6、Web-RestfulCRUD

2020年6月6日 11:09

- 1、默认访问首页

```
@Bean//将组件注册在容器中
publicWebMvcConfigurermyMvcConfig1(){
WebMvcConfigurerindex=newWebMvcConfigurer(){
publicvoidaddViewControllers(ViewControllerRegistryregistry){
registry.addViewController("/").setViewName("login");
registry.addViewController("/index.html").setViewName("login");
registry.addViewController("/main.html").setViewName("dashboard");
}
//注册拦截器
//@Override
//publicvoidaddInterceptors(InterceptorRegistryregistry){
////静态资源；*.css， *.js
////SpringBoot已经做好了静态资源映射
//registry.addInterceptor(newLoginHandlerInterceptor()).addPathPatterns("/
**").
//excludePathPatterns("/index.html","/","/user/login","/asserts/**","/webjars/
**","/resources/**","/hello");
//}
};
returnindex;
}
```

- 2、国际化
  - 1）编写国际化配置文件；
  - 2）使用ResourceBundleMessageSource管理国际化资源文件
  - 3）在页面使用th:text="#{login.username}"取出国际化内容

    效果：根据浏览器语言设置的信息切换了国际化；

    原理：

    国际化Locale（区域信息对象）；LocaleResolver（获取区域信息对象）；
  - 4）点击链接切换国际化

```
publicclassMyLocalResolverimplementsLocaleResolver{
@Override
publicLocaleresolveLocale(HttpServletRequestrequest){
Stringl=request.getParameter("l");
Localelocale=Locale.getDefault();
if(!StringUtils.isEmpty(l)){
String[]split=l.split("_");
locale=newLocale(split[0],split[1]);
}
returnlocale;
}
@Override
publicvoidsetLocale(HttpServletRequesthttpServletRequest,HttpServletRe
sponsehttpServletResponse,Localelocale){
}
```

```
}
```

- 3、登录

  开发期间模板引擎页面修改以后，要实时生效

  - 1）、禁用模板引擎的缓存

    ```
    #禁用模板引擎缓存
    spring.thymeleaf.cache=false
    ```

  - 2）、页面修改完成以后ctrl+f9：重新编译；

    登录错误消息的显示

    ```
    <pstyle="color:red"th:text="${msg}"th:if="${not#strings.isEmpty(msg)}">
    </p>
    ```

- 4、拦截器进行登录检查

  拦截器

  ```
  /*
  *登录检查
  **/
  publicclassLoginHandlerInterceptorimplementsHandlerInterceptor{
  //目标方法执行之前
  @Override
  publicbooleanpreHandle(HttpServletRequestrequest,HttpServletResponse
  response,Objecthandler)throwsException{
  Objectuser=request.getSession().getAttribute("loginUser");
  if(user==null){
  //未登录，返回登录页面
  request.setAttribute("msg","没有权限请先登录");
  request.getRequestDispatcher("index.html").forward(request,response);
  returnfalse;
  }else{
  //已登录
  returntrue;
  }
  }
  @Override
  publicvoidpostHandle(HttpServletRequestrequest,HttpServletResponseres
  ponse,Objecthandler,ModelAndViewmodelAndView)throwsException{


  }
  @Override
  publicvoidafterCompletion(HttpServletRequestrequest,HttpServletRespon
  seresponse,Objecthandler,Exceptionex)throwsException{
  }
  }
  ```

  注册拦截器

  ```
  //注册拦截器
  @Override
  publicvoidaddInterceptors(InterceptorRegistryregistry){
  //静态资源；*.css，*.js
  //SpringBoot已经做好了静态资源映射
  registry.addInterceptor(newLoginHandlerInterceptor()).addPathPatterns("
  /**").
  excludePathPatterns("/index.html","/","/user/login","/asserts/
  **","/webjars/**","/resources/**","/hello");
  ```

```
}
};
```

- 5、CRUD-员工列表

  实验要求

  - 1）RestfulCRUD：CRUD满足Rest风格；

    URI： /资源名称/资源标识 HTTP请求方式区分对资源CRUD操作

普通CRUD（uri来区分操作）	**RestfulCRUD**	
查询	getEmp	emp---GET
添加	addEmp?xxx	emp---POST
修改	updateEmp?id=xxx&xxx=xx	emp/{id}---PUT
删除	deleteEmp?id=1	emp/{id}---DELETE

  - 2）实验的请求架构

实验功能	**请求URI**	**请求方式**
查询所有员工	emps	GET
查询某个员工(来到修改页面)	emp/1	GET
来到添加页面	emp	GET
添加员工	emp	POST
来到修改页面（查出员工进行信息回显）	emp/1	GET
修改员工	emp	PUT
删除员工	emp/1	DELETE

  - 3）员工列表

    thymeleaf公共页面元素抽取

    1、抽取公共片段

```
<div th:fragment="copy">
&copy; 2011 The Good Thymes Virtual Grocery
</div>
```

    2、引入公共片段

```
<div th:insert="~{footer :: copy}"></div>
```

    ~{templatename::selector}：模板名::选择器

    ~{templatename::fragmentname}:模板名::片段名

    3、默认效果：

    insert的公共片段在div标签中

    如果使用th:insert等属性进行引入，可以不用写~{}：

    行内写法可以加上：[[~{}]];[(~{})]；

    三种引入公共片段的th属性：

th:insert：将公共片段整个插入到声明引入的元素中

th:replace：将声明引入的元素替换为公共片段

th:include：将被引入的片段的内容包含进这个标签中

```
<footer th:fragment="copy">
&copy; 2020 The Good Thymes Virtual Grocery
</footer>
```

引入方式

```
<div th:insert="footer :: copy"></div>
<div th:replace="footer :: copy"></div>
<div th:include="footer :: copy"></div>
```

效果

```
<div>
   <footer>
   &copy; 2020 The Good Thymes Virtual Grocery
   </footer>
</div>

<footer>
&copy; 2020 The Good Thymes Virtual Grocery
</footer>

<div>
&copy; 2020 The Good Thymes Virtual Grocery
</div>
```

引入片段的时候传入参数：

```
<nav class="col-md-2 d-none d-md-block bg-light sidebar" id="sidebar">
   <div class="sidebar-sticky">
      <ul class="nav flex-column">
         <li class="nav-item">
            <a class="nav-link active"
               th:class="${activeUri=='main.html'?'nav-link active':'nav-link'}"
               href="#" th:href="@{/main.html}">
                <svg xmlns="http://www.w3.org/2000/svg" width="24"
height="24" viewBox="0 0 24 24" fill="none" stroke="currentColor" stroke-
width="2" stroke-linecap="round" stroke-linejoin="round" class="feather
feather-home">
                   <path d="M3 9l9-7 9 7v11a2 2 0 0 1-2 2H5a2 2 0 0 1-2-2z">
</path>
                   <polyline points="9 22 9 12 15 12 15 22"></polyline>
                </svg>
                Dashboard <span class="sr-only">(current)</span>
            </a>
         </li>

<!--引入侧边栏;传入参数-->
<div th:replace="commons/bar::#sidebar(activeUri='emps')"></div>
```

- 6、员工增加

  添加页面

```
<!--引入抽取的topbar-->
<!--模板名：会使用thymeleaf的前后缀配置规则进行解析-->
<divth:replace="commons/bar::topbar"></div>
<divclass="container-fluid">
```

```
<divclass="row">
<!--引入侧边栏-->
<divth:replace="commons/bar::#sidebar(activeUri='emps')"></div>
<mainrole="main"class="col-md-9ml-sm-autocol-lg-10pt-3px-4">
<!--需要区分是修改还是添加-->
<formth:action="@{/emp}"method="post">
<!--发送put请求修改员工数据-->
<!--
1、SpringMVC中配置HiddenHttpMethodFilter（SpringBoot配置好的）
2、页面创建一个post表单
3、创建一个input项，name="_method";值就是我们指定的请求方式
-->
<inputtype="hidden"name="_method"value="put"th:if="${emp!=null}">
<inputtype="hidden"name="id"th:if="${emp!=null}"th:value="${emp.id}">
<divclass="form-group">
<label>LastName</label>
<inputth:value="${emp!=null}?
${emp.lastName}"name="lastName"type="text"class="form-
control"placeholder="zhangsan">
</div>
<divclass="form-group">
<label>Email</label>
<inputth:value="${emp!=null}?
${emp.email}"name="email"type="email"class="form-
control"placeholder="zhangsan@atguigu.com">
</div>
<divclass="form-group">
<label>Gender</label><br/>
<divclass="form-checkform-check-inline">
<inputclass="form-check-
input"type="radio"name="gender"value="1"th:checked="${emp!=null}?
${emp.gender==1}">
<labelclass="form-check-label">男</label>
</div>
<divclass="form-checkform-check-inline">
<inputclass="form-check-
input"type="radio"name="gender"value="0"th:checked="${emp!=null}?
${emp.gender==0}">
<labelclass="form-check-label">女</label>
</div>
</div>
<divclass="form-group">
<label>department</label>
<selectclass="form-control"name="department.id">
<!--提交部门id-->
<optionth:selected="${emp!=null}?
${dept.id==emp.department.id}"th:value="${dept.id}"th:each="dept:
${depts}"th:text="${dept.departmentName}"name="department.id"></option>
</select>
</div>
<divclass="form-group">
<label>Birth</label>
<inputth:value="${emp!=null}?${#dates.format(emp.birth,'yyyy-MM-
ddhh:MM:ss')}"name="birth"type="text"class="form-
control"placeholder="zhangsan">
```

```
</div>
<buttontype="submit"class="btnbtn-primary"th:text="${emp!=null}?'修改':'添加'">添加</button>
</form>
</main>
</div>
</div>
```

提交的数据格式不对：生日：日期；

2017-12-12；2017/12/12；2017.12.12；

日期的格式化；SpringMVC将页面提交的值需要转换为指定的类型;

2017-12-12---Date； 类型转换，格式化;

默认日期是按照/的方式；

- 7、员工修改

  与员工添加是二合一页面

- 8、员工删除

  删除按钮

```
<buttonth:attr="delete_uri=@{/emp/}+${emp.id}"type="submit"class="btnbtn-smbtn-dangerdeleteBtn">删除</button>
```

绑定表单

```
<formid="deleteEmpForm"method="post">
<inputtype="hidden"name="_method"value="delete">
</form>
```

删除脚本

```
<script>
$(".deleteBtn").click(function(){
//删除当前员工
$("#deleteEmpForm").attr("action",$(this).attr("delete_uri")).submit();
returnfalse;
})
</script>
```