1，技术架构：

Spring+mybatis+jsf，框架使用；

Shiro； 权限系统；

页面框架；

2，配置：

3，页面；

4，数据操作；

5：特殊实现；

Ajax；

Jquery：弹出层，日期；Ztree；

弹出层选取人员；

-------------------

目录：

配置；

///////////////////////////////////////////////////////////////////////////////////////////////

Maven:---或直接jar；

Jar；

配置：

配置文件：

Web.xml；

faces-config.xml:只需要和spring集成的配置；| 自定义组件是另外的；

Spring配置，applicationContext.xml；

Shiro配置，applicationContext-shiro.xml

Shiro需要的缓存，ehcache-shiro.xml

Web.xml：

// JSF2 设置(当前JVM)默认时区

<context-param>

<param-name>javax.faces.DATETIMECONVERTER\_DEFAULT\_TIMEZONE\_IS\_SYSTEM\_TIMEZONE</param-name>

<param-value>true</param-value>

</context-param>

//

<context-param>

<param-name>javax.faces.PROJECT\_STAGE</param-name>

<param-value>Development</param-value>

</context-param>

//Richfaces皮肤，

<context-param>

<param-name>org.richfaces.SKIN</param-name>

<param-value>blueSky</param-value>

</context-param>

//<param-name>org.richfaces.enableControlSkinning</param-name> 值对 RichFaces 和（出乎意料的是）标准 JSF 组件的外观有影响。如果其值为 true，标准控件都涂色。如果省略该元素，则默认值为 true。

<context-param>

<param-name>org.richfaces.enableControlSkinning</param-name>

<param-value>false</param-value>

</context-param>

//过滤器配置：

richfaces内置了几个解析器，TIDY解析器用于复杂的或非标准的标记，这些标记由过滤器做严格效验并更正。NEKO过滤器在解析代码时没有严格效验，大大加快了ajax请求的处理，但使用时你要确保你的应用程序标记是严格用于这个过滤器，否则可能会导致很多错误和恶化布局结果。

所以TIDY解析器相对NEKO解析器来说会慢很多。默认是TIDY解析器。

<context-param>

<param-name>org.ajax4jsf.xmlparser.ORDER</param-name>

<param-value>NEKO</param-value>

</context-param>

<context-param>

<param-name>org.ajax4jsf.xmlparser.NEKO</param-name>

<param-value>.\*\..\*</param-value>

</context-param>

<context-param>

//该配置将强制容器在发送到客户端之前压缩所有的Richfaces资源，包括图片、样式表、JavaScript等。它将大大的减少页面的加载时间。

<context-param>

<param-name>org.ajax4jsf.COMPRESS\_SCRIPT</param-name>

<param-value>true</param-value>

</context-param>

//该配置将保证所有和样式相关的文件在第一次访问Richfaces应用的请求到达时全部加载到客户端。

<context-param>

<param-name>org.richfaces.LoadStyleStrategy</param-name>

<param-value>all</param-value>

</context-param>

//JSF优化技巧将保证所有和Richfaces相关的JavaScript、文件、库等在第一次请求到达时被下载到客户端。

<context-param>

<param-name>org.richfaces.LoadScriptStrategy</param-name>

<param-value>all</param-value>

</context-param>

//最大上传大小20m

<param-name>org.richfaces.fileUpload.maxRequestSize</param-name>

<param-value>20971520</param-value>

</context-param>

//上传文件的存储方式：true，临时文件；false，内存中；

<context-param>

<param-name>org.richfaces.fileUpload.createTempFiles</param-name>

<param-value>true</param-value>

</context-param>

//指定facelets路径

<context-param>

<param-name>javax.faces.FACELETS\_LIBRARIES</param-name>

<param-value>/WEB-INF/shiro-face.taglib.xml</param-value>

</context-param>

//通过ContextLoaderListener加载contextConfigLocation指定的Spring配置文件。

<context-param>

<param-name>contextConfigLocation</param-name>

<param-value>

classpath\*:/applicationContext.xml,

classpath\*:/applicationContext-shiro.xml

</param-value>

</context-param>

//jsf servlet配置，

<listener>

<listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>

</listener>

<listener>

<listener-class>org.springframework.web.context.request.RequestContextListener</listener-class>

</listener>

//路径映射配置

<servlet>

<servlet-name>Faces Servlet</servlet-name>

<servlet-class>javax.faces.webapp.FacesServlet</servlet-class>

<load-on-startup>1</load-on-startup>

</servlet>

<servlet-mapping>

<servlet-name>Faces Servlet</servlet-name>

<url-pattern>\*.xhtml</url-pattern>

</servlet-mapping>

//session和首页配置

<session-config>

<session-timeout>

30

</session-timeout>

</session-config>

<welcome-file-list>

**<welcome-file>greeting.xhtml</welcome-file>**

</welcome-file-list>

//shiro过滤器；

<filter>

<filter-name>shiroSecurityFilter</filter-name>

<filter-class>

org.springframework.web.filter.DelegatingFilterProxy

</filter-class>

</filter>

<filter-mapping>

<filter-name>shiroSecurityFilter</filter-name>

<url-pattern>/\*</url-pattern>

</filter-mapping>

//字体编码过滤器；

<filter>

<filter-name>CharacterEncodingFilter</filter-name>

<filter-class>org.springframework.web.filter.CharacterEncodingFilter</filter-class>

<init-param>

<param-name>encoding</param-name>

<param-value>UTF-8</param-value>

</init-param>

<init-param>

<param-name>forceEncoding</param-name>

<param-value>true</param-value>

</init-param>

</filter>

<filter-mapping>

<filter-name>CharacterEncodingFilter</filter-name>

<url-pattern>/\*</url-pattern>

</filter-mapping>

//错误页面

<error-page>

<exception-type>com.sun.faces.context.FacesFileNotFoundException</exception-type>

<location>/sessionout.xhtml</location>

</error-page>

<error-page>

<error-code>404</error-code>

<location>/sessionout.xhtml</location>

</error-page>

faces-config.xml，

<?xml version="1.0" encoding='UTF-8'?>

<faces-config version="2.2" xmlns="http://xmlns.jcp.org/xml/ns/javaee"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee

http://xmlns.jcp.org/xml/ns/javaee/web-facesconfig\_2\_2.xsd">

...

</faces-config>

//自定义组件；

<component>

<component-type>hasRole</component-type>

<component-class>com.tag.HasRole</component-class>

</component>

<render-kit>

<renderer>

<component-family>hasRole</component-family>

<renderer-type>hasRoleRenderer</renderer-type>

<renderer-class>com.tag.HasRoleRenderer</renderer-class>

</renderer>

</render-kit>

<component>

<component-type>token</component-type>

<component-class>com.tag.Token</component-class>

</component>

<component>

<component-type>htmlRefresh</component-type>

<component-class>com.tag.HtmlRefresh</component-class>

</component>

//jsf,spring集成；DelegatingVariableResolver

<application>

<variable-resolver>

org.springframework.web.jsf.DelegatingVariableResolver

</variable-resolver>

<action-listener>com.cxdai.filter.ActionFilter</action-listener>

<locale-config>

<default-locale>en</default-locale>

<supported-locale>de</supported-locale>

<supported-locale>zh</supported-locale>

</locale-config>

</application>

~

ActionFilter类，

Action过滤，权限验证；

shiro-face.taglib.xml：

//自定义组件对应的标签，

<tag>

<tag-name>hasRole</tag-name>

<component>

<component-type>hasRole</component-type>

<renderer-type>hasRoleRenderer</renderer-type>

</component>

<attribute>

<description></description>

<name>name</name>

<required>true</required>

</attribute>

</tag>

<tag>

<tag-name>token</tag-name>

<component>

<component-type>token</component-type>

</component>

</tag>

<tag>

<tag-name>htmlRefresh</tag-name>

<component>

<component-type>htmlRefresh</component-type>

</component>

</tag>