学习成果（知识梳理）

最初的spring

刚开始接触spring，学习到xml配置，刚开始并不是难以理解，标签属性，都接触蛮多了，如最开始的Hello，以及排序

```
<!--通过bean标签创建一个bean对象，名字叫‘hello’，单例
    <bean id="hello" class="com.spring.hello.Hello"/>-->
    <!--通过bean标签创建一个bean对象,名字叫"bubbleSort"-->
    <bean id="bubbleSort"
class="com.spring.bubble_sort.BubbleSort"/>
```

再往后就是，实体类的交集了，如student和phone

```
<bean id="phone1" class="com.soft1851.spring.Phone">
        <constructor-arg name="brand" value="iPhone11"/>
        <constructor-arg name="price" value="8888.8"/>
    </bean>
    <bean id="phone2" class="com.soft1851.spring.Phone">
    <property name="brand" value="iPhoneX"/>
    <property name="price" value="6666.6"/>
    </bean>
    <bean id="phone3" class="com.soft1851.spring.Phone">
        <constructor-arg name="brand" value="iPhoneXmax"/>
        <constructor-arg name="price" value="1111.1"/>
    </bean>

    <bean id="student1"
class="com.soft1851.spring.Student">
        <property name="name" value="王锋"/>
        <property name="phone" ref="phone1"/>
    </bean>
    <bean id="student2"
class="com.soft1851.spring.Student">
        <constructor-arg name="name" value="颜子浩"/>
        <constructor-arg name="phone" ref="phone2"/>
    </bean>
    <bean id="student3"
class="com.soft1851.spring.Student">
        <property name="name" value="刘恋"/>
```

```
        <property name="phone" ref="phone3"/>
    </bean>
```

就这样进一步的学习，了解并手动搭建了maven

后来的spring

开始了容器的学习，学习了如何建多模块项目

首先是aop（依赖注入：setter注入和构造器注入）

利用依赖关系注入的方式，实现对象之间的解耦。

这里，学习了账号密码验证

测试类

```
ApplicationContext ac = new
ClassPathXmlApplicationContext("beans.xml");
        UserLogin userLogin = (UserLogin)
ac.getBean("userLogin");
        User user = (User) ac.getBean("user");
        user.setAccount("admin");
        user.setPassword("1234");
```

xml配置

```
<bean id="userLogin"
class="com.soft1851.ioc.entity.UserLogin">
     <property name="user" ref="user" />
 </bean>
    <bean id="user" class="com.soft1851.ioc.entity.User"
/>
```

其次是orm

对象关系映射,是一种程序技术,用于实现面向对象编程语言里不同类型系统的数据之间的转换.从效果上说,它其实是创建了一个可在编程语言里使用的--"虚拟对象数据库". ORM的方法论基于三个核心原则:

- 简单:以最基本的形式建模数据
- 传达性:数据库结构被任何人都能理解的语言文档化
- 精确性:基于数据模型创建正确标准化的结构

数据库xml配置

```
<context:component-scan base-
package="com.soft1851.orm.config"/>
    <context:property-placeholder
location="db.properties"/>
    <bean id="dataSource"
class="com.alibaba.druid.pool.DruidDataSource" init-
method="init">
        <property name="driverClassName"
value="${jdbc.driverClassName}"/>
        <property name="url" value="${jdbc.url}"/>
        <property name="username"
value="${jdbc.username}"/>
        <property name="password"
value="${jdbc.password}"/>
        <property name="initialSize" value="8"/>
    </bean>
    <bean id="jdbcTemplate"
class="org.springframework.jdbc.core.JdbcTemplate">
        <property name="dataSource" ref="dataSource"/>
    </bean>
```

自定义日志

```
<?xml version="1.0" encoding="UTF-8" ?>
<configuration>
    <!-- 设置日志输出根目录 -->
    <property name="log.dir" value="logs" />
    <property name="encoding" value="UTF-8" />
    <property name="pattern"
              value="%d{yyyy-MM-dd.HH:mm:ss} %-5level
[%thread] %logger.%M:%L  %msg%n" />

    <!-- 控制台输出日志 -->
    <appender name="STDOUT"
class="ch.qos.logback.core.ConsoleAppender">
        <layout
class="ch.qos.logback.classic.PatternLayout">
            <pattern>${pattern}</pattern>
        </layout>
    </appender>
    <!-- 主日志 -->
    <appender name="FILE"

class="ch.qos.logback.core.rolling.RollingFileAppender">
        <File>main.log</File>
        <filter
class="ch.qos.logback.classic.filter.ThresholdFilter">
```

```
            <level>INFO</level>
        </filter>
        <file>${log.dir}/main.log</file>
        <prudent>false</prudent>
        <encoder charset="UTF-8">
            <pattern>${pattern}</pattern>
        </encoder>
        <rollingPolicy
class="ch.qos.logback.core.rolling.TimeBasedRollingPolicy"
>
            <fileNamePattern>${log.dir}/main.%d{yyyy-MM-
dd-HH}.log.gz</fileNamePattern>
        </rollingPolicy>
    </appender>
    <!-- 错误日志 -->
    <appender name="ERROR_FILE"

class="ch.qos.logback.core.rolling.RollingFileAppender">
        <File>error.log</File>
        <filter
class="ch.qos.logback.classic.filter.ThresholdFilter">
            <level>ERROR</level>
        </filter>
        <file>${log.dir}/error.log</file>
        <prudent>false</prudent>
        <encoder charset="UTF-8">
            <pattern>${pattern}</pattern>
        </encoder>
        <rollingPolicy
class="ch.qos.logback.core.rolling.TimeBasedRollingPolicy"
>
            <fileNamePattern>${log.dir}/error.%d{yyyy-MM-
dd-HH}.log.gz</fileNamePattern>
        </rollingPolicy>
    </appender>

    <logger name="com.soft1851.spring.com.soft1851.ioc">
        <level value="INFO" />
        <appender-ref ref="STDOUT" />
        <appender-ref ref="FILE" />
        <appender-ref ref="ERROR_FILE" />
    </logger>
</configuration>
```

注解配置

如:

```
package com.soft1851.orm.config;

import com.alibaba.druid.pool.DruidDataSource;
import com.soft1851.orm.dao.ForumDao;
import com.soft1851.orm.dao.impl.ForumDaoImpl;
import org.springframework.context.annotation.Bean;
import
org.springframework.context.annotation.ComponentScan;
import
org.springframework.context.annotation.Configuration;
import org.springframework.jdbc.core.JdbcTemplate;

@Configuration
@ComponentScan("com.soft1851.orm")
public class JDbcConfig {
    @Bean
    public DruidDataSource dataSource() {
        DruidDataSource source = new DruidDataSource();

source.setDriverClassName("com.mysql.jdbc.Driver");

source.setUrl("jdbc:mysql://localhost:3306/db_java_basic?
useUnicode=true&characterEncoding=utf8&useSSL=false&autoRe
connect=true");
        source.setUsername("root");
        source.setPassword("123456");
        //配置初始化大小、最小、最大
        source.setInitialSize(8);
        source.setMinIdle(1);
        source.setMaxActive(20);
        //配置获取连接等待超时的时间
        source.setMaxWait(60000);
        //配置间隔多久进行一次检测，检测需要关闭的空间连接，单位是毫
秒
        source.setTimeBetweenEvictionRunsMillis(60000);
        //配置一个连接在池中最小生成的时间，单位是毫秒
        source.setMinEvictableIdleTimeMillis(300000);
        //禁止自动提交，实现事务管理
        source.setDefaultAutoCommit(false);
        //设置连接池启用预处理事务集
        source.setPoolPreparedStatements(true);
        return source;
    }

    @Bean
    public JdbcTemplate jdbcTemplate(DruidDataSource
dataSource){
        return new JdbcTemplate(dataSource);
```

```
    }

    @Bean
    public ForumDao forumDao(JdbcTemplate jdbcTemplate) {
        return new ForumDaoImpl(jdbcTemplate);
    }
}
```

通过dao层实现数据库命令，再由测试类实现，还有就是通过xml配置传参

然后就是web模块

学习了爬虫、引用模块、service层事务回滚、controller层，util层

部分代码如下

```
    <script
src="https://cdn.jsdelivr.net/npm/popper.js@1.16.0/dist/um
d/popper.min.js" integrity="sha384-
Q6E9RHvbIyZFJoft+2mJbHaEWldlvI9IOYy5n3zV9zzTtmI3UksdQRVvox
MfooAo" crossorigin="anonymous"></script>
```

```
@Transactional(rollbackFor = Exception.class)
//事务回滚
```

```
@Controller
@RequestMapping(value = "/topic")
public class TopicController {
    @Resource
    private TopicService topicService;

    @GetMapping(value = "",produces =
"application/json;charset=utf-8")
    @ResponseBody
    public String getTopics(){
        ResponseObject rs = new
ResponseObject(1,"success",topicService.queryAll());
        return JSONObject.toJSONString(rs);
    }

    @GetMapping("/")
    public String topic(Model model){
        model.addAttribute("topics",
topicService.queryAll());
```

```
        return "templates/topic";
    }
}
```

```
@Data
@AllArgsConstructor
@NoArgsConstructor
public class ResponseObject {
    private Integer code;
    private String msg;
    private Object data;
}
```

mybatis

使用xml配置的方式实现了数据库的功能、类型别名及一些配置、mapper层，部分代码如下：

```
<configuration>
    <!--属性-->
    <!--    <properties></properties>-->
    <!--设置-->
    <settings>
        <!--设置启用数据库字段下划线映射到java对象的驼峰式命名属性，默认为false-->
        <setting name="mapUnderscoreToCamelCase" value="true"/>
    </settings>
    <!--类型命名-->
    <typeAliases>
        <typeAlias type="com.soft1851.spring.mybatis.entity.Forum" alias="Forum"/>
        <typeAlias type="com.soft1851.spring.mybatis.entity.Teacher" alias="Teacher"/>
        <typeAlias type="com.soft1851.spring.mybatis.entity.Clazz" alias="Clazz"/>
        <typeAlias type="com.soft1851.spring.mybatis.entity.Student" alias="Student"/>
        <typeAlias type="com.soft1851.spring.mybatis.entity.Course" alias="Course"/>
```

```
        <typeAlias
type="com.soft1851.spring.mybatis.entity.MatterDto"
alias="MatterDto"/>
        <typeAlias
type="com.soft1851.spring.mybatis.entity.ClazzVo"
alias="ClazzVo"/>
        <typeAlias
type="com.soft1851.spring.mybatis.entity.MatterVo"
alias="MatterVo"/>
        <typeAlias
type="com.soft1851.spring.mybatis.entity.CourseStudent"
alias="CourseStudent"/>
    </typeAliases>
    类型处理器
        <typeHandlers></typeHandlers>
    对象工厂
        <objectFactory type=""/>
    插件
        <plugins>
            <plugin interceptor=""></plugin>
        </plugins>
    配置环境
        <environments default="">
    环境变量
            <environment id="">
    事务管理器
                <transactionManager type="">
</transactionManager>
    数据源
                <dataSource type="">
                </dataSource>
            </environment>
        </environments>
    数据库厂商标识
        <databaseIdProvider type=""/>
    映射器
        <mapperss></mappers>
</configuration>
```

```
 <!-- 在springIOC容器中创建mybatis核心类sqlSessionFactor -->
    <bean id="sqlSessionFactory"

class="org.mybatis.spring.SqlSessionFactoryBean">
        <!-- 需要 dataSource -->
        <property name="dataSource" ref="dataSource"/>
        <!-- 引入mybatis配置文件 -->
```

```
        <property name="configLocation"
value="classpath:mybatis-config.xml"/>
        <!--指定实体类所在包-->
        <property name="typeAliasesPackage"
value="com.soft1851.spring.mybatis.entity" />
        <!-- 自动扫描mapping.xml文件 -->
        <property name="mapperLocations"
value="classpath:mappers/*.xml"/>
    </bean>

    <!-- 通过Mapper扫描器MapperScannerConfigurer，批量将
basePackage指定包中的接口全部生成Mapper动态代理对象 -->
    <bean
class="org.mybatis.spring.mapper.MapperScannerConfigurer">
        <property name="basePackage"
value="com.soft1851.spring.mybatis.mapper"/>
        <property name="sqlSessionFactoryBeanName"
value="sqlSessionFactory">
        </property>
    </bean>

    <!--事务管理器配置 -->
    <bean id="manager"
class="org.springframework.jdbc.datasource.DataSourceTrans
actionManager"
          p:dataSource-ref="dataSource"/>

    <!-- 使用声明式事务 -->
    <tx:annotation-driven transaction-manager="manager"/>
```

```
<mapper
namespace="com.soft1851.spring.mybatis.mapper.TeacherMappe
r">
    <resultMap type="Teacher" id="teacherMap">
        <id property="teacherId" column="teacher_id"/>
        <result property="teacherName"
column="teacher_name"/>
        <association property="clazz" javaType="Clazz">
            <id property="clazzId" column="clazz_id"/>
            <result property="clazzName"
column="clazz_name"/>
        </association>
    </resultMap>
    <select id="getTeacherOneToOne" resultMap="teacherMap"
parameterType="int">
```

```
        SELECT t.teacher_id , t.teacher_name , c.clazz_id
,c.clazz_name
        From t_teacher t,
             t_clazz c
        WHERE t.teacher_id = ${teacherId}
          AND t.clazz_id = c.clazz_id

    </select>
</mapper>
```

aop模块

AOP是Spring框架面向切面的编程思想，AOP采用一种称为“横切”的技术，将涉及多业务流程的通用功能抽取并单独封装，形成独立的切面，在合适的时机将这些切面横向切入到业务流程指定的位置中。

使用到的xml依赖

```
<properties>
        <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
        <project.reporting.outputEncoding>UTF-
8</project.reporting.outputEncoding>
        <java.version>1.11</java.version>

<maven.compiler.source>1.11</maven.compiler.source>

<maven.compiler.target>1.11</maven.compiler.target>
        <spring.version>5.2.4.RELEASE</spring.version>
        <mysql.version>5.1.48</mysql.version>
        <lombok.version>1.18.10</lombok.version>
        <junit.version>4.12</junit.version>
        <logback.version>1.2.3</logback.version>
        <slf4j.version>1.7.30</slf4j.version>
        <druid.version>1.1.21</druid.version>
        <jackson.version>2.10.3</jackson.version>
        <http.version>4.4.10</http.version>
    <httpClient.version>4.5.6</httpClient.version>
    <jsoup.version>1.10.2</jsoup.version>
        <mybatis.version>3.5.4</mybatis.version>
        <spring-mybatis.version>2.0.4</spring-
mybatis.version>
    </properties>

    <dependencies>
        <!-- Spring上下文模块,级联引入了aop、beans、core、
expression-->
```

```
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-context</artifactId>
            <version>${spring.version}</version>
        </dependency>


        <!--JUnit，级联引入了hamcrest -->
        <dependency>
            <groupId>junit</groupId>
            <artifactId>junit</artifactId>
            <version>4.12</version>
            <scope>test</scope>
        </dependency>
        <!--Spring-test模块依赖-->
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-test</artifactId>
            <version>${spring.version}</version>
        </dependency>

        <!--  日志依赖：logback和slf4j-->
        <dependency>
            <groupId>ch.qos.logback</groupId>
            <artifactId>logback-classic</artifactId>
            <version>${logback.version}</version>
        </dependency>
        <dependency>
            <groupId>org.slf4j</groupId>
            <artifactId>slf4j-api</artifactId>
            <version>${slf4j.version}</version>
        </dependency>

        <!-- spring-web-mvc -->
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-web</artifactId>
            <version>${spring.version}</version>
        </dependency>

        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-webmvc</artifactId>
            <version>${spring.version}</version>
        </dependency>


        <!--Spring JDBC，级联引入了spring-tx事务依赖-->
```

```
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-jdbc</artifactId>
            <version>${spring.version}</version>
        </dependency>
        <!--MySQL依赖-->
        <dependency>
            <groupId>mysql</groupId>
            <artifactId>mysql-connector-java</artifactId>
            <version>${mysql.version}</version>
            <scope>runtime</scope>
        </dependency>

        <!-- lombok依赖-->
        <dependency>
            <groupId>org.projectlombok</groupId>
            <artifactId>lombok</artifactId>
            <version>${lombok.version}</version>
            <optional>true</optional>
        </dependency>

        <!-- druid数据库连接池依赖 -->
        <dependency>
            <groupId>com.alibaba</groupId>
            <artifactId>druid</artifactId>
            <version>${druid.version}</version>
        </dependency>

        <dependency>
            <groupId>com.alibaba</groupId>
            <artifactId>fastjson</artifactId>
            <version>1.2.66</version>
        </dependency>

        <dependency>

<groupId>com.fasterxml.jackson.dataformat</groupId>
            <artifactId>jackson-dataformat-
xml</artifactId>
            <version>2.8.5</version>
        </dependency>

        <dependency>
            <groupId>javax.servlet</groupId>
            <artifactId>javax.servlet-api</artifactId>
            <version>3.1.0</version>
            <scope>provided</scope>
        </dependency>
```

```
        <dependency>
            <groupId>javax.servlet.jsp.jstl</groupId>
            <artifactId>jstl</artifactId>
            <version>1.2</version>
        </dependency>

        <dependency>
            <groupId>javax.servlet.jsp</groupId>
            <artifactId>jsp-api</artifactId>
            <version>2.2</version>
            <scope>provided</scope>
        </dependency>

        <dependency>
            <groupId>com.fasterxml.jackson.core</groupId>
            <artifactId>jackson-annotations</artifactId>
            <version>${jackson.version}</version>
        </dependency>

        <dependency>
            <groupId>com.fasterxml.jackson.core</groupId>
            <artifactId>jackson-databind</artifactId>
            <version>${jackson.version}</version>
        </dependency>

        <!--
https://mvnrepository.com/artifact/org.thymeleaf/thymeleaf
-spring5 -->
        <dependency>
            <groupId>org.thymeleaf</groupId>
            <artifactId>thymeleaf-spring5</artifactId>
            <version>3.0.11.RELEASE</version>
        </dependency>

        <!--jsoup爬虫-->
        <dependency>
            <groupId>org.jsoup</groupId>
            <artifactId>jsoup</artifactId>
            <version>${jsoup.version}</version>
        </dependency>

        <!--httpclient依赖-->
        <dependency>
            <groupId>org.apache.httpcomponents</groupId>
            <artifactId>httpcore</artifactId>
            <version>${http.version}</version>
        </dependency>
```

```
        <!--
https://mvnrepository.com/artifact/org.apache.httpcomponen
ts/httpclient -->
        <dependency>
            <groupId>org.apache.httpcomponents</groupId>
            <artifactId>httpclient</artifactId>
            <version>${httpClient.version}</version>
        </dependency>

        <!--MyBatis依赖-->
        <dependency>
            <groupId>org.mybatis</groupId>
            <artifactId>mybatis</artifactId>
            <version>${mybatis.version}</version>
        </dependency>
        <!--Spring整合Mybatis依赖-->
        <dependency>
            <groupId>org.mybatis</groupId>
            <artifactId>mybatis-spring</artifactId>
            <version>${spring-mybatis.version}</version>
        </dependency>
        <!--切面配置所需的aop和aspects依赖-->
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-aop</artifactId>
            <version>${spring.version}</version>
        </dependency>
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-aspects</artifactId>
            <version>${spring.version}</version>
        </dependency>

    </dependencies>
```