- * 学习目标
- * 能够掌握登录的案例
 - * Lombok, MVC, Web的三层架构
 - * bean,dao(impl),service(impl),controller
- *能够掌握Servlet3.0的使用
 - * @WebServlet,@WebInitParams
- * 能够理解请求和响应的概述
 - * Request --- Response
- *能够掌握Request
 - * ServletRequest,HttpServletReuest
- * 能够编写注册案例
- *能够掌握Repsonse

* 能够掌握登录的案例

```
1 * 数据库: mysql-lg01-t user(id,username,password)
 2 * 代码层次
    * com.lg.bean
 3
4
      * User
    * com.lg.dao
 5
 6
      * UserDao
7
       * UserDaoImpl
    * com.lg.service
8
9
      * UserService
10
       * UserServiceImpl
    * com.lg.servlet(controller)
11
      * LoginServlet
12
    * 页面: login.html,success.html,fail.html
13
   * MVC 模式: Model-View-Controller
14
      * 用户发送请求-到控制层(接受用户的数据,调用业务逻辑方法,根据放回结果跳转到不同)
15
   * Web三层架构
16
```

```
17
       * 数据持久层, Web层次(业务逻辑层),页面层
18
19 * 代码
20 @Data
21 @AllArgsConstructor
22 @NoArgsConstructor
   public class User {
23
       private int id;
24
       private String username;
25
       private String password;
26
27
28 }
29 public interface UserDao {
30
        User getUser(String username, String password);
   }
31
32 public class UserDaoImpl implements UserDao {
33
       @Override
34
       public User getUser(String username, String password) {
           String sql="SELECT * FROM t_user WHERE username=? AND password=? ";
35
           QueryRunner runner=new QueryRunner(DataSourceUtils.getDataSource());
36
37
           try {
38
               User user = runner.query(sql, new BeanHandler<User>(User.class), us
39
               return user;
           } catch (SQLException e) {
40
               e.printStackTrace();
41
42
           }
           return null;
43
       }
44
45 }
46
   public interface LoginService {
47
       boolean login(String username, String password);
48
49
       User loginUser(String username, String password);
50
   }
51
52
   public class LoginServiceImpl implements LoginService {
53
54
       private UserDao mUserDao;
       public LoginServiceImpl(){
55
           mUserDao=new UserDaoImpl();
56
```

```
57
       }
       @Override
58
       public boolean login(String username, String password) {
59
           User user = loginUser(username,password);
60
           return user!=null;
61
62
       }
63
       @Override
64
       public User loginUser(@NonNull String username, @NonNull String password) {
65
           User user = mUserDao.getUser(username, password);
66
           return user;
67
       }
68
69 }
70
  @WebServlet(value = "/login")
71
   public class LoginServlet extends HttpServlet {
72
       @Override
73
       protected void doGet(HttpServletRequest req, HttpServletResponse resp) thro
74
          // 1 获取用户从页面输入的数据: 例如表单提交的数据
75
           String username = req.getParameter("username");
76
           String password = req.getParameter("password");
77
78
           System.out.println("username = " + username);
           System.out.println("password = " + password);
79
80
           // 2 调用登录业务--调用dao--查询数据库
81
           LoginService loginService=new LoginServiceImpl();
82
           boolean isOk = loginService.login(username, password);
83
           if(is0k){
84
               getServletContext().setAttribute("msg","login success");
85
               getServletContext().getRequestDispatcher("/success").forward(req,re
86
           }else{
87
               getServletContext().setAttribute("msg","login fail");
88
               getServletContext().getRequestDispatcher("/fail").forward(req,resp)
89
           }
90
       }
91
92
       @Override
93
94
       protected void doPost(HttpServletRequest req, HttpServletResponse resp) thr
           doGet(req,resp);
95
       }
96
```

```
97 }
98 @WebServlet(value = "/success")
99 public class SuccessServlet extends HttpServlet {
        @Override
100
        protected void doGet(HttpServletRequest req, HttpServletResponse resp) thro
101
            // 获得信息,响应给页面
102
            String msg = (String) getServletContext().getAttribute("msg");
103
104
            resp.getWriter().write(msg);
105
        }
106
107
        @Override
        protected void doPost(HttpServletRequest req, HttpServletResponse resp) thr
108
           doGet(req, resp);
109
110
        }
111 }
112
113 @WebServlet(value = "/fail")
114 public class FailServlet extends HttpServlet {
115
        @Override
        protected void doGet(HttpServletRequest req, HttpServletResponse resp) thro
116
            // 获得信息,响应给页面
117
            String msg = (String) getServletContext().getAttribute("msg");
118
119
            resp.getWriter().write(msg);
120
        }
121
122
        @Override
        protected void doPost(HttpServletRequest req, HttpServletResponse resp) thr
123
           doGet(req, resp);
124
125
        }
126 }
127
128
```

^{*}能够掌握Servlet3.0的使用

String name() default "";// servlet-name

String[] value() default {};//url-pattern的简单写法

String[] urlPatterns() default {};//url-pattern的写法

int loadOnStartup() default -1;// tomcat启动时候 , 实例化Servlet并且调用init

WebInitParam[] initParams() default {};//配置初始化 参数

常用的方法

WebServlet注解

WebInitParam注解

String name();// 初始化参数的名字 String value();// 初始化参数的值

```
1 案例一:
  @WebServlet(name="hello",value= {"/hello"})
  public class HelloServlet extends HttpServlet{
 4
 5
       @Override
       protected void doGet(HttpServletRequest req, HttpServletResponse resp) thro
 6
 7
           resp.getWriter().append("Hello Servlet3.0");
       }
 8
 9
       @Override
10
       protected void doPost(HttpServletRequest req, HttpServletResponse resp) thr
11
12
       }
13 }
14
15 案例二:
  @WebServlet(name="hello",urlPatterns= {"/hello"})
16
   public class HelloServlet extends HttpServlet
17
18
19 案例三:
   @WebServlet(name="hello", value= {"/hello"}, loadOnStartup=1)
   public class HelloServlet extends HttpServlet{
21
22
23
       @Override
       public void init(ServletConfig config) throws ServletException {
24
25
           super.init(config);
           System.out.println("HelloServlet init 3.00000000");
26
27
       }
```

```
28
       @Override
29
       protected void doGet(HttpServletRequest req, HttpServletResponse resp) thro
           resp.getWriter().append("Hello Servlet3.0");
30
       }
31
32
33
       @Override
34
       protected void doPost(HttpServletRequest req, HttpServletResponse resp) thr
35
36 }
37
```

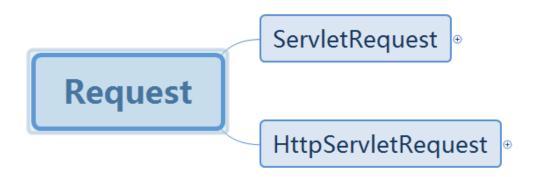
* 案例四

```
Markers Properties Servers Data Source Explorated v8.0 Server at localhost [Apache Tomcat] C:\Program Fixiaobai
```

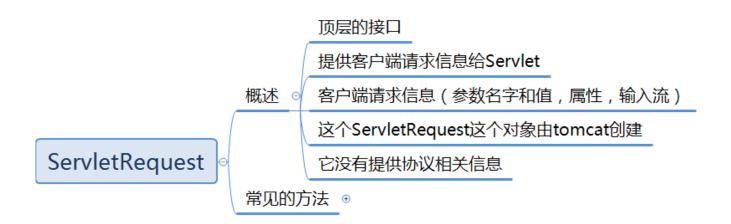
* 能够理解请求和响应的概述



*能够掌握Request



- * ServletRequest
 - * 概述



* 常见方法

```
远程地址 • public String getRemoteAddr();
            远程主机 Θ
                       public String getRemoteHost();
            远程端口号 ⊙ public int getRemotePort();
            请求体长度 ⊙ public int getContentLength();
            请求MIME类型 🕤 public String getContentType();
            请求体输入流 🏻 public ServletInputStream getInputStream();
                    获取请求参数 🌼 public String getParameter(String name);
                    获取请求参数名字 ◎ public Enumeration<String> getParameterNames();
            参数
                    获取请求参数(数组) 🏻 public String[] getParameterValues(String name);
常见的方法
                    获取请求参数(Map) 🌼 public Map<String, String[]> getParameterMap();
                    获取在Request范围内容属性值 ◎ public Object getAttribute(String name);
                    设置Request范围内容属性值 o public void setAttribute(String name, Object o);
            属性
                    移除Request范围内容属性值 o public void removeAttribute(String name);
                    获取在Request范围内容属性值集合 ◎ public Enumeration<String> getAttributeNames();
                                   public String getCharacterEncoding();
            编码
                    设置请求编码,可以解决post中文乱码问题 🂿 public void setCharacterEncoding(String env);
                         public ServletContext getServletContext();
            获取上下文 ⊕
                             public RequestDispatcher
            获取请求转发器 🏻 getRequestDispatcher(String path);
```

* HttpServletRequest

```
基础ServletRequest
                                提供Http协议相关信息
                        概述
                                它有tomcat创建对象,并且作为参数传递Serlvet的
                                doGet , doPost
                                     通过名字获取请求头的内容 💿 public String getHeader(String name);
                                     请求头的名字集合 o public Enumeration < String > getHeaderNames();
HttpServletRequest
                                     请求方法 o public String getMethod();
                                     请求上下文的路径 • public String getContextPath();
                         常见的方法
                                     查询字符 o public String getQueryString();
                                     请求URL o public StringBuffer getRequestURL();
                                     请求URI o public String getRequestURI();
                                     获取请求的Cookie值 ⊚ public Cookie[] getCookies();
                                                   public HttpSession getSession();
                                     获取Session ⊖
```

```
* 案例一

protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws

String remoteAddr = req.getRemoteAddr();

String remoteHost = req.getRemoteHost();

int remotePort = req.getRemotePort();
```

```
6
           System.out.println("remoteAddr = " + remoteAddr);
7
           System.out.println("remoteHost = " + remoteHost);
           System.out.println("remotePort = " + remotePort);
8
9
           int contentLength = req.getContentLength();
10
11
           System.out.println("contentLength = " + contentLength);
           String contentType = req.getContentType();
12
           System.out.println("contentType = " + contentType);
13
             ServletInputStream is = req.getInputStream();
14 //
             byte[] b=new byte[1024];
15 //
16 //
             int len=is.read(b);
             String msg=new String(b,0,len);
17 //
             System.out.println("msg = " + msg);
18 //
19
           String username = req.getParameter("username");
20
           String password = req.getParameter("password");
21
           System.out.println("username = " + username);
22
           System.out.println("password = " + password);
23
             String[] hobbies = req.getParameterValues("hobby");
24 //
            for (String hobby : hobbies) {
25 //
                 System.out.println("hobby = " + hobby);
26 //
27 //
             }
           Enumeration<String> names = req.getParameterNames();
28
           while (names.hasMoreElements()){
29
               System.out.println(names.nextElement());
30
31
           }
           Map<String, String[]> map = req.getParameterMap();
32
           Set<Map.Entry<String, String[]>> entries = map.entrySet();
33
           for(Map.Entry<String, String[]> entry:entries){
34
               System.out.println(entry.getKey()+":"+entry.getValue()[0]);
35
           }
36
37
       }
38 * 案例二:
39 public class MBeanUtils {
       public static void populate(HttpServletRequest req,Object obj){
40
           Map<String,String> map=new HashMap<String,String>();
41
           Enumeration<String> names = req.getParameterNames();
42
43
           while(names.hasMoreElements()){
               String name=names.nextElement();
44
               String value=req.getParameter(name);
45
```

```
46
               map.put(name, value);
           }
47
           try {
48
               BeanUtils.populate(obj,map);
49
           } catch (IllegalAccessException e) {
50
51
               e.printStackTrace();
           } catch (InvocationTargetException e) {
52
               e.printStackTrace();
53
           }
54
55
       }
56 }
57
    @Override
58
59
   protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws S
           User user=new User();
60
           MBeanUtils.populate(req,user);
61
           System.out.println("user = " + user);
62
63
    }
64
65
  * 案例三:
66
67
     @Override
       protected void doGet(HttpServletRequest req, HttpServletResponse resp) thro
68
           User user=new User();
69
           MBeanUtils.populate(req,user);
70
           System.out.println("user = " + user);
71
           boolean isRegOk=true;
72
           if(isRegOk){
73
               req.setAttribute("msg", "reg success");
74
               req.getRequestDispatcher("/success").forward(req,resp);
75
           }else{
76
               req.setAttribute("msg", "reg fail");
77
               req.getRequestDispatcher("/fail").forward(req,resp);
78
           }
79
       }
80
   * 案例四:
81
     @Override
82
83
       protected void doGet(HttpServletRequest req, HttpServletResponse resp) thro
             String header = req.getHeader("Accept-Encoding");
84 //
           Enumeration<String> headerNames = req.getHeaderNames();
85
```

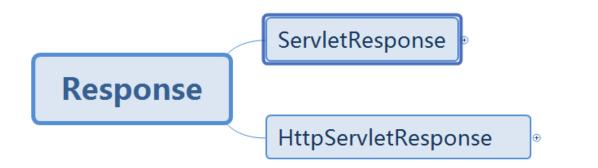
```
86
           while(headerNames.hasMoreElements()){
               String name=headerNames.nextElement();
87
               String value=req.getHeader(name);
88
               System.out.println(name+":"+value);
89
90
           }
             System.out.println("header = " + header);
91
           System.out.println(req.getMethod());
92
           System.out.println(req.getRequestURL());
93
           System.out.println(req.getRequestURI());
94
           System.out.println(req.getContextPath());
95
           System.out.println(req.getQueryString());
96
97
       }
```

* 能够编写注册案例

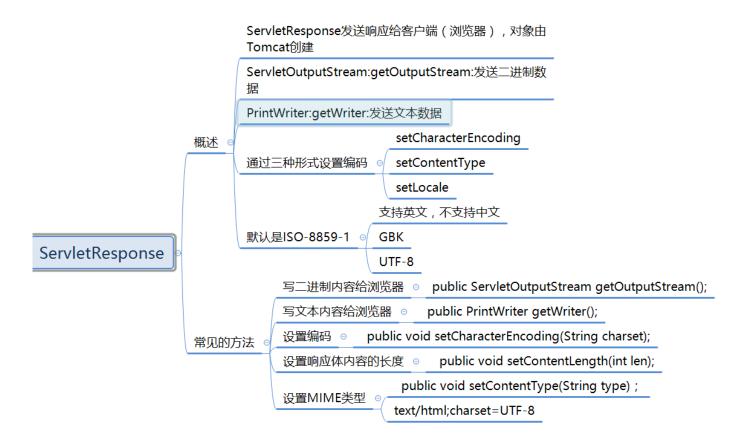
* 作业



*能够掌握Repsonse



* ServletResponse



* HttpServletResponse

```
添加Cookie给浏览器 。 public void addCookie(Cookie cookie);

编码url 。 public String encodeURL(String url);

重定向 。 public void sendRedirect(String location);

设置响应头 。 public void setHeader(String name, String value);

添加响应头 。 public void addHeader(String name, String value);

设置状态码 。 public void setStatus(int sc);/
```

```
案例一:
1
2
   @Override
3
      protected void doGet(HttpServletRequest req, HttpServletResponse resp) thro
            // ISO-8859-1
4
5
  //
           resp.setCharacterEncoding("GBK");
          resp.setContentType("text/html;charset=UTF-8");
6
          resp.getWriter().write("亮哥教育,做教育我们是认真的");
8
      }
```