1.staticsql="select StudentID,Name,pycode,case when Gender='f' then '女' else '男' end as
gender,"+"birthdate,province+' '+city as hometown,mobile,homephone,email,"+
"weixin,qq from students";  //原始不变的查询语句
pmyGrid1.activesql=pmyGrid1.staticsql;  //根据过滤条件动态变化的查询语句
pmyGrid1.gridfields="[@l%c#90]姓名/name;[@c%c#110,2]拼音/pycode;
[#90@c]出生日期/birthdate;[@c#100]所属城市/hometown;"+"[110]联系电话/mobile;
[110]家庭电话/homephone;[140]Email/email;"+
"[130]微信号/weixin;[100]QQ 号/qq";  //各列标题、字段名、宽度、类型等
pmyGrid1.fixedfields='[90]学号/studentid';  //固定列的标题、字段名、宽度、类型等
pmyGrid1.title='学生列表';  //标题
pmyGrid1.checkbox='single';   //单选还是多选或无复选框
pmyGrid1.pagesize=10;     //每页显示行数
```

②调用函数 myGridColumns()，从固定列和滚动列字符串中分割提取各个列名标题、字段名、列宽度、对其方式等，生成符合 EasyUI 网格要求的列集对象。如果需要行复选框，则将复选框加入到固定列中。将所有的列合并加载到 pmyGrid1.columns 对象数组中，以备数据过滤中使用。

```
pmyGrid1.columns=[];
pmyGrid1.scrollcolumns=[];
pmyGrid1.fixedcolumns=[];
pmyGrid1.checkboxcolumn=[];
//生成滚动列
pmyGrid1.scrollcolumns=myGridColumns(pmyGrid1.gridfields);
//生成固定列
pmyGrid1.fixedcolumns=(pmyGrid1.fixedcolumns).concat(myGridColumns(pmyGrid1.fixedfields));;
//所有列合并到 pmyGrid1.columns 中
pmyGrid1.columns=(pmyGrid1.fixedcolumns).concat(pmyGrid1.scrollcolumns);
//判断和处理复选框列
if (pmyGrid1.checkbox!=''){
    pmyGrid1.checkboxcolumn.push({field:"checkboxid", width: 20, checkbox: true, align:"center"});
}
pmyGrid1.fixedcolumns=pmyGrid1.checkboxcolumn.concat(pmyGrid1.fixedcolumns)
```

③定义网格控件 myGrid1，并设置其主要属性。在设置属性时会使用到 pmyGrid1 对象变量中的许多参数。

```
var str='<div id="myGrid1" class="EasyUI-datagrid" data-options=""></div>';
$("#"+pmyGrid1.parent).append($(str)); //$("#main").append($(str));
if (pmyGrid1.width==0) pmyGrid1.width='100%';
if (pmyGrid1.height==0) pmyGrid1.height='100%';
var myGrid1=pmyGrid1.id;
```

```
$("#myGrid1").datagrid({
    title: ' '+pmyGrid1.title,
    width: pmyGrid1.width,
    height: pmyGrid1.height,
    iconCls: "panelIcon",
    nowrap: true,
    collapsible:true,
    pagePosition: 'bottom',   //top,both
    autoRowHeight: false,
    rownumbers:true,
    pageNumber: 1,
    striped: true,
    columns: [pmyGrid1.scrollcolumns],
    frozenColumns: [pmyGrid1.fixedcolumns]
});
if (pmyGrid1.checkbox='single'){
    $("#myGrid1").datagrid({singleSelect: true});
}else if (pmyGrid1.checkbox!=''){
    $("#myGrid1").datagrid({singleSelect: false});
}
```

④定义分页模式，使用 pmyGrid1 对象变量中的 pagesize 等参数。

```
if (pmyGrid1.pagesize>0){
    $("#myGrid1").datagrid({
        pagination: true,
        pageSize: pmyGrid1.pagesize
    });
    var pg = $("#"+pmyGrid1.id).datagrid("getPager");
    (pg).pagination({
        showPageList: false,   //是否显示设置每页显示行数的组合框
        beforePageText: '第', //页数文本框前显示的汉字
        afterPageText: '页      共 {pages} 页',
        displayMsg: '当前显示{from}～{to}行，共{total}行',
        onRefresh:function(pageNumber,pageSize){
            fnLoadGridData(pmyGrid1,pageNumber);
        },
        onChangePageSize:function(pageSize){   //变换每页显示行数时触发
            var opts = $('#myGrid1').datagrid('options');
            opts.pageSize=pageSize;
            fnLoadGridData(pmyGrid1,1);
        },
        onSelectPage:function(pageNumber,pageSize){   //变换页码时触发
            var opts = $('#myGrid1').datagrid('options');
            opts.pageNumber=pageNumber;
            opts.pageSize=pageSize;
            fnLoadGridData(pmyGrid1,pageNumber);
        }
    });
}
```

⑤定义网格其属性和事件之后，调用 fnLoadGridData()函数从服务器中提取第一页，显示在网格中。注意：本例 fnLoadGridData()函数代码与上个实例 32 不同。

```
var opts =$("#myGrid1").datagrid('getPager').data("pagination").options;
```

```
opts.pageNumber=1;  //设置当前页
opts.pageSize=pmyGrid1.pagesize; //设置每页显示行数
fnLoadGridData(pmyGrid1,1);  //从服务器其中取数
```

⑥编写 fnLoadGridData()函数程序，输入一个对象变量和当前页码。使用 datagrid('getPager')方法将分页码设置到当前页。

```
function fnLoadGridData(pmyGrid1,pageNumber){  //数据库中分页取数据
        var myGrid1=pmyGrid1.id;
        var pager =$("#myGrid1").datagrid('getPager');
        pager.pagination('refresh',{     //改变选项，并刷新分页栏信息
            pageNumber: pageNumber
        });
        var pageSize=opts.pageSize;
        $.ajax({
            type: "Post",
            url: "system/Easyui_getGridData.jsp",
            data: {
                database: sysdatabasestring,
                selectsql: pmyGrid1.activesql,
                keyfield: pmyGrid1.keyfield,
                sortfield: '',
                limit: pageSize,
                start: (pageNumber-1)*pageSize,
                summaryfields:''
            },
            async: false,
            success: function(data) {
                eval("var source="+data+";");
                $('#myGrid1').datagrid( "loadData", source );  //必须用 loaddata
                //根据总行数改变行号的列宽度，改 css
                var rowcount=$("#myGrid1").datagrid('getData').total+'';  //转换为字符型
                var width=(rowcount.length*6+8);
                if (width<25) width=25;
                var px=width+'px';
                $('.datagrid-header-rownumber,.datagrid-cell-rownumber').css({"width": px});
                $('#myGrid1').datagrid('resize');  //必须写
                $('#myGrid1').datagrid('selectRow',0); //选中某行
            }
        });
    }
```

⑦定义关键字组合框和查询文本框。组合框中关键字选项来自于网格中的各个列标题，用 tmp 变量存储。

```
//从网格中提取列标题作为组合框选项
var tmp='';
for (var i=0;i<pmyGrid1.columns.length;i++){
    if (tmp!='') tmp+=';';
    tmp+=pmyGrid1.columns[i].title;
}
myComboField('searchfield','toolbar','选择关键字:',75,4,10,0,260,tmp,'','checkbox');
myTextField('searchtext','toolbar','快速过滤：',65,4,380,0,200,'','');
```

⑧编写文本框小按钮的点击事件开始过滤数据库。先在组合框中找出所有选中的关键字，根据文本框内容，生成一个查询语句的 where 条件。由于组合框给出的选中值是列标题，而查询条

件需要的是字段名，这时需要逐个循环到数据网格列中找到选中列标题所对应的字段名。本例 where 条件中使用的是模糊匹配查找，各个条件之间用 or 连接。实施过滤时，调用 fnLoadGridData() 函数，重新从数据库加载满足条件的第一页记录。

```
$('#searchtext').textbox({
    buttonIcon:'locateIcon',   //定义小按钮
    onClickButton: function(e){   //小按钮点击事件
        var xtext=$('#searchtext').textbox("getValue");   //查找的字符串
        var xfields=';'+$('#searchfield').combobox("getText")+';';   //选中的列标题
        var wheresql='';   //循环根据列标题获取对应的字段名，生成模糊匹配查询条件
        for (var i=0;i<pmyGrid1.columns.length;i++){
            if (xfields.indexOf(';'+pmyGrid1.columns[i].title+';')>=0){
                if (wheresql!='') wheresql+=' or ';
                wheresql+=pmyGrid1.columns[i].field+" like '%"+xtext+"%'";
            }
        }
        if (wheresql!='' && xtext!='') pmyGrid1.activesql='select * from ('+pmyGrid1.staticsql+') as p
        where '+wheresql;
        else pmyGrid1.activesql=pmyGrid1.staticsql;
        var opts =$("#myGrid1").datagrid('getPager').data("pagination").options;
        opts.pageNumber=1;
        fnLoadGridData(pmyGrid1,1);
    }
});
```

本实例可直接调用 myGrid()等函数构造控件及其相关事件，使得代码更加简化，具体程序见 example33_dbgridsearch.jsp 文件。

相关知识点：

①获取一个网格的列信息，包括固定列信息。

```
var opts = $('#myGrid1').datagrid('options');
var gridcols = opts.frozenColumns[0];
for (var i=0;i<gridcols.length;i++){
    if (i>0) tmp+=';';
    tmp+=gridcols[i].title;
}
var gridcols = opts.columns[0];   //opts.frozenColumns[0]
for (var i=0;i<gridcols.length;i++){
    if (i>0) tmp+=';';
    tmp+=gridcols[i].title;
}
```

②数组合并可使用 concat()函数，例如：

```
pmyGrid1.fixedcolumns=pmyGrid1.checkboxcolumn.concat(pmyGrid1.fixedcolumns)
```

③提取分页模式中的相关参数。设置当前起始页的页码，使用下列两种方式在 pagination 中刷新页码：

```
var pager =$("#myGrid1").datagrid('getPager');
pager.pagination('refresh',{    //改变选项，并刷新分页栏信息
    pageNumber: pageNumber   //设置页码
});
var opts =$("#myGrid1").datagrid('getPager').data("pagination").options;
opts.pageNumber=pageNumber;
```

实例 34. 可编辑数据网格及其数据增删改操作的实现。

可编辑网格是指网格从数据库中提取数据后，在相关的单元格上可直接新增、修改和删除数据，并将编辑过的数据保存到数据库中去。在 EasyUI 中，可编辑网格分为行编辑和单元格编辑两种模式，主要方法是为可编辑列添加一个编辑器（editor），在编辑器中指定控件的类型和属性。此外还需要在 onCellClick 事件中使用 beginEdit 和 endEdit 方法控制行的编辑状态。

本例构造一个有关教师信息的数据网格，每个列都定义为可编辑列，其中教师编码、姓名、拼音、联系电话为文本框（textbox），出生日期为日期框（datebox），性别、职称和党派为组合框（combobox），所属城市为组合树（combotree），研究方向为复选组合框（combobox）。

除此之外，该数据网格还包含一个工具栏，点击菜单中“新增”按钮，在网格的第一行插入一条新空白记录，点击菜单中“删除”按钮，直接删除该条记录，并将光标定位到相邻的记录上；点击“保存”按钮，先将本页原有记录全部删除，再重新在数据库中插入本页全部记录。

本例实现换页时自动保存当前页数据的功能，为保证网格中当前正在编辑的行数据不被丢失，需要在换页事件和各个按钮的点击事件中使用 endEdit 结束编辑当前行。当前行用 pmyGrid1.rowindex 变量全程记录和跟踪。

本实例程序运行界面如图 4-9 所示，具体实现过程及方法如下：

图 4-9 可编辑数据网格程序运行界面

①定义一个对象变量 pmyGrid1，其中 pmyGrid1.rowindex 用于保存网格中正在聚焦编辑的行序号（其初值为-1）。当用户点击网格的某个单元格时其 onCellClick 事件被触发，pmyGrid1.rowindex 值随之改变。pmyGrid1.staticsql 为数据网格的查询语句，pmyGrid1.data 用于存储网格当前页中从数据库提取的原始数据，即网格被修改过之前的数据。

```
var pmyGrid1={};
pmyGrid1.id='myGrid1';  //网格名称
pmyGrid1.parent='main';  //父类容器标识符
pmyGrid1.staticsql="select teacherid,Name,pycode,case when Gender='f' then '女' else '男' end as gender,"+
"birthdate,province,city,title,party,research,mobile from teachers";
pmyGrid1.activesql=pmyGrid1.staticsql;  //根据过滤条件动态变化的查询语句
pmyGrid1.pagesize=10;      //每页显示行数
pmyGrid1.keyfield='teacherid';  //数据表关键字
pmyGrid1.sql_party="select Pycode as id,description as party from dictionary where Type='党派'";
pmyGrid1.sql_title="select Pycode as id,description as title from dictionary where Type='职称'";
pmyGrid1.sql_city="select AreaName as text,AreaName as id,areaname as city,* from Areas";
pmyGrid1.rowindex=-1;
pmyGrid1.data=[];
//定下网格中的组合框（下拉框）的数据源
pmyGrid1.gendersource=[{gender:'男'},{gender:'女'}];
```

```
pmyGrid1.researchsource=[{research:'区域经济学'},{research:'国际经济贸易'},{research:'财务管理'},
{research:'电子商务'},{research:'企业信息化'}];
```

②调用 myRunSelectSql()函数，执行查询语句从服务器端数据库中提取“职称”、“党派”组合框的选项；调用 myLoadComboTreeData()函数，从数据库中获取城市组合树的节点，并采用分层逐级展开各级子节点。

```
pmyGrid1.titlesource=myRunSelectSql(sysdatabasestring, pmyGrid1.sql_title);
pmyGrid1.partysource=myRunSelectSql(sysdatabasestring, pmyGrid1.sql_party);
pmyGrid1.citysource=myLoadComboTreeData('city',pmyGrid1.sql_city,'areaid',' ',' ');
```

③定义网格列属性，在编辑器 editor 中指定可编辑列的类型、属性及事件。限于篇幅，下面仅给出文本框、组合框、组合树的定义方法。在定义组合树和复选组合框时使用了多个事件，具体见本例完整程序代码。

```
var xcolumns=[[{
  title:'教师编码', field:'teacherid', width:75, halign:'center', align: 'left',
  editor:{
      type : 'textbox', options : {required: true}
  }
},{   title: '性别', field: 'gender', width: 55, halign:'center', align: 'center',
  editor:{
      type: 'combobox',
      options:{
            valueField:'gender',
            textField:'gender',
            panelHeight:'auto',
            data: pmyGrid1.gendersource
       }
  }
},{   title: '出生日期', field: 'birthdate', width: 95, halign:'center', align: 'center',
  editor:{
      type : 'datebox'
  }
},{ title: '职称', field: 'title', width: 95, halign:'center', align: 'left',
  editor : {
      type : 'combobox',
      options : {
            valueField:'title',
            textField:'title',
            panelHeight:'auto',
                  data: pmyGrid1.titlesource
      }
  }
},{ title: '出生地', field: 'city', width: 150, halign:'center', align: 'left',
  editor : {
      type : 'combotree',
      options : {
            valueField: 'city',
            textField: 'city',
            panelHeight: 255,
            panelWidth: 180, //'auto',
            data: pmyGrid1.citysource,
            onBeforeExpand: fnExpandComboTree   //点击展开事件
      }
  }
}]];
```

④定义网格控件 myGrid1，指定显示行号和行单选框，并使用 toolbar 属性指定工具栏 myToolBar1。为节省篇幅，这里使用函数 myGridPaging(pmyGrid1)定义网格分页模式。网格定义和使用下列四个事件，其主要作用如下：

◆ onClickCell 事件。当用户点击网格中的一个单元格时（包括换行时），开启这行的编辑状态，关闭之前行的编辑状态，使得整个网格始终只有一行处于编辑状态。使用 pmyGrid1.rowindex 变量记录当前光标所聚焦的行号，为下一次关闭这行的编辑状态作准备。

```
onClickCell: function(index,data){
    $("#myGrid1").datagrid('beginEdit', index);   //新行可以编辑
    if (pmyGrid1.rowindex!=index && pmyGrid1.rowindex>=0) {
        $("#myGrid1").datagrid('endEdit', pmyGrid1.rowindex);
    }
    pmyGrid1.rowindex=index;   //记录新行号
},
```

◆ onSelect 事件。当用户点击行左端的选择框（checkbox）时，网格将光标聚焦到这行，并开启这行的编辑状态，关闭之前行的编辑状态，与 onClickCell 事件的程序代码相同。

```
onSelect: function(index,data){
    if (pmyGrid1.rowindex!=index && pmyGrid1.rowindex>=0) {
        $("#myGrid1").datagrid('endEdit', pmyGrid1.rowindex);
    }
    pmyGrid1.rowindex=index;   //记录新行号
}
```

◆ onLoadSuccess 事件。当网格从数据库中成功加载一页数据后，使用 pmyGrid1.Data 变量将这一页数据保存起来，而网格数据可以由用户编辑修改。网格数据经过修改需要保存时，先在数据库中删除原有数据，再插入新数据。这时需要借助于 pmyGrid1.Data 这个值。

```
onLoadSuccess: function(data){
    pmyGrid1.data=data.rows;
},
```

◆ onBeforeLoad 事件。网格从数据库中加载新一页数据之前（即换页前触发），先需要将旧一页的数据保存到数据库，这里直接调用 fn_save 函数。在换页时为保存当前正在编辑的这一行数据不被丢失，需要在 datagrid("getPager")换页和刷新页的 onSelectPage 和 onRefresh 事件中添加结束行编辑语句：if (pmyGrid1.rowindex>=0) $("#myGrid1").datagrid('endEdit',pmyGrid1.rowindex); 具体参见 Easyui_function.js 文件中的 myGridPaging()函数。

```
onBeforeLoad:function(){
    fn_save();   //换页之前保存当前页的记录
}
```

⑤由于教师“出生地”城市组合树时分层逐级展开，在其 onBeforeExpand 事件中编写程序，点击展开某个父节点的子节点。在这个事件中，关键技术是通过 getEditors 方法，获取网格中 city 这一列对应的编辑器名称或序号，根据这个名称通过 editors[i].target.combotree('tree')语句获取组合框中的树 tree 对象，以便之后使用 append 和 remove 增删树中的子节点。

```
function fnExpandComboTree(node){
    var pid=eval("node.areaid");
    var editors = $('#myGrid1').datagrid('getEditors', pmyGrid1.rowindex);
    for (var i=0;i<editors.length;i++){
        if (editors[i].field=='city'){
            var xcbtree=editors[i].target.combotree('tree');
            break;
```

```
            }
        }
        var child_node = xcbtree.tree('getChildren', node.target);
        if (child_node.length==1 && child_node[0].id=='_'+pid){ //生成子节点
            var xsql="select * from ("+pmyGrid1.sql_city+") as p where parentnodeid='"+pid+"'";
            $.ajax({
                url: "system\\Easyui_getChildNodes.jsp",
                data: { database: sysdatabasestring, selectsql: xsql, keyfield:'areaid', sortfield:'' },
                async: false,
                success: function(data) {
                    var source=eval(data);
                    xcbtree.tree('remove', child_node[0].target); //删除子节点
                    xcbtree.tree('append', {   //增加数据作为子节点
                        parent: node.target,
                        data: source
                    });
                }
            });
        }
    }
```

⑥点击新增记录时，使用 endEdit 方法关闭原来行的编辑状态，否则网格不会保存这行数据。之后使用 insert 方法在第一行之前插入一个新空行，并将光标聚焦到这一行。设置新增行的 rownumber 列值为-1，作为新增行与原有行的差异标记。

```
function fn_add(){
    $("#myGrid1").datagrid('endEdit', pmyGrid1.rowindex);
    $("#myGrid1").datagrid('insertRow', { //插入一行到最前行
        index: 0,
        row: {rownumber:"-1"}
    });
    $("#myGrid1").datagrid('beginEdit', 0);
    pmyGrid1.rowindex = 0;
    $("#myGrid1").datagrid('selectRow', 0);
}
```

⑦点击删除行时，使用 deleteRow 方法从网格中删除当前行。删除前先判断网格是否为空（当网格有没有记录时予以提示不能删除记录），删除后将光标指向下一行，没有下一行时指向上一行。

```
function fn_delete(){
    var rows=$("#myGrid1").datagrid('getRows');
    if (rows.length>0){
        $('#myGrid1').datagrid('deleteRow', pmyGrid1.rowindex);
        if (pmyGrid1.rowindex>rows.length-1 && rows.length>0){
            pmyGrid1.rowindex--;   //选中上一行，否则自动选下一行
        }
        $("#myGrid1").datagrid('selectRow', pmyGrid1.rowindex);//选中下一行
    }else{
        myMessagebox('系统提示','记录已经被全部删除！');
    }
}
```

⑧点击保存记录时，先从 pmyGrid1.data 中提取本页原始数据中的教师编码值（即主码值），生成 delete 语句，从数据库中删除原来记录；然后为网格（可能修改过）每行数据生成一条 Insert 语句，向数据库插入记录。注意，点击“保存”按钮时，必须结束当前行的编辑状态，否则网格不保存当前行的修改数据。在生成 Insert 语句时，先调用 Easyui_getEditableFields.jsp 程序从服务器

数据库获取表中所有可编辑的列，再判断这些列在网格中是否进行了编辑修改。

```
if (pmyGrid1.rowindex>=0) $("#myGrid1").datagrid('endEdit',pmyGrid1.rowindex);
    var sql='';   //生成 delete 语句删除旧记录
    for (var i=0;i<pmyGrid1.data.length;i++){
        sql+="\n delete teachers where teacherid='"+pmyGrid1.data[i].teacherid+"'";
    }
    var fieldset='';    //从服务器提取数据表所有可编辑的列名信息
    $.ajax({
        type: "Post",
        url: "system//Easyui_getEditableFields.jsp",
        data: {
            database: sysdatabasestring,
            tablename: 'teachers'
        },
        async: false,
        success: function(data) {
            eval("fieldset="+data+";");
        }
    });
    var rows = $("#myGrid1").datagrid('getRows');   //获取本页全部行
    for (var i=0;i<rows.length;i++){
        var record=rows[i];
        var sql1='';
        var sql2='';
        for (var j=0; j<fieldset.length; j++){                var field=fieldset[j].field;
            var value=eval("record."+field);
            if (value!=undefined){ //判断网格中有没有可编辑的列
                if (sql1!=''){
                    sql1+=',';   sql2+=',';
                }
                sql1+=field;
                sql2+="'"+value+"'";
            }
        }
    sql+="\n insert into teachers ("+sql1+") values ("+sql2+")";
    myRunUpdateSql(sysdatabasestring, sql);   //合起来执行 delete 和 insert 语句
}
```

作业题：

1．编写程序，在保存网格数据时先对数据的正确性进行验证。

2．网格编辑数据时，如果发现教师编码和姓名输入错误时（例如值为空时），编写程序让光标一直聚焦在当前单元格上，直到输入正确为止。

3．编写程序实现功能：教师编码这一列只有在新增记录时才可能编辑，其他情况下不能编辑。

实例 35. 数据网格的表单关联扩展。

在 EasyUI 中，数据网格除了可以用行列显示数据外，还可以点击展开以表单或其他形式显示某一行的详细信息，这是数据网格的一种扩展功能，称为数据网格详细视图（DataGrid DetailView）。由于数据网格本身不带行扩展的 DetailView 功能，因此在使用这项功能时需要引用 EasyUI 另外一个插件（datagrid-detailview.js）。

本例在一个数据网格中列表显示商品信息，通过 view 属性引入网格详细视图 detailview，在 detailFormatter 属性中定义一个 DIV 容器，以表单形式显示商品详细信息。这时数据网格每行左侧会出现一个加号小图标（+），点击这个加号将展开一行，并触发 onExpandRow 事件。在这个

事件中，可以编写程序将更多商品信息加载到表单中展示出来。本例程序运行界面如图 4-10 所示，具体实现过程和方法如下：

图 4-10 数据网格行扩展程序运行界面

①在 jqEasyUI/jQuery.EasyUI. min.js 之后引入插件 datagrid-detailview.js，否则无法调用 detailview 实现行扩展功能。

```
<head>
    …
    <script type="text/javascript" src="jqEasyUI/jQuery.min.js"></script>
    <script type="text/javascript" src="jqEasyUI/jQuery.EasyUI.min.js"></script>
    <script type="text/javascript" src="jqEasyUI/datagrid-detailview.js"></script>
    …
</head>
```

②定义一个数据网格 myGrid1，除其他属性外，将 view 属性值设为 detailview，将 detailFormatter 属性值设为一个“myForm”字符串开头的 div 容器（本例使用的是一个 panel 类型控件）。由于不同的行在扩展时需要不同名称的容器，因此每个 div 命名时添加一个行号参数 index。

```
$("#myGrid1").datagrid({
    title: ' 产品列表',
    iconCls: "panelIcon",
    width:800,
    height:555,
    ……
    frozenColumns: xfixedcolumns,
    columns: xcolumns,
    view: detailview,
    detailFormatter:function(index,row){
        return '<div id="myForm' + index + '" style="position:relative; padding:5px 0"></div>';
    },
    ……
});
```

③定义数据网格的行扩张器事件 onExpandRow。在该事件中，首先设置在 detailFormatter 中指定的 div 表单容器（myForm1,2..,n）的属性（如高度、宽度、边框等），然后在表单容器内添加两个 fieldset 子容器。一个子容器用 label 控件显示商品详细信息，另一个子容器显示商品图片。由于不同行不能使用相同名称的控件，因此在触发 onExpandRow 时需要将两个子容器内的各个控件用 remove()方法全部清除后再重新定义。当然，这种方法限制了数据网格同时只能展开一行，即在展开一行时，将之前展开的其他行收缩关闭。需要指出的是，onExpandRow 事件中，提取行信息的主要对象是 row 参数。

```
$("#myGrid1").datagrid({
        title: ' 产品列表',
        width:800,
        height:555,
        ……
        onExpandRow: function(index,row){
            //同一页只能同时展开一行
            if(pmyGrid1.rowindex>=0 && pmyGrid1.rowindex!=index) {
                $('#myGrid1').datagrid('collapseRow',pmyGrid1.rowindex);
            }
            pmyGrid1.rowindex=index;
            //设置 div 容器的属性
            $('#myForm'+index).panel({
                border:true,
                cache:false,
                height:140,
                width:'100%',   //655
                onLoad:function(){
                    $('#myGrid1').datagrid('fixDetailRowHeight', index);
                }
            });
            $('#myGrid1').datagrid('fixDetailRowHeight', index);
            //删除 myFieldset1 和 myFieldset2 控件，其容器内的所有控件也同时被删除
            if ($("#myFieldset1").length>0) $("#myFieldset1").remove();
            if ($("#myFieldset2").length>0) $("#myFieldset2").remove();
            //调用自定义函数，添加两个容器控件，其类型为 fieldset，没有边框 border
            myGroupbox('myFieldset1','myForm'+index,0,100,0,0);
            myGroupbox('myFieldset2','myForm'+index,0,0,0,0);
            //在子容器 1 内添加文字 label 控件显示商品信息。
            myLabelField('productid','myFieldset1','商品编码：'+row.productid,25*0+10,18,0,120);
            myLabelField('productname','myFieldset1','商品名称：'+row.productname,25*0+10,228,0,0);
            myLabelField('englishname','myFieldset1','英文名称：'+row.englishname,25*1+10,18,0,0);
            myLabelField('quantityperunit','myFieldset1','规格型号：'+row.quantityperunit,25*2+10,
            18,0,220);
            myLabelField('unit','myFieldset1','计量单位：'+row.unit,25*2+10,228,0,130);
            myLabelField('suppliername','myFieldset1','供应商：'+row.suppliername,25*3+10,18,0,0);
            myLabelField('unitprice','myFieldset1','单价：'+row.unitprice,25*4+10,18,0,100);
            //在子容器 2 中添加一个图形控件显示商品 jpg 图片
            var src='mybase/products/'+row.productid+'.jpg';
            myImageField('picture','myFieldset2','',0,2,1,134,98,src);
        }
    });
```

相关知识点：

①容器自定义函数 function myGroupbox(id, parent, top, left, height, width)生成一个 fieldset 类型的无边框容器，用于存放其他控件，其主要参数的含义依次为容器标识符、父类容器、起始位置坐标 y 与 x、容器大小高度与宽度。这是一个没有边框的容器，采用绝对地址定位，具体代码见 Easyui_function.js 文件。应用示例：

```
myGroupbox('myFieldset1','myForm1', 10, 10, 500,300);
myGroupbox('myFieldset2','myForm1', 0, 0, 0, 0);
```

②数据网格行展开使用 expand 方法，行收缩使用 collapse 方法；提取当前行中的字段值，使用 row 参数。

③div 容器及其元素的动态定义方法。如果要实现同时多行展开，可以将两个 fieldset 子容器和所有 label 控件的标识符都改成与行号（index）相关的名称。

实例 36. 树形网格（TreeGrid）的综合应用。

树形网格（treegrid）以网格形式显示分层数据。它基于数据网格（datagrid），并结合树视图（treeview）和可编辑网格的特征。树形网格允许创建可定制的、可异步展开的行，并以多列形式显示分层数据。树形网格在属性、事件和方法上继承数据网格或树控件，在节点处理上与树的主要差异在于它依赖于节点的 id 值，而非节点的 target 值。

本例定义一个树形网格控件 myTreeGrid1，按树形结构分层显示商品信息。该控件绑定一个右键菜单，具有新增、修改、删除和保存商品记录等功能。在新增和修改商品时，使用一个子窗口编辑商品信息，其数据更新处理方式与树控件相似。本例程序运行界面如图 4-11 所示，具体实现过程与方法如下：

图 4-11. 树形网格分层展开节点程序运行界面

①定义树形网格 myTreeGrid1 的各个列，其属性和方法与数据网格（datagrid）相同；在定义单价时使用 formatter 为其保留两位小数和非零显示。

```
sql_products="select productid+' '+productname as text,* from products";
var xcolumns=[[
    { title: '商品编码', field:'productid', width: 120, halign:'center', align: 'left' },
    { title: '商品名称', field:'productname', width: 160, halign:'center', align: 'left' },
    { title: '英文名称', field:'englishname', width: 200, halign:'center', align: 'left' },
    { title: '规格型号', field:'quantityperunit', width: 110, halign:'center', align: 'left' },
    { title: '计量单位', field:'unit', width: 100, halign:'center', align: 'left' },
    { title: '单价', field: 'unitprice', width: 75, halign:'center', align: 'right',
```

```
            formatter: function(value){
                if (value==undefined || value==0) value='';   //非零显示
                else value=(1.0*value).toFixed(2);
                return '<div align="right">' + value+'</div>';
            }
        }
    ]];
```

②定义树形网格控件 myTreeGrid1 及其属性，其中 treeField 和 idField 是必选项。定义固定列 frozenColumns 时必须设置 rownumbers 为 true。定义一个右键菜单，利用树形网格的 onContextMenu 事件与 myTreeGrid1 绑定。

```
var str='<table id="myTreeGrid1" class="EasyUI-treegrid" style="overflow:auto"></table>';
$("#main").append($(str));
trGrid=$("#myTreeGrid1");
 myMenu('myMenu1','新增节点/mnadd/addIcon;新增子节点/mnaddsub/addIcon;
 修改/mnedit/editIcon; - ; 删除/mndelete/deleteIcon;-;刷新/mnrefresh/refreshIcon',");
 trGrid.treegrid({
     title: ' 商品分类列表',
     iconCls: "panelIcon",
     width:785,
     height:350, //'100%',
     collapsible: false,
     singleSelect:true,
     rownumbers: true,
     treeField: 'productid',
     idField: 'id',
     frozenColumns:[[{ title: '商品编码', field:'productid', width: 120, halign:'center', align: 'left' }]],
     columns: xcolumns,
     onContextMenu: function(e, row){   //定义右键菜单
         e.preventDefault();
         trGrid.treegrid("select", row.id);   //点击右键同时选中光标所在的行
         $("#myMenu1").menu('show', {
             left: e.pageX,
             top: e.pageY
         });
     }
});
```

③定义一个信息编辑窗口 myEditWin1，在窗口内添加商品信息编辑框，其中包括一个商品图片列。四个隐藏列仅用于保存树形网格中每个节点的父节点信息，其作用类似于变量，但不受变量作用域的影响。调用 mySelectOnFocus()函数可以光标聚焦选中控件的全部文本字符；调用函数 myKeyDownEvent()可以让 productid、productname、englishname、quantityperunit、unit 及 unitprice 等文本框控件绑定 keydown 事件，实现表单输入数据过程中的回车键和上下键的键盘移动控制。

```
//定义窗口及其编辑控件
myWindow('myEditWin1','编辑商品',0,0,300,660,'save;cancel','modal;close;drag');
myFieldset('myFieldset1','myEditWin1','商品信息',10,10,205,410);
myFieldset('myFieldset2','myEditWin1','',17,435,198,194);
myTextField('productid','myFieldset1','商品编码：',70,36*0+20,18,0,120,'');
myTextField('productname','myFieldset1','商品名称：',70,36*1+20,18,0,310,'');
myTextField('englishname','myFieldset1','英文名称：',70,36*2+20,18,0,310,'');
myTextField('quantityperunit','myFieldset1','规格型号：',70,36*3+20,18,0,310,'');
myTextField('unit','myFieldset1','计量单位：',70,36*4+20,18,0,130,'');
myNumberField('unitprice','myFieldset1','单价：',40,36*4+20,258,24,100,12,2,'0');
myFileField('filename','myFieldset1',36*5+20,10,0,320,'right','fnPictureupLoad');
```

```
myImageField('picture','myFieldset2','',0,1,1,231,198,'');
myHiddenFields("addoredit;level;parentnodeid;ancester"); //添加隐藏列，用作变量
mySelectOnFocus();//调用通用函数，实现聚焦时全选文本框
//调用通用函数，下列各列实现键盘光标上下移动聚焦
myKeyDownEvent('productid;productname;englishname;quantityperunit;unit;unitprice');
```

④从后台数据库中提取商品数据，为树形网格 myTreeGrid1 添加第一层节点，其后台程序与树控件相同，只是虚拟空子节点的 ID 值不能为空；

定义 myTreeGrid1 节点点击展开事件 onBeforeExpand，当用户点第一次点击某个父节点时，程序从后台数据库提取这个父节点的所有下级子节点显示出来；同样地，定义 onDblClickRow 事件，双击父节点时展开或收缩其下级子节点，这里，expand 和 collapse 使用的参数是.id，而不是树控件中的.target。

```
//提取 TreeGrid1 第一层节点
var sqlx="select * from ("+sql_products+") as p where parentnodeid="";   //第一层
$.ajax({
    url: "system\\Easyui_getChildNodes.jsp",
    data: { database: sysdatabasestring, selectsql: sqlx, keyfield:'productid', sortfield:''},
    async: false,
    success: function(data) {
        var source=eval(data);
        trGrid.treegrid({ data: source });   //加载 JSON 数据到树形网格
    }
});
//定义 myTreeGrid1 父节点点击展开事件
trGrid.treegrid({
    onBeforeExpand: function (node){   //点击展开事件
        var pid=node.id;
        var sql=sql_products;
        var sqlx="select * from ("+sql+") as p where parentnodeid='"+pid+"'";
        var child_node = trGrid.treegrid('getChildren', pid);
        if (child_node.length==1 && child_node[0].id=='_'+pid){ //生成子节点
            $.ajax({
                url: "system\\Easyui_getChildNodes.jsp",
                data: {database: sysdatabasestring, selectsql: sqlx, keyfield:'productid', sortfield:'' },
                async: false,
                success: function(data) {
                    var source=eval(data);
                    trGrid.treegrid('remove', child_node[0].id); //删除虚拟子节点
                    trGrid.treegrid('append', {  //增加数据作为子节点
                        parent: pid,   //这里不能用 node.target,
                        data: source
                    });
                }
            });
        }
    },
    //双击展开或收缩节点事件
    onDblClickRow: function(row){
        if (row.state=='closed') $(this).treegrid('expand', row.id);
        else $(this).treegrid('collapse', row.id);
    }
});
```

⑤定义“新增节点”（即新增兄弟节点）的点击事件。因为新节点总是在父节点之下添加的，因此在新增节点时需要提取其父节点，如父节点为空，则增加根节点。新增节点时，首先在树形网格的父节点下添加一个临时空节点，并用隐藏控件变量记录新节点的层次号、父节点及其祖先节点等值；然后将窗口 myEditWin1 中的所有商品信息清空，打开窗口后由用户输入信息。这时商品编码是可以编辑的，树控件聚焦（选中）在新增加的那个空节点上（如图 4-12 所示）。

图 1-12　新增商品节点时树控件的聚焦位置

```
$('#mnadd').bind('click',function(item){
    $("#addoredit").val('add');   //标志值，区分新增节点、子节点或修改节点三种状态值
    var records=trGrid.treegrid("getSelections");   //取当前节点
    var pnode=trGrid.treegrid("getParent",records[0].id); //取其父节点
    if (pnode!=null){
        var parentstr=pnode.productid+'-'+pnode.productname;
        $("#myEditWin1").window('setTitle','新增商品【所属类别：'+parentstr+'】');
        trGrid.treegrid('expand', pnode.id);   //展开父节点
        trGrid.treegrid('append',{   //添加一个空的临时子节点
            parent:pnode.id,
            data:[{id:'_'+pnode.id, productid:' '}]
        });
        trGrid.treegrid('select','_'+pnode.id);   //聚焦临时子节点
        $("#parentnodeid").val(pnode.productid); //记录新节点的父节点
        $("#level").val(1*pnode.level+1); //记录新节点的层次号
        $("#ancester").val(pnode.ancester+pnode.id+"#"); //记录新节点的祖先节点
    }else{
        $("#myEditWin1").window('setTitle','新增商品【所属类别：无】');
        trGrid.treegrid('append',{    //在根节点中增加一个兄弟节点，空的临时节点
            parent:null,
            data:[{id:'_x',productid:' '}]
        });
        trGrid.treegrid('select','_x');
        $("#parentnodeid").val("");   //记录新节点的父节点
        $("#level").val(1);    //记录新节点的层次号
        $("#ancester").val("");    //记录新节点的祖先节点
    }
    //清空商品信息
    $("#productid").textbox("setValue","");
    $("#productname").textbox("setValue","");
```

```
        $("#englishname").textbox("setValue","");
        $("#quantityperunit").textbox("setValue","");
        $("#unit").textbox("setValue","");
        $("#unitprice").textbox("setValue","");
        $("#picture").attr('src','')textbox("setValue","");
        $("#productid").textbox("readonly",false);   //设置主键可编辑
        $('#myEditWin1').window('open');        //打开窗口
    });
```

⑥定义“新增子节点”的点击事件。新增子节点的实现方式与新增新节点相似，不同只是父节点及其相关值（当前节点即为父节点）。本例使用一个 addoredit 隐藏变量记录和区分三种不同数据编辑状态（即新增节点、新增子节点、修改节点），并在不同的编辑状态下，对窗口 myEditWin1 的标题进行不同设置。

```
    $('#mnaddsub').bind('click',function(item){
        $("#addoredit").val('addsub');
        var records=trGrid.treegrid("getSelections");
        var pnode=records[0];
        var parentstr=pnode.productid+'-'+pnode.productname;
        $("#myEditWin1").window('setTitle','新增商品【所属类别：'+parentstr+'】');
        trGrid.treegrid('expand',pnode.id);
        trGrid.treegrid('append',{
            parent:pnode.id,
            data:[{id:'_'+pnode.id,productid:' '}]
        });
        trGrid.treegrid('select','_'+pnode.id);
        $("#parentnodeid").val(pnode.productid);
        $("#level").val(1*pnode.level+1);
        alert(pnode.level);
        $("#ancester").val(pnode.ancester+pnode.id+"#");
        $("#productid").textbox("setValue","");
        $("#productname").textbox("setValue","");
        $("#englishname").textbox("setValue","");
        $("#quantityperunit").textbox("setValue","");
        $("#unit").textbox("setValue","");
        $("#unitprice").textbox("setValue","");
        $("#picture").attr('src','');
        $("#productid").textbox("readonly",false);   //设置主键可编辑
        $('#myEditWin1').window('open');
    });
```

⑦定义“修改节点”的点击事件。修改节点时将当前节点（records[0]）中的值提取出来复制到窗口 myEditWin1 的各个相应控件中去，这时商品编码（主键）是只读的（不能修改），右侧图片从服务器中提取 src 地址值，调用 myResizeImage()函数显示出来。Src 文件路径为/mybase/products/，文件名由商品编码与 jpg 组成（如 70201.jpg）

```
    $('#mnedit').bind('click',function(item){
        $("#addoredit").val('edit');
        var records=trGrid.treegrid("getSelections");
        if (records.length>0){
            var pnode=trGrid.treegrid("getParent",records[0].id); //取其父节点
            if (pnode!=null){
                var parentstr=pnode.productid+'-'+pnode.productname;
                $("#myEditWin1").window('setTitle','修改商品【所属类别：'+parentstr+'】');
            }else{
                $("#myEditWin1").window('setTitle','修改商品【所属类别：无】');
```

```
            }
            $("#parentnodeid").val(records[0].parentnodeid);
            $("#level").val(records[0].level);
            $("#ancester").val(records[0].ancester);
            $("#productid").textbox("setValue",records[0].productid);
            $("#productname").textbox("setValue",records[0].productname);
            $("#englishname").textbox("setValue",records[0].englishname);
            $("#quantityperunit").textbox("setValue",records[0].quantityperunit);
            $("#unit").textbox("setValue",records[0].unit);
            $("#unitprice").textbox("setValue",records[0].unitprice);
            $("#productid").textbox("readonly",true);   //设置主键只读
            var src='mybase/products/'+records[0].productid+'.jpg';
            $("#picture").attr('src',src);
            myResizeImage('picture',src,231,198);
            $("#myEditWin1").window("open");
        }
    }
```

⑧定义“删除节点”的点击事件。删除节点之前先判断确定，如果商品编码是一个父节点，则删除一个大类的商品，即包括删除商品的所有层次的子节点（包括子节点的子节点）。这个过程借助祖先列 ancester 的值实现，例如删除编码为 7 的商品，则 SQL 语句为 DELETE Products WHERE ProductID='7' or Ancester Like '7#%' 。

节点删除后，需要处理两个环节：i）树形网格中光标的聚焦问题。一般在删除节点后将光标聚焦到下一个兄弟节点上，如果没有兄弟节点时则聚焦到父节点。这个过程的实现有点繁琐。ii）存在这样的可能，即一个节点被删除后，其父节点可能再也没有子节点而成为叶子节点，这时需要将原来父节点的 isparentflag 标志值从 1 改为 0。

```
    $('#mndelete').bind('click',function(item){
        var records=trGrid.treegrid("getSelections");
        if (records.length>0){
            if (records[0].isparentflag==0){
                var msg=' 是否确定删除商品【'+records[0].productid+'】？';
            }else{
                var msg=' 是否确定删除商品大类【'+records[0].productid+'】？';
            }
            $.messager.confirm('系统提示', msg, function(r){
                if (r){
                    errmsg=[];
                    var pnode=trGrid.treegrid('getParent',records[0].id);   //求出父节点
                    var sql="delete products where productid='"+records[0].productid+"'";
                    sql+=" or ancester like '"+records[0].productid+"#%'";
                    var result=myRunUpdateSql(sysdatabasestring, sql); //执行删除
                    if (result.error!='') errmsg.push(result.error);
                    //判断删除之后原来父节点是否还有子节点，如果没有则将其 isparentflag 设置为 0
                    if (pnode!=null){
                        sql="update products set isparentflag=0 where productid='"+pnode.productid+"'";
                        sql+=" and not exists(select 1 from products where
                          parentnodeid='"+pnode.productid+"')";
                        var result=myRunUpdateSql(sysdatabasestring, sql); //执行删除
                        if (result.error!='') errmsg.push(result.error);   //记录 sql 语句运行错误
                    }
                    if (errmsg.length>0) myMessageBox('系统提示','记录删除失败！',errmsg);
                    else{
                        //删除之后将光标聚焦到兄弟节点，没有兄弟节点时，聚焦到父节点
                        var pnode=trGrid.treegrid('getParent',records[0].id);   //求出父节点
```

```
                    if (pnode!=null) var children=trGrid.treegrid('getChildren',pnode.id);   //求子节点
                    else var children=trGrid.treegrid('getRoots');   //求出子节点,即为所有根节点
                    var priorbrother=null;
                    var nextbrother=null;
                    for (var i=0;i<children.length;i++){
                        if (children[i].id==records[0].id) break;
                    }
                    if (i<children.length-1) nextbrother=children[i+1];
                    if (i>0) priorbrother=children[i-1];
                    trGrid.treegrid("remove",records[0].id);   //删除节点
                    if (nextbrother!=null){
                        trGrid.treegrid('select',nextbrother.id);   //定位到下一个兄弟节点
                    }else if (priorbrother!=null){
                        trGrid.treegrid('select',priorbrother.id); //定位到上一个兄弟节点
                    }else if (pnode!=null){
                        trGrid.treegrid('select',pnode.id);   //定位到父节点
                    }
                    if (pnode!=null && children.length==1){//判断父节点是否变成叶子节点
                        trGrid.treegrid('update',{
                            id:pnode.id,
                            row: {"isparentflag":"0"}
                        });
                    }
                }
            }
        });
    }else{
        $.messager.confirm('系统提示','不存在记录，删除失败！');
    }
}
```

⑨定义窗口“保存”按钮的点击事件。在保存数据时，先从 myEditWin1 各个控件中取值存放在 9 个变量中。然后根据 addoredit 值判断是修改节点之后还是新增节点之后保存数据，两种操作的 SQL 语句不同。修改节点使用 update 语句，只需对 6 个列进行修改，而新增节点则使用 insert 语句，还需要对另外四个列（parentnodeid、level、isparentflag 及 ancester）进行赋值。新增节点的 isparentflag 值总是为 0，但其父节点之前可能是叶子节点，这时需要将父节点的 isparentflag 值由 0 改为 1。

在数据库中保存节点记录之后，还需要对树形网格当前节点的值进行刷新，这个过程通过 treegrid('update',{id:xid, row:data});方法实现。如果放弃保存，则删除原来新增的空节点。

```
$('#myEditWin1SaveBtn').bind('click',function(item){
    var errmsg=[];
    var addoredit=$("#addoredit").val();
    var records=trGrid.treegrid("getSelections");
    var node=records[0];
    var pnode=trGrid.treegrid("getParent",node.id);
    var s1=$("#productid").textbox('getValue');
    var s2=$("#productname").textbox('getValue');
    var s3=$("#englishname").textbox('getValue');
    var s4=$("#quantityperunit").textbox('getValue');
    var s5=$("#unit").textbox('getValue');
    var s6=$("#unitprice").textbox('getValue');
    if (s6=='') s6='0';   //输错时值为空,不必 iaNaN 判断
    var s01=$("#parentnodeid").val();
    var s02=$("#level").val();
```

```
        var s03=$("#ancester").val();
        if (addoredit=='edit'){   //修改记录
            var sql="update products set productname='"+s2+"',";
            sql+="englishname='"+s3+"',";
            sql+="quantityperunit='"+s4+"',";
            sql+="unit='"+s5+"',";
            sql+="unitprice="+s6;
            sql+=" where productid='"+s1+"'";
        }else{      //新增记录
var sql="insert into products (productid,productname,englishname,quantityperunit, unit,unitprice,
            parentnodeid, isparentflag,level,ancester)";
            sql+="values('"+s1+"','"+s2+"','"+s3+"','"+s4+"','"+s5+"','"+s6+"','"+s01+"',0,"+s02+",'"+s03+"')";
            //设置父节点的 isparentflag=1
            if (pnode!=null) sql+="\n update products set isparentflag=1 where
            productid='"+pnode.productid+"'";
        }
        var result=myRunUpdateSql(sysdatabasestring, sql); //执行删除
        if (result.error!=''){
            errmsg.push(result.error);
            myMessagebox('系统提示','记录保存失败！',errmsg);
        }else{   //修改或插入成功之后更新节点
            $("#parentnodeid").val(s01);
            $("#level").val(s02);
            $("#ancester").val(s03);
            var xid=records[0].id;
            var data={'id':s1,
                'productid':s1,
                'productname':s2,
                'englishname':s3,
                'quantityperunit':s4,
                'unit':s5,
                'unitprice':s6,
                'parentnodeid':s01,
                'level':s02,
                'ancester':s03,
                'isparentflag':'0'
            };
            trGrid.treegrid('update',{id:xid, row:data});   //修改更新当前节点
            //修改树中节点之后关闭窗口
            $('#myEditWin1').window('close');
        }
    });
```

⑩编写窗口 myEditWin1 的 onClose 事件。由于新增节点和子节点时，在树形网格中添加了一个临时的空节点，这个空节点在 myEditWin1 数据没有被保存的情况下是需要被删除的。有两种情况会关闭窗口，一是点击“取消”按钮，二是点击窗口标题上的小×。这两种情况都会触发 myEditWin1 的 onClose 事件，因此只要在这个事件中删除空节点就可以保证树形网格中的节点与后台数据库节点的一致性。临时空节点被删除后，光标将聚焦到其父节点上。

```
    //定义窗口“取消”按钮事件
    $('#myEditWin1CancelBtn').bind('click',function(item){
        $('#myEditWin1').window('close');
    });
    //定义窗口关闭事件
    $("#myEditWin1").window({
```

```
        //如果在窗口编辑商品信息时用户选择“取消”，则关闭窗口，并将空节点删除。
        onClose:function(){
            var records=trGrid.treegrid("getSelections");
            var records=trGrid.treegrid("getSelections");
            if (records.length>0 && records[0].productid.trim()==''){
                var pnode=trGrid.treegrid('getParent',records[0].id);
                if (pnode==null) pnode=trGrid.treegrid('getRoot');
                trGrid.treegrid('remove',records[0].id);
                if (pnode!=null) trGrid.treegrid('select',pnode.id);
            }
        }
    });
```

相关知识点：

①TreeGrid 控件的定义和使用：idField 和 treeField 的用途；固定列定义时必须将 rownumbers 设置为 true（这可能是个 bug）。

②TreeGrid 在节点增删改、取父节点、取子节点等操作语句与 Tree 和 DataGrid 控件存在区别。

③树形网格的右键定义方法（与 datagrid 和 tree 也有所不同）。

④树形网格分层逐级展开时需要为每一个父节点添加一个临时空节点，这些空节点的 id 值不能为空（tree 控件可以为空），否则会因为空值重复而无法聚焦定位空节点而进行删除操作。

⑤EasyUI 的 numberbox 控件不能输入字符，但存在 bug 可以输入汉字或空格，这时控件返回的值为空，而不是用 isNaN 去判断。

⑥网格控件（包括数据网格和树形网格）中列数据的格式化 formatter 方法。

作业题：

1．编写程序，实现实例中“刷新”按钮对应的功能，即重新从数据库中提取所有节点的数据。

2．编写程序在节点数据保存到数据库之前先验证数据，包括新增商品节点时，商品编码是否重复、商品编码和名称不能为空、单价不能为负值。

3．编写程序，在实例 myTreeGrid1 的双击事件（onDblClickRow）中编写程序，但用户点双击叶子节点时，弹出 myEditWin1 窗口，显示商品的详细信息（只读，vuneng 修改）。

4．调用后台程序 Easyui_getAllTreeNodes.jsp，编写程序实现 TreeGrid 控件一次性从数据库全部提取节点。

第 5 章、图表统计及其应用

实例 37. FusionCharts 单序列图表及应用。

FusionCharts 是一个跨平台、跨浏览器的 flash 图表组件解决方案，能够被 JSP 、ASP.NET、PHP、Ruby on Rails、简单 HTML 页面甚至 PPT 调用。FusionCharts 可以接受多种格式的数据，其中最常用之一是 XML 格式。

FusionCharts 单序列图表是指一个图表中只显示一个指标值，图表分为 X 和 Y 两个（二维）坐标。常用的 Fusioncharts 单序列图表包括折线图、圆柱图（直方图）、面积图、栏位图、圆饼图和环饼图等。生成一个单序列图表需要设置下列参数：①图表的标题；②X 及 Y 轴坐标的名称；③图表曲线依赖的数据；④图表外观形状参数（包括标题和注释等）。上述五种单序列图表其标题、注释和数据格式基本相同。下面举例给出单序列图表的 XML 数据描述规范。

```
<chart
    //图表标题
    caption="2012 年 01 月 01 日～2012 年 12 月 31 日销售额变化图果蔬汁饮料"
    XAxisName="月份" YAxisName="销售额" numberPrefix="" showValues="0"
    formatNumberScale="0" showAboutMenuItem="0" seriesNameInToolTip="1"
    chartTopMargin="10" chartBottomMargin="10" chartLeftMargin="10" chartRightMargin="10"
    baseFont="宋体" baseFontSize="10"
    useRoundEdges="1" animation="0" palette="2" >
    //图表数据
    <set label='201201' value='621733.16' />
    <set label='201202' value='547200.86' />
    <set label='201203' value='634607.84' />
    <set label='201204' value='590234.98' />
    <set label='201205' value='600609.30' />
    <set label='201206' value='550518.67' />
    <set label='201207' value='510044.83' />
    <set label='201208' value='528015.67' />
    <set label='201209' value='677015.57' />
    <set label='201210' value='523706.70' />
    <set label='201211' value='558824.63' />
    <set label='201212' value='636873.32' />
    //图表注释
    <styles>
      <definition>
        <style type="font" color="666666" name="CaptionFont" font="黑体" size="16" />
          <style type="font" name="SubCaptionFont" font="宋体" size="12" bold="0" />
      </definition>
      <application>
          <apply toObject="caption" styles="CaptionFont" />
        <apply toObject="SubCaption" styles="SubCaptionFont" />
      </application>
  </styles>
</chart>
```

可以发现，Fushionchart 单序列图表的关键技术是将数据库数据按照上述 XML 格式提取出来。

本例使用 FusionCharts 图表显示商品销售额的月度变化趋势。页面布局时使用 EasyUI-layout 控件将屏幕分成三个区域，顶部工具栏输入一个统计查询日期区间，屏幕上方显示商品分类树，屏幕下方用标签页（tabs）分别显示商品月度销售额的折线图、圆柱图（直方图）、面积图、栏位图、圆饼图和环饼图。所有数据取自数据库中商品表（Products）、订单表（Orders）和订单明细

表（Orderitems）。为方便和规范地描述图表属性，本例定义一个图表对象变量 pmyChart11，其主要参数及含义举例如下：

```
var pmyChart11={};
pmyChart11.colorset=' ';      //图表中每段 F6C11F';
pmyChart11.type='single';   //图表类型（单序列、多序列、圆锥、仪表盘等）
pmyChart11.title=date1+"～"+date2+"销售额变化图\n"   //图表标题
pmyChart11.xAxisName="月份";        //X 轴坐标名称
pmyChart11.yAxisName='销售额';     // Y 轴坐标名称
pmyChart11.labelfield='month';        // X 轴坐标取值对应的列名
pmyChart11.valuefield='amount';     //Y 轴坐标取值对应的列名
pmyChart11.data=source;          //图表数据，是一个 JSON 数组
pmyChart.type=='single'          //单序列图表标志值，多序列为 multiple
pmyChart11.height=350;        //图表显示的高度
pmyChart11.width=550;       //图表显示的宽度
```

本例程序运行界面如图 5-1 所示，具体实现过程与方法如下：

图 5-1 FusionCharts 单序列图表程序运行界面

①引入 FusionCharts 插件 fusioncharts.js；定义树控件 myTree1，使用 tree1=$("#myTree1")，在随后引用时可使用 tree1 简化书写过程。定义日期区间、标签页等其他相关控件。

```
var pmyTree1={};
pmyTree1.sql="select cast(productid as varchar(16))+' '+productname as text,* from products";
pmyTree1.keyfield='productid';
myDBTree('myTree1','treepanel','商品分类',0,0,0,0,pmyTree1.sql,pmyTree1.keyfield,",");
myTabs('myTab','main','折线图;直方图;面积图;栏位图;圆饼图;环饼图',0,0,0,0,");
```

```
myDateField('datefrom','toolbar','设置区间：',70,4,16,0,100,'2012-01-01','');
myDateField('dateto','toolbar','～',16,4,198,0,100,'2012-12-31','');
myCheckBoxField('x3d','toolbar','',0,5,340,0,1,'是否 3D 显示？');
myButton('cmdok','toolbar','开始',4,450,25,68,'searchIcon','','');
var tree1=$("#myTree1");
```

②根据日期区间编写查询语句，从数据库中统计检索某段时间内某个商品或某类商品的月度销售汇总额，并将每月销售额数据保存到图表变量 pmyChart1.data 中。在统计某类商品销售时需要按祖先列 ancester 的值进行查询。例如，计算 2012 年至 2013 年编码为 1 的这个商品大类的销售额，其查询语句如下：

```
select sum(a.amount) as amount,100*year(orderdate)+month(orderdate) as month from orderitems a
join orders b on a.orderid=b.orderid
where productid in (select productid from products where ancester '1#%' and isparentflag=0)
and orderdate between '2012-01-01' and '2013-12-31'
group by year(orderdate),month(orderdate)
order by year(orderdate),month(orderdate)
```

将上述查询转换成带日期区间参数和商品编码参数之后，其 SELECT 语句描述如下。

```
var node=tree1.tree('getSelected');
var flag=node.isparentflag;
var xid=node.productid;      //取树节点对应的商品编码
//var xid=eval("node."+pmyTree1.keyfield);   //取树节点对应的商品编码
if (flag>0){     //是父节点，即一个商品大类
    var sql="select sum(a.amount) as amount,100*year(orderdate)+month(orderdate) as month
    from orderitems a join orders b on a.orderid=b.orderid ";
    sql+="\n where productid in (select productid from products where ancester like '"+xid+"#%'
    and sparentflag=0)";
    sql+="\n and orderdate between '"+$("#datefrom").textbox("getValue")+"' and ";
    sql+=" '"+$("#dateto").textbox("getValue")+"'";
}else{   //是叶子节点，即一个具体商品
    var sql="select sum(a.amount) as amount,100*year(orderdate)+month(orderdate) as month
    from orderitems a join orders b on a.orderid=b.orderid ";
    sql+=" where productid='"+xid+"'";
    sql+=" and orderdate between '"+$("#datefrom").textbox("getValue")+"' and ";
    sql+=" '"+$("#dateto").textbox("getValue")+"'";
}
sql+="\n group by year(orderdate),month(orderdate)";
sql+="\n order by year(orderdate),month(orderdate)";
var date1=myDateboxValue('datefrom','');
var date2=myDateboxValue('dateto','');
var pmyChart1={};
pmyChart1.data=myRunSelectSql(sysdatabasestring, sql);   //执行查询语句，将结果放在 data 变量中
pmyChart1.type='single';   //单序列
pmyChart1.title=date1+"～"+date2+"销售额变化图\n"+node.productname;
pmyChart1.xAxisName="月份";
pmyChart1.yAxisName='销售额';
pmyChart1.labelfield='month';
pmyChart1.valuefield='amount';
pmyChart1.height=0;
pmyChart1.width=0;
```

③根据查询语句结果和 pmyChart1 其他参数值生成单序列图表的 xml。首先获取图表 XML 文件头部分（是一个字符串，调用 myGetChartXMLHeader()函数获取），然后将每一行的查询结果转换成XML语句，最后获取XML文件尾部注释部分（也是一个字符串，调用函数myGetChartXMLFooter()获取）。

```
pmyChart1.xml=myGetChartXMLHeader(pmyChart1);   //取 xml 的文件头部分
var xml='\n';
for (var i=0;i<pmyChart1.data.length;i++){
    var s1=eval('pmyChart1.data['+i+'].'+pmyChart1.labelfield);
    var s2=eval('pmyChart1.data['+i+'].'+pmyChart1.valuefield);
        xml+="<set label='"+eval('pmyChart1.data['+i+'].'+pmyChart1.labelfield)+"'";
        xml+=" value='"+eval('pmyChart1.data['+i+'].'+pmyChart1.valuefield)+"'";
        if (pmyChart1.sliced){
            xml+=" issliced='1'";
        }
        xml+=' />\n';
    }
    pmyChart1.xml+=xml;
    pmyChart1.xml+=myGetChartXMLFooter(pmyChart1);
```

④显示图表。以显示折线图为例（其 flash 图片文件为 Line.swf），第一条语句为新建一个 FusionCharts，包括 flash 文件名称和路径、图表大小；第二条语句将 XML 语句加载到图表中；第三条语句将 flash 图表在一个层（myTab1）中渲染显示出来。

```
var chart = new FusionCharts("jqEasyUI/fusioncharts/swf/Line.swf","chartid1","550","350","0","1");
chart.setDataXML(pmyChart1.xml);
chart.render('myTab1');
```

实例 38. FusionCharts 多序列图表及应用。

多序列 FusionCharts 图表是指在一个图表内显示多条折线图、圆柱图、栏位图或面积图，其实现原理与单序列图表基本相同，不同的只是 XML 文件格式和 flash 图表文件（单序列与多序列图表的 XML 文件头和文件尾也相同，但序列数据格式不同）。常用的多序列 flash 文件包括：折线图（MSSpline.swf、MSLine.Swf）、圆柱图（直方图，MSColumn3D.swf、MSColumn2D.swf）、圆柱折线图（MSColumnLine3D.swf、MSColumn3DLineDY.swf、MSColumnLine3D.swf）、面积图（MSArea.swf、MSSplineArea.swf）、栏位图（MSBar3D.swf、MSBar2D.swf）。下面给出多序列图表 XML 文件的数据格式。

```
<chart
….
  <categories>
        <category label="201201" />
        <category label="201202" />
        <category label="201203" />
        <category label="201204" />
        <category label="201205" />
        <category label="201206" />
        <category label="201207" />
        <category label="201208" />
        <category label="201209" />
        <category label="201210" />
        <category label="201211" />
        <category label="201212" />
  </categories>
  <dataset seriesname="销售额" >
        <set    value="621733.16" />
```

```
            <set    value="547200.86" />
            <set    value="634607.84" />
            <set    value="590234.98" />
            <set    value="600609.30" />
            <set    value="550518.67" />
            <set    value="510044.83" />
            <set    value="528015.67" />
            <set    value="677015.57" />
            <set    value="523706.70" />
            <set    value="558824.63" />
            <set    value="636873.32" />
        </dataset>
        <dataset seriesname="利润额" parentYAxis="S" >
            <set    value="136178.66" />
            <set    value="128875.86" />
            <set    value="146734.59" />
            <set    value="131835.73" />
            <set    value="134418.80" />
            <set    value="117936.67" />
            <set    value="118966.58" />
            <set    value="121128.67" />
            <set    value="153139.57" />
            <set    value="113197.95" />
            <set    value="129061.13" />
            <set    value="143710.07" />
        </dataset>
        <styles>
        ….
        </styles>
</chart>
```

图 5-2 FusionCharts 多序列图表程序运行界面

本例程序功能与界面设计与实例 37 相似，程序运行界面如图 5-2 所示。下面给出多序列 XML 文件生成程序。

```
xml+="\n<categories>";
for (var i=0;i<pmyChart.data.length;i++){
    var s1=eval('pmyChart.data['+i+'].'+pmyChart.labelfield);
    xml+="\n <category label=\""+s1+"\" />";
}
xml+="\n</categories>";
var yfields=pmyChart.valuefield.split(';');
var ylabels=pmyChart.yAxisName.split(';');
//处理某个指标
var xcolor=[]
if (pmyChart.colorset!=undefined) xcolor=pmyChart.colorset.split(';');
j=0;
for (var k=0;k<yfields.length;k++){   //第一个序列指标
    xml+="\n<dataset seriesname=\""+ylabels[k]+"\"";
    if (k>0) xml+=" parentYAxis=\"S\"";
    xml+=" >";
    for (var i=0;i<pmyChart.data.length;i++){
        var s1=eval('pmyChart.data['+i+'].'+pmyChart.labelfield);
        var s2=eval('pmyChart.data['+i+'].'+yfields[k]);
        xml+="\n <set ";
        if (xcolor.length>0){
            xml+=" color=\""+xcolor[j]+"\"";
            j++;
            if (j>=xcolor.length) j=0;
        }
        xml+=" value=\""+s2+"\"";
        //处理钻取的链接命令
        if (pmyChart.drilldown!=undefined){
        xml+=" link='javascript:"+pmyChart.drilldown+"(\""+s1+"\")'";
        }
        xml+=" />";
    }
    xml+="\n</dataset>";
}
pmyChart.xml=myGetChartXMLHeader(pmyChart);   //取 xml 的文件头部分
pmyChart.xml+=xml;
pmyChart.xml+='\n'+myGetChartXMLFooter(pmyChart);
return pmyChart;
```

实例 39. FusionCharts 图表向下钻取。

在多维分析中，钻取可以改变维的层次，变换分析的粒度，包括向上钻取（roll up）和向下钻取（drill down）两种方式。向下钻取是指从汇总数据（summery data）深入到细节数据进行分析或增加新维。例如，用户分析“各地区、城市的销售情况”时，可以对某一个城市的销售额细分为各个年度的销售额，对某一年度的销售额，可以继续细分为各个季度的销售额。

FusionCharts 图表钻取时可以单击父表的圆柱图、圆饼图或折线图等，向下钻取到子图表从而查看更为详细的信息。它允许利用单个数据源无限层级向下钻取图表，可以在同一窗口、新窗口、指定的 frame 中实现钻取，实现各种链接功能，也可以触发 JavaScript 函数。

FusionCharts 实现图表钻取的关键是在 set 上定义一个 link 属性标签。下列三种形式，分别向下钻取链接一个网页、一个后台程序和一个 JS 函数。

```
<set label='Jan' value='17400' link='DemoLinkPages/DemoLink1.html'/>     //链接网页
<set label='Mar' value='21800' link=' link='ShowDetails.jsp?Month=6' >   //链接后台程序
<set label='Apr' value='23800' link='javascript:fnGenReport("6")' />     //链接 JS 函数
```

本例在页面上创建一个环饼图和一个圆饼图，分别显示销售额和利润额最大前 n 个客户的份额比重。点击环饼图上的某个客户，向下钻取该客户每个月份的销售额，并在屏幕下方的一个圆柱折线图中显示出来。点击圆柱折线图中的某个月份，可以向下进行二级钻取，显示该月份这个客户购买的具体商品的销售额及利润额。点击圆饼图上的某个客户，向下钻取该客户每个月份的利润额和利润率，并在屏幕上弹出两个子窗体，分别以数据网格和仪表盘形式显示出来。本例程序运行界面如图 5-3 所示，程序实现的主要过程与方法如下。

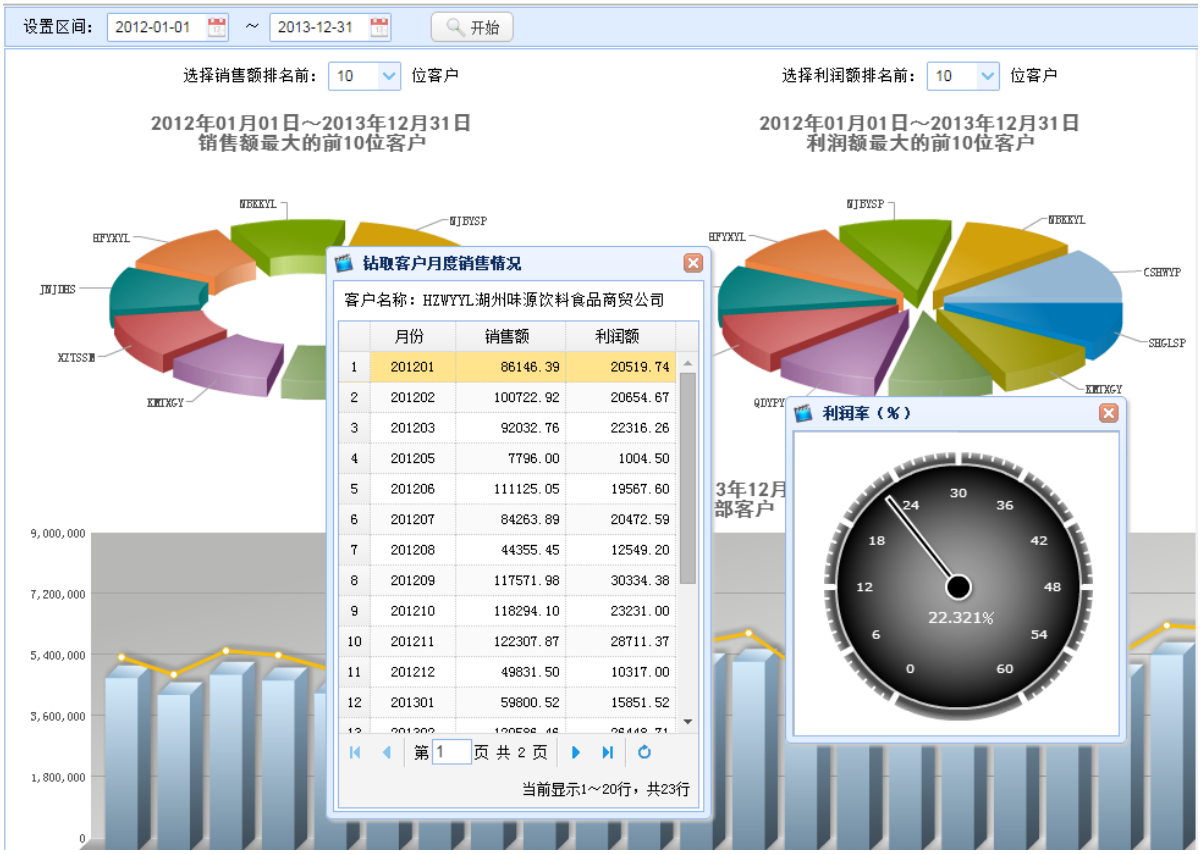


图 5-3. FusionCharts 向下钻取程序运行界面图

①页面布局。定义三个层，分别用来显示环饼图、圆饼图和圆柱折线图；定义三个窗口控件，分别用来显示客户销售情况月报表（数据网格）、客户购买商品情况表（数据网格）和客户利润率（仪表盘）；定义日期区间和排名下拉框等控件。

```
$("#pie1").css(myTextCss(40,2,300,490));
$("#pie2").css(myTextCss(40,500,300,490));
$("#chartdiv1").css(myTextCss(340,5,400,600));
myDateField('datefrom','toolbar','设置区间:',65,6,16,0,100,'2012-01-01','');
myDateField('dateto','toolbar','～',16,6,198,0,100,'2013-12-31','');
myButton('cmdok','toolbar','开始',5,350,25,68,'searchIcon','');
myComboField('rank1','main','选择销售额排名前:',115,10,145,0,60,'10;20;30;40;50','');
myComboField('rank2','main','选择利润额排名前:',115,10,635,0,60,'10;20;30;40;50','');
myLabel('rankx1','main','位客户',10+4,332);
myLabel('rankx2','main','位客户',10+4,822);
```

```
myWindow('myGridWin1','钻取客户月度销售情况',220,200,470,316,'','close;drag');
myWindow('myGridWin2','钻取商品销售情况',0,0,595,455,'','close;drag');
myWindow('myChartWin1','利润率（%）',320,640,285,280,'','drag;close');
```

②画环饼图（其 flash 图形文件名为 Doughnut3D.swf）。编写查询语句，计算指定日期区间内的销售额最大的前 n 个客户的销售额，将查询结果保存到数组变量 source1 中；定义环饼图对象变量 pmyChart1，其中 pmyChart1.title 为环饼图的标题，pmyChart1.labelfield 和 pmyChart1.valuefield 分别为环饼图中 xml 文件对应的 label 和 value 标签取值的列，pmyChart1.data 为环饼图对应的数据源（即查询结果集 source1），pmyChart1.sliced 表示环饼图处于切开状态。pmyChart1.drilldown 指定环饼图的 link 标签在向下钻取时调用 fnDrillDownChart(customerid)函数处理数据，将客户编码值（参数为 customerid）传递给向下钻取的程序，生成一个圆柱折线图显示下钻客户每个月的销售额和利润额。在设置完这些参数后，调用 myGetChartXML(pmyChart1)函数生成 XML 语句，调用 myShowFusionChart()函数在 pie1 层上显示环饼图。

```
//取日期区间
var date1=myDateboxValue('datefrom','');
var date2=myDateboxValue('dateto','');
//计算销售额和利润额
var sql="select top "+$("#rank1").textbox("getValue")+" customerid,sum(a.amount) as amount
from orderitems a join orders b on a.orderid=b.orderid ";
sql+="\n where orderdate between '"+$("#datefrom").textbox("getValue")+"' and ";
sql+=" '"+$("#dateto").textbox("getValue")+"'";
sql+="\n group by customerid";
sql+="\n order by amount desc ";
var source1=myRunSelectSql(sysdatabasestring, sql);   //将销售额和利润额的数据保存到数组变量
var pmyChart1={};
pmyChart1.title=date1+"～"+date2+"\n 销售额最大的前"+$("#rank1").textbox("getValue")+"位客户";
pmyChart1.xAxisName="月份";   //x 轴坐标名称
pmyChart1.yAxisName='销售额';   //Y 轴坐标名称
pmyChart1.labelfield='customerid';   //label 标签对应的列名
pmyChart1.valuefield='amount';      //value 标签对应的列名
pmyChart1.data=source1;           //图表数据源
pmyChart1.type='single';           //圆饼图为单序列图表
pmyChart1.sliced=true;           //圆饼图展开
pmyChart1.height=300;       //圆饼图高度
pmyChart1.width=490;              //圆饼图宽度
pmyChart1.drilldown='fnDrillDownChart(customerid)';    //向下钻取时调用的 js 函数名称及参数
pmyChart1=myGetChartXML(pmyChart1);       //生成圆饼图的 xml 语句
myShowFusionChart(pmyChart1,'Doughnut3D','pie1');       //显示圆饼图
```

③画圆饼图（其 flash 图形文件名为 PieD.swf）。处理过程和参数设置方法与环饼图相同，在向下钻取时调用 fnDrillDownCustomerGrid(customerid)函数处理数据，将客户编码值（参数为 customerid）传递给向下钻取的程序，生成一个数据网格显示下钻客户每个月的销售额和利润额、一个仪表盘显示客户的利润率。在设置完这些参数后，调用 myGetChartXML(pmyChart1)函数生成 xml 语句，调用 myShowFusionChart()函数在 pie2 层上显示环饼图。

```
//画圆饼图，先计算每个客户的销售额和利润额
var sql="select top "+$("#rank2").textbox("getValue")+" customerid,sum(a.amount) as amount,
SUM(a.amount-a.Quantity*c.unitprice) as profit from orderitems a ";
sql+="\n join orders b on a.orderid=b.orderid ";
sql+="\n join products c on a.productid=c.productid ";
sql+="\n where orderdate between '"+$("#datefrom").textbox("getValue")+"' and ";
```

```
sql+=" '"+$("#dateto").textbox("getValue")+"'";
sql+="\n group by customerid";
sql+="\n order by profit desc ";
var source2=myRunSelectSql(sysdatabasestring, sql);//将查询结果保存到数组变量 source2
var pmyChart2={};
pmyChart2.title=date1+"～"+date2+"\n 利润额最大的前"+$("#rank2").textbox("getValue")+"位客户";
pmyChart2.xAxisName="月份";
pmyChart2.yAxisName='利润额';
pmyChart2.labelfield='customerid';
pmyChart2.valuefield='profit';
pmyChart2.data=source2;
pmyChart2.sliced=true;
pmyChart2.height=300;
pmyChart2.width=490;
pmyChart2.type='single';
pmyChart2.drilldown='fnDrillDownCustomerGrid(customerid)'; //钻取时使用的函数和传递的参数
pmyChart2=myGetChartXML(pmyChart2);
myShowFusionChart(pmyChart2,'Pie3D','pie2');
```

④环饼图钻取生成圆柱折线图（flash 文件名为 MSColumn3DLineDY.swf）。根据上级图表钻取时 link 标签中指定客户编码值（cid），计算出每个客户每个月份的销售额和利润额，存放到 pmyChart2.data 变量中。由于圆柱折线图为多序列图表，因此需要将 pmyChart3.type 设置为'multiple'，将 pmyChart3.valuefield 设置为 amount;profit 两个列。

由于圆柱折线图本身也具有向下钻取的功能，因此将 pmyChart3.drilldown 设置为 fnGenProductGrid(month,"'+cid+'")函数。该函数包含月份和客户编码两个参数，向下钻取时可以网格形式得到当前客户某个月份所购买的商品信息。这里，客户编码是上一级钻取时传递过来的参数，在下一次钻取时将是一个常量，也就是说圆柱折线图下钻时只有月份是可以选择变的。pmyChart3.drilldown 值举例：pmyChart3.drilldown=fnGenProductGrid(month, "HFYXYL")。

```
function fnDrillDownColumnLineChart(cid){   //向下钻取圆柱折线图
    var date1=myDateboxValue('datefrom',");
    var date2=myDateboxValue('dateto',");
    //计算钻取客户每个月的销售额和利润额
    var sql="select sum(a.amount) as amount,SUM(a.amount-a.Quantity*c.unitprice) as profit,
    100*year(orderdate)+month(orderdate) as month from orderitems a ";
    sql+="\n join orders b on a.orderid=b.orderid ";
    sql+="\n join products c on a.productid=c.productid";
    sql+="\n where orderdate between '"+$("#datefrom").textbox("getValue")+"' and ";
    sql+=" '"+$("#dateto").textbox("getValue")+"'";
    if (cid!=") sql+=" and customerid='"+cid+"'";
    sql+="\n group by year(orderdate),month(orderdate)";
    sql+="\n order by year(orderdate),month(orderdate)";
    var source3=myRunSelectSql(sysdatabasestring, sql);   //将结果集保存到一个数组变量
    var pmyChart3={};
    pmyChart3.title=date1+"～"+date2+"销售额变化图";
    if (cid!="){   //根据客户编码提取客户名称
        sql="select customerid,customername from customers where customerid='"+cid+"'";
        var customer=myRunSelectSql(sysdatabasestring, sql);
        pmyChart3.title+="\n"+customer[0].customerid+customer[0].customername;
    }else{
        pmyChart3.title+="\n 全部客户";
```

```
        }
        pmyChart3.xAxisName="月份";
        pmyChart3.labelfield='month';
        pmyChart3.yAxisName='销售额;利润额';
        pmyChart3.valuefield='amount;profit'; //两个序列的列名
        pmyChart3.data=source3;
        pmyChart3.type='multiple';
        pmyChart3.height=0;
        pmyChart3.width=800;
        pmyChart3.drilldown='fnGenProductGrid(month,'"+cid+"')';  //进行而二级钻取时将月份和客户编码传递给函数
        pmyChart3=myGetChartXML(pmyChart3);
        myShowFusionChart(pmyChart3,'MSColumn3DLineDY','chartdiv1');
        //钻取开始后，关闭三个无关的子窗口
        $("#myGridWin1").window("close");
        $("#myGridWin2").window("close");
        $("#myChartWin1").window("close");
    }
```

⑤圆柱折线图钻取生成数据网格。该数据网格获取固定客户（上一级钻取时确定的，本级钻取无法改变）每个月购买的商品销售额和销售量。编写查询语句，调用 myGrid()函数定义数据网格，并在一个窗口显示出来。

```
    //二级钻取，取客户购买的商品销售额和销售量
    var xyear=("+month).substr(0,4);
    var xmonth=("+month).substr(4,2);
    var pmyGrid2={};
    pmyGrid2.id='myGrid2';
    pmyGrid2.parent='myGridWin2';
    pmyGrid2.staticsql="select top 100 percent c.productid,c.productname,sum(a.amount) as amount,
    SUM(a.Quantity) as qty from orderitems a "+
    "\n join orders b on a.orderid=b.orderid "+
    "\n join products c on a.productid=c.productid "+
    "\n where year(orderdate)="+xyear+" and month(orderdate)="+xmonth;
    if (cid!='') pmyGrid2.staticsql+=" \n and b.customerid='"+cid+"'";
    pmyGrid2.staticsql+="\n group by c.productid,c.productname order by amount desc";
    pmyGrid2.activesql=pmyGrid2.staticsql;
    pmyGrid2.gridfields='[60]商品编码/productid;[180]商品名称/productname;[%n75,2]销售额
    /amount;[%n75]销售额/qty';
    pmyGrid2.fixedfields='';
    pmyGrid2.title='';
    pmyGrid2.checkbox=''; //'single';
    pmyGrid2.pagesize=20;
    pmyGrid2.keyfield='productid';
    pmyGrid2.sortfield='-amount';
    pmyGrid2.rownumbers=true;
    pmyGrid2.collapsible=false;
    pmyGrid2.height=524;
    pmyGrid2.width=435;
    pmyGrid2.top=32;
    pmyGrid2.left=3;
    pmyGrid2.rowindex=0;
```

```
$("#myGridWin2").empty();
myGrid(pmyGrid2);
myLoadGridData(pmyGrid2,1);
if (cid!=''){
    var sql="select customerid,customername from customers where customerid='"+cid+"'";
    var customer=myRunSelectSql(sysdatabasestring, sql);
    myLabel('xcustomer2','myGridWin2','客户名称：'+cusomer[0].customerid+customer[0].customername,8,8);
}else{
    myLabel('xcustomer2','myGridWin2','客户名称：全部客户',8,8);
}
$("#myGridWin2").window("setTitle",'钻取'+xyear+'年'+xmonth+'月份商品销售情况');
$("#myGridWin2").window("open");
```

⑥圆饼图钻取生成数据网格和仪表盘（flash 文件名为 AngularGauge.swf）。数据网格显示钻取客户每个月的销售额和利润额，仪表盘显示客户的利润率。

根据上级图表下钻时传递的客户编码（cid）编写查询语句，调用 myGrid()函数生成数据网格 myGrid1，并在窗口 myGridWin1 中显示。

```
//显示钻取的报表，客户每月销售额+利润额
var date1=myDateboxValue('datefrom','');
var date2=myDateboxValue('dateto','');
var sql="select sum(a.amount) as amount,SUM(a.amount-a.Quantity*c.unitprice) as profit,
100*year(orderdate)+month(orderdate) as month from orderitems a ";
sql+="\n join orders b on a.orderid=b.orderid ";
sql+="\n join products c on a.productid=c.productid";
sql+="\n where orderdate between '"+$("#datefrom").textbox("getValue")+"' and ";
sql+=" '"+$("#dateto").textbox("getValue")+"'";
if (cid!='') sql+=" and customerid='"+cid+"'";
sql+="\n group by year(orderdate),month(orderdate)";
var pmyGrid1={};
pmyGrid1.id='myGrid1';
pmyGrid1.parent='myGridWin1';
pmyGrid1.staticsql=sql;
pmyGrid1.activesql=pmyGrid1.staticsql;
pmyGrid1.gridfields='[@c#70]月份/month;[%n90,2]销售额/amount;[%n90,2]利润额/profit';
pmyGrid1.fixedfields='';
pmyGrid1.title=''; //不显示标题
pmyGrid1.checkbox=''; //'single';
pmyGrid1.pagesize=20;
pmyGrid1.keyfield='month';
pmyGrid1.sortfield='';
pmyGrid1.rownumbers=true;
pmyGrid1.collapsible=false;
pmyGrid1.height=400;
pmyGrid1.width=296;
pmyGrid1.top=32;
pmyGrid1.left=3;
pmyGrid1.rowindex=0;
$("#myGridWin1").empty();   //清空窗口中的所有子控件
var sql="select customerid,customername from customers where customerid='"+cid+"'";
var customer=myRunSelectSql(sysdatabasestring, sql);
myLabel('xcustomer1','myGridWin1','客户名称：'+customer[0].customerid+customer[0].customername,8,8);
```

```
//调用函数定义 myGrid1
myGrid(pmyGrid1);    //定义数据网格
myLoadGridData(pmyGrid1,1);    //提取 myGrid1 数据
$("#myGridWin1").window("open");  //打开窗口
```

根据上级图表下钻时传递的客户编码（cid）编写查询语句计算当前客户的销售额和利润额的汇总值保存到 source4，然后计算出利润率保存到 pmyChart4.value 变量中。由于仪表盘的 XML 语句与其他图表不同，故将 pmyChart4.type 设置为'angular'，而不能为'single'。仪表盘的其他主要参数还包括其半径大小、取值范围和当前值。仪表盘生成后在一个窗口中显示。

```
//计算客户利润率，画仪表盘
var sql="select sum(a.amount) as amount,SUM(a.amount-a.Quantity*c.unitprice) as profit from orderitems a ";
sql+="\n join orders b on a.orderid=b.orderid ";
sql+="\n join products c on a.productid=c.productid";
sql+="\n where orderdate between '"+$("#datefrom").textbox("getValue")+"' and ";
sql+=" '"+$("#dateto").textbox("getValue")+"'";
sql+=" and customerid='"+cid+"'";
var source4=myRunSelectSql(sysdatabasestring, sql);
//生成仪表盘
var pmyChart4={};
pmyChart4.title=date1+"～"+date2+"销售额变化图";
    if (cid!=''){
        sql="select customerid,customername from customers where customerid='"+cid+"'";
        var customer=myRunSelectSql(sysdatabasestring, sql);
        pmyChart4.title+="\n"+customer[0].customerid+customer[0].customername;
    }else{
        pmyChart4.title+="\n 全部客户";
    }
    pmyChart4.type='angular';
    pmyChart4.height=255;
    pmyChart4.width=270;
    pmyChart4.max=60;    //仪表盘显示的最大值
    pmyChart4.min=0;     //仪表盘显示的最小值
    pmyChart4.model=1;
    pmyChart4.radius=110;    //仪表盘的半径大小
    pmyChart4.value=(100.00*source4[0].profit/source4[0].amount).toFixed(3);  //利润率
    pmyChart4.label=pmyChart4.value+'%';    //仪表盘中心点显示的数值文字
    pmyChart4=myGetChartXML(pmyChart4);  //生成仪表盘的 xml 语句
    myShowFusionChart(pmyChart4,'AngularGauge','myChartWin1');  //显示仪表盘
    $("#myChartWin1").window("open");    //打开窗口显示仪表盘
    $("#myGridWin2").window("close");    //关闭其他窗口
```

实例 40. 带多层表头和汇总行的数据网格及其 Excel 文件输出。

在 EasyUI 中，数据网格多层叠加表头的定义方式是依次分层平列定义各个列集，而不是嵌套定义。例如定义表 5-1 所示的多行叠加列表头，那么第一层为[A, B, C, D]，第二层为[B1, B2, C1, C2]，第三层为[B11, B12, D1, D2]。

表 5-1. 多行叠加数据网格列表头

<table>
<tr><td rowspan="3">A</td><td colspan="3">B</td><td colspan="2">C</td><td colspan="2" rowspan="2">D</td></tr>
<tr><td colspan="2">B1</td><td rowspan="2">B2</td><td rowspan="2">C1</td><td rowspan="2">C2</td></tr>
<tr><td>B11</td><td>B12</td><td>D1</td><td>D2</td></tr>
</table>

除此之外，EasyUI 在定义每个列单元格时还需要使用 rowspan 和 colspan 两个参数，分别用来指定单元格所包含和合并的行数和列数。表 5-1 对应的多行叠加列表头的 EasyUI 语句描述如下：

```
var cols=[
[
        { title: 'A', rowspan:3 },
        { title: 'B', colspan:3 },
        { title: 'C', colspan:2},
        { title: 'D', rowspan:2, colspan:2}
],
[
        { title: 'B1', colspan:2 },
        { title: 'B2', rowspan:2, colspan:1 },
        { title: 'C1', rowspan:2, colspan:1},
        { title: 'C2', rowspan:2, colspan:1}
],
[
        { title: 'B11, colspan:1},
        { title: 'B12', colspan:1 },
        { title: 'D1', colspan:1},
        { title: 'D2', colspan:1}
] ];
```

本例创建一个二层叠加表头的数据网格，显示商品销售情况表。这里，“商品编码”为固定列，本月销售情况和本年销售情况为二层标题。本例程序运行界面如图 5-4 所示，其表头列的 EasyUI 语句描述如下：

商品销售情况月报表（2012年12月份）

输出 刷新

		商品编码	商品名称	规格型号	计量单位	单价	本月销售情况		本年销售情况	
							销售量	销售额	销售量	销售额
1	✔	101	汇源100%橙汁	1L	瓶	9.95	3395	42232.81	52875	679957.34
2		102	汇源桃汁果肉果汁	1L	瓶	7.90	7055	70143.34	48955	490976.21
3		103	味全每日C系列100%橙汁	300ml	瓶	6.80	4090	35952.19	49200	434030.20
4		104	味全每日C系列100%胡萝卜果蔬混合汁	900ml	瓶	19.00	2405	58750.85	28105	687914.55
5		105	农夫果园30%混合果蔬饮料（胡橙）	500ml*15瓶	瓶	45.00	1740	101866.70	17145	1002765.56
6		106	韩国乐天芒果汁	180ml*15罐	罐	45.00	1220	72479.12	18085	1046265.74
7		107	韩国乐天葡萄果肉果汁	238ml*12灌	灌	44.50	1635	94988.01	19485	1120136.67
8		108	西班牙富禄特桃子味天然果汁饮料	200ml	瓶	14.60	3190	60059.29	27405	515823.22
9		109	香港鸿福堂杨枝甘露原味	450ml	瓶	26.00	2985	100401.01	29625	1001516.04
10		201	红牛维生素功能饮料	250g	瓶	5.20	4325	29691.25	42645	288935.15
			* 全部商品合计 *				152320	5449647.32	1680740	60558571.96

第 1 页 共 8 页　当前显示1～10行，共74行

图 5-4 带多层叠加表头和汇总行的数据网格

```
var xcolumns=[
[   //第一层开始
        { title: '商品名称', field:'productname', width: 200, halign:'center', align: 'left', rowspan:2 },
        { title: '规格型号', field:'quantityperunit', width: 130, halign:'center', align: 'left', rowspan:2 },
        { title: '计量单位', field:'unit', width: 80, halign:'center', align: 'center', rowspan:2},
        { title: '单价', field: 'unitprice', width: 70, halign:'center', align: 'right', rowspan:2, decimal:2,
          formatter: fnSetNumberFormat},
        { title: '本月销售情况',"colspan":2},
        { title: '本年销售情况',"colspan":2}
],   //第一层结束
[   //第二层开始
```

```
        { title: '销售量', field:'qty1', width: 70, halign:'center', align: 'right', decimal:0, formatter:
fnSetNumberFormat    },
        { title: '销售额', field:'amount1', width: 80, halign:'center', align: 'right', decimal:2, formatter:
fnSetNumberFormat },
        { title: '销售量', field:'qty2', width: 70, halign:'center', align: 'right', decimal:0, formatter:
fnSetNumberFormat },
        { title: '销售额', field:'amount2', width: 80, halign:'center', align: 'right', decimal:2, formatter:
fnSetNumberFormat }
] ];
var xfixedcolumns=[[
        { title: 'id', field: 'id', width: 20, checkbox: true, sortable: true, align: 'center'},
        { title: '商品编码', field:'productid', width: 70, halign:'center', align: 'left' }
] ];
```

许多插件可以在数据网格的尾部添加固定的汇总行信息，EasyUI 实现汇总行的方式比较简单，可以通过网格对应的 JSON 数据中添加一个 footer（页脚、注释）属性来实现。例如，下列 JSON 数据中，footer 属性在 rows 属性之后指定一个“全部商品合计”的汇总行，这个汇总行仅包含 5 个列的数据，数据网格在显示这行数据时自动将其他列设置为空。


```
data：{"total":"74",
"rows":[{"rownumber":"1","productid":"101","productname":"汇源 100%橙汁","quantityperunit":"1L","unit":"瓶",
"unitprice":"9.95","qty1":"3395","amount1":"42232.81","qty2":"52875","amount2":"679957.34","sortflag":"0"}
,{"rownumber":"2","productid":"102","productname":"汇源桃汁果肉果汁","quantityperunit":"1L","unit":"瓶
","unitprice":"7.90","qty1":"7055","amount1":"70143.34","qty2":"48955","amount2":"490976.21","sortflag":"0"}
……
,{"rownumber":"10","productid":"201","productname":"红牛维生素功能饮料","quantityperunit":"250g","unit":"瓶
","unitprice":"5.20","qty1":"4325","amount1":"29691.25","qty2":"42645","amount2":"288935.15","sortflag":"0"}
],
"footer":[{"productname":"<center>*  全部商品合计  *</center>", "qty1":"152320", "amount1":"5449647.32",
"qty2":"1680740", "amount2":"60558571.96"}]
}
```


除此之外，footer 还可以显示除汇总行之外的其他信息，而且还可以不止一行，但必须在网格定义时将 showFooter 属性设置为 true，否则 footer 不会显示。

```
$("#myGrid1").datagrid({
        title: ' 商品销售情况月报表（2012 年 12 月份）',
        width:'100%',
        height:'100%',
        ……
        columns: xcolumns,
        showFooter: true
});
```

数据网格只是数据输出的一种形式，很多情况下需要将数据库中数据以报表形式打印出来。在 Web 程序设计中，直接报表打印是比较复杂的，常用的做法是将数据库中数据导出到文件中（例如 Office 文件和 PDF 文件等）再去打印。本例介绍如何将商品销售统计表中的数据按照指定的模板格式导出到一个 Excel 文件中，这个功能实现包括一个前台和一个后台程序，前台程序设计过程和方法如下：

	A	B	C	D	E	F	G	H	I
1	商品****报表								
2	20**年**月								
3	商品编码	商品名称	规格型号	计量单位	单价	本月销售情况		本年累计销售情况	
4						销售量	销售额	销售量	销售额
5					1.00	1	1.00	1	1.00

图 5-5 商品销售月报表 Excel 模板

①创建一个 Excel 模板文件 TProductSales.xls（如图 5-5 所示，文件所在路径为\system\templates\）。该模板将报表分为表头、标题和表尾三个部分，第 1~4 行为报表表头（标题），在每页打印时都会出现；第 5 行为表体中的第一行，之后将按照这行格式重复加载数据库中记录，打印满页时 Excel 自动分页；表尾部分为页脚，在后面程序中将设置其输出内容和格式。

②在客户端调用服务器端序 example40_toExcel.jsp 生成 Excel 文件，前台客户端传递的变量包括：数据查询 SQL 语句、模板名称等。服务器端程序将数据查询结果转换成 Excel 文件并保存在一个临时文件中返回给客户端，之后客户端再调用服务器端程序 Easyui_fileDownLoad.jsp，下载和打开生成的 Excel 文件。

```
var pmyReport1={};
pmyReport1.sql=pmyGrid1.staticsql+" union all ";
pmyReport1.sql+="\n select ",'*  全部商品合计  *','','', 0, sum(qty1), sum(amount1), sum(qty2),
sum(amount2), 1 from (";
pmyReport1.sql+="\n "+pmyGrid1.staticsql+") as p";
pmyReport1.sql+="\n order by sortflag,productid";
pmyReport1.targetfilename="商品销售情况月报表.xls";
pmyReport1.template="TProductSales.xls";
var filename="";
$.ajax({
    url:"example40_toExcelFile.jsp",   //调用后台程序生成 excel 文件
    data:{ database:sysdatabasestring,
        template:pmyReport1.template,
        selectsql:pmyReport1.sql
    },
    async:false,
    success:function(data){
        eval("var result="+data);
        filename=result[0].filename;
    }
});
var xsourcefilename=filename;
//调用后台程序，下载 excel 文件。
var url='system//Easyui_fileDownLoad.jsp?source='+filename+'&target='+pmyReport1.targetfilename;
window.location.href=url;
```

服务器端程序实现过程和方法如下：

①在 jQDemos 项目的\WEB-INF\lib\文件夹下引入下列三个 POI 文件，并在在服务器端程序 example40_toExcelFile.jsp 中进行引用。

poi-3.2-FINAL-20081019	2015/8/22 17:53	Executable Jar File	1,396 KB
poi-contrib-3.2-FINAL-20081019	2015/8/22 17:53	Executable Jar File	62 KB
poi-scratchpad-3.2-FINAL-20081019	2015/8/22 17:53	Executable Jar File	772 KB

```
<%@ page import="com.StringUtil,java.net.URLEncoder,
    org.apache.poi.hssf.usermodel.HSSFWorkbook,
    org.apache.poi.hssf.usermodel.HSSFSheet,
    org.apache.poi.hssf.usermodel.HSSFRow,
    org.apache.poi.hssf.usermodel.HSSFCell,
    org.apache.poi.hssf.util.Region,
    org.apache.poi.hssf.util.*,
    org.apache.poi.hssf.usermodel.HSSFFont,
    org.apache.poi.hssf.usermodel.HSSFPrintSetup,
    org.apache.poi.hssf.usermodel.HSSFRichTextString,
```

```
        org.apache.poi.hssf.usermodel.*,
        org.apache.poi.hssf.*,
        org.apache.poi.poifs.filesystem.POIFSFileSystem,
        java.io.* "
%>
```

②连接数据库，执行查询语句，将查询结果保存到 grid_rs 中。

```
request.setCharacterEncoding("ISO-8859-1");
request.setCharacterEncoding("utf8");
String database=StringUtil.getUrlCHN(request.getParameter("database"));
String xtemplate=StringUtil.getUrlCHN(request.getParameter("template"));
String xselectsql=StringUtil.getUrlCHN(request.getParameter("selectsql"));
xselectsql=xselectsql.trim();
//数据库连接
DBConn con=new DBConn();
Connection connection=con.getConnection(database);
Statement stmt=connection.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE,ResultSet.
CONCUR_READ_ONLY);
//执行查询语句
stmt.executeQuery(xselectsql);
ResultSet grid_rs=stmt.getResultSet();
```

③打开 Excel 模板文件，定义一个 Excel 工作表（workbook）变量 wb，选取第一个 sheet，并通过 HSSFRow 和 HSSFCell 定义行变量 row 和单元格变量 cell。将 Excel 表中第 1~4 行设置标题行，（即在每页打印时都出现）。将模板中的第 5 行作为格式参照行保存到变量 formatrow 中，表体中之后新增行和单元格的格式（包括行高、单元格的字体即大小、数据类型、小数位数即对齐方式等）都从这行中复制。

```
HSSFRow firstrow = sheet.getRow(4);
//取excel模板文件的路径和文件
String root = application.getRealPath("/");
String templatefile=root+"system/templates/"+xtemplate;
//定义poi
POIFSFileSystem fs=new POIFSFileSystem(new FileInputStream(templatefile));
HSSFWorkbook wb = new HSSFWorkbook(fs); //创建工作表
HSSFSheet sheet = wb.getSheetAt(0);   //选取第一个sheet
//处理表头，每页重复打印第1行第1列~第4行第9列这个区域
wb.setRepeatingRowsAndColumns(0,0,8,0,3);//(0,x1,x2,y1,y2)
HSSFRow row = null;
HSSFCell cell = null;
//记录模板表体中的第 5 行，其他行的单元格格式以第 5 为标准
HSSFRow formatrow = sheet.getRow(4);
```

④利用循环提取查询结果集 grid_rs 中的行记录和列数据，每条记录生成 Excel 的一行，每个数据表列对应 Excel 的一个单元格。标题中的第一行（即 Excel 中的第 5 行、格式行）可以直接引用，只需要设置单元格的值，而其他行都必须先创建再设置值和格式。

```
grid_rs.beforeFirst();
i=1;
while(grid_rs.next()) { //遍历记录进行数据组合
    if (i==1){
        row = sheet.getRow(4); //表体中第 1 行（即模板中第 5 行）不用创建，因为原来就存在
    }else{
        row = sheet.createRow(i+3);   //表体中第 2 行（模板中的第 6 行）之后的行需要新创建
    }
    for (j=1;j<=fielddim.length;j++) {   //循环处理一行中的每个列
```

```
            if (i==1){
                cell = row.getCell(j-1);
            }else{
                cell = row.createCell(j-1);
            }
            value=grid_rs.getString(fielddim[j-1]).trim();   //提取取数据表中某个列的值
            cell.setCellValue(new HSSFRichTextString(value));   //将数据表某列值填充到单元格
            //将模板第 5 行 formatrow 的 cellstyle 复制过来存放在新的行中
            cell.setCellStyle(formatrow.getCell(j-1).getCellStyle());
            cell.setCellType(formatrow.getCell(j-1).getCellType());
        }
        row.setHeight(formatrow.getHeight());   //按模板设置行高
        i++;
    }
```

⑤设置模板中的标题表头和表尾（页脚）内容。将模板中的第一行报表标题修设置为“商品销售情况月报表”，由于第一行已合并为一个单元格，因此使用 getCell(0)。同样方法将第二行第一列设置为“2012 年 12 月 31 日”。报表页脚只能设置左中右三个值，其中当前页和总页数分别使用 HSSFFooter.page()和 HSSFFooter.numPages()系统变量。

```
row = sheet.getRow(0);
cell = row.getCell(0);
cell.setCellValue("商品销售月报表");   //修改表头第 1 行
row = sheet.getRow(1);
cell = row.getCell(0);
cell.setCellValue("2012 年 12 月 31 日");   //修改表头第 2 行
//写页脚 footer 的内容
HSSFFooter footer=sheet.getFooter();
footer.setLeft("制表人：诸葛亮");    //在页脚左边设置文字
//在页脚中间显示页码，即页码居中
footer.setCenter("第"+HSSFFooter.page()+"页/共"+HSSFFooter.numPages()+"页");
```

⑥模板修改结束，将内容复制到一个临时 Excel 文件中。临时文件的路径为 system\\temp\\，文件名为模板名，最后将临时文件返回给客户端。

```
String filename=root+"\\system\\temp\\"+xtemplate;
FileOutputStream fo=new FileOutputStream(filename);
wb.write(fo);
fo.close();
filename="\\\\system\\\\temp\\\\"+xtemplate;   //需要4根斜杠。去掉root，在下载时自动会加上
response.getWriter().write("[{\"totalcount\":\""+(i-1)+"\",\"filename\":\""+filename+"\"}]");
grid_rs.close();
stmt.close();
connection.close();
```

将数据库中数据填充倒模本文件，从而导出到 Excel 文件，可通过调用通用函数 myExportExcelReport() 直接实现，该函数调用一个比较通用的 Excel 导出程序 Easyui_toExcelFile.jsp。除 SQL 查询语句和输出列名之外，该函数还要求提供表头、表体中需要修改的行列单元位置及修改内容。例如，本例导出 Excel 文件的过程可用下列语句实现。

```
pmyReport1.headerrange='<1-4>';   //标题为第 1 行~第 4 行，每页重复
pmyReport1.headercells="<1,1>商品销售情况月报表;<2,1>2012 年 12 月份";
pmyReport1.footercells="<l>制单人：诸葛亮;<c>第 pageno 页/共 pagecount 页";
pmyReport1.fields="productid;productname;quantityperunit;unit;unitprice;qty1;amount1;qty2;amount2";
var r=myExportExcelReport(sysdatabasestring,pmyReport1);
```

附录 1、jQDemos 自定义控件函数

1．function myForm(id,parent,title,top,left,height,width,style)

功能：定义和生成一个表单（tree）。

参数：表单标识符、父类容器标识符（缺省时为 main）、表单标题、起始坐标（y,x）位置，表单的高度与宽度、样式。Style 选项：close; collapse;min;max 等。当高度与宽度为 0 时，自动取父类容器的高度与宽度。

举例：myForm('myForm1','main','学生信息编辑',0,0,490,695,'close;max');

2．function myLabel(id,parent,label,top,left)或
function myLabelField(id,parent,label,top,left,height,width)

功能：定义和生成一个 HTML 的 label 文字标签。

参数：控件标识符、父类容器标识符（缺省时为 main）、文字内容、起始坐标（y,x）位置，文字的高度和宽度是可选项，一般不设置即显示文字全部内容。

举例：myLabel('customerx','main','客户名称',10,20);

3．function myTextField(id,parent,label,labelwidth,top,left,height,width,value,style)

功能：定义和生成一个文本框控件（textbox），输入文本内容。

参数：控件标识符、父类容器标识符（缺省时为 main）、输入框之前的提示性文字、起始坐标（y,x）位置，文本框的高度和宽度、初始值和样式。一般高度是可选的，默认为系统指定的高度。style 选项值为 readonly 时表示文本框只读，选项值为 email 和 url 时分别验证文本框中输入的内容是否符合邮箱地址和网址的格式要求，选项值为 password 时文本框输入内容为不可见的星号。

举例：myTextField('stuname','myFieldset1','姓名：',70,33*1+20,12,0,160,'贾宝玉', ' ');

myTextField('email','myFieldset2','Email：',70,33*3+20,12,0,180,'zxywolf@163.com', 'e

4．function myDateField(id,parent,label,labelwidth,top,left,height,width,value,style)

功能：定义和生成一个日期控件（datebox），按中文本地化格式输入或选择一个日期。

参数：控件标识符、父类容器标识符（缺省时为 main）、输入框之前的提示性文字、起始坐标（y,x）位置，日期框的高度和宽度、初始值和样式。一般高度是可选的，默认为系统指定的高度。style 选项值为 readonly。

举例：myDateField('birthdate','myFieldset1','出生日期：',70,33*3+20,12,0,120,'1997-2-17', ' ');

5. function myMemoField(id,parent,label,labelwidth,top,left,height,width,value,style)
function myTextareaField(id,parent,label,labelwidth,top,left,height,width,value,style)

功能：定义和生成一个多行文本框控件，即 HTML 的 textarea 控件。

参数：控件标识符、父类容器标识符（缺省时为 main）、输入框之前的提示性文字、起始坐标（y,x）位置，多行文本框高度和宽度、初始值和样式。一般高度是可选的，默认为系统指定的高度。style 选项值为 readonly。当 labelwidth 值为 0 时，则提示性文字与文本框分上下两行显示。

举例：myMemoField('notes','myFieldset3','个人简介：',0,33*2+5,10,100,273,' ',' ');

6. function myNumberField(id,parent,label,labelwidth,top,left,height,width,length,decimal,value,min,max,style)

功能：定义和生成一个数值框控件（numberbox），只能输入只能输入数字和小数点。

参数：控件标识符、父类容器标识符（缺省时为 main）、输入框之前的提示性文字、起始坐标（y,x）位置，数值框的高度和宽度、数值总位数、小数位数、初始值、最小值、最大值和样式。style 选项值为 readonly 时表示文本框只读。当用户输入非数值型数据（实际上只有汉字可以输入）时，返回空格；当小数位数不足时会自动补加 0。

举例：myNumberField('unitprice','myFieldset1','单价：',70,120,10,0,120,8,2,'7.25',0,500,' ');

7. function myCheckBoxField(id,parent,label,labelwidth,top,left,rowheight,cols,items)

功能：定义和生成一个可以有多个复选框的组合控件（多个 HTML 的 checkbox）。

参数：控件标识符、父类容器标识符（缺省时为 main）、输入框之前的提示性文字、起始坐标（y,x）位置，复选框换行时的行间距、每列显示复选框的个数、复选框选项内容（多个选项之间用分号分隔）。选项内容 items 中可以包含方括号（[ ]），方括号内可以有 u、c 和数字。数字表示某个选项的显示宽度，[u]表示初始状态不选中，[c]表示选中，默认为选中状态。

myCheckBoxField('research','myFieldset3','个人特长：',70,33*0+20,12,24,3,'围棋;国际象棋;足球;长跑;数学;信息技术');

举例：myCheckBoxField('research','myFieldset3','个人特长：',70,20,12,24,3,'围棋;国际象

myCheckBoxField('research','myTab3','研究方向：',0,120,12,24,3,'[u120]企业管理;[120]区域经济学;[120]管理信息系统;[120]电子商务;[u150]信息系统开发技术');

这里，每行显示 3 个复选框选项, 企业管理和信息系统开发技术初始状态为不选中。

当用户点击复选框，改变其选中状态时，所有选中选项的文本内容会赋值到一个隐藏控件中，其名称为 id+'_value'。

8. function myComboField(id,parent,label,labelwidth,top,left,height,width,items,value,style)

功能：定义和生成一个定义静态组合框（下拉框，combobox）控件，其选项数据直接由程序员指定，与数据库无关。

参数：控件标识符、父类容器标识符（缺省时为 main）、输入框之前的提示性文字、起始坐标（y,x）位置，组合框的高度和宽度、选项条目（多个选项之间用分号分隔）和样式。style 选项值为 readonly 时表示组合框只读，选项值为 autodrop 表示组合框只读并直接展开下拉。

举例：myComboField('gender','myFieldset1','性别：',70,rowheight*3+25,16,0,120,'男;女','男','autodrop');

9. function myDBComboField(id,parent,label,labelwidth,top,left,height,width,sql,textfield, masterfield,style)

功能：定义和生成一个定义数据库动态组合框（下拉框，combobox）控件，其选项数据从数据库后台取数。

参数：控件标识符、父类容器标识符（缺省时为 main）、输入框之前的提示性文字、起始坐标（y,x）位置，组合框的高度和宽度、select 查询语句、组合框显示内容对应的字段名、联动时对应的主组合框标识符和样式。style 选项值为 readonly 时表示组合框只读，选项值为 autodrop 表示组合框只读并直接展开下拉。当 masterfield 为非空时，主组合框值变化时，子组合框值自动变化。

举例：myDBComboField('degree','myFieldset1','学历：',70,33*9+20,16,0,160,jqsql.degree,'degree',"");

myDBComboField('city','myFieldset1','所在城市：',70,33*8+20,16,0,160,jqsql.city,'city', 'province(provinceid)',' ');

这里，主组合框的标识符为 province，联动取值自主主组合框的 procinceid 列的值。

10．function myImageField(id,parent,label,labelwidth,top,left,height,width,src)

功能：定义和生成一个 HTML 的图形（img）控件。

参数：控件标识符、父类容器标识符（缺省时为 main）、输入框之前的提示性文字、起始坐标（y,x）位置，图形显示高度和宽度、图形对应的路径及文件。在定义图形控件后一般需要调用 myResizeImage()，实现图形等比例缩放显示。

举例：myImageField('picture','myFieldset4',",0,80,6,140,340,'mybase//products//101.jpg'); ,

11．function myResizeImage(img,src,winheight,winwidth)

功能：给定一个图形控件及其文件名，按规定大小等比例缩放图形，然后显示图形。

参数：图形控件标识符、图形路径文件名、显示区域的高度和宽度。

举例：myResizeImage('picture','mybase//products//101.jpg',180,230);

12．function myFileField(id,parent,top,left,height,width,buttonalign,event)

功能：定义和生成一个文件上传控件（HTML 控件）及其事件。同时生成一个上传文件的按钮，点击按钮将调用后台 Easyui_fileUpLoad.jsp 程序实现文件上传。“上传”按钮的名称为 id+'upload'，其点击事件中一般可以调用 myFileupLoad()函数开始文件上传。

参数：控件标识符、父类容器标识符（缺省时为 main）、输入框之前的提示性文字、起始坐标（y,x）位置，文件框的高度和宽度、“上传”按钮的位置（right 或 left 两个选项）、按钮点击触发的事件名。event 一般不能缺省，否则点击“上传”按钮不会将文件上传到服务器。

举例：myFileField('photo','myFieldset4',20,6,0,240,' ','fnupLoadPhoto');

这里，点击“上传”按钮时触发 fnupLoadPhoto 函数，因此需要在这个函数中调用 myFileupLoad()函数实现文件上传。

13．function myFileupLoad(id,targetpath,targetname)

功能：调用后台 Easyui_upFileupLoadl.jsp 程序实现文件上传到指定路径。

参数：文件控件标识符、上传文件存放的路径、上传文件保存的文件名。一般与 **myFileField** 控件配套使用，也可以单独调用。

举例：var result=myFileupLoad('photo','mybase/students/',targetname);

14．function myDBTree(id,parent,title,top,left,height,width,sql,keyfield,sortfield,style)

功能：定义和生成一个动态数据库树控件（tree），根据查询语句从数据库中提取节点，可以一次性提取全部节点，也可以分层逐级展开父节点提取子节点。

参数：控件标识符、父类容器标识符（缺省时为 main）、输入框之前的提示性文字、起始坐标（y,x）位置，文件框的高度和宽度、数据库取数的查询语句、节点关键字、排序列名和样式。style 有两种选项，当其值为 full 时，系统调用后台程序 Easyui_getAllTreeNodes.jsp，一次性从数据库提取所有节点加载到树控件中，否则调用后台程序 Easyui_getAllTreeNodes.jsp 提取当前父节点的下级子节点。style 其他选项包括 checkbox（是否显示复选框）、line（是否显示节点之间的连接线）、animate（节点展开是否需要动画效果）、edit（是否可以编辑节点）等。除此之外，sql 查询语句中必须包含 text 列，作为节点中显示的文字。

举例：var sql="select rtrim(areaid)+' '+rtrim(areaname) as 'text', areaid, areaname, parentnodeid, isparentflag, level, ancester from Areas";

myDBTree('myTree1', 'myForm1', '地区分类',0,0,0,0, sql,'areaid',' ',' full');

15．function myLoadTreeData(id,sql,keyfield,sortfield,style)

功能：加载树结点，分一次性全部加载和分层逐级加载两种模式。与 myDBTree()配套使用，也可以独立调用。

参数：控件标识符、数据库取数的查询语句、节点关键字、排序列名和样式。style 有两个选项，其含义与 **myDBTree** 相同。

举例：var sql="select rtrim(areaid)+' '+rtrim(areaname) as 'text', areaid, areaname, parentnodeid, isparentflag, level, ancester from Areas";

myLoadTreeData('myTree1',sql,'areaid',' ',' full');

16．function myDBComboTreeField(id,parent,label,labelwidth,top,left,height,width,sql, keyfield,sortfield,style)

功能：定义和生成一个动态数据库组合树控件（combotree），根据查询语句从数据库中提取节点，可以一次性提取全部节点，也可以分层逐级展开父节点提取子节点。

参数：与 **myDBTreeField** 相同。

举例：var sql="select rtrim(areaid)+' '+rtrim(areaname) as 'text', areaid, areaname, parentnodeid, isparentflag, level, ancester from Areas";

myDBComboTreeField('areaid', 'myFieldset1', '出生地：',70,33*5+20, 16,0,220, sql, 'areaid', ' ','full');

17．function myComboLoadTreeData(id,sql,keyfield,sortfield,style)

与 **myLoadTreeData** 功能基本相同。与 myDBComboTree()配套使用，也可以独立调用。

18．function myFieldset(id,parent,title,top,left,height,width)

功能：定义和生成一个 HTML 的 fieldset 控件，是一个容器，用于存放其他控件。

参数：控件标识符、父类容器标识符（缺省时为 main）、标题、起始坐标（y,x）位置，容器的高度和宽度。

举例：myFieldset('myFieldset1','myForm1','基本信息',010,10,230,290);
myFieldset('myFieldset2','myForm1','通信信息',010,320,230,350);

19．function myGroupbox(id,parent,top,left,height,width,border)

功能：定义和生成一个 HTML 的 fieldset 控件，是一个容器，用于存放其他控件。与 myFieldset 不同的是这个控件没有标题，也可以没有边框。

参数：控件标识符、父类容器标识符（缺省时为 main）、标题、起始坐标（y,x）位置，容器的高度和宽度、边框的大小。

举例：myGroupbox('myFieldset1','myForm1',10,10,230,290,0);

20．function myButton(id,parent,text,top,left,height,width,icon,style,event)

功能：定义和生成一个链接按钮（linkbutton）。

参数：控件标识符、父类容器标识符（缺省时为 main）、按钮文字、起始坐标（y,x）位置，按钮的高度和宽度、按钮对应的小图标、样式和点击按钮触发的事件函数名称。style 选项值为 plain 时，按钮将显偏平形态。

举例：myButton('cmdok','toolbar','查找',4,450,25,68,'searchIcon',' ','fnOkClick ');

21．function myTabs(id,parent,items,top,left,height,width,style)

功能：定义和生成一个标签页/选项卡控件（tabs），分页显示表单内容。

参数：控件标识符、父类容器标识符（缺省时为 main）、标签页标题（多个之间用分号分隔）、起始坐标（y,x）位置，标签页的高度和宽度和样式。style 选项值为 close 时，每个选项卡标题上有一个关闭的小图标。当高度和宽度值为 0 时，自动与父类容器的大小相适应。

举例：myTabs('myTab', 'main', '基本信息;通信信息;个人简历;上传照片', 0, 0, 285, 385, ' ');

22．**function myTabForm(id,parent,title,tabs,top,left,height,width,style)**

功能：同时定义和生成一个表单控件和一个标签页/选项卡控件（tabs），标签页嵌入在表单中，其大小与表单相适应。

参数：控件标识符、父类容器标识符（缺省时为 main）、表单的标题、标签页标题（多个之间用分号分隔）、起始坐标（y,x）位置，标签页的高度和宽度和样式。参数含义与 **myTabs()**函数相同。

举例：myTabForm('myTab', 'main', '学生信息','基本信息;通信信息;个人简历', 0, 0, 285, 385, ' ');

23．function myMenu(id,items,bindobj)

功能：定义和生成一个菜单控件（menu），并可与其他表单、树、数据网格等绑定，作为右键菜单使用。

参数：菜单标识符、菜单条目（多个之间用分号分隔）、绑定的对象标识符。Bindobj 为空时不绑定对象。菜单条目包括三个参数用斜杠分隔：菜单条目名称、标识符和小图标（可缺省）。

举例： myMenu('myMenu1','新增结点/mnaddnode/addIcon;新增子结点/mnaddsubnode/addIcon;修改结点/mneditnode/editIcon;-;删除结点/mndeletenode/deleteIcon','myTree1');

24．function myWindow(id,title,top,left,height,width,buttons,style)

功能：定义和生成一个窗口控件（window），可在屏幕弹出一个子窗口，可分为 modal 和非 modal 模式。当 buttons 参数非空时，窗口下方可以出现保存、确定或取消等按钮。

参数：窗口标识符、窗口标题、起始坐标（y,x）位置，窗口的高度和宽度、显示的按钮种类

和样式。按钮选项有 ok、cancel 和 save 三种及其组合，style 选项值包括 resize、drag、close、collapse、max、min、modal，分别表示窗口是否可以改变大小、拖动、关闭、收缩、最大化、最小化和多模显示。

举例：myWindow('myWin1','编辑商品信息',0,0,300,660,'save;cancel','modal;close;drag');

25．function myGrid(pmyGrid1)

功能：定义和生成一个动态数据网格控件（datagrid）及其相关事件。一旦定义后，数据网格可自动分页从数据库中提取数据。

参数：参数较多，由一个对象变量（pmyGrid①统一管理，主要包括：id（网格的标识符）、parent（父类容器标识符）、staticsql（查询语句）、activesql（动态变化的查询语句）、gridfields（网格各个列描述字符串）、fixedfields（固定列描述字符串）、title（标题）、menu（绑定的右键菜单）、checkbox（是否带行复选框）、pagesize（分页时每页显示行数）、keyfield（查询语句的主键）、rownumbers（是否显示行序号）、collapsible（是否可收缩）、height（网格高度）、width（网格宽度）等。其中列描述字符串包括：列格式、列标题和字段名三部分，格式用方括号包含、列标题与字段名之间用斜杠（/）分隔。列格式又分为列对齐方式、数据类型、宽度和小数位数四部分，其中[@r]、[@c]、[@l]分别表示列右对齐、居中和左对齐（默认为左对齐），[%c]、[%d]、[%f]、[%f]分别表示列数据类型为字符型、日期型、浮点型和数值型（默认为字符型）；[#w]表示列像素宽度为 w（默认为 70 像素）。一个列只有一个格式方括号，多种格式合并在一个方括号内，例如'[@l%c#90]姓名/name。

举例：var pmyGrid1={};

```
pmyGrid1.id='myGrid1';
pmyGrid1.parent='main';
pmyGrid1.staticsql="select StudentID,Name,pycode,gender,birthdate,province+' '+city as hometown,";
pmyGrid1.staticsql+="mobile,homephone,email,weixin,qq from students";
pmyGrid1.activesql=pmyGrid1.staticsql;
pmyGrid1.gridfields='[@l%c#90]姓名/name;[@c%c#110,2]拼音/pycode;[%d#90@c]出生日期
/birthdate;[@c#100]所属城市/hometown; '[110]联系电话/mobile;[110]家庭电话/homephone;
[140]Email/email;[130]微信号/weixin;[100]QQ 号/qq';
pmyGrid1.fixedfields='[%c#90]学号/studentid';
pmyGrid1.title='学生列表';
pmyGrid1.menu='myMenu1';
pmyGrid1.checkbox='single';
pmyGrid1.pagesize=20;
pmyGrid1.keyfield='studentid';
pmyGrid1.rownumbers=true;
pmyGrid1.collapsible=false;
pmyGrid1.height=600;
pmyGrid1.width=780;
myGrid(pmyGrid1);
```

26．function myGridColumns(fieldset)

功能：定义数据网格（datagrid）列集，返回符合数据网格的各个列标题。一般与 myGrid()配套使用，也可以单独使用。

参数：与 myGrid()函数中的 gridfields 相同。

27．myLoadGridData(pmyGrid1,pageNumber)

功能：从数据库中提取某一页的数据显示在数据网格上，通常与 myGrid()配套使用，也可以单独使用。

参数：pmyGrid1 对象变量，与 myGrid()函数相同。

28．function myGridPaging(pmyGrid1)

功能：定义数据网格的分页模式，通常与 myGrid()配套使用，也可以单独使用。

参数：主要为 pmyGrid1 对象变量中的 pagesize 等，其他与 myGrid()函数相同。

29．function mySetFormValuesbyJSON(fieldset, source)

功能：将符合 JSON 格式的数据提取出来赋值到表单中 fieldset 指定的各个控件中。

参数：控件字段集、JSON 数据源。字段集为空时将自动在表单中查找控件赋值。

举例：var source= {"teacherid":"FJP12M","name":"费剑平","pycode":"bijianping","gender":"男","birthdate":"1966-06-01","title":"教授"};

mySetFormValuesbyJSON("teacherid;name;gender;birthdate;pycode;title;birthdate", source)

31．function mySetFormReadonly(flag)

功能：设置表单中所有控件只读或取消只读状态。

参数：标志值（rue 或 false）。true 时表单中所有控件只读不能编辑，false 时控件可以编辑。

32．function mySetReadonly(fieldset,flag)

功能：将 fieldset 指定的控件设置为只读或可编辑状态

参数：控件字段集、标志值（rue 或 false）。多个控件之间用分号分隔，flag 值为 true 时表单中所有控件只读不能编辑，false 时控件可以编辑。fielset 为空时，对所有表单控件设置只读状态。

33．function myreSetForm()

功能：重置表单，将表单中所有值恢复到控件定义时的初始值

34．function myClearForm()

功能：清空表单，将表单中所有值表尾空值

35．function myGetInputValue(id)

功能：根据控件的标识符值，返回其输入的内容值

参数：控件标识符。

36. function myGetSelectedComboIndex(field)

功能：获取多选组合框（下拉框中）选中条目的下标值，存放到一个数组变量中返回。

参数：组合框的标识符。

举例：myGetSelectedComboIndex('research');

假设 research 组合框有 5 个选项，而用户选中的只有第 1、2、4 个，则返回的值为[1,2,4]。

37．function myKeyDownEvent(fieldset)

功能：将一个或多个可编辑控件绑定 keydown 事件，控制页面键盘回车和移动操作。

参数：控件标识符集。多个控件之间用分号分割。当 c 为空时，则将表单中的所有可编辑控件与 keydown 事件绑定。

37．function myBindKeyDownEvent(id)

功能：将某个控件绑定 keydown 事件，控制页面键盘回车和移动操作。

参数：控件标识符。

38．function myDisableCmp(fieldset,flag)

功能：将一个或多个控件设置为激活或非激活状态。

参数：控件字段集、标志值（true 或 false）。flag 为 true 时取消激活状态，false 时激活状态。

39．function myFocus(id)

功能：在表单编辑过程中将光标聚焦到某一个控件上。

参数：控件标识符。

40．function mySelectOnFocus()

功能：表单中可编辑控件在聚焦时全选中其文本框内容（selectAll）。

41．function myGetChartXML(pmyChart)

功能：生成 FusionCharts 图表的 XML 格式语句。

参数：由一个对象变量（pmyTree）统一管理，主要包括：title（图表标题）、xAxisName（X 轴坐标名）、yAxisName （Y 轴坐标名）、labelfield（XML 文件中 label 标签对应的列名）、valuefield（XML 文件中 value 标签对应的列名）、data（图表数据源）、type（单序列、多序列、仪表盘、圆锥图等）、sliced（展开状态）、height（高度）、width（宽度）、drilldown（向下钻取时调用的 js 函数名称及参数）等

举例：var pmyChart1={};

```
pmyChart1.title="销售情况圆柱图";
pmyChart1.xAxisName="月份";
pmyChart1.yAxisName='销售额';
pmyChart1.labelfield='month';
pmyChart1.valuefield='amount';
pmyChart1.data=source1;
pmyChart1.type='single';
pmyChart1.sliced=true;
pmyChart1.height=300;
pmyChart1.width=490;
pmyChart1.drilldown='fnDrillDownColumnLineChart(month)';
pmyChart1=myGetChartXML(pmyChart1);
```

40．function myGetChartXMLHeader(pmyChart)

功能：返回 FusionCharts 中折线图、圆柱图、栏位图、圆饼图等图表的 XML 文件头部参数语句，一般被 **myGetChartXML()**调用而不单独使用。

参数：与 **myGetChartXML()**相同。

41．function myGetChartXMLFooter(pmyChart)

功能：返回 FusionCharts 中折线图、圆柱图、栏位图、圆饼图等图表的 XML 文件尾部参数语句，一般被 **myGetChartXML()**调用而不单独使用。

参数：与 **myGetChartXML()**相同。

43．function myShowFusionChart(pmyChart,swf,div)

功能：将 FusionCharts 的图表显示出来。

参数：pmyChart 与 **myGetChartXML()**相同、需要展示的 flash 图表源文件、图表渲染的区域（通常是一个层）。

举例：myShowFusionChart(pmyChart1,'Pie3D','piediv1');　　//显示一个圆饼图

44．function myExportExcelReport(sysdatabasestring,pmyReport1)

功能：从数据库中提取数据按照模板格式以 Excel 文件输出下载。

参数：由一个对象变量（pmyReport）统一管理，主要包括：sql（查询语句）、template（Excel 莫办文件）、targetfilename（下载的 Excel 文件名）、headerrange（每页重复打印的报表表头）、headercells（报表标题中需要修改的单元格及其值）、footercells（报表页脚中需要修改的单元格及其值）、fields（标题打印的字段集，缺省时输出 select 语句中的全部列）。headcells 格式为<row_1, col_1>文字 1; <row_2, col_2>文字 1…<row_n, col_n>文字 n。footercells 格式为：<l>页脚左侧文字; <c>页脚中间文字; <r>页脚右侧。

举例：var pmyReport1={};

```
pmyReport1.sql=pmyGrid1.staticsql+" union all ";
pmyReport1.sql+="\n select ","* 全部商品合计 *","  ","  ",0,sum(qty1),sum(amount1),sum(qty2),
sum(amount2),1 from ("+pmyGrid1.staticsql+") as p";
```

```
pmyReport1.sql+="\n order by sortflag,productid";
pmyReport1.targetfilename="商品销售情况月报表.xls";
pmyReport1.template="TProductSales.xls";
pmyReport1.headerrange='<1-4>'; //标题为第 1 行到第 4 行，每页重复
pmyReport1.headercells="<1,1>商品销售情况月报表;<2,1>2012 年 12 月份";
pmyReport1.footercells="<l>制单人：诸葛亮;<c>第 pageno 页/共 pagecount 页";
pmyReport1.fields="productid;productname;quantityperunit;unit;unitprice;qty1;amount1;qty2;amount2"
;
var r=myExportExcelReport(sysdatabasestring,pmyReport1);
```

45．function myRunSelectSql(databasestring, sql)

功能：给定一条 select 查询语句，从服务器后台执行查询操作后，返回 JSON 格式的查询结果集，以数组形式存放。

参数：数据库连接字符串、查询 sql 语句。

举例：var data=myRunSelectSql(sysdatabasestring, sql);

```
eval("result="+data+";");   //将查询结果存放在result数组中
```

46．function myRunUpdateSql(databasestring, sql)

功能：给定一条 insert、update、create 等 select 之外的语句，从服务器后台执行相应操作后，返回操作是否成功的信息。

参数：数据库连接字符串、数据更新 sql 语句。

举例：var updatesql="TRUNCATE TABLE myProvinces;\n INSERT INTO myProvinces (ProvinceID, ProvinceName) SELECT areaid,areaname FROM areas WHERE ParentnodeID="";

```
var result=myRunUpdateSql(sysdatabasestring,sql);
if (result.error==''){
    fnRefresh();
}
```

附录 2、JavaScript 常用函数

abs：返回一个数的绝对值。

acos：返回一个数的反余弦。

anchor：在对象的指定文本两端加上一个带 NAME 属性的 HTML 锚点。

asin：返回一个数的反正弦。

atan：返回一个数的反正切。

atan2：返回从 X 轴到点（y,x）的角度（以弧度为单位）。

atEnd：返回一个表明枚举算子是否处于集合结束处的 Boolean 值。

big：在 String 对象的文本两端加入 HTML 的<BIG>标识。

blink：将 HTML 的<BLINK>标识添加到 String 对象中的文本两端。

bold：将 HTML 的<B>标识添加到 String 对象中的文本两端。

ceil：返回大于或等于其数值参数的最小整数。

charAt：返回位于指定索引位置的字符。

charCodeAt：返回指定字符的 Unicode 编码。

compile：将一个正则表达式编译为内部格式。

concat：（Array）返回一个由两个数组合并组成的新数组。

concat：（String）返回一个包含给定的两个字符串的连接的 String 对象。

cos：返回一个数的余弦。dimensions：返回 VBArray 的维数。

escape：对 String 对象编码，以便在所有计算机上都能阅读。

eval：对 JScript 代码求值然后执行之。

exec：在指定字符串中执行一个匹配查找。

exp：返回 e（自然对数的底）的幂。

fixed：将 HTML 的<TT>标识添加到 String 对象中的文本两端。

floor：返回小于或等于其数值参数的最大整数。

fontcolor：将 HTML 带 COLOR 属性的<FONT>标识添加到 String 对象中的文本两端。

fontsize：将 HTML 带 SIZE 属性的<FONT>标识添加到 String 对象中的文本两端。

fromCharCode：返回 Unicode 字符值的字符串。

getDate：使用当地时间返回 Date 对象的月份日期值。

getDay：使用当地时间返回 Date 对象的星期几。

getFullYear：使用当地时间返回 Date 对象的年份。

getHours：使用当地时间返回 Date 对象的小时值。

getItem：返回位于指定位置的项。

getMilliseconds：使用当地时间返回 Date 对象的毫秒值。

getMinutes：使用当地时间返回 Date 对象的分钟值。

getMonth：使用当地时间返回 Date 对象的月份。

getSeconds：使用当地时间返回 Date 对象的秒数。

getTime：返回 Date 对象中的时间。

getTimezoneOffset：返回主机的时间和全球标准时间（UTC）之间的差（以分钟为单位）。

getUTCDate：使用全球标准时间（UTC）返回 Date 对象的日期值。

getUTCDay：使用全球标准时间（UTC）返回 Date 对象的星期几。

getUTCFullYear：使用全球标准时间（UTC）返回 Date 对象的年份。

getUTCHours：使用全球标准时间（UTC）返回 Date 对象的小时数。
getUTCMilliseconds：使用全球标准时间（UTC）返回 Date 对象的毫秒数。
getUTCMinutes：使用全球标准时间（UTC）返回 Date 对象的分钟数。
getUTCMonth：使用全球标准时间（UTC）返回 Date 对象的月份值。
getUTCSeconds：使用全球标准时间（UTC）返回 Date 对象的秒数。
getVarDate：返回 Date 对象中的 VT_DATE。
getYear：返回 Date 对象中的年份。
indexOf：返回在 String 对象中第一次出现子字符串的字符位置。
isFinite：返回一个 Boolean 值，表明某个给定的数是否是有穷的。
isNaN：返回一个 Boolean 值，表明某个值是否为保留值 NaN（不是一个数）。
italics：将 HTML 的<I>标识添加到 String 对象中的文本两端。
item：返回集合中的当前项。
join：返回一个由数组中的所有元素连接在一起的 String 对象。
lastIndexOf：返回在 String 对象中子字符串最后出现的位置。
lbound：返回在 VBArray 中指定维数所用的最小索引值。
link：将带 HREF 属性的 HTML 锚点添加到 String 对象中的文本两端。
log：返回某个数的自然对数。
match：使用给定的正则表达式对象对字符串进行查找，并将结果作为数组返回。
max：返回给定的两个表达式中的较大者。
min：返回给定的两个数中的较小者。
moveFirst：将集合中的当前项设置为第一项。
moveNext：将当前项设置为集合中的下一项。
parse：对包含日期的字符串进行分析，并返回该日期与 1970 年 1 月 1 日零点之间相差的毫秒数。
parseFloat：返回从字符串转换而来的浮点数。
parseInt：返回从字符串转换而来的整数。
pow：返回一个指定幂次的底表达式的值。
random：返回一个 0 和 1 之间的伪随机数。
replace：返回根据正则表达式进行文字替换后的字符串的拷贝。
reverse：返回一个元素反序的 Array 对象。
round：将一个指定的数值表达式舍入到最近的整数并将其返回。
search：返回与正则表达式查找内容匹配的第一个子字符串的位置。
setDate：使用当地时间设置 Date 对象的数值日期。
setFullYear：使用当地时间设置 Date 对象的年份。
setHours：使用当地时间设置 Date 对象的小时值。
setMilliseconds：使用当地时间设置 Date 对象的毫秒值。
setMinutes：使用当地时间设置 Date 对象的分钟值。
setMonth：使用当地时间设置 Date 对象的月份。
setSeconds：使用当地时间设置 Date 对象的秒值。
setTime：设置 Date 对象的日期和时间。
setUTCDate：使用全球标准时间（UTC）设置 Date 对象的数值日期。
setUTCFullYear：使用全球标准时间（UTC）设置 Date 对象的年份。

setUTCHours：使用全球标准时间（UTC）设置 Date 对象的小时值。
setUTCMilliseconds：使用全球标准时间（UTC）设置 Date 对象的毫秒值。
setUTCMinutes：使用全球标准时间（UTC）设置 Date 对象的分钟值。
setUTCMonth：使用全球标准时间（UTC）设置 Date 对象的月份。
setUTCSeconds：使用全球标准时间（UTC）设置 Date 对象的秒值。
setYear：使用 Date 对象的年份。sin：返回一个数的正弦。
slice：（Array）返回数组的一个片段。
slice：（String）返回字符串的一个片段。
small：将 HTML 的<SMALL>标识添加到 String 对象中的文本两端。
sort：返回一个元素被排序了的 Array 对象。
split：将一个字符串分割为子字符串，然后将结果作为字符串数组返回。
sqrt：返回一个数的平方根。
strike：将 HTML 的<STRIKE>标识添加到 String 对象中的文本两端。
sub：将 HTML 的<SUB>标识放置到 String 对象中的文本两端。
substr：返回一个从指定位置开始并具有指定长度的子字符串。
substring：返回位于 String 对象中指定位置的子字符串。
sup：将 HTML 的<SUP>标识放置到 String 对象中的文本两端。
tan：返回一个数的正切。
test：返回一个 Boolean 值，表明在被查找的字符串中是否存在某个模式。
toArray：返回一个从 VBArray 转换而来的标准 JScript 数组。
toGMTString：返回一个转换为使用格林威治标准时间（GMT）的字符串的日期。
toLocaleString：返回一个转换为使用当地时间的字符串的日期。
toLowerCase：返回一个所有的字母字符都被转换为小写字母的字符串。
toString：返回一个对象的字符串表示。
toUpperCase：返回一个所有的字母字符都被转换为大写字母的字符串。
toUTCString：返回一个转换为使用全球标准时间（UTC）的字符串的日期。
ubound：返回在 VBArray 的指定维中所使用的最大索引值。
unescape：对用 escape：编码的 String 对象进行解码。
UTC：返回 1970 年 1 月 1 日零点的全球标准时间（UTC）（或 GMT）与指定日期之间的毫秒数.
valueOf：返回指定对象的原始值。

浙江理工大学
ZHEJIANG SCI-TECH UNIVERSITY
版权所有，未经许可不得复制

网盘下载地址：http://pan.baidu.com/s/1dDuHMa1 ，密码：65h4