1. 过滤器

依赖于servlet容器，在实现上基于函数回调,可以对几乎所有请求进行过滤，但是缺点就是在一个过滤器中只能在容器初始化调用一次，使用过滤器的目的是用来做一些过滤操作，获取我们想要的数据。

Filter存在的意义就好比你要去重庆，它是你的目的地，但是提供一个机制让你在去的途中可以做一些拦截工作，如可以将你的一些行李存放在某个”存放处”，当你返回时你可以再从这个地方取回，总之它可以在你的途中增加一些东西，或者减少一些东西。

1. 拦截器

依赖于web框架，在实现上基于Java的反射机制，属于AOP的一种运用，由于拦截器是基于web框架的调用，因此可以使用spring的DI进行业务操作。同时一个拦截器实例在一个controller生命周期内可以多次调用，但是缺点是只能对controller进行拦截，对于其他的一些比如直接访问静态资源的请求则没办法进行拦截处理。

区别：

1. 拦截器是基于Java的反射机制的，而过滤器是基于函数回调的
2. 拦截器不依赖于servlet容器，过滤器依赖于servlet容器
3. 拦截器只能对action请求起作用，而过滤器则可以对几乎所有的请求起作用
4. 拦截器可以访问action上下文，堆栈里的对象，而过滤器不能访问
5. 在action的生命周期，拦截器可以多次被调用，而过滤器只能在容器初始化时被调用一次。
6. 拦截器可以获取IOC容器中的各个bean，而过滤器则不行，在拦截器里注入一个service，可以调用业务逻辑。

1.过滤器和拦截器触发时机不一样:

　　过滤器是在请求进入容器后，但请求进入servlet之前进行预处理的。请求结束返回也是，是在servlet处理完后，返回给前端之前。

1.过滤器和拦截器触发时间和地点不一样:

　　过滤器是在请求进入容器后，但请求进入servlet之前进行预处理的。请求结束返回也是，是在servlet处理完后，返回给前端之前。

1.过滤器和拦截器触发时间和地点不一样:

　　过滤器是在请求进入容器后，但请求进入servlet之前进行预处理的。请求结束返回也是，是在servlet处理完后，返回给前端之前。

总结：过滤器包裹住servlet，servlet包裹住拦截器。