01_过滤器的概述

- 什么是过滤器:
  - 对客户端向服务器发送的请求进行过滤.
  - Filter和Listener都属于Servlet中的高级部分.Filter是Servlet中最为实用的技术.
- 过滤器链
  - 过滤器链指的是 一组过滤器对某些WEB资源进行过滤.那么这组过滤器称为是过滤器链.
  - 这组过滤器链执行的顺序与配置有关!
    - 注解配置
      - 按照命名的字符串的字典顺序排序，值小的先执行。
      - 比如：AFilter和BFilter，AFiler先执行,后回来；BFilter后执行，先回来
    - web.xml配置
      - 先配置的先执行,后回来。
      - 后配置的后执行,先回来。
- 过滤器的入门案例
  - 自定义类实现Filter接口
  - 在web.xml中配置过滤器（或注解）

02_过滤器的相关配置

- 过滤器的生命周期
  - 指的是Filter从创建到销毁的过程!
  - Filter何时创建和销毁:
    - 创建:
      - 服务器启动的时候创建Filter的对象.
    - 销毁:
      - 服务器关闭的时候或者是项目移除!
  - 生命周期方法
    - init方法:过滤器初始化
    - doFilter方法:过滤器放行资源
    - destroy方法:过滤器销毁
- FilterConfig过滤器配置对象
  - 获取过滤器的初始化参数
- 过滤器路径配置详解
  - 和Servlet的配置一致,分为三种:
    - 完全路径匹配
      - 以"/"开始。比如：

```
/aa、/bb、/cc
```

- 目录匹配
  - 以"/"开始，需要以"*"结束。比如：

```
/*、/aa/*、/aa/bb/*
```

- 扩展名匹配
  - 不能以"/"开始，需要以"*"开始。比如：

```
*.jsp、*.do、*.action
```

03_过滤器案例之中文乱码

04_过滤器案例之自动登录

- 实现步骤
  - 登录账户后，根据是否勾选了自动登录选项框，
  - 判断是否访问和登录相关资源
    - 如果是，直接放行
    - 如果不是，判断是否已经在登录状态
      - 如果是，直接放行
      - 如果不是，需要从cookie中取出存储的用户信息，进行登录操作
        - 如果登录成功，直接放行
        - 如果登录失败，就跳转到登录页面
- 代码实现
  - 登录功能

```
@WebServlet(name = "LoginServlet",urlPatterns = "/login")
public class LoginServlet extends HttpServlet {
    protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        String username = request.getParameter("username");
        String password = request.getParameter("password");
        String autoLogin = request.getParameter("autoLogin");
        if ("root".equals(username) && "root".equals(password)) {
            if ("auto".equals(autoLogin)) {
                Cookie cookie = new Cookie("autoLogin",username+"-"+password);
                cookie.setMaxAge(7*24*60*60);
                response.addCookie(cookie);
            }
            //登录成功，重定向到
            User user = new User();
            user.setUsername(username);
            user.setPassword(password);
            request.getSession().setAttribute("existUser",user);
            response.sendRedirect("/myweb/showIndex");
        } else {
            //登录失败，转发登录页面
```

```
request.getRequestDispatcher("/login.html").forward(request,response);
        }
    }
    protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        doPost(request, response);
    }
}
```

- LoginFilter自动登录

```
@WebFilter(filterName = "LoginFilter",urlPatterns = "/*")
public class LoginFilter implements Filter {
    public void destroy() {
    }

    public void doFilter(ServletRequest req, ServletResponse resp, FilterChain
chain) throws ServletException, IOException {
        HttpServletRequest request = (HttpServletRequest) req;
        HttpServletResponse response = (HttpServletResponse) resp;
        String requestURI = request.getRequestURI();
        if (requestURI.contains("login")){
            chain.doFilter(request,response);
        } else {
            HttpSession session = request.getSession();
            Object existUser = session.getAttribute("existUser");
            if (null != existUser) {
                //已经在登录状态,直接放行
                chain.doFilter(request,response);
            } else {
                //不在登录状态，自动登录
                Cookie cookie = CookieUtil.getCookie(request.getCookies(),
"autoLogin");
                if ( null != cookie) {
                    String infoStr = cookie.getValue();
                    String[] infos = infoStr.split("-");
                    String username = infos[0];
                    String password = infos[1];
                    if ("root".equals(username) && "root".equals(password))  {
                        //自动登录成功
                        User user = new User();
                        user.setUsername(username);
                        user.setPassword(password);
                        request.getSession().setAttribute("existUser",user);
                        chain.doFilter(request,response);
                    } else {
                        //自动登录失败,跳转到登录页面
                        response.sendRedirect("login.html");
                    }
                } else {
                    //cookie失效
                    response.sendRedirect("login.html");
                }

            }
```

```
        }

    }

    public void init(FilterConfig config) throws ServletException {

    }

}
```

05_过滤器案例之敏感词过滤

- 对request对象进行增强。增强获取参数相关方法
- 放行。传递增强的请求方法

```
@WebFilter(filterName = "SensitivedWordsFilter",urlPatterns = "/*",initParams =
{@WebInitParam(name = "word1",value = "笨蛋"),@WebInitParam(name = "word2",value =
"坏蛋")})
public class SensitivedWordsFilter implements Filter {

    private List<String> sensitivedWords = new ArrayList<>();
    public void destroy() {
    }

    public void doFilter(ServletRequest req, ServletResponse resp, FilterChain chain)
throws ServletException, IOException {
        ServletRequest request = (ServletRequest)
Proxy.newProxyInstance(req.getClass().getClassLoader(),
            req.getClass().getInterfaces(),
            new InvocationHandler() {
                @Override
                public Object invoke(Object proxy, Method method, Object[] args)
throws Throwable {
                    if ("getParameter".equals(method.getName())) {
                        //获取到请求参数值
                        String obj = (String) method.invoke(req, args);
                        if (obj != null || !obj.equals("")) {
                            for (String sensitivedWord : sensitivedWords) {
                                if (obj.contains(sensitivedWord)) {
                                    obj = obj.replace(sensitivedWord, "");
                                }
                            }
                        }
                        return obj;
                    }
                    return method.invoke(req, args);

                }
            });
        chain.doFilter(request, resp);
    }

    public void init(FilterConfig config) throws ServletException {
        Enumeration<String> initParameterNames = config.getInitParameterNames();
        while (initParameterNames.hasMoreElements()){
```

```
            String parameterName = initParameterNames.nextElement();
            String parameterValue = config.getInitParameter(parameterName);
            sensitivedWords.add(parameterValue);
        }
    }

}
```