day01_框架概述、Mybatis快速入门、自定义Mybatis

一.框架概述

问题一：什么是框架?

框架（Framework）是整个或部分系统的可重用设计，表现为一组抽象构件以及构件实例间交互的方法；另一种定义认为，框架是可被应用开发者定制的应用骨架。前者是从应用方面给出的定义，而后者是从目的方面给出的定义。简而言之，框架其实就是某种应用的半成品，就是一组组件供你选用来完成自己的系统。框架中封装了很多的细节，使开发者可以使用极简的方式实现功能，大大提高开发效率。

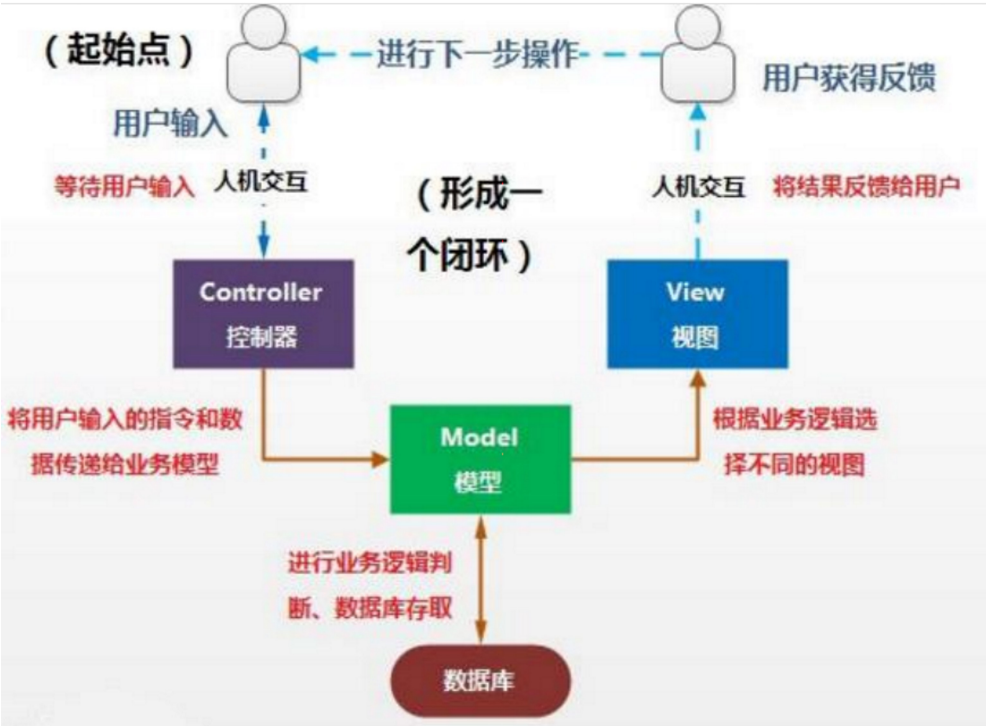
问题二：框架要解决的问题是什么?

框架要解决的最重要的一个问题是技术整合的问题，在 J2EE 的 框架中，有着各种各样的技术，不同的软件企业需要从 J2EE 中选择不同的技术，这就使得软件企业最终的应用依赖于这些技术，技术自身的复杂性和技术的风险性将会直接对应用造成冲击。而应用是软件企业的核心，是竞争力的关键所在，因此应该将应用自身的设计和具体的实现技术解耦。这样，软件企业的研发将集中在应用的设计上，而不是具体的技术实现，技术实现是应用的底层支撑，它不应该直接对应用产生影响。

框架一般处在低层应用平台（如 J2EE）和高层业务逻辑之间的中间层。

(1).软件开发的分层架构

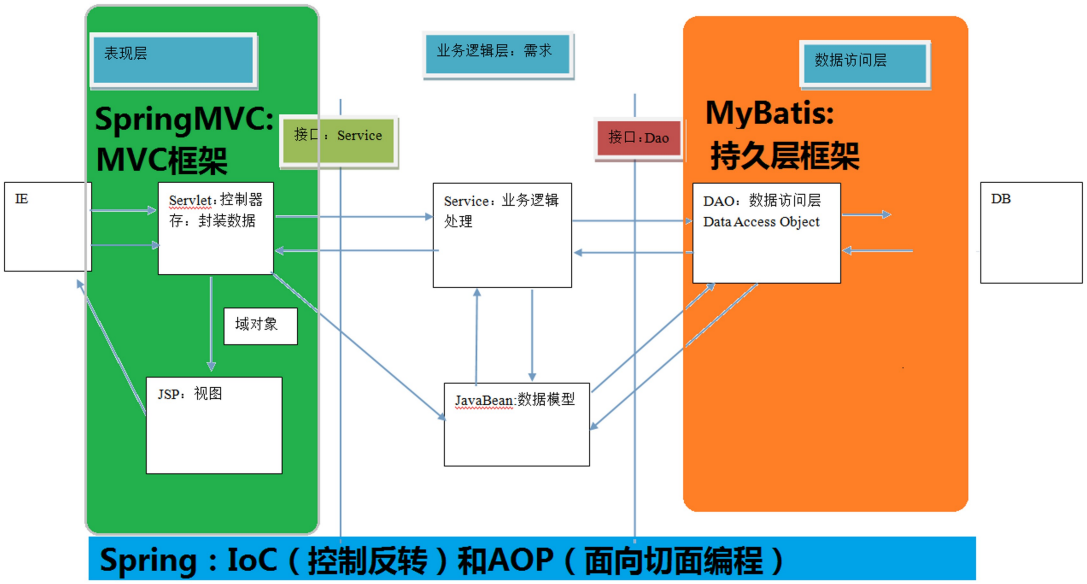
框架的重要性在于它实现了部分功能，并且能够很好的将低层应用平台和高层业务逻辑进行了缓和。为了实现软件工程中的“高内聚、低耦合”。把问题划分开来各个解决，易于控制，易于延展，易于分配资源。我们常见的 MVC 软件设计思想就是很好的分层思想。



- 表现层(View)：是用于展示数据的
- 业务层(Controller)：是处理业务需求
- 持久层(Model)：是和数据库交互的

通过分层更好的实现了各个部分的职责，在每一层将再细化出不同的框架，分别解决各层关注的问题。

(2).分层开发下的常见框架



a.解决数据的持久化问题的框架 Mybatis：

MyBatis 本是apache的一个开源项目iBatis，2010年这个项目由apache software foundation 迁移到了google code，并且改名为MyBatis 。2013年11月迁移到Github。

iBATIS一词来源于“internet”和“abatis”的组合，是一个基于Java的持久层框架。iBATIS提供的持久层框架包括SQL Maps和Data Access Objects (DAOs)

b.解决WEB层问题的MVC框架 Spring MVC：

Spring MVC属于SpringFrameWork的后续产品，已经融合在Spring Web Flow里面。Spring框架提供了构建Web应用程序的全功能MVC模块。使用Spring可插入的MVC架构，从而在使用Spring进行WEB开发时，可以选择使用Spring的Spring MVC框架或集成其他MVC开发框架，如Struts1(现在一般不用)、Struts2(一般老项目使用)等等。

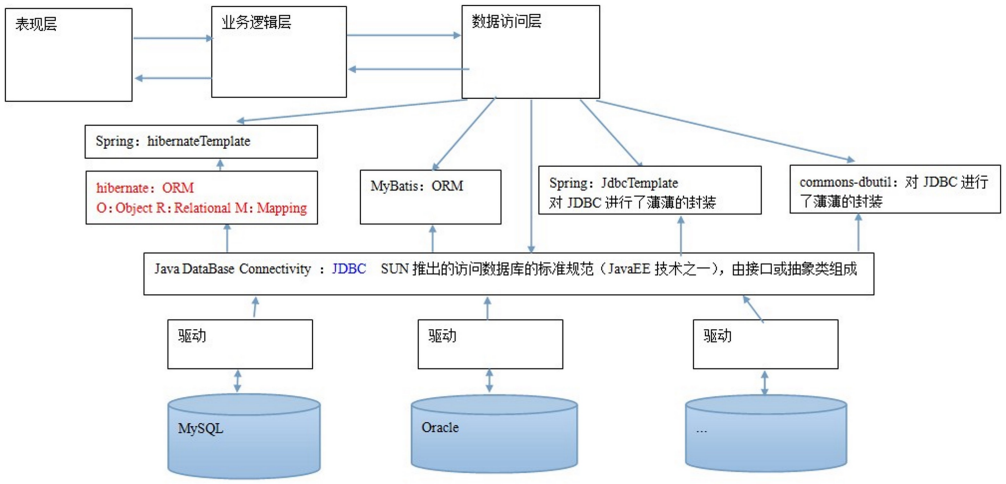
c.解决技术整合问题的框架 Spring：

Spring框架是由于软件开发的复杂性而创建的。Spring使用的是基本的JavaBean来完成以前只可能由EJB完成的事情。然而，Spring的用途不仅仅限于服务器端的开发。从简单性、可测试性和松耦合性角度而言，绝大部分Java应用都可以从Spring中受益。

Spring是一个轻量级的控制反转(IOC)和面向切面(AOP)的容器框架。

1、JDBC编程的分析

(1).持久层技术解决方案



- JDBC技术：Connection、PreparedStatement、ResultSet
- Spring的JdbcTemplate：Spring中对jdbc的简单封装
- Apache的DBUtils：它和Spring的JdbcTemplate很像，也是对Jdbc的简单封装

以上这些都不是框架，JDBC是规范、Spring的JdbcTemplate和Apache的DBUtils都只是工具类

(2).JDBC程序回顾与问题分析

a.数据库链接创建、释放频繁造成系统资源浪费从而影响系统性能，如果使用数据库链接池可解决此问题。

b.sql语句在代码中硬编码，造成代码不易维护，实际应用sql变化的可能较大，sql变动需要改变java代码。

c.使用preparedStatement向占有位符号传参数存在硬编码，因为sql语句的where条件不一定，可能多也可能少，修改sql还要修改代码，系统不易维护。

d.对结果集解析存在硬编码（查询列名），sql变化导致解析代码变化，系统不易维护，如果能将数据库记录封装成pojo对象解析比较方便。

2、Mybatis框架概述

mybatis是一个优秀的基于java的持久层框架，它内部封装了jdbc，使开发者只需要关注sql语句本身，而不需要花费精力去处理加载驱动、创建连接、创建statement等繁杂的过程。

mybatis通过xml或注解的方式将要执行的各种statement配置起来，并通过java对象和statement中sql 的动态参数进行映射生成最终执行的sql语句，最后由mybatis框架执行sql并将结果映射为java对象并返回。

采用ORM思想解决了实体和数据库映射的问题，对jdbc进行了封装，屏蔽了jdbc api底层访问细节，使我们不用与jdbc api打交道，就可以完成对数据库的持久化操作。

ORM：Object Relational Mappging 对象关系映射

简单的说：

就是把数据库表和实体类及实体类的属性对应起来，让我们可以操作实体类就实现操作数据库表。

```
user          User
id            userId
user_name     userName
```

今天我们需要做到：实体类中的属性和数据库表的字段名称保持一致。

```
user          User
id            id
user_name     user_name
```

二.Mybatis框架快速入门

1、Mybatis的环境搭建

(1).创建maven工程，并在pom.xml中添加 Mybatis、mysql、junit、log4j的坐标

- src\main下：java包用来存放源代码，resources包用来存放配置文件
- src\test下：java包用来存放测试代码，resources包用来存放测试配置文件

(2).创建User实体类，实现Serializable序列化接口

```
public class User implements Serializable{

    private Integer id;
    private String username;
    private Date birthday;
    private String sex;
    private String address;
```

- 在 src\main\java\com\itheima\domain 路径下创建（实体类一般存放在 domain 包下）
- User类中的属性与数据库user表中的列名一一对应，且名称完全一致

(3).创建持久层接口 IUserDao

```
public interface IUserDao {

    /** 查询所有操作 ...*/
    List<User> findAll();
}
```

- 在 src\main\java\com\itheima\dao 路径下创建（持久层接口一般存放在 dao 包下，并且domain包、dao包尽量在同一目录下）
- 持久层接口IUserDao与实体类User一一对应，可以在接口中定义用来操作该实体类(数据库user表)的抽象方法

(4).创建持久层接口的映射配置文件 IUserDao.xml

```
<mapper namespace="com.itheima.dao.IUserDao">
    <!--配置查询所有-->
    <select id="findAll" resultType="com.itheima.domain.User">
        select * from user
    </select>
</mapper>
```

- 在 src\main\resources\com\itheima\dao 路径下创建（映射配置文件IUserDao.xml的位置，必须和持久层接口IUserDao.java的包结构相同）
- 映射配置文件 IUserDao.xml 与持久层接口 IUserDao.java 一一对应，可以在其中配置该接口中的抽象方法所要执行的sql语句
- 映射配置文件的mapper标签：namespace属性的取值必须是IUserDao接口的全限定类名
- 映射配置文件的操作配置select标签：id属性的取值必须是IUserDao接口的方法名、resultType属性必须是实体类User的全限定类名(用来告知Mybatis要将sql的查询结果封装到哪个实体类的对象中)

#.注意：创建 IUserDao.xml、IUserDao.java时，使用此文件名称是为了和之前的知识保持一致。在Mybatis中，把持久层的接口名称和映射配置文件叫做：Mapper。所以 IUserDao.xml、IUserDao.java也可以命名为：IUserMapper.xml、IUserMapper.java

(5).创建Mybatis的主配置文件 SqlMapConfig.xml

```
<configuration>
    <!-- 配置环境 -->
    <environments default="mysql"...>

    <!-- 指定映射配置文件的位置，映射配置文件指的是每个dao独立的配置文件 -->
    <mappers>
        <mapper resource="com/itheima/dao/IUserDao.xml"/>
    </mappers>
</configuration>
```

- 在 src\main\resources 路径下创建（主配置文件的名称可以自定义），用来：配置数据库连接信息、指定映射配置文件的位置
- <environments>标签：配置数据库环境信息
- <mappers>标签：使用resource属性来指定映射配置文件的位置

2、Mybatis的入门案例

在 src\test\java\com\itheima\test 路径下创建 MybatisTest 测试类，在测试类的main方法中编写测试代码。（此处也可使用 junit 单元测试框架来执行测试代码，无需编写main方法）

(1).读取Mybatis的主配置文件 SqlMapConfig.xml

```
InputStream in = Resources.getResourceAsStream("SqlMapConfig.xml");
```

(2).创建SqlSessionFactory工厂

```
SqlSessionFactoryBuilder builder = new SqlSessionFactoryBuilder();
SqlSessionFactory factory = builder.build(in);
```

(3).使用工厂生产SqlSession对象

```
SqlSession session = factory.openSession();
```

(4).使用SqlSession对象创建IUserDao接口的代理对象

```
IUserDao userDao = session.getMapper(IUserDao.class);
```

(5).使用代理对象执行IUserDao接口中的方法

```
List<User> users = userDao.findAll();
for(User user : users){
    System.out.println(user);
}
```

(6).释放资源

```
session.close();
in.close();
```

3、Mybatis注解开发的入门案例

Mybatis注解开发的过程与上述入门案例的步骤基本一致，做出以下改变即可：

(1).把 IUserDao.xml 文件移除，在IUserDao接口的方法上使用 @Select("") 注解，并且指定SQL语句

```
public interface IUserDao {

    /** 查询所有操作 ...*/
    @Select("select * from user")
    List<User> findAll();
}
```

(2).同时需要在SqlMapConfig.xml中的mapper配置时，使用class属性指定IUserDao接口的全限定类名

```
<configuration>
    <!-- 配置环境 -->
    <environments default="mysql"...>

    <!-- 指定映射配置文件的位置，映射配置文件指的是每个dao独立的配置文件
         如果是用注解来配置的话，此处应该使用class属性指定被注解的dao全限定类名
    -->
    <mappers>
        <mapper class="com.itheima.dao.IUserDao"/>
    </mappers>
</configuration>
```

4、Mybatis编写dao实现类的入门案例

不管使用XML还是注解配置，Mybatis都是支持写dao实现类的。我们在实际开发中，都是越简便越好，所以都是采用不写dao实现类的方式。

三.自定义Mybatis框架