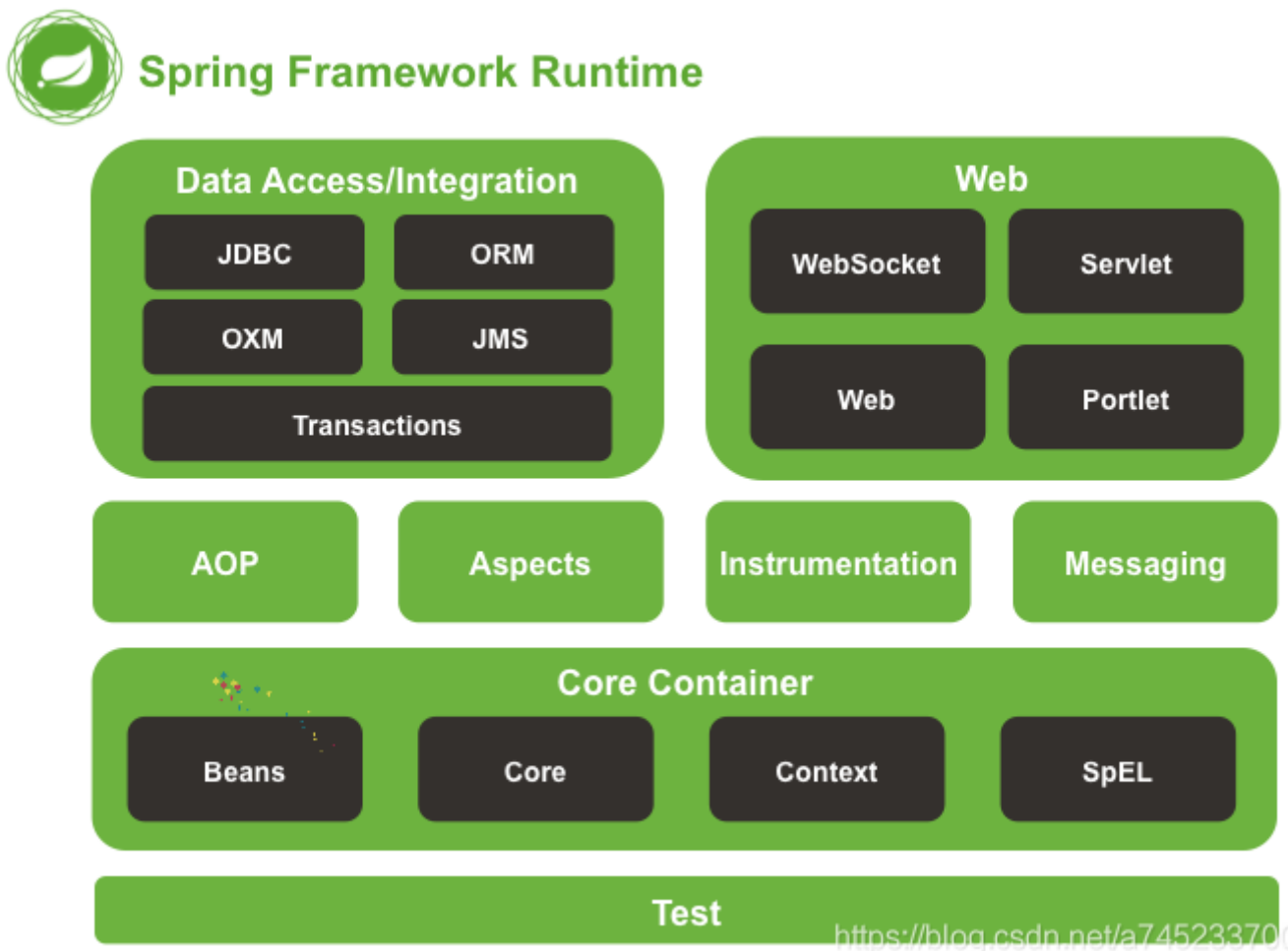
Spring是什么

Spring是一个轻量级的IoC和AOP容器框架。是为Java应用程序提供基础性服务的一套框架，目的是用于简化企业应用程序的开发，它使得开发者只需要关心业务需求。

轻量级IOC和AOP容器框架,基础性服务框架,目的简化开发.

主要包括以下七个模块：

- Spring Context：提供框架式的Bean访问方式，以及企业级功能（JNDI、定时任务等）；
- Spring Core：核心类库，所有功能都依赖于该类库，提供IOC和DI服务；
- Spring AOP(切面)：AOP服务；
- Spring Web：提供了基本的面向Web的综合特性，提供对常见框架如Struts2的支持，Spring能够管理这些框架，将Spring的资源注入给框架，也能在这些框架的前后插入拦截器；
- Spring MVC：提供面向Web应用的Model-View-Controller，即MVC实现。
- Spring DAO(数据)：对JDBC的抽象封装，简化了数据访问异常的处理，并能统一管理JDBC事务；
- Spring ORM：对现有的ORM框架的支持；对象关系映射
- 

Spring的IoC理解：

（1）什么是IOC：

将对象的控制权交给spring

IOC，Inversion of Control，控制反转，指将对象的控制权转移给Spring框架，由 Spring 来负责控制对象的生命周期（比如创建、销毁）和对象间的依赖关系。

最直观的表达就是，以前创建对象的时机和主动权都是由自己把控的，如果在一个对象中使用另外的对象，就必须主动通过new指令去创建依赖对象，使用完后还需要销毁（比如Connection等），对象始终会和其他接口或类耦合起来。而 IOC 则是由专门的容器来帮忙创建对象，将所有的类都在 Spring 容器中登记，当需要某个对象时，不再需要自己主动去 new 了，只需告诉 Spring 容器，然后 Spring 就会在系统运行到适当的时机，把你想要的对象主动给你。也就是说，对于某个具体的对象而言，以前是由自己控制它所引用对象的生命周期，而在IOC中，所有的对象都被 Spring 控制，控制对象生命周期的不再是引用它的对象，而是Spring容器，由 Spring 容器帮我们创建、查找及注入依赖对象，而引用对象只是被动的接受依赖对象，所以这叫控制反转。

总结: 以前都是new出来的对象,而现在对象需要提前在spring容器中登记,需要用的时候总结去容器中找就可以.

（2）什么是DI(依赖注入)：

IoC 的一个重点就是在程序运行时，动态的向某个对象提供它所需要的其他对象，这一点是通过DI（Dependency Injection，依赖注入）来实现的，即应用**程序在运行时依赖 IoC 容器来动态注入对象所需要的外部依赖**。而 Spring 的 DI 具体就是通过反射实现注入的，反射允许程序在运行的时候动态的生成对象、执行对象的方法、改变对象的属性

（3）IoC的原理：

Spring 的 IoC 的实现原理就是工厂模式加反射机制

Spring的AOP理解：

OOP面向对象，允许开发者定义纵向的关系，但并不适用于定义横向的关系，会导致大量代码的重复，而不利于各个模块的重用。

AOP，一般称为面向切面，作为面向对象的一种补充，用于将那些与业务无关，但却对多个对象产生影响的公共行为和逻辑，抽取并封装为一个可重用的模块，这个模块被命名为“切面”（Aspect），减少系统中的重复代码，降低了模块间的耦合度，提高系统的可维护性。可用于权限认证、日志、事务处理。

Spring容器的启动流程：

1）初始化Spring容器

初始化Spring容器，注册内置的BeanPostProcessor的BeanDefinition到容器中：

2) 将配置类的BeanDefinition注册到容器中：

3）调用refresh()方法刷新容器：

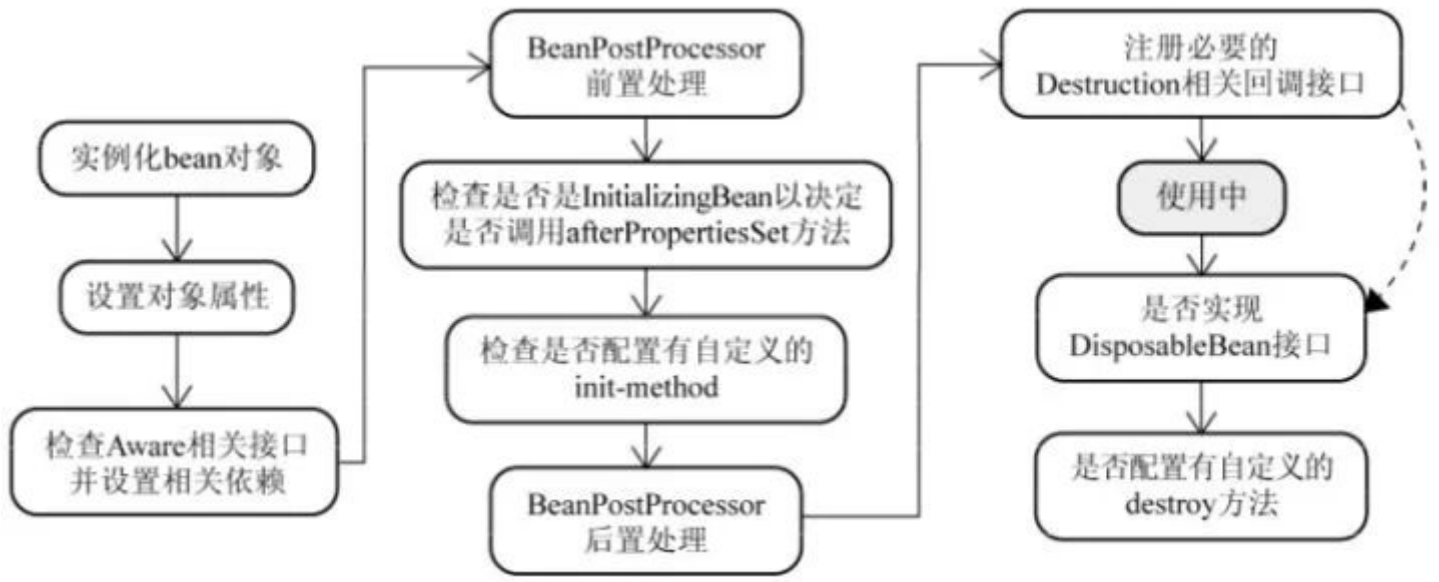
SpringBoot启动流程

1.SpringBoot启动的时候，会构造一个SpringApplication的实例，然后调用这个实例的run方法

Spring Bean的生命周期?

简单来说，Spring Bean的生命周期只有四个阶段：实例化 Instantiation --> 属性赋值 Populate --> 初始化 Initialization --> 销毁 Destruction

但具体来说，Spring Bean的生命周期包含下图的流程：



Spring中bean的作用域：

（1）singleton：默认作用域，单例bean，每个容器中只有一个bean的实例。

（2）prototype：为每一个bean请求创建一个实例。

（3）request：为每一个request请求创建一个实例，在请求完成以后，bean会失效并被垃圾回收器回收。

（4）session：与request范围类似，同一个session会话共享一个实例，不同会话使用不同的实例。

（5）global-session：全局作用域，所有会话共享一个实例。如果想要声明让所有会话共享的存储变量的话，那么这全局变量需要存储在global-session中。

Bean是线程安全的么?

Spring框架中的Bean是线程安全的么？如果线程不安全，那么如何处理?

Spring容器本身并没有提供Bean的线程安全策略，因此可以说Spring容器中的Bean本身不具备线程安全的特性，但是具体情况还是要结合Bean的作用域来讨论。

（1）对于prototype作用域的Bean，每次都创建一个新对象，也就是线程之间不存在Bean共享，因此不会有线程安全问题。

（2）对于singleton作用域的Bean，所有的线程都共享一个单例实例的Bean，因此是存在线程安全问题的。但是如果单例Bean是一个无状态Bean，也就是线程中的操作不会对Bean的成员执行查询以外的操作，那么这个单例Bean是线程安全的。比如Controller类、Service类和Dao等，这些Bean大多是无状态的，只关注于方法本身。

```
有状态Bean(Stateful Bean) ：就是有实例变量的对象，可以保存数据，是非线程安全的。

无状态Bean(Stateless Bean)：就是没有实例变量的对象，不能保存数据，是不变类，是线程安全的。
```

对于有状态的bean（比如Model和View），就需要自行保证线程安全，最浅显的解决办法就是将有状态的bean的作用域由“singleton”改为“prototype”。

也可以加锁

Spring基于xml注入bean的几种方式：

- set()方法注入；
- 构造器注入：①通过index设置参数的位置；②通过type设置参数类型；

springboot注入

- @Component
- 注解 @Bean

一.通过注解注入的一般形式

Configuration类

@Configuration注解去标记了该类，这样标明该类是一个Spring的一个配置类，在加载配置的时候会去加载他。

@Bean的注解，标明这是一个注入Bean的方法，会将下面的返回的Bean注入IOC。

```
//创建一个class配置文件
@Configuration
public class TestConfiguration{
 //将一个Bean交由Spring进行管理
    @Bean
    public TestBean myBean(){
        return new TestBean();
    }
}
```

二.通过构造方法注入Bean

我们在生成一个Bean实例的时候，可以使用Bean的构造方法将Bean实现注入

Bean类

```
@Component
public class TestBeanConstructor {

 private AnotherBean anotherBeanConstructor;

 @Autowired
 public TeanBeanConstructor(AnotherBean anotherBeanConstructor){
     this.anotherBeanConstructor = anotherBeanConstructor;
 }

 @Override
 public String toString() {
     return "TeanBean{" +
         "anotherBeanConstructor=" + anotherBeanConstructor +
         '}';
 }
}
```

AnotherBean

```
@Component(value="Bean的id，默认为类名小驼峰")
public class AnotherBean {
}
```

@Component（默认单例模式）

@Component 被称为元注释，它是@Repository、@Service、@Controller、@Configuration的父类，理论上可以使用@Component来注释任何需要Spring自动装配的类。但每个注释都有他们自己的作用，用来区分类的作用，Spring文档上也说明后续版本中可能为@Repository、@Service、@Controller、@Configuration这些注释添加其他功能，所以建议大家还是少使用@Component。

@Repository注解是任何满足存储库角色或构造型(也称为数据访问对象或DAO)的类的标记。该标记的用途之一是异常的自动翻译，如异常翻译中所述。

@Service注解一般使用在Service层。

@Controller注解一般使用在Controller层的类，@RestController继承了@Controller。

@Configuration注解一般用来配置类，用来项目启动时加载的类。

三.通过set方法注入Bean

我们可以在一个属性的set方法中去将Bean实现注入

Bean类

```
@Component
public class TestBeanSet {

 private AnotherBean anotherBeanSet;

 @Autowired
 public void setAnotherBeanSet(AnotherBean anotherBeanSet) {
     this.anotherBeanSet = anotherBeanSet;
 }

 @Override
 public String toString() {
     return "TestBeanSet{" +
         "anotherBeanSet=" + anotherBeanSet +
         '}';
 }
}
```

Spring事务的实现方式和实现原理：

Spring事务的本质其实就是数据库对事务的支持，没有数据库的事务支持，spring是无法提供事务功能的。Spring只提供统一事务管理接口，具体实现都是由各数据库自己实现，数据库事务的提交和回滚是通过 redo log 和 undo log实现的。Spring会在事务开始时，根据当前环境中设置的隔离级别，调整数据库隔离级别，由此保持一致。

（1）Spring事务的种类：

spring支持编程式事务管理和声明式事务管理两种方式：

①编程式事务管理使用TransactionTemplate。

②声明式事务管理建立在AOP之上的。其本质是通过AOP功能，对方法前后进行拦截，将事务处理的功能编织到拦截的方法中，也就是在目标方法开始之前启动一个事务，在执行完目标方法之后根据执行情况提交或者回滚事务。

（2）spring的事务传播机制：

spring事务的传播机制说的是，当多个事务同时存在的时候，spring如何处理这些事务的行为。事务传播机制实际上是使用简单的ThreadLocal实现的，所以，如果调用的方法是在新线程调用的，事务传播实际上是会失效的。

Spring中的隔离级别：

① ISOLATION_DEFAULT：这是个 PlatfromTransactionManager 默认的隔离级别，使用数据库默认的事务隔离级别。

② ISOLATION_READ_UNCOMMITTED：读未提交，允许事务在执行过程中，读取其他事务未提交的数据。

③ ISOLATION_READ_COMMITTED：读已提交，允许事务在执行过程中，读取其他事务已经提交的数据。

④ ISOLATION_REPEATABLE_READ：可重复读，在同一个事务内，任意时刻的查询结果都是一致的。

⑤ ISOLATION_SERIALIZABLE：所有事务逐个依次执行。

Spring 框架中都用到了哪些设计模式?

1）工厂模式：Spring使用工厂模式，通过BeanFactory和ApplicationContext来创建对象

（2）单例模式：Bean默认为单例模式

（3）策略模式：例如Resource的实现类，针对不同的资源文件，实现了不同方式的资源获取策略

（4）代理模式：Spring的AOP功能用到了JDK的动态代理和CGLIB字节码生成技术

（5）模板方法：可以将相同部分的代码放在父类中，而将不同的代码放入不同的子类中，用来解决代码重复的问题。比如RestTemplate, JmsTemplate, JpaTemplate

（6）适配器模式：Spring AOP的增强或通知（Advice）使用到了适配器模式，Spring MVC中也是用到了适配器模式适配Controller

（7）观察者模式：Spring事件驱动模型就是观察者模式的一个经典应用。

（8）桥接模式：可以根据客户的需求能够动态切换不同的数据源。比如我们的项目需要连接多个数据库，客户在每次访问中根据需要会去访问不同的数据库

注解的原理：

（1）什么是注解：

Java 注解就是代码中的一些特殊标记（元信息），用于在编译、类加载、运行时进行解析和使用，并执行相应的处理。它本质是继承了 Annotation 的特殊接口，其具体实现类是 JDK 动态代理生成的代理类，通过反射获取注解时，返回的也是 Java 运行时生成的动态代理对象 $Proxy1。通过代理对象调用自定义注解的方法，会最终调用 AnnotationInvocationHandler 的 invoke 方法，该方法会从 memberValues 这个Map中查询出对应的值，而 memberValues 的来源是Java常量池。

(2) 如何自定义注解?

① 创建一个自定义注解：与创建接口类似，但自定义注解需要使用 @interface

② 添加元注解信息，比如 @Target、@Retention、@Document、@Inherited 等

③ 创建注解方法，但注解方法不能带有参数

④ 注解方法返回值为基本类型、String、Enums、Annotation 或其数组

⑤ 注解可以有默认值;

```
@Target(FIELD)
@Retention(RUNTIME)
@Documented
public @interface CarName {
    String value() default "";
}
```

Spring的自动装配:

在spring中，使用autowire来配置自动装载模式，对象无需自己查找或创建与其关联的其他对象，由容器负责把需要相互协作的对象引用赋予各个对象。

(1) 在Spring框架xml配置中共有5种自动装配:

- no：默认的方式是不进行自动装配的，通过手工设置ref属性来进行装配bean。
- byName：通过bean的名称进行自动装配，如果一个bean的 property 与另一bean 的name 相同，就进行自动装配。
- byType：通过参数的数据类型进行自动装配。
- constructor：利用构造函数进行装配，并且构造函数的参数通过byType进行装配。
- autodetect：自动探测，如果有构造方法，通过 construct的方式自动装配，否则使用 byType的方式自动装配。

(2) 基于注解的自动装配方式:

使用@Autowired、@Resource注解来自动装配指定的bean。在使用@Autowired注解之前需要在Spring配置文件进行配置，。在启动spring IoC时，容器自动装载了一个AutowiredAnnotationBeanPostProcessor后置处理器，当容器扫描到@Autowied、@Resource或@Inject时，就会在IoC容器自动查找需要的bean，并装配给该对象的属性。在使用@Autowired时，首先在容器中查询对应类型的bean:

如果查询结果刚好为一个，就将该bean装配给@Autowired指定的数据;

如果查询的结果不止一个，那么@Autowired会根据名称来查找;

如果上述查找的结果为空，那么会抛出异常。解决方法时，使用required=false。

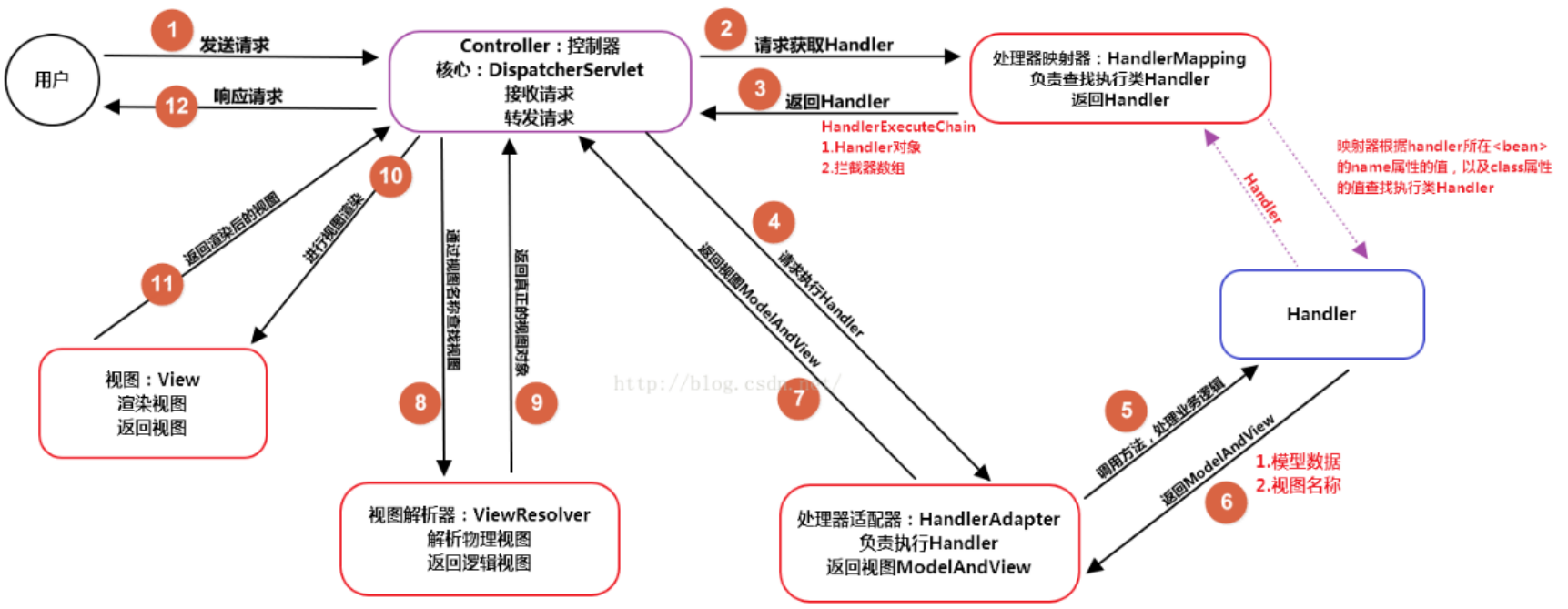
什么是Spring MVC ？

Spring MVC是一个基于Java的实现了MVC设计模式的请求驱动类型的轻量级Web框架，通过把Model，View，Controller分离，将web层进行职责解耦，把复杂的web应用分成逻辑清晰的几部分，简化开发，减少出错，方便组内开发人员之间的配合。

Spring MVC是基于java,实现mvc设计模式的web框架

SpringMVC的流程?

- （1）用户发送请求至前端控制器DispatcherServlet；
- （2）DispatcherServlet收到请求后，调用HandlerMapping处理器映射器，请求获取Handler；
- （3）处理器映射器根据请求url找到具体的处理器Handler，生成处理器对象及处理器拦截器(如果有则生成)，一并返回给DispatcherServlet；
- （4）DispatcherServlet 调用 HandlerAdapter处理器适配器，请求执行Handler；
- （5）HandlerAdapter 经过适配调用 具体处理器进行处理业务逻辑；
- （6）Handler执行完成返回ModelAndView；
- （7）HandlerAdapter将Handler执行结果ModelAndView返回给DispatcherServlet；
- （8）DispatcherServlet将ModelAndView传给ViewResolver视图解析器进行解析；
- （9）ViewResolver解析后返回具体View；
- （10）DispatcherServlet对View进行渲染视图（即将模型数据填充至视图中）
- （11）DispatcherServlet响应用户。



- 前端控制器 DispatcherServlet：接收请求、响应结果，相当于转发器，有了DispatcherServlet 就减少了其它组件之间的耦合度。
- 处理器映射器 HandlerMapping：根据请求的URL来查找Handler
- 处理器适配器 HandlerAdapter：负责执行Handler
- 处理器 Handler：处理器，需要程序员开发

- 视图解析器 ViewResolver：进行视图的解析，根据视图逻辑名将ModelAndView解析成真正的视图（view）
- 视图View：View是一个接口， 它的实现类支持不同的视图类型，如jsp，freemarker，pdf等等

Spring Boot、Spring MVC 和 Spring 有什么区别?

1、Spring

Spring最重要的特征是依赖注入。所有 SpringModules 不是依赖注入就是 IOC 控制反转。

当我们恰当的使用 DI 或者是 IOC 的时候，我们可以开发松耦合应用。松耦合应用的单元测试可以很容易的进行。

2、Spring MVC

Spring MVC 提供了一种分离式的方法来开发 Web 应用。通过运用像 DispatcherServelet，MoudlAndView 和 ViewResolver 等一些简单的概念，开发 Web 应用将会变的非常简单。

3、SpringBoot

Spring 和 SpringMVC 的问题在于需要配置大量的参数。Spring Boot 通过一个自动配置和启动的项来目解决这个问题。为了更快的构建产品就绪应用程序，Spring Boot 提供了一些非功能性特征。

什么是自动配置?

Spring 和 SpringMVC 的问题在于需要配置大量的参数。

Spring 查看（CLASSPATH 上可用的框架）已存在的应用程序的配置。在此基础上，Spring Boot 提供了配置应用程序和框架所需要的基本配置。这就是自动配置。

springboot自动配置的原理

在spring程序main方法中 添加@SpringBootApplication或者@EnableAutoConfiguration

会自动去maven中读取每个starter中的spring.factories文件 该文件里配置了所有需要被创建spring容器中的bean

Spring Boot 的核心注解是哪个?

启动类上面的注解是@SpringBootApplication，它也是 Spring Boot 的核心注解，主要组合包含了以下 3 个注解：

@SpringBootConfiguration：组合了 @Configuration 注解，实现配置文件的功能。

@EnableAutoConfiguration：打开自动配置的功能，也可以关闭某个自动配置的选项，如关闭数据源自动配置功能：

@ComponentScan：Spring组件扫描。