**一、SpringMVC**

<http://blog.csdn.net/evankaka/article/details/45501811>

Spring Web MVC是一种基于Java的实现了Web MVC[设计模式](http://www.amazon.cn/gp/product/B001130JN8/ref=as_li_qf_sp_asin_il_tl?ie=UTF8&tag=importnew-23&linkCode=as2&camp=536&creative=3200&creativeASIN=B001130JN8)的请求驱动类型的轻量级Web框架，即使用了MVC架构模式的思想，将web层进行职责解耦，基于请求驱动指的就是使用请求-响应模型，框架的目的就是帮助我们简化开发，Spring Web MVC也是要简化我们日常Web开发的。

**模型（Model ）**封装了应用程序的数据和一般他们会组成的POJO。

**视图（View）**是负责呈现模型数据和一般它生成的HTML输出，客户端的浏览器能够解释。

**控制器（Controller ）**负责处理用户的请求，并建立适当的模型，并把它传递给视图渲染。

Spring的web模型 – 视图 – 控制器（MVC）框架是围绕着处理所有的HTTP请求和响应的DispatcherServlet的设计。

Spring Web MVC处理请求的流程



具体执行步骤如下：

1、 首先用户发送请求————>前端控制器，前端控制器根据请求信息（如URL）来决定选择哪一个页面控制器进行处理并把请求委托给它，即以前的控制器的控制逻辑部分；图2-1中的1、2步骤；

2、 页面控制器接收到请求后，进行功能处理，首先需要收集和绑定请求参数到一个对象，这个对象在Spring Web MVC中叫命令对象，并进行验证，然后将命令对象委托给业务对象进行处理；处理完毕后返回一个ModelAndView（模型数据和逻辑视图名）；图2-1中的3、4、5步骤；

3、 前端控制器收回控制权，然后根据返回的逻辑视图名，选择相应的视图进行渲染，并把模型数据传入以便视图渲染；图2-1中的步骤6、7；

4、 前端控制器再次收回控制权，将响应返回给用户，图2-1中的步骤8；至此整个结束。

**二、Spring**

<http://blog.csdn.net/cainiaowys/article/details/7107925>

2.1、IOC容器：<http://www.cnblogs.com/linjiqin/archive/2013/11/04/3407126.html>

IOC容器就是具有依赖注入功能的容器，IOC容器负责实例化、定位、配置应用程序中的对象及建立这些对象间的依赖。应用程序无需直接在代码中new相关的对象，应用程序由IOC容器进行组装。在Spring中BeanFactory是IOC容器的实际代表者。

2.2、AOP：<http://blog.csdn.net/moreevan/article/details/11977115>

简单地说，就是将那些与业务无关，却为业务模块所共同调用的逻辑或责任封装起来，便于减少系统的重复代码，降低模块间的耦合度，并有利于未来的可操作性和可维护性。AOP代表的是一个横向的关系

AOP用来封装横切关注点，具体可以在下面的场景中使用:

Authentication 权限  
Caching 缓存  
Context passing 内容传递  
Error handling 错误处理  
Lazy loading　懒加载  
Debugging　　调试  
logging, tracing, profiling and monitoring　记录跟踪　优化　校准  
Performance optimization　性能优化  
Persistence　　持久化  
Resource pooling　资源池  
Synchronization　同步  
Transactions 事务

**三、Mybatis**

<http://blog.csdn.net/u013142781/article/details/50388204>

MyBatis 是支持普通 SQL查询，存储过程和高级映射的优秀持久层框架。MyBatis 消除了几乎所有的JDBC代码和参数的手工设置以及结果集的检索。MyBatis 使用简单的 XML或注解用于配置和原始映射，将接口和 Java 的POJOs（Plain Old Java Objects，普通的 Java对象）映射成数据库中的记录。

总体流程：

(1)加载配置并初始化  
触发条件：加载配置文件  
将SQL的配置信息加载成为一个个MappedStatement对象（包括了传入参数映射配置、执行的SQL语句、结果映射配置），存储在内存中。

(2)接收调用请求  
触发条件：调用Mybatis提供的API  
传入参数：为SQL的ID和传入参数对象  
处理过程：将请求传递给下层的请求处理层进行处理。

(3)处理操作请求 触发条件：API接口层传递请求过来  
传入参数：为SQL的ID和传入参数对象

处理过程：

(A)根据SQL的ID查找对应的MappedStatement对象。

(B)根据传入参数对象解析MappedStatement对象，得到最终要执行的SQL和执行传入参数。

(C)获取数据库连接，根据得到的最终SQL语句和执行传入参数到数据库执行，并得到执行结果。

(D)根据MappedStatement对象中的结果映射配置对得到的执行结果进行转换处理，并得到最终的处理结果。

(E)释放连接资源。

(4)返回处理结果将最终的处理结果返回。

MyBatis 最强大的特性之一就是它的动态语句功能。如果您以前有使用JDBC或者类似框架的经历，您就会明白把SQL语句条件连接在一起是多么的痛苦，要确保不能忘记空格或者不要在columns列后面省略一个逗号等。动态语句能够完全解决掉这些痛苦。

**四、Dubbo**

<http://blog.csdn.net/u013142781/article/details/50387583>

Dubbo是一个分布式服务框架，致力于提供高性能和透明化的RPC（远程过程调用协议）远程服务调用方案，以及SOA服务治理方案。简单的说，dubbo就是个服务框架，如果没有分布式的需求，其实是不需要用的，只有在分布式的时候，才有dubbo这样的分布式服务框架的需求，并且本质上是个服务调用的东东，说白了就是个远程服务调用的分布式框架。

1、透明化的远程方法调用，就像调用本地方法一样调用远程方法，只需简单配置，没有任何API侵入。

2、软负载均衡及容错机制，可在内网替代F5等硬件负载均衡器，降低成本，减少单点。

3、 服务自动注册与发现，不再需要写死服务提供方地址，注册中心基于接口名查询服务提供者的IP地址，并且能够平滑添加或删除服务提供者。

节点角色说明：  
Provider: 暴露服务的服务提供方。  
Consumer: 调用远程服务的服务消费方。  
Registry: 服务注册与发现的注册中心。  
Monitor: 统计服务的调用次调和调用时间的监控中心。  
Container: 服务运行容器。

**五、Maven**

<http://blog.csdn.net/u013142781/article/details/50316383>

Maven这个个项目管理和构建自动化工具，越来越多的开发人员使用它来管理项目中的jar包。但是对于我们程序员来说，我们最关心的是它的项目构建功能。

**六、RabbitMQ**

<http://blog.csdn.net/u013142781/article/category/6061896>

消息队列一般是在项目中，将一些无需即时返回且耗时的操作提取出来，进行了异步处理，而这种异步处理的方式大大的节省了服务器的请求响应时间，从而提高了系统的吞吐量。

RabbitMQ是用Erlang实现的一个高并发高可靠AMQP消息队列服务器。

Erlang是一门动态类型的函数式编程语言。对应到Erlang里，每个Actor对应着一个Erlang进程，进程之间通过消息传递进行通信。相比共享内存，进程间通过消息传递来通信带来的直接好处就是消除了直接的锁开销(不考虑Erlang虚拟机底层实现中的锁应用)。

AMQP(Advanced Message Queue Protocol)定义了一种消息系统规范。这个规范描述了在一个分布式的系统中各个子系统如何通过消息交互。

**七、Log4j**

<http://blog.csdn.net/u013142781/article/category/6045728>

日志记录的优先级，分为OFF、FATAL、ERROR、WARN、INFO、DEBUG、ALL或者您定义的级别。

**八、Ehcache**

<http://blog.csdn.net/u013142781/article/category/6066337>

EhCache 是一个纯Java的进程内缓存框架，具有快速、精干等特点，是Hibernate中默认的CacheProvider。Ehcache是一种广泛使用的开源Java分布式缓存。主要面向通用缓存,Java EE和轻量级容器。它具有内存和磁盘存储，缓存加载器,缓存扩展，缓存异常处理程序，一个gzip缓存servlet过滤器，支持REST和SOAP api等特点。

优点：  
1、 快速  
2、 简单  
3、 多种缓存策略  
4、缓存数据有两级：内存和磁盘，因此无需担心容量问题  
5、 缓存数据会在虚拟机重启的过程中写入磁盘  
6、可以通过RMI、可插入API等方式进行分布式缓存  
7、 具有缓存和缓存管理器的侦听接口  
8、支持多缓存管理器实例，以及一个实例的多个缓存区域  
9、提供Hibernate的缓存实现

缺点：  
1、使用磁盘Cache的时候非常占用磁盘空间：这是因为DiskCache的算法简单，该算法简单也导致Cache的效率非常高。它只是对元素直接追加存储。因此搜索元素的时候非常的快。如果使用DiskCache的，在很频繁的应用中，很快磁盘会满。

2、 不能保证数据的安全：当突然kill掉java的时候，可能会产生冲突，EhCache的解决方法是如果文件冲突了，则重建cache。这对于Cache数据需要保存的时候可能不利。当然，Cache只是简单的加速，而不能保证数据的安全。如果想保证数据的存储安全，可以使用Bekeley DB Java Edition版本。这是个嵌入式数据库。可以确保存储安全和空间的利用率。

**九、Redis**

<http://blog.csdn.net/u013142781/article/category/6067864>

redis是一个key-value存储系统。和Memcached类似，它支持存储的value类型相对更多，包括string(字符串)、list(链表)、set(集合)、zset(sorted set –有序集合)和hash（哈希类型）。这些数据类型都支持push/pop、add/remove及取交集并集和差集及更丰富的操作，而且这些操作都是原子性的。在此基础上，redis支持各种不同方式的排序。与memcached一样，为了保证效率，数据都是缓存在内存中。区别的是redis会周期性的把更新的数据写入磁盘或者把修改操作写入追加的记录文件，并且在此基础上实现了master-slave(主从)同步。

Redis数据库完全在内存中，使用磁盘仅用于持久性。相比许多键值数据存储，Redis拥有一套较为丰富的数据类型。Redis可以将数据复制到任意数量的从服务器。

1.2、Redis优点：

（1）异常快速：Redis的速度非常快，每秒能执行约11万集合，每秒约81000+条记录。

（2）支持丰富的数据类型：Redis支持最大多数开发人员已经知道像列表，集合，有序集合，散列数据类型。这使得它非常容易解决各种各样的问题，因为我们知道哪些问题是可以处理通过它的数据类型更好。

（3）操作都是原子性：所有Redis操作是原子的，这保证了如果两个客户端同时访问的Redis服务器将获得更新后的值。

（4）多功能实用工具：Redis是一个多实用的工具，可以在多个用例如缓存，消息，队列使用(Redis原生支持发布/订阅)，任何短暂的数据，应用程序，如Web应用程序会话，网页命中计数等。

1.3、Redis缺点：  
（1）单线程  
（2）耗内存

**十、Shiro**

<http://blog.csdn.net/u013142781/article/details/50629708>

Apache Shiro是Java的一个安全框架，旨在简化身份验证和授权。Shiro在JavaSE和JavaEE项目中都可以使用。它主要用来处理身份认证，授权，企业会话管理和加密等。Shiro的具体功能点如下：

（1）身份认证/登录，验证用户是不是拥有相应的身份；

（2）授权，即权限验证，验证某个已认证的用户是否拥有某个权限；即判断用户是否能做事情，常见的如：验证某个用户是否拥有某个角色。或者细粒度的验证某个用户对某个资源是否具有某个权限；

（3）会话管理，即用户登录后就是一次会话，在没有退出之前，它的所有信息都在会话中；会话可以是普通JavaSE环境的，也可以是如Web环境的；

（4）加密，保护数据的安全性，如密码加密存储到数据库，而不是明文存储；

（5）Web支持，可以非常容易的集成到Web环境；  
Caching：缓存，比如用户登录后，其用户信息、拥有的角色/权限不必每次去查，这样可以提高效率；

（6）shiro支持多线程应用的并发验证，即如在一个线程中开启另一个线程，能把权限自动传播过去；

（7）提供测试支持；

（8）允许一个用户假装为另一个用户（如果他们允许）的身份进行访问；

（9）记住我，这个是非常常见的功能，即一次登录后，下次再来的话不用登录了。

文字描述可能并不能让猿友们完全理解具体功能的意思。下面我们以登录验证为例，向猿友们介绍Shiro的使用。至于其他功能点，猿友们用到的时候再去深究其用法也不迟。

**十一、设计模式**

这个算不上框架，可自行忽略，不过博主认为设计模式的思想很有必要了解一下。

<http://blog.csdn.net/u013142781/article/details/50816245>  
<http://blog.csdn.net/u013142781/article/details/50821155>  
<http://blog.csdn.net/u013142781/article/details/50825301>

思想：

开闭原则：开闭原则就是说对扩展开放，对修改关闭。在程序需要进行拓展的时候，不能去修改原有的代码。

针对接口编程，真对接口编程，依赖于抽象而不依赖于具体。

尽量使用合成/聚合的方式，而不是使用继承。

一个实体应当尽量少的与其他实体之间发生相互作用，使得系统功能模块相对独立。

使用多个隔离的接口，比使用单个接口要好。

里氏代换原则：（1）子类的能力必须大于等于父类，即父类可以使用的方法，子类都可以使用。（2）返回值也是同样的道理。假设一个父类方法返回一个List，子类返回一个ArrayList，这当然可以。如果父类方法返回一个ArrayList，子类返回一个List，就说不通了。这里子类返回值的能力是比父类小的。（3）还有抛出异常的情况。任何子类方法可以声明抛出父类方法声明异常的子类。  
而不能声明抛出父类没有声明的异常。