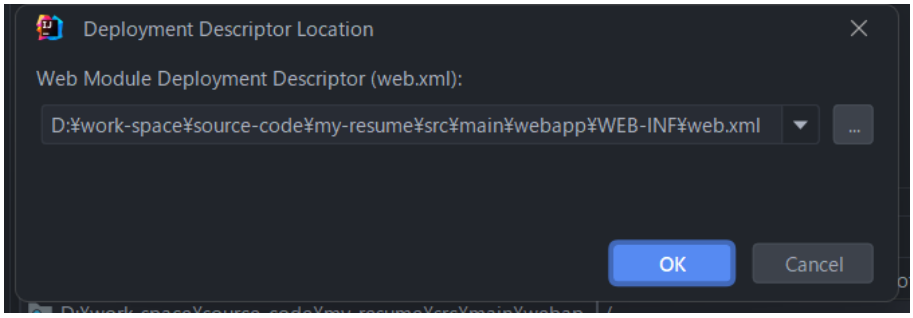
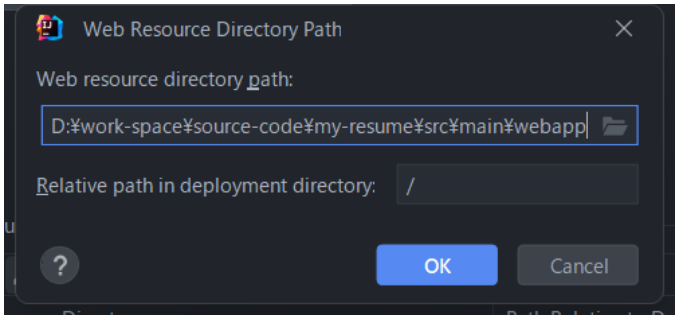
一、创建Web项目

- 通过Idea（或者eclipse）创建普通Maven项目
- 通过对pom.xml进行以下基本的设置，让Idea可以识别当前项目为潜在web项目

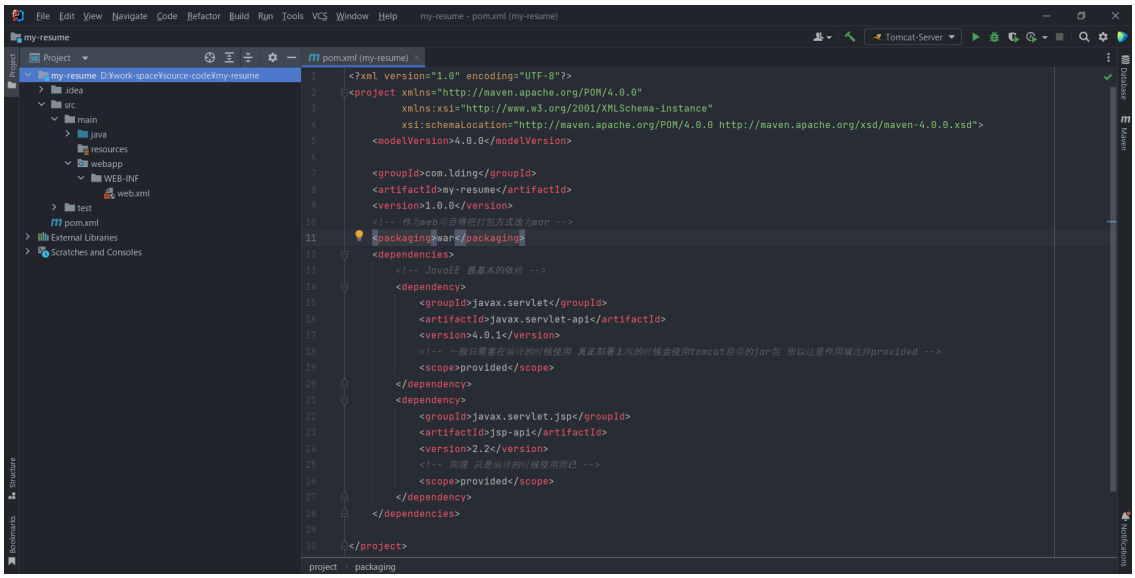
```
<!-- 作为web项目得把打包方式改为war -->
<packaging>war</packaging>
<dependencies>
    <!-- JavaEE 最基本的依赖 -->
    <dependency>
        <groupId>javax.servlet</groupId>
        <artifactId>javax.servlet-api</artifactId>
        <version>4.0.1</version>
        <!-- 一般只需要在编译的时候使用 真正部署上线的时候会使用tomcat自带的jar包 所以这里作用域选择provided -->
        <scope>provided</scope>
    </dependency>
    <dependency>
        <groupId>javax.servlet.jsp</groupId>
        <artifactId>jsp-api</artifactId>
        <version>2.2</version>
        <!-- 同理 只是编译的时候使用而已 -->
        <scope>provided</scope>
    </dependency>
</dependencies>
```

- 通过对「Project Structure > Facets」的web模块进行以下修改（添加WEB-INF\web.xml），让Idea正式识别当前项目为web项目





- 最终得到如下基本结构



二、添加Tomcat服务器

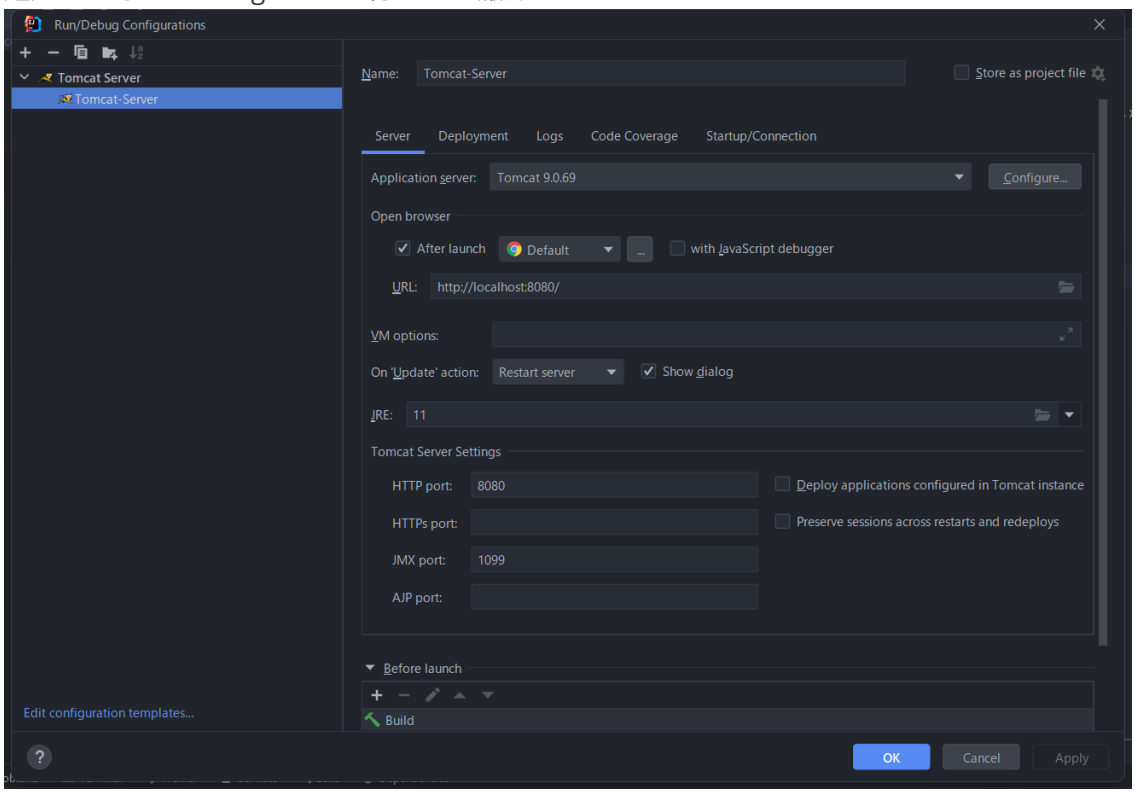
- 在官网下载Tomcat

```
https://tomcat.apache.org/download-90.cgi
```

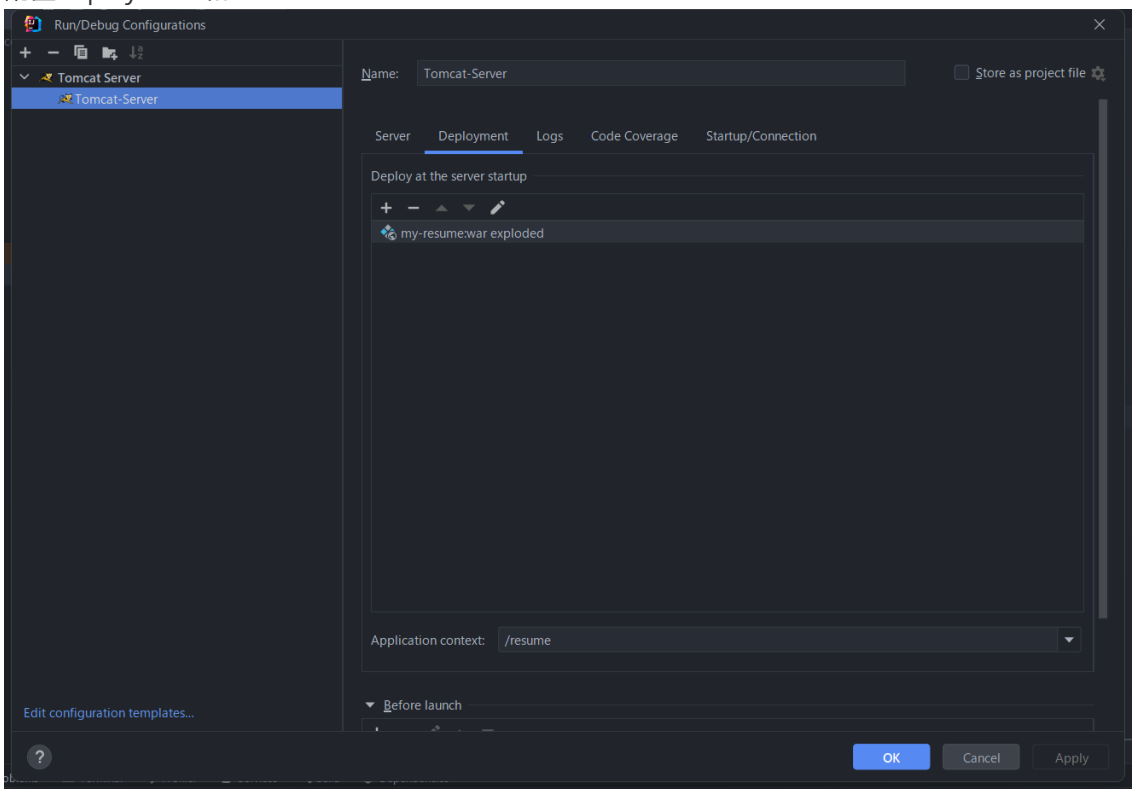
- 解压安装即可，另外要注意：

```
# 由于我是日文系统 为了防止乱码 一般将conf/logging.properties 里面的控制台打印配置从默认的UTF8改为SJIS
# java.util.logging.ConsoleHandler.encoding = UTF-8
java.util.logging.ConsoleHandler.encoding = SJIS
```

- 通过Idea的edit Configurations将tomcat服务

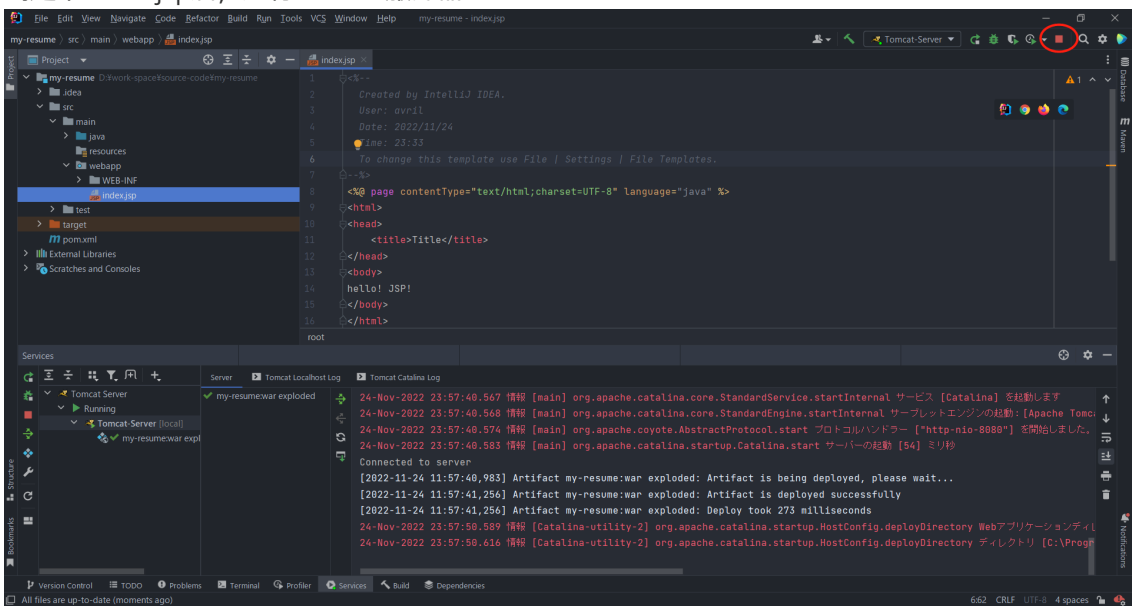


- 配置deployment和context

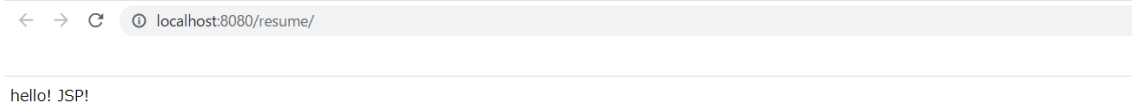


三、开发环境构建完毕

- 创建个index.jsp后，运行Tomcat服务器



- 通过浏览器可以发现项目正常启动了



hello! JSP!

四、引入相关jar包

- 在pom.xml文件中添加以下依赖

```
<dependencies>
    <!-- JavaEE 最基本的依赖 -->
    <dependency>
        <groupId>javax.servlet</groupId>
```

```
        <artifactId>javax.servlet-api</artifactId>
        <version>4.0.1</version>
        <!-- 一般只需要在编译的时候使用 真正部署上线的时候会使用tomcat自带的jar包 所以这里作用域选择provided -->
        <scope>provided</scope>
    </dependency>
    <dependency>
        <groupId>javax.servlet</groupId>
        <artifactId>jstl</artifactId>
        <version>1.2</version>
    </dependency>
    <dependency>
        <groupId>javax.servlet.jsp</groupId>
        <artifactId>jsp-api</artifactId>
        <version>2.2</version>
        <!-- 同理 只是编译的时候使用而已 -->
        <scope>provided</scope>
    </dependency>
    <!-- SpringMVC + Spring -->
    <dependency>
        <!-- SpringMVC会自动依赖Spring的包 -->
        <groupId>org.springframework</groupId>
        <artifactId>spring-webmvc</artifactId>
        <version>5.3.18</version>
    </dependency>
    <!-- 面向切面编程相关 -->
    <dependency>
        <groupId>org.aspectj</groupId>
        <artifactId>aspectjrt</artifactId>
        <version>1.9.9.1</version>
        <scope>runtime</scope>
    </dependency>
    <dependency>
        <groupId>org.aspectj</groupId>
        <artifactId>aspectjweaver</artifactId>
        <version>1.9.9.1</version>
        <scope>runtime</scope>
    </dependency>
    <!-- json相关 -->
    <dependency>
        <groupId>com.fasterxml.jackson.core</groupId>
        <artifactId>jackson-databind</artifactId>
        <version>2.14.0</version>
    </dependency>
    <!-- Mybatis（整合Spring）相关 -->
    <dependency>
        <groupId>org.mybatis</groupId>
        <artifactId>mybatis</artifactId>
        <version>3.5.11</version>
    </dependency>
    <dependency>
        <groupId>org.mybatis</groupId>
        <artifactId>mybatis-spring</artifactId>
        <version>2.0.7</version>
    </dependency>
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-jdbc</artifactId>
```

```
            <version>5.3.18</version>
        </dependency>
        <!-- 连接池和JDBC驱动 -->
        <dependency>
            <groupId>com.alibaba</groupId>
            <artifactId>druid</artifactId>
            <version>1.2.15</version>
        </dependency>
        <dependency>
            <groupId>mysql</groupId>
            <artifactId>mysql-connector-java</artifactId>
            <version>8.0.30</version>
        </dependency>
        <dependency>
            <groupId>org.projectlombok</groupId>
            <artifactId>lombok</artifactId>
            <version>1.18.24</version>
        </dependency>
        <!-- 操作IO流的工具库 -->
        <dependency>
            <groupId>commons-fileupload</groupId>
            <artifactId>commons-fileupload</artifactId>
            <version>1.4</version>
        </dependency>
        <!-- 添加分页 -->
        <dependency>
            <groupId>com.github.pagehelper</groupId>
            <artifactId>pagehelper</artifactId>
            <version>5.3.2</version>
        </dependency>
        <!-- Junit -->
        <dependency>
            <groupId>junit</groupId>
            <artifactId>junit</artifactId>
            <version>4.13.2</version>
            <scope>test</scope>
        </dependency>
    </dependencies>
    <!-- 构建信息 -->
    <build>
        <!-- 打包后的文件名 -->
        <finalName>resume</finalName>
    </build>
```

五、准备好基本的domain（领域模型）、dao、controller以及service

- 这里以项目里面的WebSite为例进行说明

```java
package com.lding.resume.domain;

import lombok.Data;

@Data
public class Website extends BaseDto {
    private String footer;
}
```

```java
package com.lding.resume.domain;

import com.fasterxml.jackson.annotation.JsonIgnore;
import com.fasterxml.jackson.databind.ObjectMapper;
import lombok.Data;
import java.text.SimpleDateFormat;
import java.util.Date;

@Data
public class BaseDto {
    protected Integer id;
    protected Date createdTime;
    // 该注解代表Json字符串的对象外方法
    @JsonIgnore
    public String getJSON() throws Exception {
        ObjectMapper mapper = new ObjectMapper();
        // 指定日期的格式
        mapper.setDateFormat(new SimpleDateFormat("yyyy-MM-dd"));
        return mapper.writeValueAsString(this).replace("\"", "'");
    }
}
```

```java
package com.lding.resume.dao;

import com.lding.resume.domain.Website;
import org.apache.ibatis.annotations.Insert;
import org.apache.ibatis.annotations.Select;
import org.apache.ibatis.annotations.Update;

public interface WebsiteDao {
    @Select("SELECT * FROM website")
    Website get();

    @Insert("INSERT INTO website(footer) VALUES (#{footer})")
    int insert(Website website);

    @Update("UPDATE website SET footer = #{link} WHERE id = #{id}")
    int update(Website website);
}
```

```java
package com.lding.resume.service;

import com.lding.resume.domain.Website;

public interface WebsiteService {
    Website get();

    boolean save(Website website);
}
```

```java
package com.lding.resume.service.impl;

import com.lding.resume.dao.WebsiteDao;
import com.lding.resume.domain.Website;
import com.lding.resume.service.WebsiteService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

@Service
public class WebsiteServiceImpl implements WebsiteService {
    @Autowired
    private WebsiteDao dao;

    @Override
    public Website get() {
        return this.dao.get();
    }

    @Override
    public boolean save(Website website) {
        int count = 0;
        if (website.getId() == null) {
            count = this.dao.insert(website);
        } else {
            count = this.dao.update(website);
        }
        return count > 0;
    }
}
```

```java
package com.lding.resume.controller;

import com.fasterxml.jackson.core.JsonProcessingException;
import com.fasterxml.jackson.databind.ObjectMapper;
import com.lding.resume.domain.Website;
import com.lding.resume.service.WebsiteService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;
import org.springframework.web.servlet.ModelAndView;
import java.util.HashMap;
import java.util.Map;

/**
 * '@RestController
```

```
 * 将 Controller 的方法返回的对象，通过适当的转换器转换为指定的格式之后，写入到HTTP 响应(Response)对象的 body 中，
 * 通常用来返回 JSON 或者 XML 数据，返回 JSON 数据的情况比较多
 */
@RestController
@RequestMapping("/admin")
public class WebsiteController {
    @Autowired
    private WebsiteService websiteService;

    @GetMapping("/website")
    public ModelAndView website() {
        // 设置数据
        ModelAndView view = new ModelAndView();
        view.addObject("website", this.websiteService.get());
        view.setViewName("website");
        return view;
    }

    @PostMapping("/website/save")
    public String save(@RequestBody Website website) throws
JsonProcessingException {
        // 1、当客户端的HTTP请求参数contentType设置为：application/json
        // 2、服务端的处理接口参数使用：@RequestBody注解，就会把客户参数当成一个
javaBean进行耦合字段赋值
        //      简单的说就是，把客户的json对象转换为javaBean对象
        // 设置数据
        Map<String, Object> result = new HashMap<>();
        if (this.websiteService.save(website)) {
            result.put("success", true);
            result.put("message", "网站信息保存成功");
        } else {
            result.put("success", false);
            result.put("message", "网站信息保存失败");
        }
        return new ObjectMapper().writeValueAsString(result);
    }
}
```

```
/* website.jsp 部分代码  */
function save() {
    // 获得url
    const requestUrl = "${ctx}/admin/website/save";
    const responseUrl = "${ctx}/admin/website";
    // 获得表单数据
    const footer = $('#save-form-data #footer').val();
    const id = $('#save-form-data [name="id"]').val();
    const formData = {
        "id": id,
        "footer": footer
    };
    // 通过ajax调用后台
    sendData(requestUrl, JSON.stringify(formData), responseUrl);
}
function sendData(requestUrl, param, responseUrl) {
    $.ajax({
```

```
        method: "post", // 请求方式
        url: requestUrl, // 请求路径
        data: param, // 请求参数
        dataType: "json", // 设置接受到的响应数据的格式
        contentType: "application/json;charset=UTF-8",
        success: function (result) { //响应成功后的回调函数
            if (result.success) {
                document.location.href = responseUrl;
            } else {
                swal({
                    title: "错误",
                    text: result.message,
                    icon: "error",
                    dangerMode: true,
                    buttons: { confirm: "确定" }
                })
            }
        }
    });
}
```

六、准备好转换器和拦截器和ViewResolver（可选）

- 自定义转换器

```
package com.lding.resume.config;

import org.springframework.core.convert.converter.Converter;
import java.text.DateFormat;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.List;

public class DateConverter implements Converter<String, Date> {
    private List<String> formats = null;
    public void setFormats(List<String> formats) {
        this.formats = formats;
    }
    @Override
    public Date convert(String source) {
        for (String format: formats) {
            DateFormat df = new SimpleDateFormat(format);
            try {
                return df.parse(source);
            } catch (ParseException e) {
                // 如果发生异常 执行下一次
            }
        }
        return null;
    }
}
```

- 自定义拦截器

```
package com.lding.resume.config;
```

```
import org.springframework.stereotype.Component;
import org.springframework.web.servlet.HandlerInterceptor;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class MyInterceptor implements HandlerInterceptor {
    @Override
    public boolean preHandle(HttpServletRequest request, HttpServletResponse
response, Object handler) throws Exception {
        boolean result = true;
        // 获取IP地址之后的URL部分（contextPath~）
        String url = request.getRequestURI();
        if (url.contains("admin")) {
            if (request.getSession().getAttribute("user") == null) {
                // 没登录成功过 跳转到登录页面
                response.sendRedirect(request.getContextPath() + "/access");
                // 拦截 阻止后续当前请求后续方法被调用
                result = false;
            }
        }
        return result;
    }
}
```

- 自定义ViewResolver

```
package com.lding.resume.config;

import org.springframework.web.servlet.view.InternalResourceView;
import java.io.File;
import java.util.Locale;

/**
 * SpringMVC自带的InternalResourceViewResolver所存在的问题
 *  1. InternalResourceViewResol本身并没有重写父类AbstractUrlBaseView的
checkResource方法
 *  2. 父类AbstractUrlBaseView的checkResource方法返回值是固定为true，即视为肯定存
在目标文件
 *  3. 一旦通过order将InternalResourceViewResol设为优先级最高的话，无论找到与否，其
他的配置都不起作用了
 */
public class MyView extends InternalResourceView {
    @Override
    public boolean checkResource(Locale locale) throws Exception {
        // 获取完整path
        String path = this.getServletContext().getRealPath("/") +
this.getUrl();
        File file = new File(path);
        // 存在返回true；否在返回false
        return file.exists();
    }
}
```

- 添加所有其他模块的代码

具体请参考本项目的具体代码，地址如下：

七、配置web.xml、dispatcherServlet.xml、applicationContext.xml

- web.xml基本配置

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app 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-app_4_0.xsd"
         version="4.0">
    <!-- 配置SpringMVC自带的DispatcherServlet -->
    <servlet>
        <servlet-name>springmvc</servlet-name>
        <servlet-
class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
        <!-- SpringMVC-子容器配置文件位置 -->
        <init-param>
            <param-name>contextConfigLocation</param-name>
            <param-value>classpath:dispatcherServlet.xml</param-value>
        </init-param>
        <!-- 让项目一旦部署的服务器便创建servlet（只要>=0即可） -->
        <load-on-startup>0</load-on-startup>
    </servlet>
    <servlet-mapping>
        <servlet-name>springmvc</servlet-name>
        <!-- 拦截大多请求（非JSP） -->
        <url-pattern>/</url-pattern>
    </servlet-mapping>
    <!-- 配置父容器 -->
    <context-param>
        <param-name>contextConfigLocation</param-name>
        <param-value>classpath:applicationContext.xml</param-value>
    </context-param>
    <!-- 添加监听器监听父容器 -->
    <listener>
        <listener-
class>org.springframework.web.context.ContextLoaderListener</listener-class>
    </listener>
    <!-- 配置过滤器Filter（解决客户端 -> 后台 乱码问题） -->
    <filter>
        <filter-name>myEncodingFilter</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>
    </filter>
    <filter-mapping>
        <filter-name>myEncodingFilter</filter-name>
        <url-pattern>/*</url-pattern>
    </filter-mapping>
</web-app>
```

- dispatcherServlet.xml的基本配置

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:context="http://www.springframework.org/schema/context"
       xmlns:mvc="http://www.springframework.org/schema/mvc"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
       http://www.springframework.org/schema/beans/spring-beans.xsd
       http://www.springframework.org/schema/context
https://www.springframework.org/schema/context/spring-context.xsd
http://www.springframework.org/schema/mvc
https://www.springframework.org/schema/mvc/spring-mvc.xsd">
    <!-- 扫描需要在IoC容器里面创建的包 -->
    <context:component-scan base-package="com.lding.resume.controller" />
    <mvc:annotation-driven conversion-service="conversionService">
        <!-- 添加message转换器 -->
        <mvc:message-converters>
            <bean
class="org.springframework.http.converter.StringHttpMessageConverter">
                <property name="defaultCharset" value="UTF-8"/>
            </bean>
            <bean
class="org.springframework.http.converter.json.MappingJackson2HttpMessageCon
verter">
                <property name="defaultCharset" value="UTF-8" />
            </bean>
        </mvc:message-converters>
    </mvc:annotation-driven>

    <!-- 将静态资源交给服务器默认Servlet去处理 -->
    <mvc:default-servlet-handler />

    <!-- 注册自定义的转换器（配置factoryBean） -->
    <!-- 固定id="conversionService" 这个不可以变，Spring内部会用到这个id -->
    <bean id="conversionService"

 class="org.springframework.context.support.ConversionServiceFactoryBean">
        <property name="converters">
            <set>
                <!-- 配置自定义转换器 -->
                <bean class="com.lding.resume.config.DateConverter" >
                    <!-- 改为通过传list的形式 -->
                    <property name="formats">
                        <list>
                            <value>yyyy/MM/dd</value>
                            <value>yyyy-MM-dd</value>
                            <value>MM/dd/yyyy</value>
                        </list>
                    </property>
                </bean>
            </set>
        </property>
    </bean>

    <!-- MultipartFile 添加CommonsMultpartResolver到IoC容器-->
```

```
    <bean id="multipartResolver"
class="org.springframework.web.multipart.commons.CommonsMultipartResolver">
        <!-- 解析multipart参数，防止乱码 -->
        <property name="defaultEncoding" value="UTF-8" />
    </bean>

    <!-- 公共前缀和后缀 -->
    <bean
class="org.springframework.web.servlet.view.InternalResourceViewResolver">
        <property name="order" value="1" />
        <property name="prefix" value="/WEB-INF/front/" />
        <property name="suffix" value=".jsp" />
    </bean>
    <bean
class="org.springframework.web.servlet.view.InternalResourceViewResolver">
        <property name="order" value="0" />
        <property name="prefix" value="/WEB-INF/admin/" />
        <property name="suffix" value=".jsp" />
        <!-- 自定义的view 优先级是0 最高 -->
        <property name="viewClass" value="com.lding.resume.config.MyView" />
    </bean>

    <!-- 配置拦截器 -->
    <mvc:interceptors>
        <mvc:interceptor>
            <!-- 需要拦截的路径：**代表当前路径下的所有内容（包括子目录） -->
            <mvc:mapping path="/**"/>
            <!-- 排除asset目录下的所有内容 -->
            <mvc:exclude-mapping path="/asset/**"/>
            <!-- 拦截器对象 -->
            <bean class="com.lding.resume.config.MyInterceptor" />
        </mvc:interceptor>
    </mvc:interceptors>
</beans>
```

- applicationContext的基本配置

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:context="http://www.springframework.org/schema/context"
xmlns:tx="http://www.springframework.org/schema/tx"
       xmlns:aop="http://www.springframework.org/schema/aop"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd
http://www.springframework.org/schema/context
https://www.springframework.org/schema/context/spring-context.xsd
http://www.springframework.org/schema/tx
http://www.springframework.org/schema/tx/spring-tx.xsd
http://www.springframework.org/schema/aop
https://www.springframework.org/schema/aop/spring-aop.xsd">
    <!-- Mybatis和事务管理 -->
    <!-- 扫描需要在IoC容器里面创建的包 -->
    <context:component-scan base-package="com.lding.resume.service" />
    <!-- mybatis基本配置 -->
    <!-- 1. 获取db配置文件 -->
```

```
    <context:property-placeholder location="classpath:db.properties" ignore-
unresolvable="true" />
    <!-- 2. 数据源配置（druid连接池）默认自动提交事务 -->
    <bean id="dataSource" class="com.alibaba.druid.pool.DruidDataSource">
        <property name="driverClassName" value="${druid.driverClassName}" />
        <property name="username" value="${druid.username}" />
        <property name="password" value="${druid.password}" />
        <property name="url" value="${druid.url}" />
        <property name="initialSize" value="${druid.initialSize}" />
        <property name="maxActive" value="${druid.maxActive}" />
        <property name="maxWait" value="${druid.maxWait}" />
    </bean>
    <!-- 3. 配置SqlSessionFactoryBean -->
    <bean id="sqlSessionFactory"
class="org.mybatis.spring.SqlSessionFactoryBean">
        <!-- 3-1. 引用数据源 -->
        <property name="dataSource" ref="dataSource" />
        <!-- 3-2. 导入领域模型所在包 -->
        <property name="typeAliasesPackage" value="com.lding.domain" />
        <!-- 3-3. 添加实体映射文件 -->
        <property name="mapperLocations">
            <array>
                <value>classpath:mappers/*.xml</value>
            </array>
        </property>
        <!-- 3-4. 开启驼峰自动映射 -->
        <property name="configuration">
            <bean class="org.apache.ibatis.session.Configuration">
                <property name="mapUnderscoreToCamelCase" value="true" />
            </bean>
        </property>
    </bean>
    <!-- 4. 配置MapperScannerConfigurer -->
    <!-- 用于扫描Dao -->
    <bean class="org.mybatis.spring.mapper.MapperScannerConfigurer">
        <!-- SqlSessionFactoryBean的id -->
        <property name="sqlSessionFactoryBeanName" value="sqlSessionFactory"
/>
        <!-- dao的包名 -->
        <property name="basePackage" value="com.lding.resume.dao"/>
    </bean>
    <!-- 5. 管理事务 -->
    <!-- 5-1. 配置事务管理器 -->
    <bean id="txManager"
class="org.springframework.jdbc.datasource.DataSourceTransactionManager">
        <!-- 数据源（这里配置的是druid） -->
        <property name="dataSource" ref="dataSource" />
    </bean>
    <tx:annotation-driven transaction-manager="txManager" />
    <!-- 5-2. 添加附加代码（声明式事务管理） -->
    <tx:advice id="txAdvice" transaction-manager="txManager">
        <tx:attributes>
            <!-- Spring会在以下方法前后增加事务管理代码 默认事务的传播行为：REQUIRED
-->
            <tx:method name="list*" timeout="10"/>
        </tx:attributes>
    </tx:advice>
    <!-- 5-3. 添加切面 -->
```

```
    <aop:config proxy-target-class="true">
        <!-- 切入点 -->
        <aop:pointcut id="txPc"
expression="within(com.lding.resume.service..*)" />
        <!-- 附加代码 + 切入点 -->
        <aop:advisor advice-ref="txAdvice" pointcut-ref="txPc" />
    </aop:config>
</beans>
```

```
## db.properties配置
druid.driverClassName=com.mysql.cj.jdbc.Driver
druid.username=root
druid.password=sysapl
druid.url=jdbc:mysql://localhost:3306/resume?
serverTimezone=Asia/Tokyo&useUnicode=true&characterEncoding=UTF8
druid.initialSize=5
druid.maxActive=10
druid.maxWait=5000
```

八、启动MySQL服务

- 启动mysql服务

```
C:\Windows\system32>net start mysql
MySQL サービスを開始します.
MySQL サービスは正常に開始されました。

C:\Windows\system32>
```

- 导入项目相关数据库和表

```
DROP TABLE IF EXISTS award;
DROP TABLE IF EXISTS education;
DROP TABLE IF EXISTS skill;
DROP TABLE IF EXISTS website;
DROP TABLE IF EXISTS experience;
DROP TABLE IF EXISTS project;
DROP TABLE IF EXISTS company;
DROP TABLE IF EXISTS user;
DROP TABLE IF EXISTS contact;

# user
CREATE TABLE user (
    id INT AUTO_INCREMENT,
    created_time DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP,
    password VARCHAR(32) NOT NULL,
    email VARCHAR(50) NOT NULL UNIQUE,
    birthday DATE,
    photo VARCHAR(100),
    intro VARCHAR(1000),
    name VARCHAR(20),
    address VARCHAR(100),
    phone VARCHAR(20),
    job VARCHAR(20),
```

```
    trait VARCHAR(100),
    interests VARCHAR(100),
    PRIMARY KEY (id)
);

# skill
CREATE TABLE skill (
    id INT AUTO_INCREMENT,
    created_time DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP,
    name VARCHAR(20) NOT NULL,
    level INT NOT NULL,
    PRIMARY KEY (id)
);

# website
CREATE TABLE website (
    id INT AUTO_INCREMENT,
    created_time DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP,
    footer VARCHAR(1000),
    PRIMARY KEY (id)
);

# company
CREATE TABLE company (
    id INT AUTO_INCREMENT,
    created_time DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP,
    name VARCHAR(20) NOT NULL UNIQUE,
    logo VARCHAR(100),
    website VARCHAR(50),
    intro VARCHAR(1000),
    PRIMARY KEY (id)
);

# award
CREATE TABLE award (
    id INT AUTO_INCREMENT,
    created_time DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP,
    name VARCHAR(20) NOT NULL,
    image VARCHAR(100),
    intro VARCHAR(1000),
    PRIMARY KEY (id)
);

# contact
CREATE TABLE contact (
    id INT AUTO_INCREMENT,
    created_time DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP,
    name VARCHAR(20) NOT NULL,
    email VARCHAR(50) NOT NULL,
    comment VARCHAR(1000) NOT NULL,
    subject VARCHAR(20),
    already_read INT NOT NULL DEFAULT 0,
    PRIMARY KEY (id)
);

# education
CREATE TABLE education (
    id INT AUTO_INCREMENT,
```

```
    created_time DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP,
    name VARCHAR(20) UNIQUE NOT NULL,
    type INT NOT NULL,
    intro VARCHAR(1000),
    begin_day DATE NOT NULL,
    end_day DATE,
    PRIMARY KEY (id)
);

# experience
CREATE TABLE experience (
    id INT AUTO_INCREMENT,
    created_time DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP,
    job VARCHAR(20) NOT NULL,
    intro VARCHAR(1000),
    begin_day DATE NOT NULL,
    end_day DATE,
    company_id INT NOT NULL,
    PRIMARY KEY (id),
    FOREIGN KEY (company_id) REFERENCES company(id)
);

# project
CREATE TABLE project (
    id INT AUTO_INCREMENT,
    created_time DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP,
    name VARCHAR(20) NOT NULL,
    intro VARCHAR(1000),
    website VARCHAR(50),
    image VARCHAR(100),
    begin_day DATE NOT NULL,
    end_day DATE,
    company_id INT NOT NULL,
    PRIMARY KEY (id),
    FOREIGN KEY (company_id) REFERENCES company(id)
);

# 初始化
INSERT INTO user(email, name, password,
    job, phone, birthday,
    address, trait, interests,
    intro) VALUES(
    'avril@qq.com', 'avril', '3f05d332600fa9d9b7837172521ffa60',
    '程序员', '9527', '1988-01-02',
    '天朝广州', '活泼,可爱', '足球,台球,电玩',
    '本人学识渊博、经验丰富，代码风骚、效率恐怖，C/C++ C#、Java、PHP、Android、iOS、Python、JavaScript，无不精通玩转，熟练掌握各种框架，并自写语言，创操作系统，写CPU处理器构架，做指令集成。深山苦练20余年，一天只睡3小时，千里之外定位问题，瞬息之间修复上线。身体强壮、健步如飞，可连续工作100小时不休息，讨论技术方案9小时不喝水，上至研发CPU芯片、带项目、出方案、弄计划，下至盗账号、黑网站、Shell提权挂马、攻击同行、拍片摄影、泡妞把妹纸、开挖掘机、威胁PM，啥都能干。'
);

INSERT INTO website(footer) VALUES(
    '<a href="https://space.bilibili.com/325538782"
target="_blank">avril</a> © All Rights Reserved 2020'
);
```

九、如何将html文件改造为jsp？（2步即可

- 步骤1：复制以下代码到html文件的最顶部

```
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
```

- 步骤2：将.html后缀改为.jsp后缀

（一）知识拓展：JSP-简介

- Java Server Pages的简称，是一种动态网页技术标准，本质也是Servlet
- 通过指令来配置JSP页面，导入资源文件

```
<%@ 指令名称 属性名1=属性值1 属性名2=属性值2 ... %>
```

指令名称	作用	用例
page	配置当前页面信息	<%@ page contentType="text/html;charset=UTF-8" language="java" %>
include	包含其他页面	<%@ include file="common/header.jsp" %>
taglib	导入标签库	<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>

- 符号 <%-- --%> 可以注释所有内容（当然，原有的html注释符号依然可以使用
- 9大内置对象：在jsp页面中不需要创建，直接使用的对象

变量名	真实类型	作用
pageContext	PageContext	当前页面共享数据，还可以获取其他八个内置对象
request	HttpServletRequest	一次请求访问的多个资源(转发)
session	HttpSession	一次会话的多个请求间
application	ServletContext	所有用户间共享数据
response	HttpServletResponse	响应对象
page	Object	当前页面(Servlet)的对象 ；this
out	JspWriter	输出对象，数据输出到页面上
config	ServletConfig	Servlet的配置对象
exception	Throwable	异常对象

- 补充说明：ServlteContext

```
一个ServlteContext对象就代表一个Web应用，可以用来与Web容器（Tomcat）通信
1. 获取ServlteContext
☛ request.getServletContext() / servlet.getServletContext()
2. 常用方法
☛ getMIMEType
   getRealPath    注意：根目录是虚拟的根目录，是相对于项目的根目录"/"相当于是webroot目录
   setAttribute / getAttribute / removeAttribute
```

（二）知识拓展：JSP-page指令详解

- contentType：等同于response.setContentType()

```
1. 设置响应体的mime类型以及字符集
2. 设置当前jsp页面的编码（只能是高级的IDE才能生效，如果使用低级工具，则需要设置pageEncoding属性设置当前页面的字符集）
```

- import：导包
- errorPage：当前页面发生异常后，会自动跳转到指定的错误页面
- isErrorPage：标识当前也是是否是错误页面

```
1. true：是，可以使用内置对象exception
2. false：否（默认值），不可以使用内置对象exception
```

```jsp
<%@ page import="java.util.ArrayList" %>
<%@ page import="java.util.List" %>
<%@ page contentType="text/html;charset=gbk" errorPage="500.jsp"
    pageEncoding="GBK" language="java" buffer="16kb" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<html>
  <head>
    <title>$Title$</title>
  </head>
  <body>
  <%
    List list = new ArrayList();
    int i = 3/0; // 发生异常，自动跳转到指定的错误页面（500.jsp）
  %>
  </body>
</html>
```

```
<%@ page contentType="text/html;charset=UTF-8" isErrorPage="true"
language="java" %>
<html>
<head>
    <title>Title</title>
</head>
<body>
    <h1>服务器正忙...</h1>
    <%
        String message = exception.getMessage();
        out.print(message);
    %>
</body>
</html>
```

（三）知识拓展：JSP-taglib指令详解

- prefix：前缀，名称任意；使用的时候通过该前缀去匹配上

```
<%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt" %>
```

```
<!-- 用法举例 -->
<input type="date" value="<fmt:formatDate
value="${contactListResult.endDay}" pattern="yyyy-MM-dd"></fmt:formatDate>"
name="endDay" class="form-control" placeholder="结束日期" >
```

（四）知识拓展：JSP-其他常用方法

- 嵌入Java代码： <% "Java代码" %>
- 声明： <%! "声明成员变量、方法" %>
- 输出： <%= "需要输出的内容" %>

（五）知识拓展：JSP-EL表达式

- Expression Language 表达式语言：替换和简化jsp页面中java代码的编写

```
语法：${表达式}
☛ jsp默认支持el表达式的。如果要忽略el表达式
   1. 设置jsp中page指令中：isELIgnored="true" 忽略当前jsp页面中所有的el表达式
   2. \${表达式} ：忽略当前这个el表达式，本质上是让字符转义
```

- 简单运算

```
1. 算数运算符： + - * /(div) %(mod)
2. 比较运算符： > < >= <= == !=
3. 逻辑运算符： &&(and) ||(or) !(not)
4. 空运算符： empty
   ☛ 功能：用于判断字符串、集合、数组对象是否为null或者长度是否为 0
      ${empty list}:判断字符串、集合、数组对象是否为null或者长度为 0
      ${not empty str}:表示判断字符串、集合、数组对象是否不为null 并且 长度 > 0
```

```
<%@ page import="java.util.List" %>
<%@ page import="java.util.ArrayList" %>
```

```jsp
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
<head>
    <title>Title</title>
</head>
<body>
<hr>
    <h3>算数运算符</h3>
    ${3 + 4}<br>            <!-- 7 -->
    ${3 / 4}<br>            <!-- 0.75 -->
    ${3 div 4}<br>          <!-- 0.75 -->
    ${3 % 4}<br>            <!-- 3 -->
    ${3 mod 4}<br>          <!-- 3 -->
    <h3>比较运算符</h3>
    ${3 == 4}<br>           <!-- false -->
    <h3>逻辑运算符</h3>
    ${3 > 4 && 3 < 4}<br>   <!-- false -->
    ${3 > 4 and 3 < 4}<br>  <!-- false -->
    <h4>empty运算符</h4>
    <%
    String str = "";
    request.setAttribute("str",str); // 空字符串

    List list = new ArrayList();
    request.setAttribute("list",list); // 长度为0的动态数组
    %>
    ${not empty str}        <!-- false -->
    ${not empty list}       <!-- false -->
</body>
</html>
```

- 获取值：el表达式只能从域对象中获取值

```
1. ${域名称.键名}：从指定域中获取指定键的值
   ☛ 域名称（优先顺位）：
      1-1. pageScope        --> pageContext
      1-2. requestScope     --> request
      1-3. sessionScope     --> session
      1-4. applicationScope --> application（ServletContext）
2. ${键名}：表示依次从最小的域中查找是否有该键对应的值，直到找到为止
3. 获取对象、List集合、Map集合的值
   ☛ 3-1. 对象：${域名称.键名.属性名}
          ☛ 本质上会去调用对象的getter方法
      3-2. List集合：${域名称.键名[索引]}
      3-3. Map集合：${域名称.键名.key名称} 或者 ${域名称.键名["key名称"]}
```

```jsp
<%@ page import="java.util.List" %>
<%@ page import="java.util.ArrayList" %>
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
<head>
    <title>el获取域中的数据</title>
</head>
<body>
<%
//在域中存储数据
```

```
session.setAttribute("name", "李四");
request.setAttribute("name", "张三");
session.setAttribute("age", "23");
request.setAttribute("str", "");
%>
<h3>el获取值</h3>
${requestScope.name}    <!-- 张三 -->
${sessionScope.age}     <!-- 23 -->
${sessionScope.haha}    <!-- 由于不存在该属性，所以是空字符串 -->

<!-- 从pageContext到applicationScope的顺序 依次查找name属性  -->
${name}                 <!-- 张三 -->
${sessionScope.name}    <!-- 李四 -->
</body>
</html>
```

```
<%@ page import="cn.itcast.domain.User" %>
<%@ page import="java.util.*" %>
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
<head>
    <title>el获取数据</title>
</head>
<body>
    <%  // Java代码
        User user = new User();
        user.setName("张三");
        user.setAge(23);
        user.setBirthday(new Date());
        request.setAttribute("u", user);

        List list = new ArrayList();
        list.add("aaa");
        list.add("bbb");
        list.add(user);
        request.setAttribute("list", list);

        Map map = new HashMap();
        map.put("sname", "李四");
        map.put("gender", "男");
        map.put("user", user);

        request.setAttribute("map", map);
    %>
    <h3>el获取对象中的值</h3>
        ${requestScope.u}<br>
    <%--
        通过的是对象的属性来获取
            setter或getter方法，去掉set或get，在将剩余部分，首字母变为小写。
            setName --> Name --> name
    --%>
        <!-- 标准写法，通过request域先获取对象再获取属性 -->
        ${requestScope.u.name}<br>  <!-- 张三 -->
        ${u.age}<br>                <!-- 23 -->
        ${u.birthday}<br>           <!-- Tue Jun 12 15:27:54 CST 2019 -->
        ${u.birthday.month}<br>     <!-- 5 -->
```

```
    <h3>el获取List值</h3>
        ${list}<br>                 <!-- [aaa, bbb] -->
        ${list[0]}<br>              <!-- [aaa] -->
        ${list[1]}<br>              <!-- [bbb] -->
        ${list[10]}<br>             <!-- 下标越界，显示空字符串 -->
        ${list[2].name}             <!-- 张三 -->

    <h3>el获取Map值</h3>
        ${map.gender}<br>           <!-- 男 -->
        ${map["gender"]}<br>        <!-- 男 -->
        ${map.user.name}            <!-- 张三 -->
</body>
</html>
```

- 隐式对象：el表达式中有11个隐式对象

```
pageContext：获取jsp其他八个内置对象
             ☛ ${pageContext.request.contextPath}：动态获取虚拟目录
```

```
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
<head>
    <title>el隐式对象</title>
</head>
<body>
    ${pageContext.request}<br>
    <h4>在jsp页面动态获取虚拟目录</h4>
    ${pageContext.request.contextPath}
</body>
</html>
```

（六）知识拓展：JSP-JSTL标签库

- JSP标准标签库，JSP Standard Tag Library的简称

```
由Apache的Jakarta小组维护，也是用于简化和替换jsp页面上的java代码
```

```
下载 JSTL 核心标签库
☛下载地址：http://tomcat.apache.org/download-taglibs.cgi
  核心标签库，只需要下载一下2个jar包即可
  taglibs-standard-impl-1.2.5.jar、taglibs-standard-spec-1.2.5.jar
```

- 使用步骤

```
1. 导入jstl相关jar包
2. 通过taglib指令来引入标签库：  <%@ taglib %>
3. 使用标签
```

- 常用的JSTL标签

1. if: 相当于java代码的if语句
   1-1. 属性：
        ☛ test 必须属性，接受boolean表达式
                如果表达式为true，则显示if标签体内容，如果为false，则不显示标签体内容
                一般情况下，test属性值会结合el表达式一起使用
   1-2. 注意：
        c:if标签没有else情况，想要else情况，则可以再定义一个c:if标签

```
<%@ page import="java.util.List" %>
<%@ page import="java.util.ArrayList" %>
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<html>
<head>
    <title>if标签</title>
</head>
<body>
    <c:if test="true">
        <h1>我是真...</h1>
    </c:if>
    <br>
    <%
        //判断request域中的一个list集合是否为空，如果不为null则显示遍历集合
        List list = new ArrayList();
        list.add("aaaa");
        request.setAttribute("list", list);
        request.setAttribute("number", 4);
    %>
    <c:if test="${not empty list}">
        遍历集合...
    </c:if>
    <br>

    <c:if test="${number % 2 != 0}">
            ${number}为奇数
    </c:if>

    <c:if test="${number % 2 == 0}">
        ${number}为偶数
    </c:if>
</body>
</html>
```

---

2. choose: 相当于java代码的switch语句
   2-1. 使用choose标签声明   相当于switch声明
   2-2. 使用when标签做判断   相当于case
   2-3. 使用otherwise标签做其他情况的声明   相当于default

```
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<%@taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<html>
<head>
    <title>choose标签</title>
</head>
```

```
<body>
    <%
        request.setAttribute("number",51);
    %>
    <c:choose>
        <c:when test="${number == 1}">星期一</c:when>
        <c:when test="${number == 2}">星期二</c:when>
        <c:when test="${number == 3}">星期三</c:when>
        <c:when test="${number == 4}">星期四</c:when>
        <c:when test="${number == 5}">星期五</c:when>
        <c:when test="${number == 6}">星期六</c:when>
        <c:when test="${number == 7}">星期天</c:when>
        <c:otherwise>数字输入有误</c:otherwise>        <!-- 数字输入有误 -->
    </c:choose>
</body>
</html>
```

---

3. foreach: 相当于java代码的for语句

```
<%@ page import="java.util.ArrayList" %>
<%@ page import="java.util.List" %>
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<%@taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<html>
<head>
    <title>foreach标签</title>
</head>
<body>
<%--
    foreach:相当于java代码的for语句
        1. 完成重复的操作
            for(int i = 0; i < 10; i ++){

            }
            * 属性：
                begin：开始值
                end：结束值
                var：临时变量
                step：步长
                varStatus:循环状态对象
                index:容器中元素的索引，从0开始
                count:循环次数，从1开始
        2. 遍历容器
            List<User> list;
            for(User user : list){

            }
            * 属性：
                items:容器对象
                var:容器中元素的临时变量
                varStatus:循环状态对象
                index:容器中元素的索引，从0开始
                count:循环次数，从1开始
--%>
    <c:forEach begin="1" end="10" var="i" step="2" varStatus="s">
```

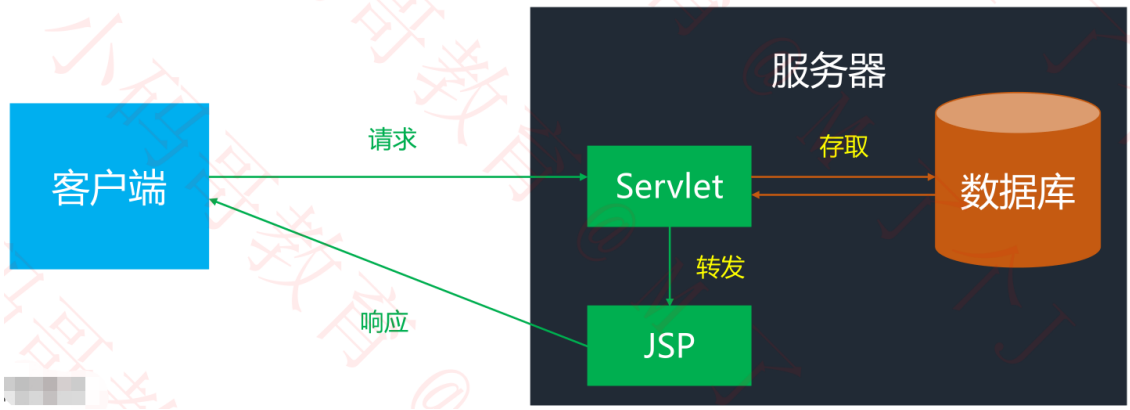
```
        ${i} <h3>${s.index}<h3> <h4> ${s.count} </h4><br>
    </c:forEach>
    <hr>
    <%
        List list = new ArrayList();
        list.add("aaa");
        list.add("bbb");
        list.add("ccc");
        request.setAttribute("list",list);
    %>
    <c:forEach items="${list}" var="str" varStatus="s">
        ${s.index} ${s.count} ${str}<br>
    </c:forEach>
</body>
</html>
```

（七）知识拓展：MVC开发模式

- jsp演变历史

```
1. 早期只有servlet，只能使用response输出标签数据，非常麻烦
2. 后来又jsp，简化了Servlet的开发，如果过度使用jsp，在jsp中即写大量的java代码，有写html表，造成难于维护，难于分工协作
3. 再后来，java的web开发，借鉴mvc开发模式，使得程序的设计更加合理性
```

- MVC开发模式




```
1. M：Model，数据模型。JavaBean
    ☛ 完成具体的业务操作，如：查询数据库，封装对象
2. V：View，视图。JSP
    ☛ 展示数据
3. C：Controller，控制器。Servlet
    ☛ 获取用户的输入
       调用模型
       将数据交给视图进行展示
```

（八）知识拓展：三层架构（软件设计架构

- 简单总结如下

```
1. 界面层(表示层)：用户看的得界面。用户可以通过界面上的组件和服务器进行交互
2. 业务逻辑层：处理业务逻辑的。
3. 数据访问层：操作数据存储文件。
```

十、拓展：文件上传

- 依赖jar包（另外还有前面已经添加的commons-fileupload）

```
<!-- 操作IO流的工具库 -->
<dependency>
    <groupId>commons-io</groupId>
    <artifactId>commons-io</artifactId>
    <version>2.6</version>
</dependency>
```

- 制作相关的工具类

```
package com.lding.resume.config;

import lombok.Data;

@Data
public class ImageUploader {
    private String basePath;
    private String imagePath;
    // 获得完整图片的路径
    public String getImageFullPath() {
        return this.basePath + this.imagePath;
    }
}
```

```
<!-- applicationContext.xml中的项目个别对应 -->
<!-- 1. 获取图片配置文件并将bean加入容器 -->
<context:property-placeholder location="classpath:fileUpload.properties"
ignore-unresolvable="true"/>
<bean id="imageUploader" class="com.lding.resume.config.ImageUploader">
    <property name="basePath" value="${fileUpload.basePath}" />
    <property name="imagePath" value="${fileUpload.imagePath}" />
</bean>
```

```
# fileUpload.properties
fileUpload.basePath=D:/work-space/tempFile
fileUpload.imagePath=/user-img/
```

```
<!-- dispatcheContext.xml中的项目个别对应 -->
<!-- 2. 配置静态资源 -->
<mvc:resources mapping="/user-img/**" location="file:///D:/work-
space/tempFile/user-img/" />
```

```
package com.lding.resume.controller;

import com.lding.resume.config.ImageUploader;
import com.lding.resume.domain.User;
import com.lding.resume.service.UserService;
import org.apache.commons.io.FileUtils;
import org.apache.commons.io.FilenameUtils;
```

```java
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;
import org.springframework.web.multipart.MultipartFile;
import org.springframework.web.servlet.ModelAndView;
import java.io.File;
import java.io.IOException;
import java.util.UUID;

@RestController
@RequestMapping("/admin")
public class UserController extends BaseController {
    @Autowired
    private ImageUploader imageUploader;
    @Autowired
    private UserService userService;

    @GetMapping("/user")
    public ModelAndView user() {
        // 设置数据
        ModelAndView view = new ModelAndView();
        view.addObject("user", this.userService.get());
        view.setViewName("user");
        return view;
    }

    @PostMapping("/user/save")
    public ModelAndView saveUser(User user, MultipartFile photoFile) throws
IOException {
        // 设置数据
        ModelAndView view = new ModelAndView();
        user.setPhoto(uploadImage(photoFile, user));
        /* 在viewName前面加上"forward:"或"redirect:" 可以排除
InternalResourceViewResolver的影响 */
        if (this.userService.save(user)) {
            view.setViewName("redirect:/admin/user");
        } else {
            view.setViewName("forward:/WEB-INF/error.jsp");
        }
        return view;
    }

    private String uploadImage(MultipartFile photoFile, User user) throws
IOException {
        // 如果FileItem为空 或者空串的话 直接返回oldDbPath
        if (photoFile == null || photoFile.getInputStream().available() ==
0) return user.getPhoto();
        // 获取文件拓展名
        String extension =
FilenameUtils.getExtension(photoFile.getOriginalFilename());
        // 获取路径和文件名
        String fileName = UUID.randomUUID() + "." + extension;
        String path = imageUploader.getImageFullPath() + fileName;
        String dbPath = imageUploader.getImagePath() + fileName;

        // 删除旧的文件
        String oldPhoto = user.getPhoto();
        if (oldPhoto != null && oldPhoto.trim().length() != 0) {
```

```
            FileUtils.deleteQuietly(new File(imageUploader.getBasePath() +
oldPhoto));
        }

        File file = new File(path);
        // 创建好目标文件所在的父目录
        FileUtils.forceMkdirParent(file);
        // 将文件数据写到目标文件
        FileUtils.copyInputStreamToFile(photoFile.getInputStream(), file);

        return dbPath;
    }
}
```

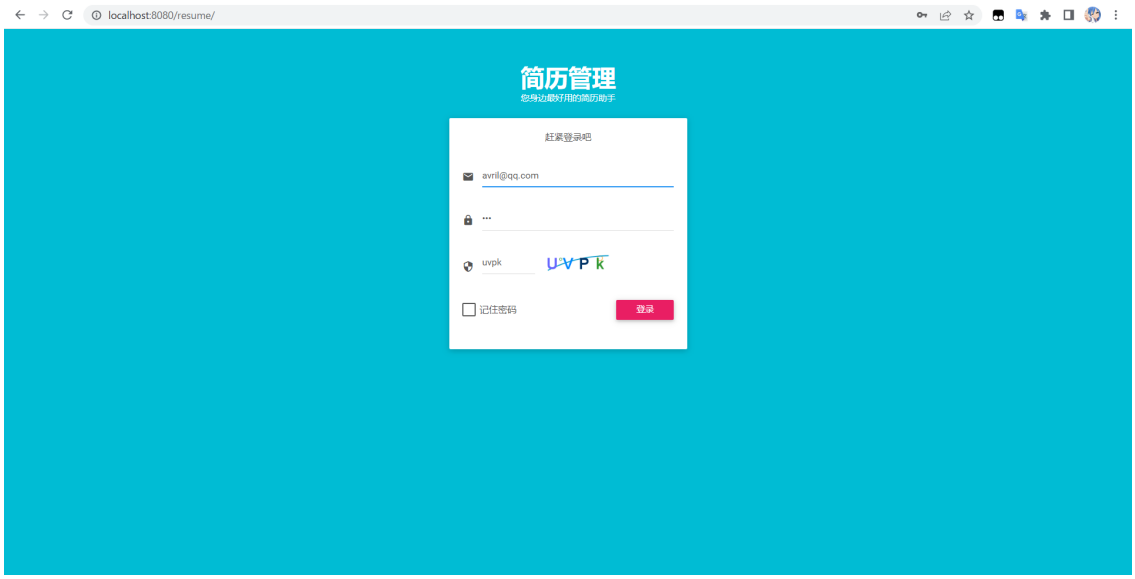
```
<!-- 通过form发送 -->
<form action="${ctx}/admin/user/save" class="form-validation" method="post"
enctype="multipart/form-data">
    <div class="row">
        <input type="hidden" name="id" value="${user.id}">
        <div class="col-lg-2 col-md-2 col-sm-3 col-xs-3 form-control-label">
            <label>头像</label>
        </div>
        <div class="col-lg-10 col-md-10 col-sm-9 col-xs-9">
            <div class="form-group">
                <div class="fileinput fileinput-new" data-
provides="fileinput">
                    <input type="text" style="display: none" name="photo"
value="${user.photo}" />
                    <div class="fileinput-new thumbnail">
                        <c:choose>
                            <c:when test="${empty user.photo}">
                                <img
src="${ctx}/asset/admin/img/noimage.png" alt="">
                            </c:when>
                            <c:otherwise>
                                <img src="${ctx}/${user.photo}" alt="">
                            </c:otherwise>
                        </c:choose>
                    </div>
                    <div class="fileinput-preview fileinput-exists
thumbnail"></div>
                    <i class="material-icons clear fileinput-exists" data-
dismiss="fileinput">close</i>
                    <input type="file" name="photoFile" accept="image/*">
                </div>
            </div>
        </div>
    </div>
    <!-- 省略大段无关项目的代码 -->
</form>
```

拓展：ajax + FormData版

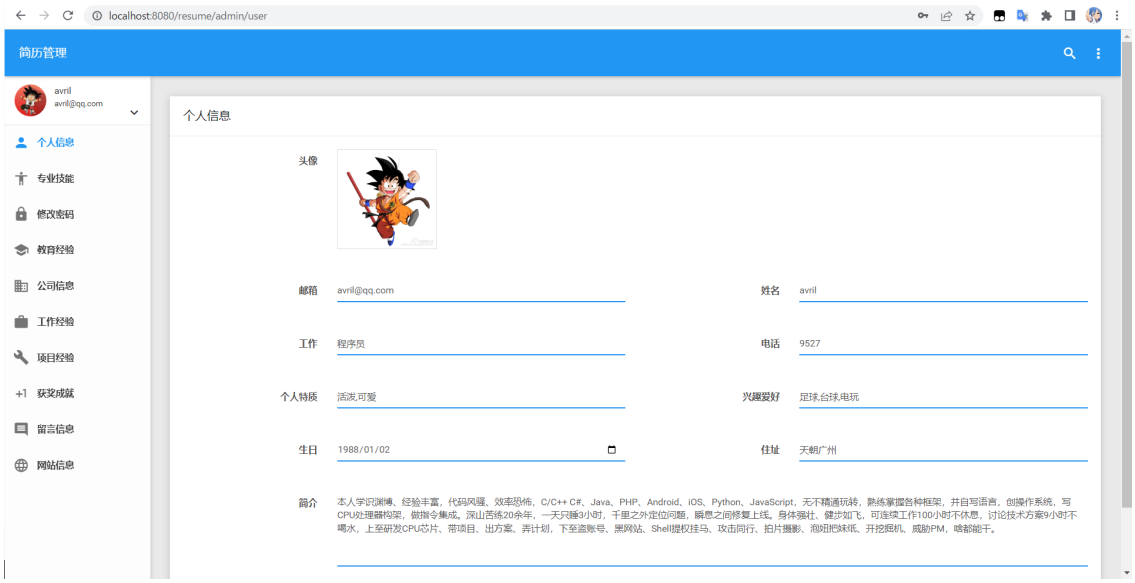
- ajax + FormData版请参考award模块相关代码

十一、本地测试疏通

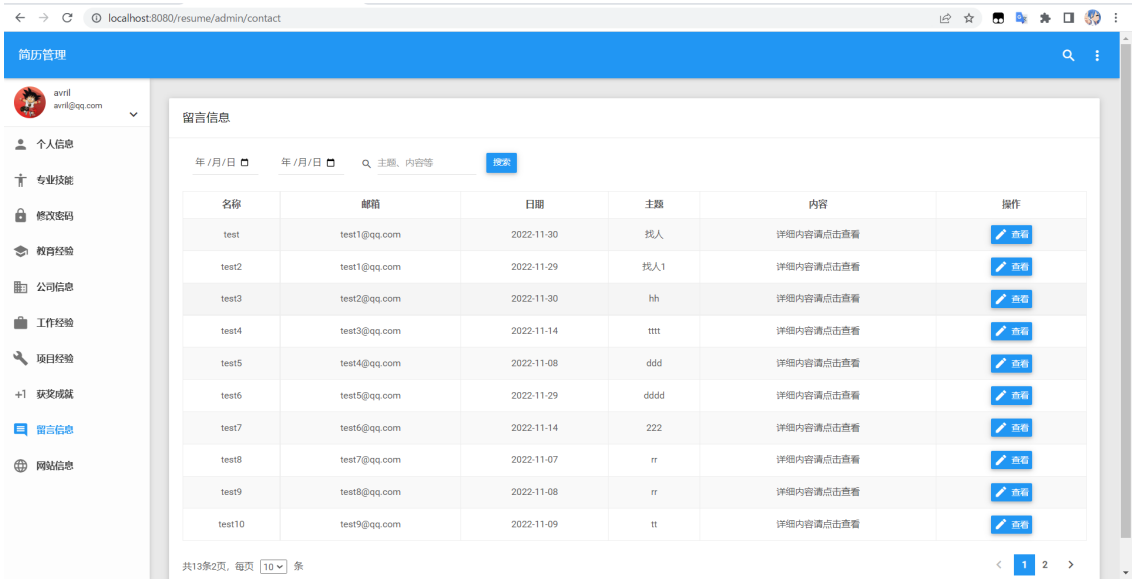
- 首页（login画面



- 后台管理-用户界面



- 后台管理-联系我吧



十二、线上发布

- 可以用虚拟机充当服务器来模拟线上发布

- 安装mysql
- 安装tomcat
- 部署发布
- 线上测试