过滤器Servlet
功能：拦截请求，检查，放行还是禁止放行

Filter的HelloWorld

① 准备一个请求发出
a. 创建index.html
b. 创建一个超链接
c. 创建一个Servlet,点击超链接访问Servlet

② 创建过滤器，过滤上述请求(hello)
a. 新建一个类
b. 实现一个接口(javax.servlet.Filter)
c. 实现接口中的所有抽象方法
d. 配置过滤器的过滤路径
位置：web.xml
代码：

```
<filter>
  <filter-name>HelloFilter</filter-name>
  <filter-class>com.atguigu.filter.HelloFilter</filter-class>
</filter>
<filter-mapping>
  <filter-name>HelloFilter</filter-name>
  <url-pattern>/hello</url-pattern>//注意:此处是要访问Servlet的访问路径
</filter-mapping>
```

③ 过滤器的原理
请求发出，如果符合过滤器的过滤要求，执行放行前代码
放行，就是去执行目标资源
目标资源执行完毕，会执行放行后的代码
最后把响应给到客户端

生命周期

3. Filter的生命周期
Filter对象什么时候创建，对象内的方法什么执行，对象什么销毁
init、doFilter、destroy
Filter对象是在特定情况下被创建，方法是在特定情况下被执行！
在web项目被加载的时候(服务器启动)过滤器对象被创建，立刻执行初始化方法
n发送符合过滤器过滤路径的请求时，直接执行n次doFilter方法
web项目被卸载的时候(服务器关闭)，对象被销毁，销毁之前，执行destroy方法

匹配规则

4. 过滤器的匹配规则
主要研究的就是filter-mapping中的url-pattern的值有几种编写方式
① 精确匹配
<url-pattern>/hello</url-pattern> 过滤项目下hello请求
<url-pattern>/a</url-pattern> 过滤项目下a请求
<url-pattern>/b</url-pattern> 过滤项目下b请求
② 目录匹配
<url-pattern>/user/*</url-pattern> 过滤项目下user请求下的所有请求
<url-pattern>/*</url-pattern> 过滤项目下所有请求
③ 后缀匹配
<url-pattern>*.abc</url-pattern> 匹配后缀为.abc的请求(注意：不需要加/)
总结：过滤器只过滤一个请求，就是用精确匹配
过滤器过滤多个请求，就是用目录匹配或者后缀匹配

过滤器链

5. 过滤器链
如果出现一个请求存在多个过滤器对其过滤，出现了过滤器链
在放行前，过滤器是正序执行，放行后过滤器是倒序执行
是什么决定了过滤器的顺序呢？和filter-mapping配置上下位置有关