# Jdbc（03）

## BaseServlet

### 作用

在一个servlet中有多种方法完成对同一个Dao的操作，其中的Service方法根据请求中携带的参数method的值，通过反射直接调用相应的方法；

同时通过方法的返回值（String），在Service中直接进行转发或者重定向，从而简化servlet的代码重复。

### 代码

* BeanServlet

package cn.itcast.web.Servlet;

import javax.servlet.ServletException;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import java.io.IOException;

import java.lang.reflect.Method;

public class BaseServlet extends HttpServlet {

@Override

protected void service(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {

req.setCharacterEncoding("utf-8");//请求编码（POST）

resp.setContentType("text/html;charset=utf-8");//响应编码

/\*

\* 1.获取参数，用来识别用户想请求的方法

\* 2.然后调用该方法

\* \*/

String methodName = req.getParameter("method");

if(methodName == null || methodName.trim().isEmpty()){

throw new RuntimeException("无method参数!");

}

/\*

\* 改进

\* 得到方法名称，通过反射来调用方法

\* 1.得到方法名，通过方法名再得到Method类的对象

\* > 需要得到class，再调用它的方法查询方法，得到method

\* > 先得到当前类的class

\* \*/

//得到class

Class aClass = this.getClass();

//得到method

Method method = null;

try {

method = aClass.getMethod(methodName,HttpServletRequest.class,HttpServletResponse.class);

} catch (Exception e) {

throw new RuntimeException("方法 "+methodName+" 不存在！");

}

/\*

\* 调用method表示的方法

\* \*/

try {

String s = (String )method.invoke(this, req, resp);//反射调用

/\*

\* 获取请求处理方法执行之后返回的字符串，他表示转发或者重定向的路径

\* 在此处完成转发或者重定向操作，简化操作。

\* \*/

/\*

\* 如果用户返回的字符串为null，或者为“”，do nothing

\* \*/

if(s == null || s.trim().isEmpty()){

return;

}

/\*

\* 查看返回的字符串中是否包含冒号，如果没有，默认表示转发

\* 如果有，使用冒号分割字符串，得到前缀和后缀

\* 若前缀为f，表示转发，若为r，表示重定向，后缀就是要转发或者重定向的路径！

\* \*/

if(s.contains(":")){

String[] strings = s.split(":");

if(strings[0].equals("f")){

req.getRequestDispatcher(strings[1]).forward(req,resp);

}else if(strings[0].equals("r")){

resp.sendRedirect(req.getContextPath()+strings[1]);

}else{

throw new RuntimeException("当前版本不支持操作 "+strings[0]+" @@.");

}

}else{

//不包含冒号，默认为转发

req.getRequestDispatcher(s).forward(req,resp);

}

} catch (Exception e) {

System.out.println(methodName+" 方法内部产生了异常！");

throw new RuntimeException(e);

}

}

}

* 之后的servlet可以直接继承BeanServlet

public class ServletA extends BaseServlet{}

写入需要的方法即可。

* 返回值

return "f:/login.jsp";

return "r:/login.jsp";

return "/login.jsp";

## Service层事务