

[文章](#) [问答](#) [论坛](#) [东西](#) [休息室](#) [?](#)

Search for articles, questions,

手表



使用 JSF 的简单 CRUD 示例



宋丽博士

2015 年 9 月 17 日 [警察](#)

评价我: 5.00/5 (4 票)

这是一个使用 JSF 的简单 CRUD 示例。

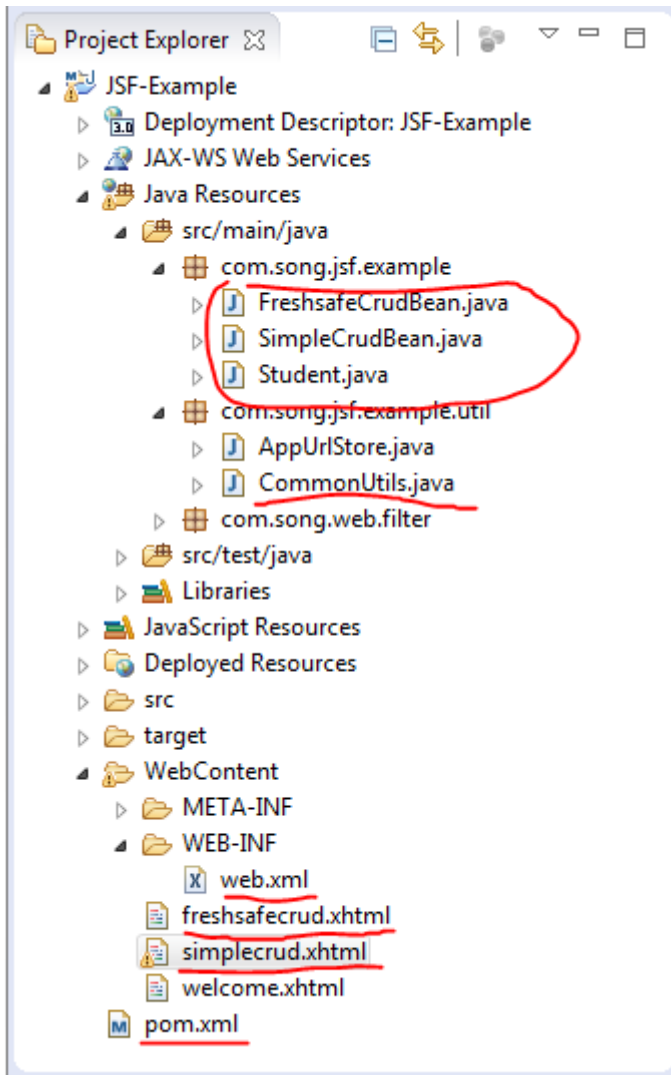
[下载 JSF-Example.zip - 10.5 KB](#)

介绍

这是一个使用 JSF 的简单 CRUD 示例。

背景

JSF 是一个 MVC 框架，但它与 Spring MVC 和 ASP.NET MVC 有很大不同。它实际上具有很强的 [ASP.NET Web 表单](#) “POSTBACK” 风味。此示例基于 [堆栈溢出示例](#)。



附件是一个Maven项目。我已经使用 Maven 3.2.1、Java 1.8.0_45、Tomcat 7 和 Eclipse Java EE IDE for Web Developers Luna Service Release 2 对其进行了测试。它实际上有两个示例。

- “simplecrud.xhtml”是简单的 CRUD 示例；
- “freshsafecrud.xhtml”是为了解决由于JSF“[POSTBACK](#)”性质引起的问题。

“welcome.xhtml”是两个例子的索引页。

Simple crud example

- [Simple crud example](#)
- [Fresh safe crud example](#)

如果您不熟悉如何将 Maven 项目导入 Eclipse，您可以查看[此链接](#)。当我尝试导入项目时，我注意到 Eclipse 向我显示了一些错误消息，告诉我一些验证错误。您可以简单地忽略这些消息并从 Eclipse 中删除标记。

pom.xml 和 web.xml

“pom.xml”声明了创建 JSF 应用程序所需的依赖项。

XML

缩小▲ 复制代码

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
    http://maven.apache.org/xsd/maven-4.0.0.xsd">
```

```

<modelVersion>4.0.0</modelVersion>
<groupId>com.song.example</groupId>
<artifactId>JSF-Example</artifactId>
<version>0.0.1-SNAPSHOT</version>
<packaging>war</packaging>

<properties>
  <tomcat.version>7.0.55</tomcat.version>
  <jsf.version>2.1.7</jsf.version>
</properties>

<dependencies>
  <!-- Sevlet jars for compilation, provided by Tomcat -->
  <dependency>
    <groupId>org.apache.tomcat</groupId>
    <artifactId>tomcat-servlet-api</artifactId>
    <version>${tomcat.version}</version>
    <scope>provided</scope>
  </dependency>

  <dependency>
    <groupId>com.sun.faces</groupId>
    <artifactId>jsf-api</artifactId>
    <version>${jsf.version}</version>
  </dependency>
  <dependency>
    <groupId>com.sun.faces</groupId>
    <artifactId>jsf-impl</artifactId>
    <version>${jsf.version}</version>
  </dependency>

  <dependency>
    <groupId>javax.validation</groupId>
    <artifactId>validation-api</artifactId>
    <version>1.1.0.Final</version>
  </dependency>
</dependencies>

<build>
  <finalName>${project.artifactId}</finalName>
  <plugins>
    <plugin>
      <artifactId>maven-compiler-plugin</artifactId>
      <version>3.1</version>
      <configuration>
        <source>1.8</source>
        <target>1.8</target>
      </configuration>
    </plugin>

    <plugin>
      <artifactId>maven-war-plugin</artifactId>
      <version>2.4</version>
      <configuration>
        <warSourceDirectory>WebContent</warSourceDirectory>
        <failOnMissingWebXml>true</failOnMissingWebXml>
      </configuration>
    </plugin>
  </plugins>
</build>
</project>

```

示例应用程序的“web.xml”如下。

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="http://java.sun.com/xml/ns/javaee"
  xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
    http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd"
  id="WebApp_ID" version="3.0">

  <display-name>JSF Example</display-name>
  <welcome-file-list>
    <welcome-file>welcome.xhtml</welcome-file>
  </welcome-file-list>

  <filter>
    <filter-name>nocachefilter</filter-name>
    <filter-class>
      com.song.web.filter.NocacheFilter
    </filter-class>
  </filter>
  <filter-mapping>
    <filter-name>nocachefilter</filter-name>
    <url-pattern>/*</url-pattern>
  </filter-mapping>

  <servlet>
    <servlet-name>jsf-servlet</servlet-name>
    <servlet-class>javax.faces.webapp.FacesServlet</servlet-class>
    <load-on-startup>1</load-on-startup>
  </servlet>
  <servlet-mapping>
    <servlet-name>jsf-servlet</servlet-name>
    <url-pattern>*.xhtml</url-pattern>
  </servlet-mapping>

  <session-config>
    <session-timeout>20</session-timeout>
    <tracking-mode>COOKIE</tracking-mode>
  </session-config>

  <context-param>
    <param-name>BaseUrl</param-name>
    <param-value>
      http://localhost:8080/JSF-Example/
    </param-value>
  </context-param>
</web-app>
```

- 从 servlet 容器的角度来看，所有 JSF 页面都将映射到由“javax.faces.webapp.FacesServlet”类实现的单个 servlet；
- 在这个示例应用程序中，我们将所有“xhtml”页面的所有请求映射到“javax.faces.webapp.FacesServlet”。

简单的 CRUD 示例

简单示例的**管理 bean**是在“SimpleCrudBean”类中实现的。

爪哇

缩小▲ 复制代码

```
package com.song.jsf.example;

import java.io.IOException;
import java.io.Serializable;
import java.util.ArrayList;
import java.util.List;

import javax.annotation.PostConstruct;
import javax.faces.bean.ManagedBean;
import javax.faces.bean.SessionScoped;
```

```
@ManagedBean
@SessionScoped
public class SimpleCrudBean implements Serializable {
    private static final long serialVersionUID = 1L;

    private List<Student> list;
    private Student item = new Student();
    private Student beforeEditItem = null;
    private boolean edit;

    @PostConstruct
    public void init() {
        list = new ArrayList<Student>();
    }

    public void add() {
        // DAO save the add
        item.setId(list.isEmpty() ? 1 : list.get(list.size() - 1).getId() + 1);
        list.add(item);
        item = new Student();
    }

    public void resetAdd() {
        item = new Student();
    }

    public void edit(Student item) {
        beforeEditItem = item.clone();
        this.item = item;
        edit = true;
    }

    public void cancelEdit() {
        this.item.restore(beforeEditItem);
        this.item = new Student();
        edit = false;
    }

    public void saveEdit() {
        // DAO save the edit
        this.item = new Student();
        edit = false;
    }

    public void delete(Student item) throws IOException {
        // DAO save the delete
        list.remove(item);
    }

    public List<Student> getList() {
        return list;
    }

    public Student getItem() {
        return this.item;
    }

    public boolean isEdit() {
        return this.edit;
    }
}
```

- “添加”、“保存编辑”和“删除”方法需要将数据与数据库等持久存储同步；
- 在此示例中，为简单起见，跳过了数据库操作。

“Student”类在“Student.java”文件中实现。

```

package com.song.jsf.example;

import java.io.Serializable;

public class Student implements Serializable {
    private static final long serialVersionUID = 1L;

    private Long id;
    private String name;

    public Student() {}
    public Student(Long id, String name) {
        this.id = id;
        this.name = name;
    }

    public Long getId() { return id; }
    public void setId(Long id) { this.id = id; }
    public String getName() { return name; }
    public void setName(String name) { this.name = name; }

    @Override
    public Student clone() {
        return new Student(id, name);
    }

    public void restore(Student student) {
        this.id = student.getId();
        this.name = student.getName();
    }
}

```

“SimpleCrudBean”类绑定到“simplecrud.xhtml”文件以使其功能齐全。

缩小▲ 复制代码

```

<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:f="http://java.sun.com/jsf/core"
      xmlns:h="http://java.sun.com/jsf/html">
<head>
    <title>Simple CRUD</title>
</head>

<body>
    <h3>List students</h3>
    <h:form rendered="#{not empty simpleCrudBean.list}">
        <h:dataTable value="#{simpleCrudBean.list}" var="item">
            <h:column><f:facet name="header">ID</f:facet>#{item.id}</h:column>
            <h:column><f:facet name="header">Name</f:facet>#{item.name}</h:column>
            <h:column>
                <h:commandButton value="edit" action="#{simpleCrudBean.edit(item)}" />
            </h:column>
            <h:column>
                <h:commandButton value="delete" action="#{simpleCrudBean.delete(item)}" />
            </h:column>
        </h:dataTable>
    </h:form>

    <h:panelGroup rendered="#{empty simpleCrudBean.list}">
        <p>No students! Please add students.</p>
    </h:panelGroup>

    <h:panelGroup rendered="#{!simpleCrudBean.edit}">
        <h3>Add student</h3>
        <h:form>
            <p>Name: <h:inputText value="#{simpleCrudBean.item.name}" /></p>
            <p>

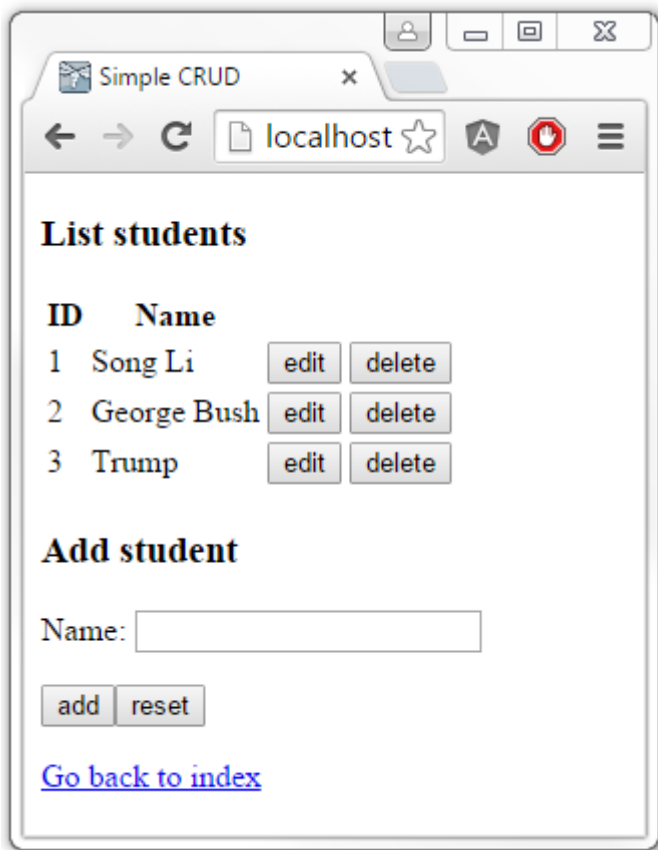
```

```

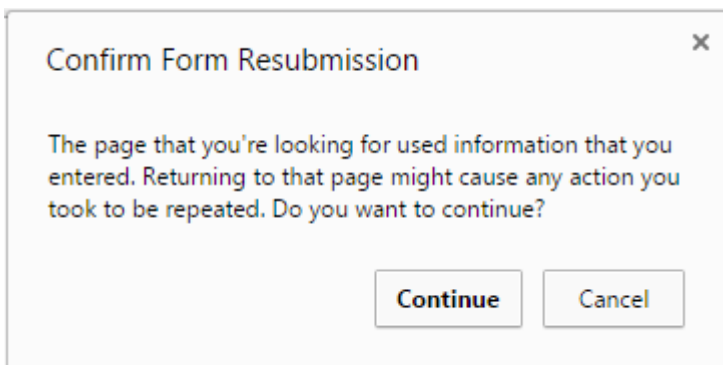
        <h:commandButton value="add" action="#{simpleCrudBean.add}" />
        <h:commandButton value="reset" action="#{simpleCrudBean.resetAdd}" />
    </p>
</h:form>
</h:panelGroup>

<h:panelGroup rendered="#{simpleCrudBean.edit}">
    <h3>Edit student #{simpleCrudBean.item.id}</h3>
    <h:form>
        <p>Name: <h:inputText value="#{simpleCrudBean.item.name}" /></p>
        <p>
            <h:commandButton value="save" action="#{simpleCrudBean.saveEdit}" />
            <h:commandButton value="cancel" action="#{simpleCrudBean.cancelEdit}" />
        </p>
    </h:form>
</h:panelGroup>
<p>
    <a href="#{appUriStore.baseUrl}">Go back to index</a>
</p>
</body>
</html>

```



如果您现在加载“simplecrud.xhtml”页面，您会发现 CRUD 操作一切正常。但是如果你在添加学生后点击浏览器上的刷新按钮，你会看到这个丑陋的弹出消息。



如果您单击“继续”按钮并继续刷新，您会注意到刚刚添加的学生又被添加了。这绝对不是一个好行为。这种行为是由于 JSF 的“POSTBACK”性质造成的。在下一个示例中，我们将尝试解决此问题。

刷新安全的 CRUD 示例

为了解决简单 CRUD 示例中的问题，我创建了“CommonUtils”类。

[缩小▲](#) [复制代码](#)

```
package com.song.jsf.example.util;

import java.io.IOException;
import java.io.Serializable;

import javax.faces.bean.ApplicationScoped;
import javax.faces.bean.ManagedBean;
import javax.faces.context.ExternalContext;
import javax.faces.context.FacesContext;
import javax.servlet.http.HttpServletRequest;

@ManagedBean(name="commonUtils")
@ApplicationScoped
public class CommonUtils implements Serializable {
    private static final long serialVersionUID = 1L;

    public void redirectWithGet() {
        FacesContext facesContext = FacesContext.getCurrentInstance();
        ExternalContext externalContext = facesContext.getExternalContext();
        HttpServletRequest request = (HttpServletRequest)externalContext.getRequest();

        StringBuffer requestURL = request.getRequestURL();
        String queryString = request.getQueryString();

        if (queryString != null) {
            requestURL.append('?').append(queryString).toString();
        }

        String url = requestURL.toString();
        try {
            externalContext.redirect(requestURL.toString());
        } catch (IOException e) {
            throw new RuntimeException("Unable to rerirect to " + url);
        }

        facesContext.responseComplete();
    }
}
```

“redirectWithGet”方法是简单地向浏览器发送重定向请求，用 GET 请求刷新浏览器。“CommonUtils”对象被注入到“FreshsafeCrudBean”类中，只要执行“POSTBACK”，就会调用“redirectWithGet”方法。

[缩小▲](#) [复制代码](#)

```
package com.song.jsf.example;

import java.io.IOException;
import java.io.Serializable;
import java.util.ArrayList;
import java.util.List;

import javax.annotation.PostConstruct;
import javax.faces.bean.ManagedBean;
import javax.faces.bean.ManagedProperty;
import javax.faces.bean.SessionScoped;

import com.song.jsf.example.util.CommonUtils;
```



```
@ManagedBean
@SessionScoped
public class FreshsafeCrudBean implements Serializable {
    private static final long serialVersionUID = 1L;

    private List<Student> list;
    private Student item = new Student();
    private Student beforeEditItem = null;
    private boolean edit;

    @ManagedProperty(value="#{commonUtils}")
    private CommonUtils util;
    public void setUtil(CommonUtils util) {
        this.util = util;
    }

    @PostConstruct
    public void init() {
        list = new ArrayList<Student>();
    }

    public void add() {
        // DAO save the add
        item.setId(list.isEmpty() ? 1 : list.get(list.size() - 1).getId() + 1);
        list.add(item);
        item = new Student();

        util.redirectWithGet();
    }

    public void resetAdd() {
        item = new Student();

        util.redirectWithGet();
    }

    public void edit(Student item) {
        beforeEditItem = item.clone();
        this.item = item;
        edit = true;

        util.redirectWithGet();
    }

    public void cancelEdit() {
        this.item.restore(beforeEditItem);
        this.item = new Student();
        edit = false;

        util.redirectWithGet();
    }

    public void saveEdit() {
        // DAO save the edit
        this.item = new Student();
        edit = false;

        util.redirectWithGet();
    }

    public void delete(Student item) throws IOException {
        // DAO save the delete
        list.remove(item);

        util.redirectWithGet();
    }

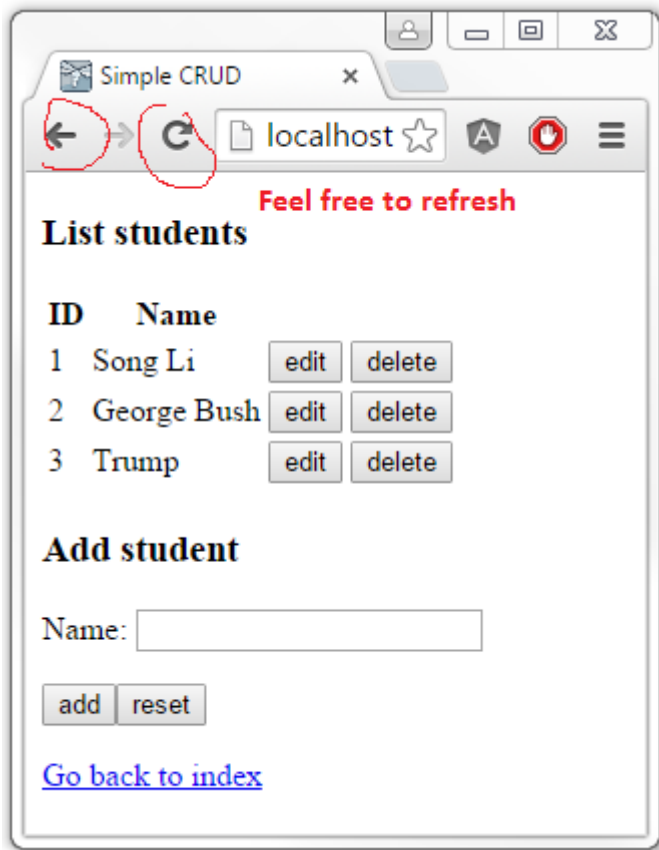
    public List<Student> getList() {
```

```
        return list;
    }

    public Student getItem() {
        return this.item;
    }

    public boolean isEdit() {
        return this.edit;
    }
}
```

“freshsafecrud.xhtml”与“simplecrud.xhtml”文件完全相同，只是它绑定到“FreshsafeCrudBean”类。



如果您现在加载“simplecrud.xhtml”，您应该可以随时刷新页面而不会看到“POSTBACK”的副作用。当然，费用是往返网络服务器的费用。

兴趣点

- 这是一个使用 JSF 的简单 CRUD 示例；
- 真正的 CRUD 操作通常涉及数据库操作，但为了简单起见，本示例跳过了它们；
- JSF 具有强大的 [ASP.NET Web Form](#) 风格，它的功能严重依赖“POSTBACK”；
- 还提供了一个额外的例子来解决“POSTBACK”的副作用，并以额外的服务器往返为代价。

历史

第一次修订 - 2015 年 9 月 16 日

执照

分享

关于作者



宋丽博士

美国 


手表
该会员

我在 IT 行业工作了一段时间。它仍然令人兴奋，我仍在学习。我是一个快乐诚实的人，我想成为你的朋友。


评论和讨论


添加评论或问题 


电子邮件提醒


Search Comments 

第一 页上一页 下一页

- Java+JPA+JSF教程 

Gerd Wagner 17-Sep-15 23:50
- Re: Java+JPA+JSF 教程 

Dr. Song Li 17-Sep-15 23:55
- Re: Java+JPA+JSF 教程 

Gerd Wagner 24-Sep-15 1:41
- Re: Java+JPA+JSF 教程 

Dr. Song Li 25-Sep-15 21:43

刷新

1

-  一般
-  新闻
-  建议
-  问题
-  错误
-  答案
-  笑话
-  赞美
-  咆哮
-  管理员

使用Ctrl+Left/Right 切换消息，Ctrl+Up/Down 切换主题，Ctrl+Shift+Left/Right 切换页面。

