* 学习目标

  * 能够理解JSTL的概述

    * JSTL:JSP标签库--核心标签库：taglib，if,foreach

  * 能够掌握JSTL核心标签

  * 能够掌握Web国际的概念

    * i18n

    * 固定文本：com.lg.resource.car :car_zh.properties,car_en.properties

      * ResourceBunlde---

      * jslt:国际标签:fmt:setBunddle,fmt message key

    * 动态

      * DateFormat,NumberFormat,MessageFormat

        * format,parse

  * 能够理解Filter的概述

    * 统一解决中文乱码问题

    * 对URL级别权限访问控制

    * 敏感词汇过滤

  * 能够掌握Filter的开发

      * init,doFIlter(FitlerChain-doFilter) ,destroy

      * 配置:xml,注解

      * 注解:@WebFilter

  * 能够使用Filter对URL级别权限访问控制

      * ok-->FitlerChain-doFilter

      * no-->重定向其他页面

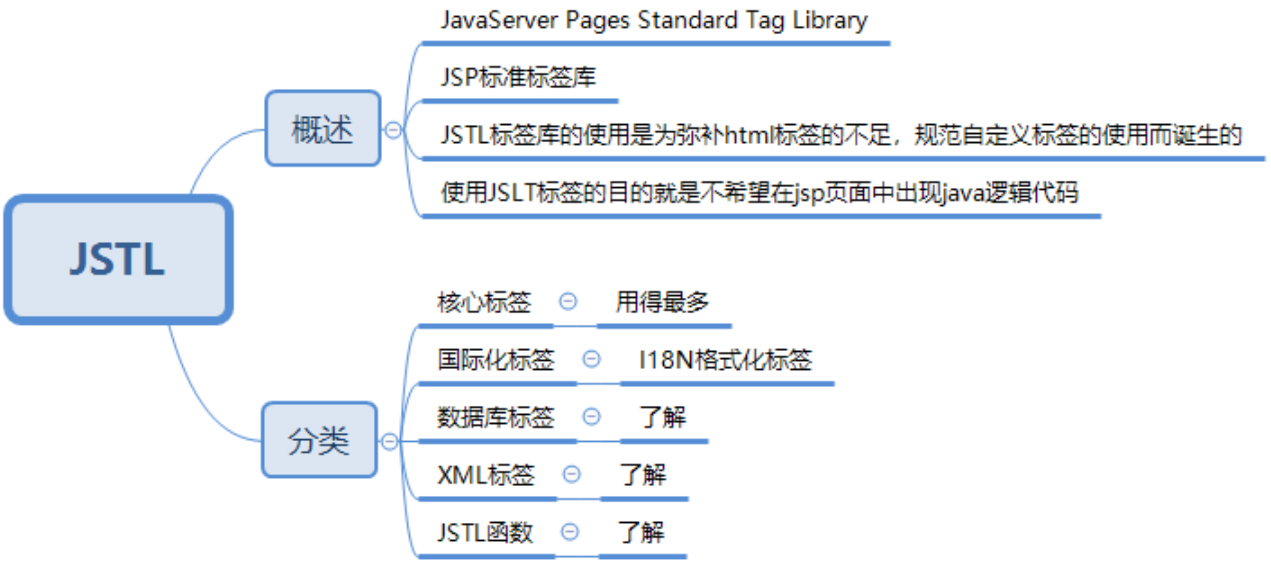
  * 能够使用Filter统一解决中文乱码问题

  * 能够使用FIlter开发敏感词汇过滤的案例

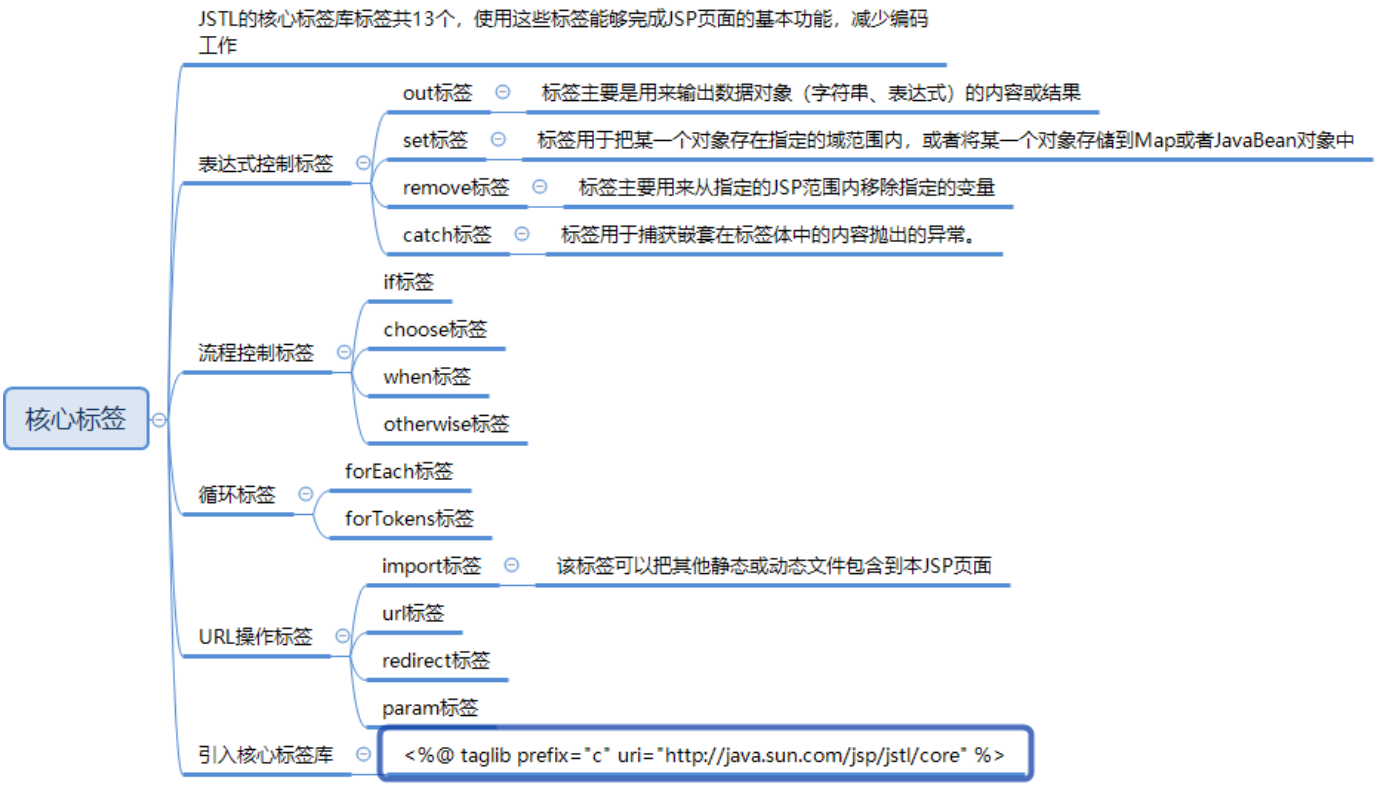
      * 装饰者模式,动态代理模式

---

* 能够理解JSTL的概述



* 能够掌握JSTL核心标签



```
* 开发案例
 * 导入jar包
    * javax.servlet.jsp.jstl-1.2.1.jar
    * javax.servlet.jsp.jstl-api-1.2.1.jar
 * 引入jstl标签库
    * <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
 * 案例一
<%@ page contentType="text/html;charset=UTF-8" language="java" isELIgnored="fal
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<html>
<head>
    <title>jstl标签</title>
</head>
<body>
    <h3>out标签测试</h3>
    <c:out value="我不是药神"></c:out>
    <h3>代码给出了给指定scope范围赋值的示例</h3>
    <c:set var="name" value="xiaohei" scope="page"></c:set>
    pageScope:${pageScope.name}<br/>
    requestScope:${requestScope.name}<br/>
    name:${name}<br/>
    <c:remove var="name"></c:remove>
    pageScope:${pageScope.name}<br/>
    requestScope:${requestScope.name}<br/>
    name:${name}<br/>
    <h3>catch标签实例</h3>
    <c:catch var="errorInfo">
        <c:set property="name" value="xiaoming"></c:set>
    </c:catch>
    <c:out value="${errorInfo}"></c:out>
    <h3>if标签演示</h3>
    <c:set var="name" value="xiaohei" scope="page"></c:set>
    <c:set var="psw" value="123" scope="page"></c:set>
    <c:if test="${(name=='xiaohei') && (psw=='123')}" var="login" scope="page">
        登录成功：${login}
    </c:if>
    <h3>choose标签演示</h3>
```

```
    <c:set var="scope" value="85"></c:set>
    <c:choose>
        <c:when test="${scope>=90}">
            您的成绩为优秀
        </c:when>
        <c:when test="${scope>=70 && scope<90}">
            您的成绩为良好
        </c:when>
        <c:when test="${scope>=60 && scope<70}">
            您的成绩为及格
        </c:when>
        <c:otherwise>
            对不起，您没能通过考试
        </c:otherwise>
    </c:choose>

</body>
</html>

* 案例二
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
<head>
    <title>JSTL标签演示二</title>
</head>
<body>
<table border="1" cellspacing="0" cellpadding="10" align="center">
    <caption>员工薪水表</caption>
    <tr>
        <th>员工编号</th>
        <th>员工姓名</th>
        <th>职位</th>
        <th>薪水</th>
    </tr>
    <c:forEach items="${emps}" var="emp">
        <tr>
            <td>${emp.id}</td>
            <td>${emp.name}</td>
            <td>${emp.job}</td>
```

```
            <td>${emp.salary}</td>
        </tr>
    </c:forEach>
</table>
<br />
<c:forTokens var="name" items="刘备,关羽,张飞" delims=",">
    ${name}<br>
</c:forTokens>
<br />
</body>
</html>

@WebServlet(value = "/emps")
public class EmpServlet extends HttpServlet {

    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp) thro
        List<Emp> emps=new ArrayList<>();
        emps.add(new Emp(10001,"刘备","皇帝","100两黄金"));
        emps.add(new Emp(10002,"关羽","将军","50两黄金"));
        emps.add(new Emp(10003,"张飞","将军","50两黄金"));
        req.setAttribute("emps",emps);
        req.getRequestDispatcher("/jstl2.jsp").forward(req,resp);
    }

    @Override
    protected void doPost(HttpServletRequest req, HttpServletResponse resp) thr
        doGet(req, resp);
    }
}

* 案例三
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<%@taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<html>
<head>
    <title>JSTL案例三</title>
</head>
<body>
   <%-- <c:import url="http://www.baidu.com" charEncoding="utf-8"></c:import>--
```

```
118     <c:url value="http://www.baidu.com" var="myurl">
119         <c:param name="username" value="xiaohei"></c:param>
120         <c:param name="psw" value="123"></c:param>
121     </c:url>
122     <c:redirect url="${myurl }">
123     </c:redirect>
124 </body>
125 </html>
126
127
```

* 案例一效果

out标签测试

我不是药神

代码给出了给指定scope范围赋值的示例

pageScope:xiaohei
requestScope:
name:xiaohei
pageScope:
requestScope:
name:

catch标签实例

javax.servlet.jsp.JspTagException

if标签演示

登录成功：true

choose标签演示

您的成绩为良好

* 案例二效果

员工薪水表

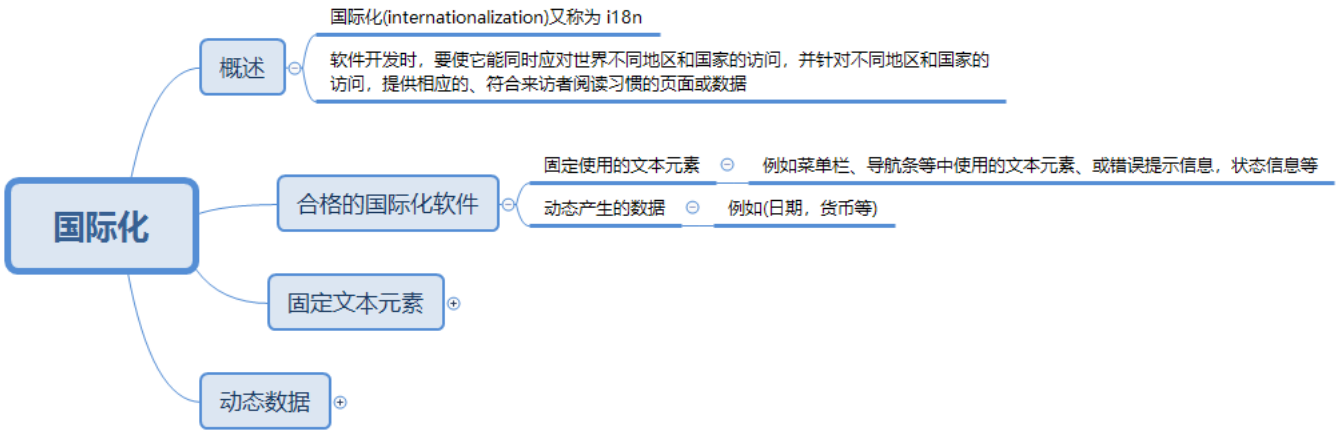
员工编号	员工姓名	职位	薪水
10001	刘备	皇帝	100两黄金
10002	关羽	将军	50两黄金
10003	张飞	将军	50两黄金

刘备
关羽
张飞

* 案例三效果

* 能够掌握Web国际的概念

* 国际化概述



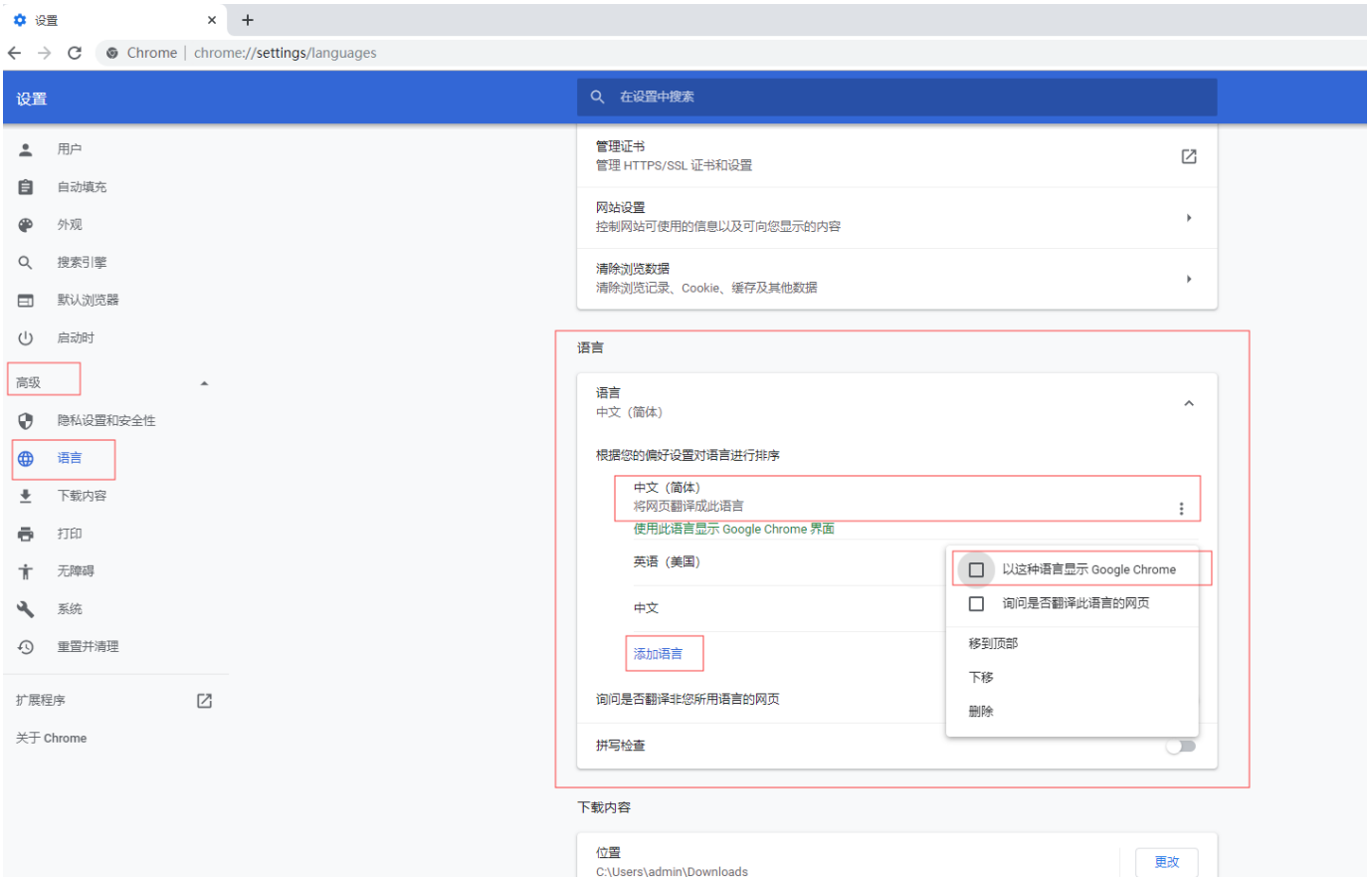
* 固定文本元素


```
* 构建资源文件
  * 在包com.lg.reource下构建三个文件
    * car_zh.properties
     * username=\u5218\u5907
    * car_en.properties
     * username=liubei
    * car.properties（默认）
     * username=xiaohei
  * 代码
    @Test
    public void test1(){
        ResourceBundle cBundle=ResourceBundle.getBundle("com.lg.resource.car",
        String username = cBundle.getString("username");
        System.out.println("username = " + username);
    }
  * 结果：不同地域显示不同文本
    Locale.CHINA: 刘备
    Locale.US: liubei
  * web的代码
<%@ page contentType="text/html;charset=UTF-8" language="java" isELIgnored="fal
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt" %>
<html>
<head>
    <title>国际化演示</title>
    <meta charset="UTF-8">
```

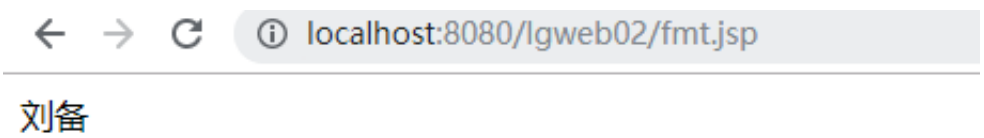
```
</head>
<body>
    <fmt:setBundle basename="com.lg.resource.car"></fmt:setBundle>
    <fmt:message key="username"></fmt:message>
</body>
</html>
```

   * 在chrome切换不同语言环境，会显示不同文本

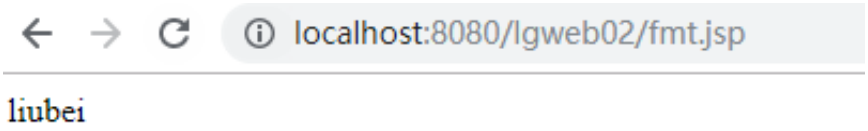
* chrome 切换用户语音环境



* 中文环境



* 英文环境



liubei

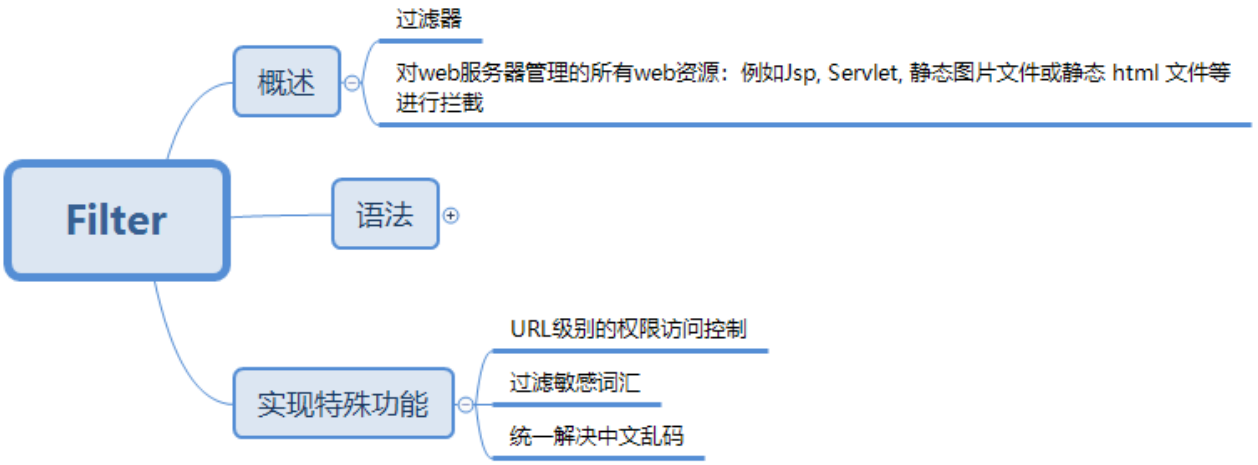
* 动态数据

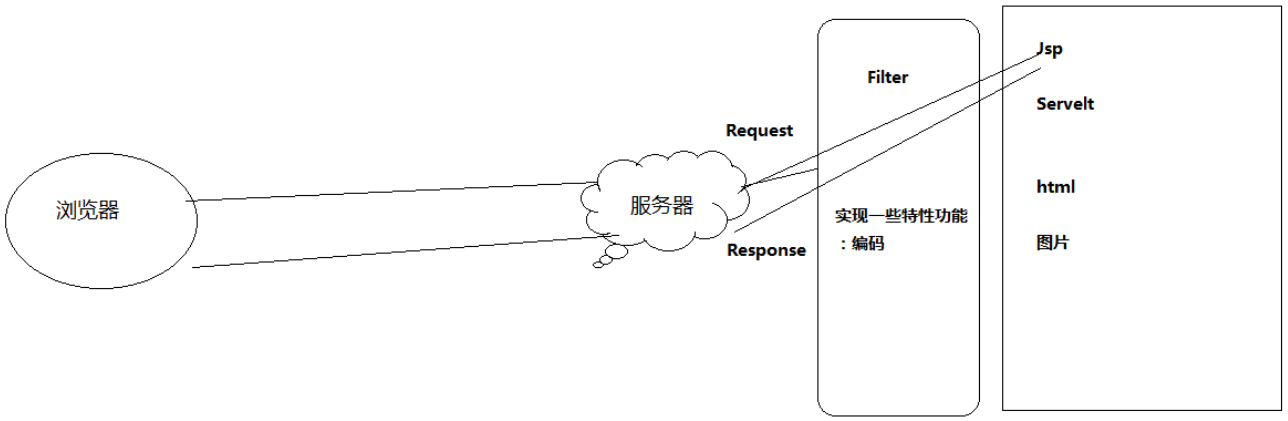

```
* 案例一
   @Test
    public void test2() throws ParseException {
        Date date=new Date();
        // 输出日期部分
        DateFormat dateFormat = DateFormat.getDateInstance(DateFormat.FULL, Loc
        String format = dateFormat.format(date);
        System.out.println("format = " + format);
        // 输出时间部分
        dateFormat=DateFormat.getTimeInstance(DateFormat.FULL, Locale.CHINA);
        format = dateFormat.format(date);
        System.out.println("format = " + format);
        // 输出日期和时间
        dateFormat=DateFormat.getDateTimeInstance(DateFormat.FULL, DateFormat.F
        format = dateFormat.format(date);
        System.out.println("format = " + format);
```

```
17
18          // 把字符串反向解析成一个date对象
19          Date d = dateFormat.parse(format);
20          System.out.println(d);
21      }
22  结果:
23  format = 2019年12月3日 星期二
24  format = 下午10时51分47秒 CST
25  format = 2019年12月3日 星期二 下午10时51分47秒 CST
26  Tue Dec 03 22:51:47 CST 2019
27
28  * 案例二
29   @Test
30      public void test3() throws ParseException {
31          // 金钱格式化
32          int price = 99;
33          NumberFormat nf = NumberFormat.getCurrencyInstance(Locale.CHINA);
34          String format = nf.format(price);
35          System.out.println("format = " + format);
36          // 字符串转换成数字
37          Number nPrice = nf.parse(format);
38          System.out.println(nPrice.intValue());
39
40          // 百分比
41          nf= getPercentInstance(Locale.US);
42          format = nf.format(0.5);
43          System.out.println("format = " + format);
44      }
45   * 结果
46   format = ￥99.00
47   99
48   format = 50%
49
50  * 案例三
51    @Test
52      public void test4() throws ParseException {
53          String pattern = "On {0}, a hurricance destroyed {1} houses and caused
54          MessageFormat format=new MessageFormat(pattern,Locale.CHINA);
55          Object[] arr={new Date(),99,100000};
56          String result = format.format(arr);
```

```
57          System.out.println(result);
58      }
59       @Test
60      public void test5() throws ParseException {
61          String pattern = "At {0, time, long} on {0, date}, a destroyed {1} hous
62          MessageFormat format=new MessageFormat(pattern,Locale.CHINA);
63          Object[] arr={new Date(),99,100000};
64          String result = format.format(arr);
65          System.out.println(result);
66      }
67  * 结果:
68      * On 19-12-4 上午11:04, a hurricance destroyed 99 houses and caused 100,000
69      * At 上午11时04分52秒 on 2019-12-4, a destroyed 99 houses and caused ￥100,0
```

* 能够理解Filter的概述





* 能够掌握Filter的开发


```
* 案例一
* 注解形式
@WebFilter(filterName = "HelloFilter",urlPatterns = "/*")
public class HelloFilter implements Filter {

```

```
    public void init(FilterConfig config) throws ServletException {
        System.out.println("初始化方法");
    }
    public void doFilter(ServletRequest req, ServletResponse resp, FilterChain
        System.out.println("访问Servlet之前会调用");
        chain.doFilter(req, resp);
    }
    public void destroy() {
        System.out.println("服务器停止的时候，调用的");
    }
}
* xml形式
public class HelloFilter2 implements Filter {
    @Override
    public void init(FilterConfig filterConfig) throws ServletException {
        System.out.println("初始化方法2");
    }

    @Override
    public void doFilter(ServletRequest request, ServletResponse response, Filt
        System.out.println("访问Servlet之前会调用2");
        chain.doFilter(request, response);
    }

    @Override
    public void destroy() {
        System.out.println("服务器停止的时候，调用的2");
    }
}
 <filter>
        <filter-name>hello2</filter-name>
        <filter-class>com.lg.filter.HelloFilter2</filter-class>
    </filter>
    <filter-mapping>
        <filter-name>hello2</filter-name>
        <url-pattern>/*</url-pattern>
 </filter-mapping>

* 案例二:
 * @WebFilter(filterName = "HelloFilter",urlPatterns = "/*",initParams = @WebIn
```

```
   public void init(FilterConfig config) throws ServletException {
        System.out.println("初始化方法");
        System.out.println("获得初始化参数"+config.getInitParameter("encoding"))
    }

* 案例三：
 * @WebFilter(filterName = "HelloFilter",urlPatterns = "/*",
          initParams = @WebInitParam(name = "encoding",value = "utf-8"),
          dispatcherTypes={DispatcherType.FORWARD})
```

* 能够使用Filter对URL级别权限访问控制

```
* 需求：没有登录成功之前，只能访问index.jsp,login.jsp,LoginServlet
        登录成功可以访问其他页面
* 在之前的登录成功案例上开发
 * 登录成功后： req.getSession().setAttribute("loginFlag","success");
 * 过滤器代码
@WebFilter(filterName = "LoginFilter",value = "/*",
          initParams = {@WebInitParam(name = "path",value = "/index.jsp,/login.
public class LoginFilter implements Filter {
    String[] paths;
    public void init(FilterConfig config) throws ServletException {
        paths=config.getInitParameter("path").split(",");
    }
    public void doFilter(ServletRequest req, ServletResponse resp, FilterChain
        //没有登录成功之前，只能访问index.jsp,login.html,login2
        //登录成功可以访问其他页面
        HttpServletRequest req1=(HttpServletRequest)req;
        HttpServletResponse resp2=(HttpServletResponse)resp;
        // 获得访问的路径
        String servletPath = req1.getServletPath();
        // 在Session获得登录标识
        Object loginFlag = req1.getSession().getAttribute("loginFlag");
        boolean pass=searchPath(servletPath);
        if(pass){
            chain.doFilter(req, resp);
        }else if(servletPath!=null && servletPath.contains("css") || servletPat
```

```
            // 放行资源
            chain.doFilter(req, resp);
        }else if("success".equals(loginFlag)){
            chain.doFilter(req, resp);
        }
        else{
            // 其他情况跳转到首页
            String contextPath = req1.getServletContext().getContextPath();
            String path=contextPath+"/login.html";
            resp1.sendRedirect(path);
        }
        chain.doFilter(req, resp);
    }

    private boolean searchPath(String servletPath) {
        if(paths!=null){
            for (String path : paths) {
                if(path.equals(servletPath)){
                    return true;
                }
            }
        }
        return false;
    }

    public void destroy() {
    }

}

```

* 能够使用Filter统一解决中文乱码问题

```
* 开发案例
@WebFilter(filterName = "EncodingFilter",value = "/*")
public class EncodingFilter implements Filter {
    public void destroy() {
    }

```

```
    public void doFilter(ServletRequest req, ServletResponse resp, FilterChain
       // 解决响应乱码问题
        resp.setContentType("text/html;charset=utf-8");
        // 解决post提交乱码问题，tomcat8不用处理get乱码
        req.setCharacterEncoding("UTF-8");
        chain.doFilter(req, resp);
    }

    public void init(FilterConfig config) throws ServletException {

    }

}

```

* 能够使用FIlter开发敏感词汇过滤的案例

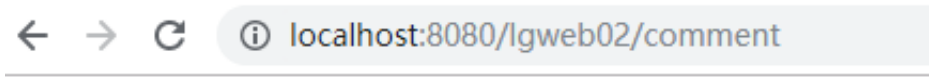
```
* 案例
 * comment.jsp
<%@ page contentType="text/html;charset=UTF-8" language="java" isELIgnored="fal
<html>
<head>
    <title>评论</title>
</head>
<body>
<form action="${pageContext.request.contextPath}/comment " method="post">
    <textarea rows="5" cols="50" name="comment">
        大家好,才是真的好,你妹,好好
    </textarea>
    <input type="submit" value="提交"/>
</form>
</body>
</html>
 * CommentServlet
@WebServlet(name = "CommentServlet",value = "/comment")
public class CommentServlet extends HttpServlet {
    protected void doPost(HttpServletRequest request, HttpServletResponse respo
        String comment=request.getParameter("comment");
        request.setAttribute("comment", comment);
```

```
        request.getRequestDispatcher("/commentResult.jsp").forward(request, res
    }

    protected void doGet(HttpServletRequest request, HttpServletResponse respor

    }
}
 * commentResult.jsp
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
<head>
    <title>显示评论信息</title>
</head>
<body>
    ${comment}
</body>
</html>

* 过滤器(使用装饰者模式)
@WebFilter(filterName = "CommnetFilter",value = "/*")
public class CommnetFilter implements Filter {
    private List<String> words=new CopyOnWriteArrayList<>();
    public void init(FilterConfig config) throws ServletException {
        words.add("你妹");
        words.add("你妈");
        words.add("贱人");
    }
    public void doFilter(ServletRequest req, ServletResponse resp, FilterChain
        // 使用装饰模式
        MHttpServletRequest req1=new MHttpServletRequest((HttpServletRequest) r
        chain.doFilter(req1, resp);
    }
    class MHttpServletRequest extends HttpServletRequestWrapper {

        /**
         * Constructs a request object wrapping the given request.
         *
         * @param request The request to wrap
         * @throws IllegalArgumentException if the request is null
         */
```

```
        public MHttpServletRequest(HttpServletRequest request) {
            super(request);
        }

        @Override
        public String getParameter(String name) {
            // 获得提交上来的内容
            String content = super.getParameter(name);
            if(content!=null) {
                // 处理内容
                for (String word : words) {
                    if (content.contains(word)) {
                        content = content.replace(word, "***");
                    }
                }
            }
            return content;
        }
    }
    public void destroy() {

    }

}

* 过滤器：动态代理
@WebFilter(filterName = "CommnetFilter1",value = "/*")
public class CommnetFilter1 implements Filter {
    private List<String> words=new CopyOnWriteArrayList<>();
    public void init(FilterConfig config) throws ServletException {
        words.add("你妹");
        words.add("你妈");
        words.add("贱人");
    }
    public void doFilter(ServletRequest req, ServletResponse resp, FilterChain
        // 使用动态代理的模式
        HttpServletRequest req1= (HttpServletRequest) req;
        Object proxy=Proxy.newProxyInstance(req1.getClass().getClassLoader(), r
            @Override
            public Object invoke(Object proxy, Method method, Object[] args) th
```

```
103                 String methodName = method.getName();
104                 if("getParameter".equals(methodName)){
105                     // 获取提交的值
106                     String content = (String) method.invoke(req1, args);
107                     if(content!=null) {
108                         for (String word : words) {
109                             if (content.contains(word)) {
110                                 content = content.replace(word, "***");
111                             }
112                         }
113                     }
114                     return content;
115                 }
116                 return method.invoke(req1,args);
117             }
118 
119         });
120         chain.doFilter((ServletRequest) proxy, resp);
121     }
122 
123     @Override
124     public void destroy() {
125 
126     }
127 }
128 
```

* 效果图



大家好,才是真的好,***,好好