《JavaEE 实验手册》

Java 教研室

版 本 1.3

文档提供：Java 教研室 孙丽萍

修 改 记 录

修改时间	修改人	修改内容
2009.9.07	孟双英	文档创建
2012.8.27	张立飞	修改
2016.3.7	孙丽萍	修改

目录

第六章 过滤器

一、内容概述

本章的教学内容是介绍 JavaWeb 开发中 Filter 过滤器相关知识，包括编写，配置，部署，运行以及 Filter 相关接口、类知识介绍。

通过本章实验应该达到的目的：掌握如何编写 Filter，以及 Filter 的常见功能用处。

二、实验一 实现字符编码过滤功能

2.1 实验目的

编写一个 Filter 过滤器
实现功能:实现字符编码过滤功能

2.2 准备

保证 Eclipse 已经正确配置 Tomcat 服务器，可以进行 JavaWeb 项目编写、编译与调试。（如果以上设置不正确，请参考《Eclipse 部署配置 Tomcat 手册》进行正确设置）

2.3 实验步骤

步骤一：

在 Eclipse 中新建 Dynamic project “02_filter“，在“02_filter/src”目录下新建 filter.EncodingFilter.java，此类实现 Filter 接口，私有成员变量 String encode，私有成员常量 String DEFAULT_ENCODING = "UTF-8"，在初始化方法中要初始化成员变量 encode 的值，使用 FilterConfig 对象的 getInitParameter()方法获取 filter 配置参数 encode 值，在 doFilter()方法中设置 response 和 request 的字符编码，关键代码如下

```
@Override
public void doFilter(ServletRequest request, ServletResponse response,
        FilterChain filterChain) throws IOException, ServletException {
    // TODO Auto-generated method stub
    request.setCharacterEncoding(this.encoding);
    response.setContentType("text/html;charset=UTF-8");
    response.setCharacterEncoding("UTF-8");
    filterChain.doFilter(request, response);

}
```

图 2-1

步骤二：

在“02_filter/WebContent/WEB-INF/”的 web.xml 文件中配置 Filter 节点，关键代码如下

```
<filter>
    <filter-name>EncodingFilter</filter-name>
    <filter-class>filter.EncodingFilter</filter-class>
    <init-param>
        <param-name>encode</param-name>
        <param-value>UTF-8</param-value>
    </init-param>
</filter>
<filter-mapping>
<filter-name>EncodingFilter</filter-name>
<url-pattern>/*</url-pattern>
</filter-mapping>
```

图 2-2

步骤三：

在“02_filter”项目中编写用户登录流程，使用 servle，示例为 Servlet+JSP 实现用户登录

步骤四：

在“02_filter/src”目录下新建 domain.Users.javaBean，使用实验手册 01-03 中的数据库表 user，新建 domian.DbConnection.java 文件，获取数据库连接(参考 01-03 实验手册中的 DbConnection 数据库连接类文件编写)，向数据库中 user 表中插入一条数据带有中文的数据“孟双英，123456”

```
package domain;

import java.io.InputStream;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.Properties;

public class DbConnection {
    private static Properties properties;
       static {
```

```
            InputStream                                is                                =
DbConnection.class.getResourceAsStream("/db.properties");
            properties = new Properties();
            try {
                properties.load(is);
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
        public static Connection getConnection() {
            Connection connection = null;
            try {
                Class.forName(properties.getProperty("dbDriver"));
                connection = DriverManager.getConnection(
                properties.getProperty("dbURL"),
                properties.getProperty("user"),
                properties.getProperty("password"));
            } catch (ClassNotFoundException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            } catch (SQLException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
            return connection;
        }
        public Users login(String name,String pwd)
        {
            Connection conn = DbConnection.getConnection();
            Statement stmt;
            Users e = null;
            try {
                stmt = conn.createStatement();
                String sql = "select * from user where name='"+name+"' and
password='"+pwd+"'";
                System.out.println("sql = "+sql);
                ResultSet rs = stmt.executeQuery(sql);
                while (rs.next()) {
                    e= new Users();
                    e.setName(rs.getString("name"));
                    e.setPassword(rs.getString("password"));
                    e.setId(rs.getInt("id"));

                }
```

```
                conn.close();
            } catch (SQLException e1) {
        // TODO Auto-generated catch block
                e1.printStackTrace();
            }

            return e;}
}
```

步骤五：

在“02_filter/src”目录下新建 servlet.LoginServlet.java 的 servlet 文件，dopost()方法中处理客户端提交的用户数据，如果用户名、密码正确 session 缓存用户信息，跳转到 success.jsp 页面，如果用户名密码错误跳转到 login.jsp 页面并提示”用户名密码错误”，关键代码如下

```
/**
 * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse
 *      response)
 */
protected void doPost(HttpServletRequest request,
        HttpServletResponse response) throws ServletException, IOException {
    // TODO Auto-generated method stub
    String userName = request.getParameter("userName");
    String password = request.getParameter("password");
    DbConnection db = new DbConnection();
    Users user = db.login(userName, password);
    System.out.println(userName);
    if (user != null) {
        request.getSession().setAttribute("user", user);
        response.sendRedirect("success.jsp");
    } else {
        request.setAttribute("result", "用户名密码错误");
        request.getRequestDispatcher("login.jsp")
                .forward(request, response);
    }
}
```

图 2-3

在 web.xml 中配置 LoginServlet 节点

```
<servlet>
  <description></description>
  <display-name>LoginServlet</display-name>
  <servlet-name>LoginServlet</servlet-name>
  <servlet-class>servlet.LoginServlet</servlet-class>
</servlet>
<servlet-mapping>
  <servlet-name>LoginServlet</servlet-name>
  <url-pattern>/LoginServlet.do</url-pattern>
</servlet-mapping>
```

图 2-4

步骤六：

编写登录页面 login.jsp 和登录成功页面 success.jsp,其中 login.jsp 页面显示登录框如果用户登录失败跳转回登录页面要提示用户登录失败信息，success.jsp 页面显示欢迎用户信息就可以了。

步骤七：

运行项目“02_filter”，在浏览器中输入 http://lcoalhost:8080/02_filter/login.jsp,回车，效果图如下：

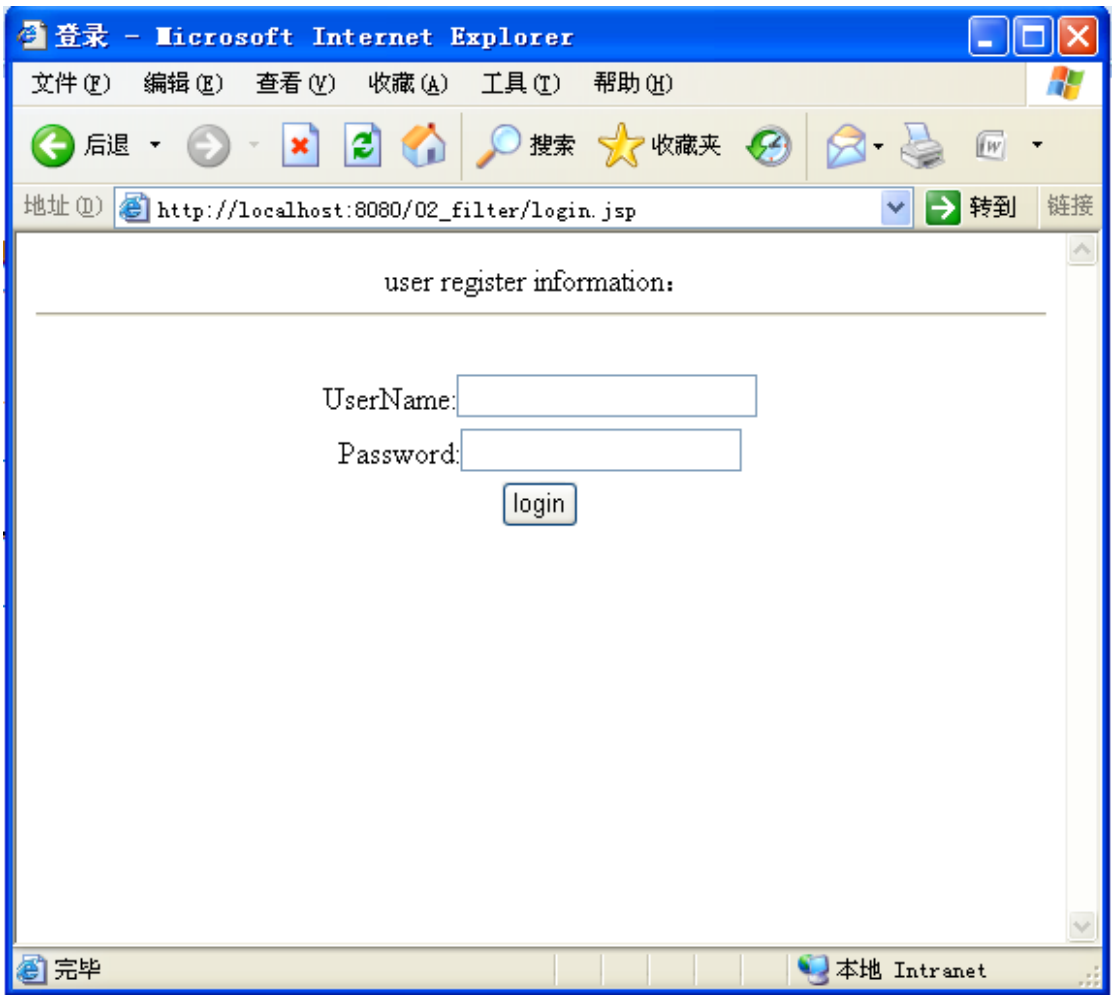

图 2-5

输入用户名、密码，先输入错误的用户名密码：admin，1234，点击 login，效果如下

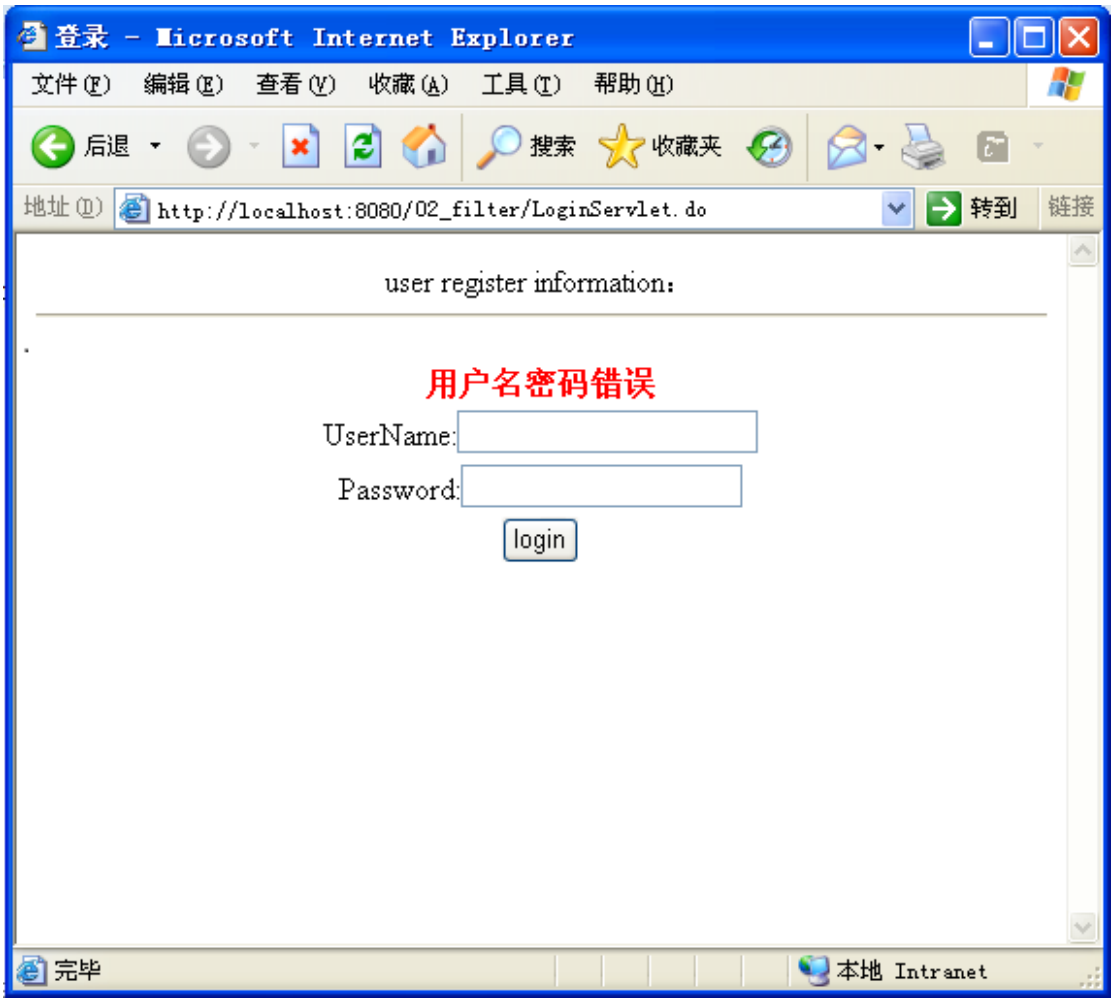

图 2-6

后台控制台显示如下

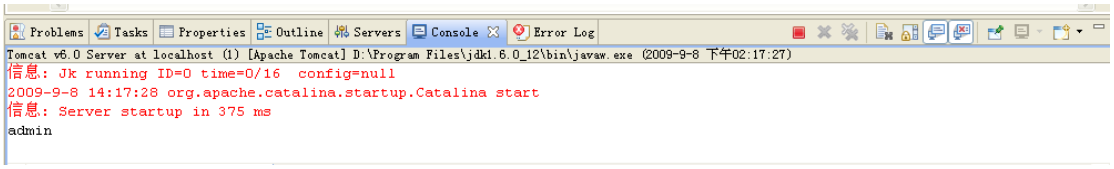

图 2-7

输入正确的用户名密码：孟双英，123456，点击 login 或者 admin，123456，点击 login，效果如下

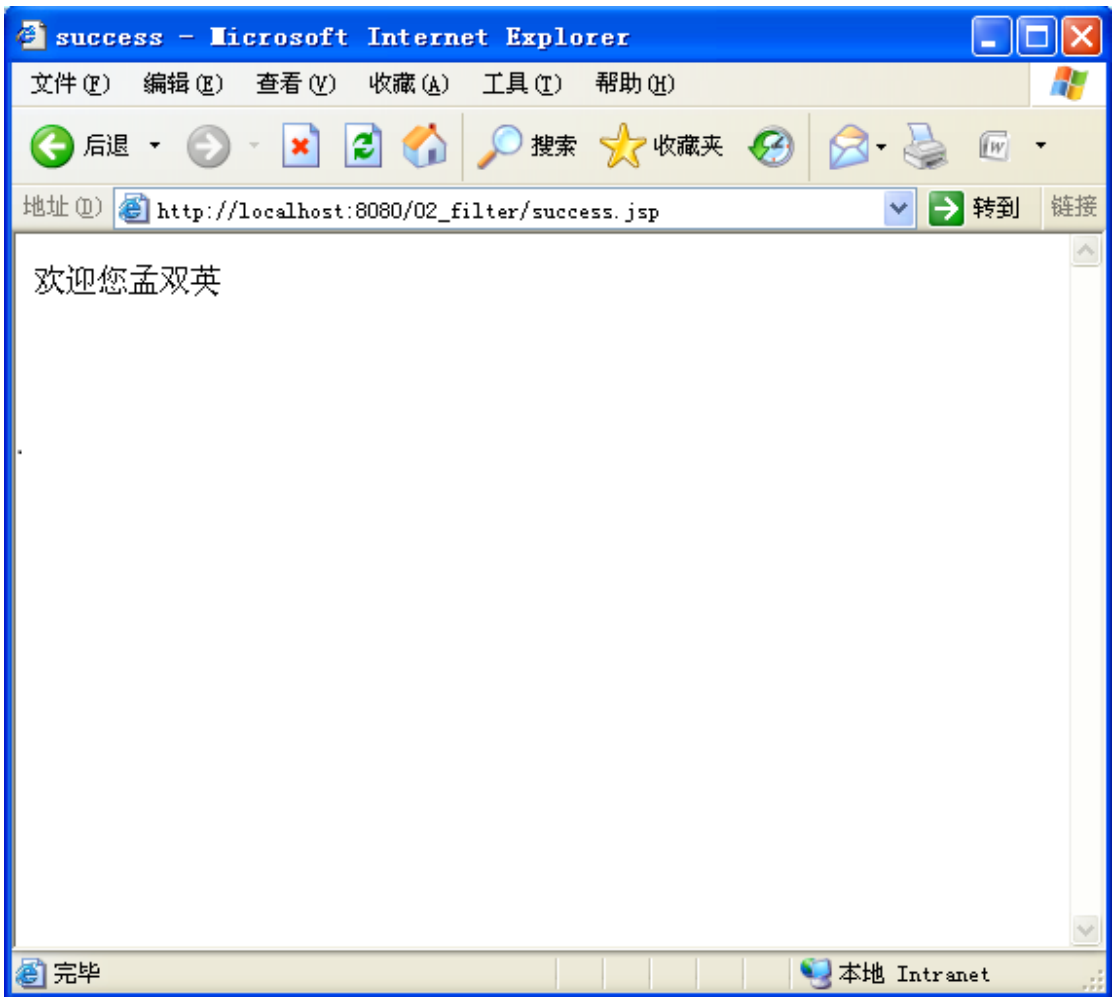
success - Microsoft Internet Explorer
文件(F) 编辑(E) 查看(V) 收藏(A) 工具(T) 帮助(H)
后退
搜索
收藏夹
地址(D) http://localhost:8080/02_filter/success.jsp
转到
链接
欢迎您孟双英
完毕
本地 Intranet

图 2-8-1

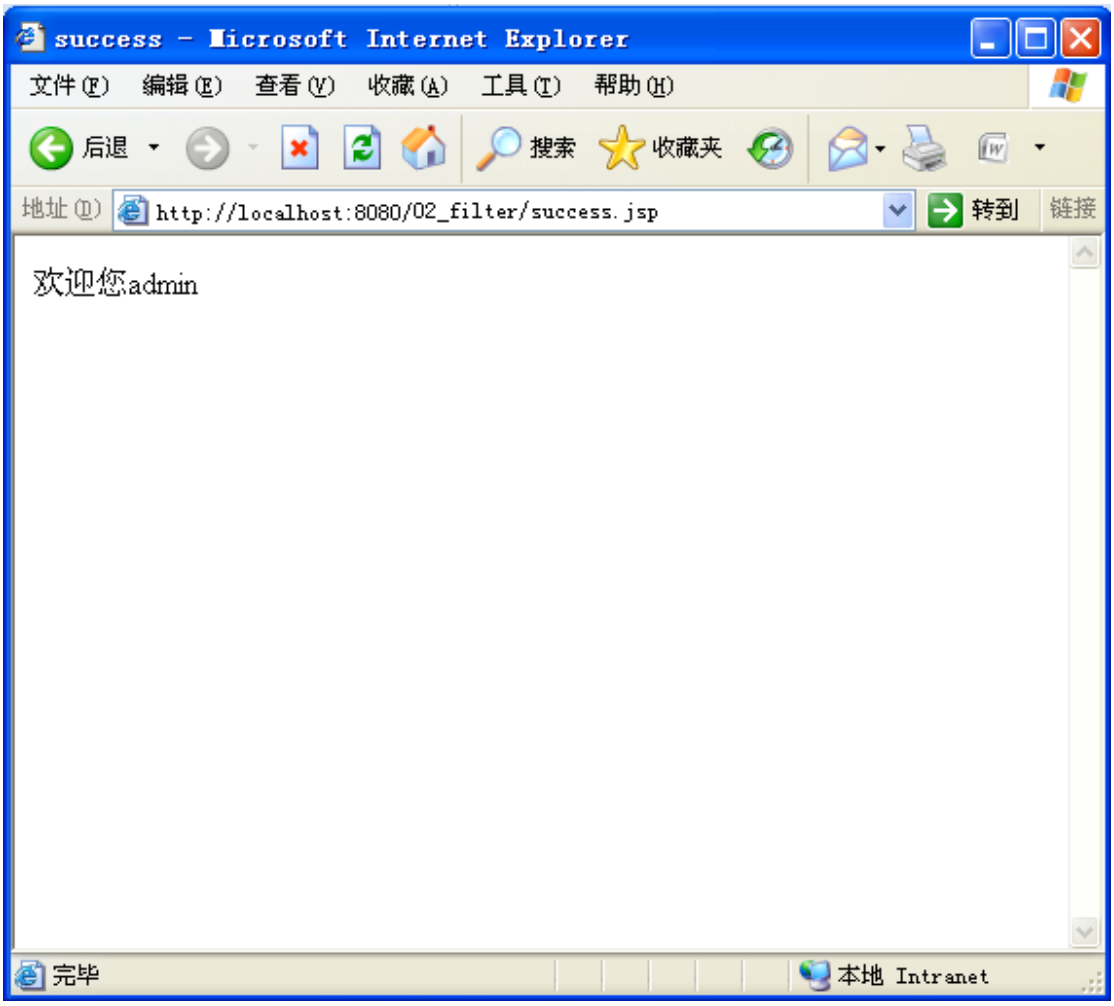

图 2-8-2

后台控制台显示如下

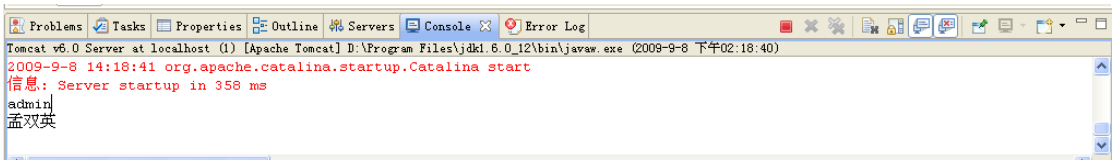

图 2-9

步骤七：

修改以下 web.xml 中 Filter 配置节点，将 Filter 的 url-pattern 节点值设置为其他的/AAA 或者/BBB，保存，重新运行项目，输入用户名密码：孟双英，123456，点击 login，再观察结果

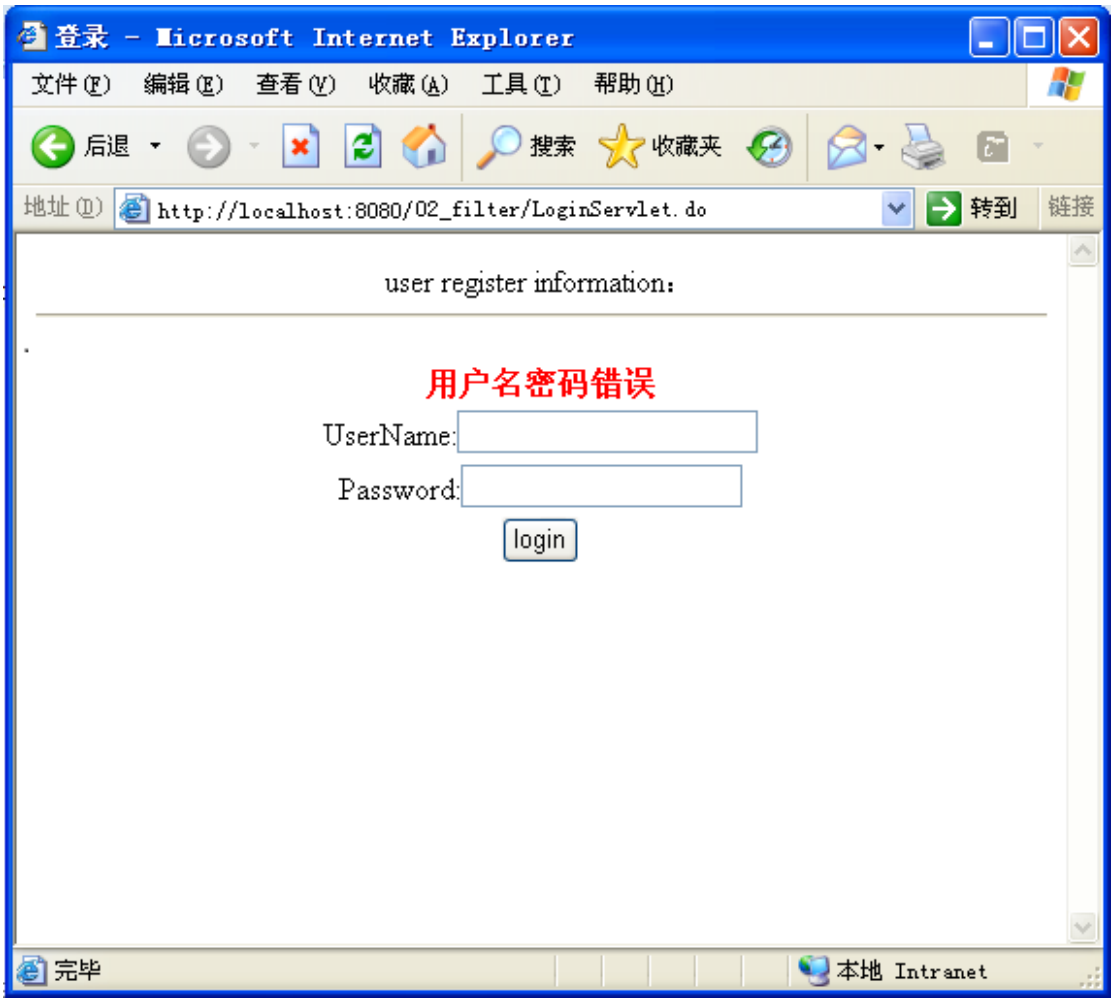

图 2-10-1

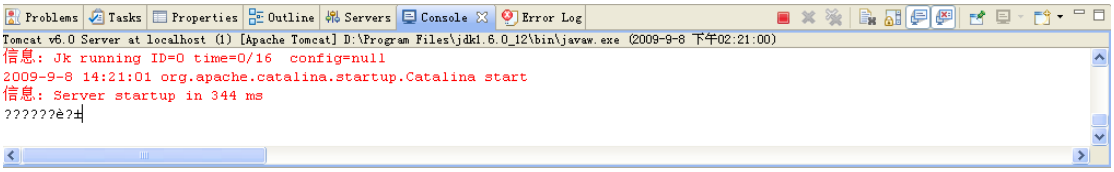

图 2-10-2

2.4实验结论

通过以上实验可以观察出，不使用过滤器中文字符会在页面到后台 servlet 的传输中出现乱码，我们的 EncodingFilter 会屏蔽或解决中文乱码问题，这是过滤器的一个广泛用途，由于一般网站的数据交换都要设计编码问题所以我们 Filter 的 url-pattern 节点值通常设置为“/*”

三、实验二 登录成功用户才能访问系统首页面

3.1 实验目的

编写 Filter 过滤器，实现 Filter 接口
实现功能:只有登录成功用户才能访问系统首页面

3.2 准备

完成实验一

3.3 实验步骤

步骤一：

在“02_filter/src/filter”目录下新建 LimitFilter.java，此过滤器实现检验用户是否成功登录，关键代码如下

```
/**
 * @see Filter#doFilter(ServletRequest, ServletResponse, FilterChain)
 */
public void doFilter(ServletRequest request, ServletResponse response,
        FilterChain filterChain) throws IOException, ServletException {
    // TODO Auto-generated method stub
    HttpServletRequest req = (HttpServletRequest) request;
    Users user = (Users) req.getSession().getAttribute("user");
    if (user != null)
        filterChain.doFilter(request, response);
    else {
        req.setAttribute("result", "您还没有登录");
        req.getRequestDispatcher("login.jsp").forward(req,
                (HttpServletResponse) response);
    }
}
```

图 3-1

步骤二：

在 web.xml 文件中配置 LimitFilter 节点

```
<filter>
  <display-name>LimitFilter</display-name>
  <filter-name>LimitFilter</filter-name>
  <filter-class>filter.LimitFilter</filter-class>
</filter>
<filter-mapping>
  <filter-name>LimitFilter</filter-name>
  <url-pattern>/index.jsp</url-pattern>
</filter-mapping>
</web-app>
```

图 3-2

步骤三：

在实验一的 success.jsp 页面新加超链接查看首页面

步骤四：

运行“02_filter”项目,在浏览器中输入“http://localhost:8080/02_filter/index.jsp”效果如下图

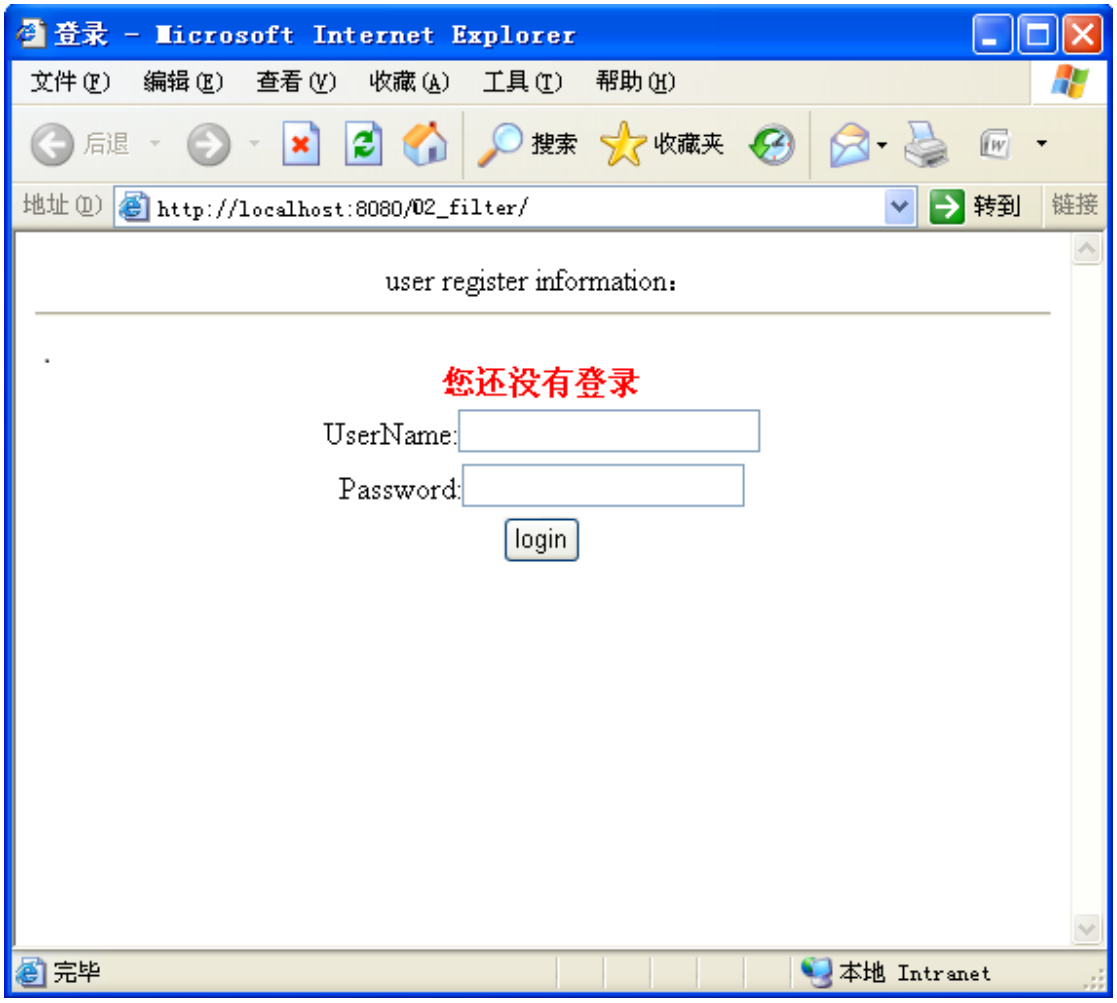

图 3-3

输入正确的用户名、密码，点击 login，效果如图

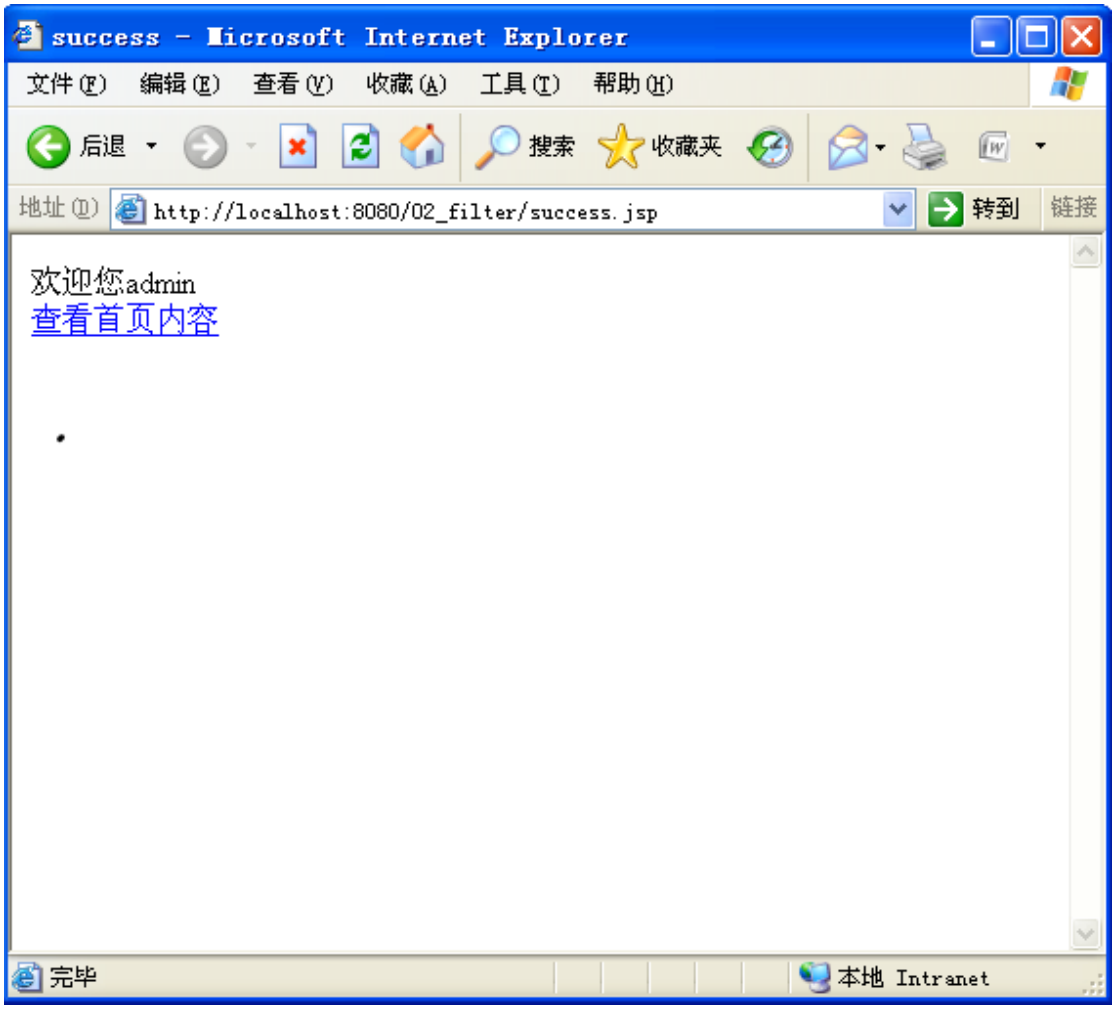

图 3-4

点击查看首页内容或者在地址栏直接输入“http://localhost:8080/02_filter/index.jsp”，效果如下图

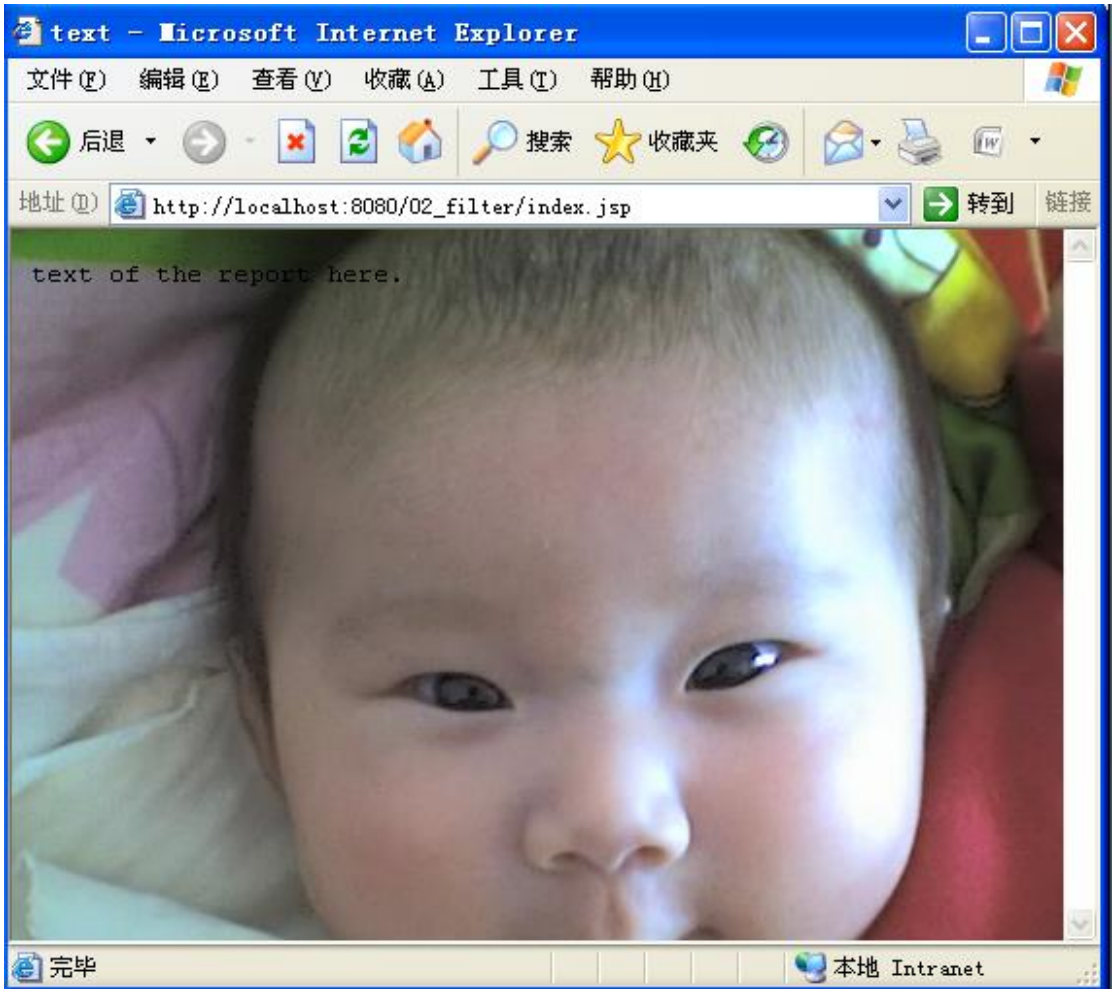

图 3-5

3.4 实验结论

通过以上实验我们再次练习 Filter 编写步骤，并演示了 Filter 过滤器的又一重要应用--权限过滤又称认证过滤，主要应用在根据权限制操作的场合