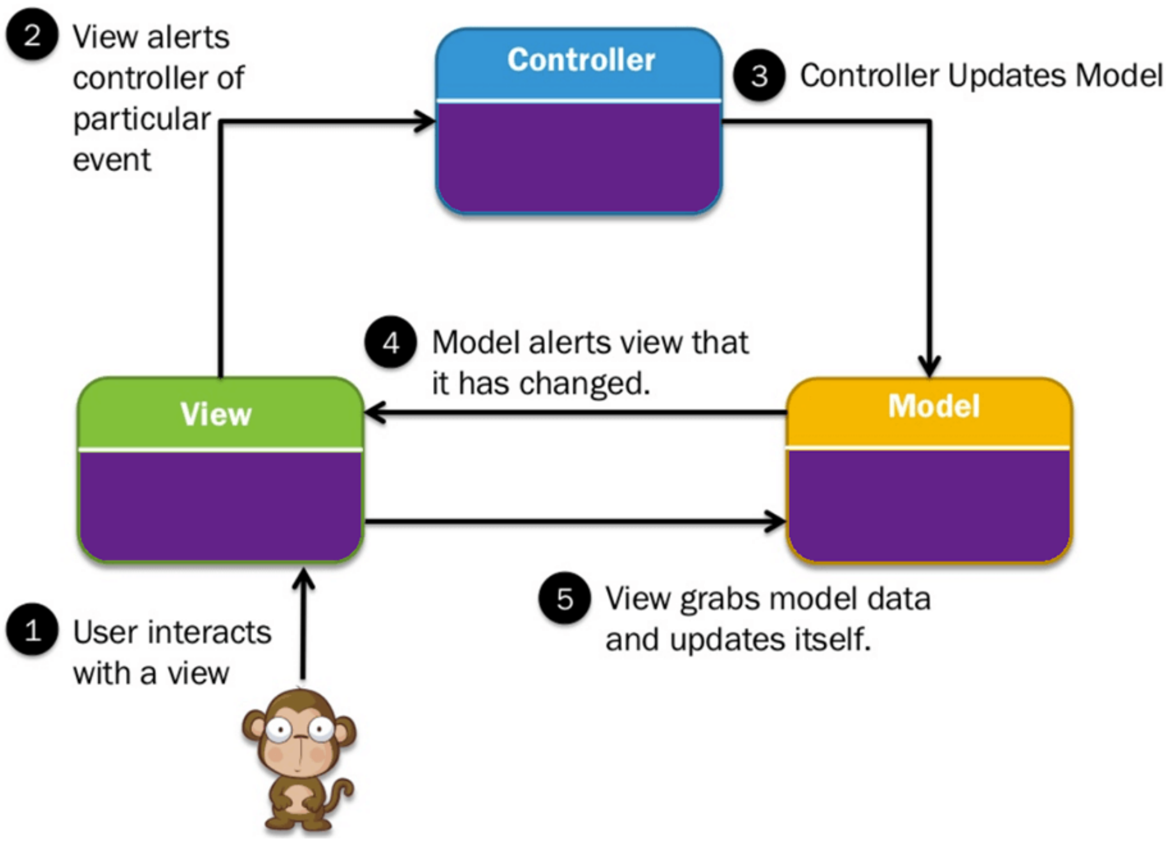
MVC学习笔记

认识

模型（Model）**、视图**（View）**和控制器**（Controller）**。**

- **模型**（Model） - **程序员编写程序应有的功能（实现算法等等）、数据库专家进行数据管理和数据库设计(可以实现具体的功能)。**
- **视图**（View） - **界面设计人员进行图形界面设计。**
- **控制器**（Controller） - **负责转发请求，对请求进行处理。**


2 View alerts controller of particular event
Controller
3 Controller Updates Model
4 Model alerts view that it has changed.
View
Model
5 View grabs model data and updates itself.
1 User interacts with a view

Filter拦截器

```
public class FilterTest implements Filter {
    @Override
    public void init(FilterConfig filterConfig) throws
ServletException {
        System.out.println("创建");
    }

    @Override
    public void doFilter(ServletRequest request, ServletResponse
response, FilterChain chain) throws IOException, ServletException
{
        request.setCharacterEncoding("utf-8");
```

```java
        response.setCharacterEncoding("utf-8");
        response.setContentType("text/html;charset=utf-8");
        chain.doFilter(request,response);
    }

    @Override
    public void destroy() {
        System.out.println("销毁");
    }
}

public class Show extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest req,
HttpServletResponse resp) throws ServletException, IOException {
        resp.getWriter().write("你好，世界");
    }

    @Override
    protected void doPost(HttpServletRequest req,
HttpServletResponse resp) throws ServletException, IOException {
        doGet(req, resp);
    }
}
```

```xml
<servlet>
        <servlet-name>show</servlet-name>
        <servlet-class>com.wll.servlet.Show</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>show</servlet-name>
        <url-pattern>/show</url-pattern>
    </servlet-mapping>
    <filter>
        <filter-name>filter</filter-name>
        <filter-class>com.wll.filter.FilterTest</filter-class>
    </filter>
    <filter-mapping>
        <filter-name>filter</filter-name>
        <!--过滤请求的 /show  页面-->
        <url-pattern>/show</url-pattern>
        <!--过滤请求的 /  下所有页面-->
        <url-pattern>/*</url-pattern>
</filter-mapping>
```

Filter实现注销功能

login.jsp（admin） → user.jsp → **注销** → loginjsp

- **注销后地址栏输入**user.jsp**不能进入，返回**404.jsp

```
<!--login.jsp-->
<!--避免路径问题，解决使用相对路径时出现的问题-->
<form action="${pageContext.request.contextPath}/servlet/login"
method="post">
    <input type="text" name="username">
    <input type="submit">
</form>

<!--user.jsp-->
<body
background="${pageContext.request.contextPath}/statics/pictures/su
ccess.png" style="background-size: 100% 100%">
    <a
href="${pageContext.request.contextPath}/servlet/login.jsp">返回登
录</a>
    <form
action="${pageContext.request.contextPath}/servlet/logout"
method="post">
        <input type="submit">
    </form>
</body>

<!--404.jsp-->
<body
background="${pageContext.request.contextPath}/statics/pictures/40
4.png" style="background-size: 100% 100%">
```

```
//Const
public class Const {
    public final static String USER_SESSION = "USER_SESSION";
}

//login servlet
@Override
protected void doGet(HttpServletRequest req, HttpServletResponse
resp) throws IOException {
    String username = req.getParameter("username");
    //如果text为admin，session中存放数据，返回用户界面，否则返回404界面
    if(username.equals("admin")){

 req.getSession().setAttribute(Const.USER_SESSION,req.getSession()
.getId());
        resp.sendRedirect("/filter_war/sys/user.jsp");
    }else {
        resp.sendRedirect("/filter_war/error-page/404.jsp");
    }
}

@Override
protected void doPost(HttpServletRequest req, HttpServletResponse
resp) throws IOException {
    doGet(req, resp);
}

//logout servlet
@Override
protected void doGet(HttpServletRequest req, HttpServletResponse
resp) throws ServletException, IOException {
    Object username =
req.getSession().getAttribute(Const.USER_SESSION);
    //session中数据不为空则移除session中的数据并返回登录界面
    if(username≠null){
        //移除session中的数据
        req.getSession().removeAttribute(Const.USER_SESSION);
        resp.sendRedirect("/filter_war/servlet/login.jsp");
    }
    //        else {
    //            //session中数据为空则返回错误界面，使用过滤器则不需要
    //            resp.sendRedirect("/filter_war/error-
page/404.jsp");
    //        }
}

@Override
protected void doPost(HttpServletRequest req, HttpServletResponse
resp) throws ServletException, IOException {
    doGet(req, resp);
}
```

```
<!—登录界面—>
<servlet>
    <servlet-name>Login</servlet-name>
    <servlet-class>com.wll.servlet.Login</servlet-class>
</servlet>
<servlet-mapping>
    <servlet-name>Login</servlet-name>
    <url-pattern>/servlet/login</url-pattern>
</servlet-mapping>
<!—注销页面—>
<servlet>
    <servlet-name>logout</servlet-name>
    <servlet-class>com.wll.servlet.Logout</servlet-class>
</servlet>
<servlet-mapping>
    <servlet-name>logout</servlet-name>
    <url-pattern>/servlet/logout</url-pattern>
</servlet-mapping>
<!-- 登录过滤-->
<filter>
    <filter-name>success</filter-name>
    <filter-class>com.wll.filter.LoginFilter</filter-class>
</filter>
<filter-mapping>
    <filter-name>success</filter-name>
    <url-pattern>/sys/*</url-pattern>
</filter-mapping>
```

HttpSessionListener监听器

```
@Override
//session创建触发此方法
public void sessionCreated(HttpSessionEvent se) {
    ServletContext session = se.getSession().getServletContext();
    Integer peopleNumber = (Integer)
session.getAttribute("peopleNumber");
    if(peopleNumber==null){
        peopleNumber = 1;
    }else{
        peopleNumber++;
    }
    session.setAttribute("peopleNumber",peopleNumber);
}

@Override
//session销毁触发此方法
public void sessionDestroyed(HttpSessionEvent se) {
    ServletContext session = se.getSession().getServletContext();
    Integer peopleNumber = (Integer)
session.getAttribute("peopleNumber");
    if(peopleNumber==null){
        peopleNumber = 0;
    }else{
        peopleNumber--;
    }
    session.setAttribute("peopleNumber",peopleNumber);
}

//login filter
@Override
```

```java
public void init(FilterConfig filterConfig) throws
ServletException {

}

@Override
public void doFilter(ServletRequest request, ServletResponse
response, FilterChain chain) throws IOException, ServletException
{
    HttpServletRequest req = (HttpServletRequest) request;
    HttpServletResponse resp = (HttpServletResponse) response;
    //如果session中数据为空，则过滤到404页面
    if(req.getSession().getAttribute(Const.USER_SESSION)==null){
        resp.sendRedirect("/filter_war/error-page/404.jsp");
    }
    chain.doFilter(request,response);
}

@Override
public void destroy() {

}
```

```xml
<listener>
    <listener-class>com.wll.listener.ListenerTest</listener-class>
</listener>
<!--设置session自动销毁-->
<session-config>
    <session-timeout>1</session-timeout>
</session-config>
```

以下两种情况下就会发生会话销毁事件：

1. **手动销毁：执行**session.invalidate()**方法时。**
2. **自动销毁：如果用户长时间没有访问服务器，超过了会话最大超时时间，服务器就会自动销毁超时的**session**；**session-config**配置。**