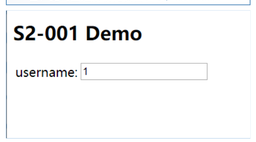
```
if ((this.username.equalsIgnoreCase( anotherString: "admin"))
        && (this.password.equals("password"))) {
    return SUCCESS;
}
return ERROR;
```

最外层程序员编写的代码（LoginAction.java）
这里会返回ERROR

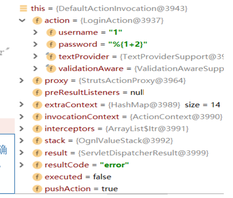
标签解析的过程其实一直在做，在调试的过程中是在页面显示之前都会对所有的值包括页面名称等信息都进行一次解析，解析完一个数据，显示一个数据例如下图，就是在解析password过程中，截得图。此时已经解析完页面名称，username。




```
private void executeResult() throws Exception {
    this.result = this.createResult();
    String timerKey = "executeResult: " + this.getResultCode();  timerKey: "executeResult: error"

    try {
        UtilTimerStack.push(timerKey);  timerKey: "executeResult: error"
        if (this.result != null) {
            this.result.execute( actionInvocation: this);  result: ServletDispatcherResult@4185
```

jar里的代码（DefaultActionInvcation.class），正如之前了解到的那样，在s2框架中的验证程序，若验证正确会根据用户写的代码进行操作，而若验证错误会保持在原页面，以及原样显示用户输入（正常情况下）。
前面的步骤就是在判断验证到底是正确还是错误。在红框这里进入成功或是失败后的操作。




```
    public void execute(ActionInvocation invocation) throws Exception {  invocation
        this.lastFinalLocation = this.conditionalParse(this.location, invocation);
        this.doExecute(this.lastFinalLocation, invocation);  invocation: DefaultAct
    }
```

这里就是判断如果验证通过会到哪个界面，而如果是验证错误就是留在当前页面。

```
public void doExecute(String finalLocation, ActionInvocation invocation) throws Exception {  finalLocation: "/index.jsp"  invocation: DefaultActionInvocation@3940
    if (log.isDebugEnabled()) {
        log.debug( o: "Forwarding to location " + finalLocation);
    }

    PageContext pageContext = ServletActionContext.getPageContext();
    if (pageContext != null) {
        pageContext.include(finalLocation);
    } else {
        HttpServletRequest request = ServletActionContext.getRequest();
        HttpServletResponse response = ServletActionContext.getResponse();
        RequestDispatcher dispatcher = request.getRequestDispatcher(finalLocation);
        if (dispatcher == null) {
            response.sendError(404, "result '" + finalLocation + "' not found");
            return;
        }

        if (!response.isCommitted() && request.getAttribute("javax.servlet.include.servlet_path") == null) {
            request.setAttribute("struts.view_uri", finalLocation);
            request.setAttribute("struts.request_uri", request.getRequestURI());
            dispatcher.forward(request, response);
```

页面之间传值

Xwork.jar-util-TextParseUtil.class文件中开始对可能存在的标签进行解析，通过UIBean.class对当前页面即index.jsp的s标签进行解析（通过调试与其说是对index.jsp里的标签进行解析，不如说是对任何传入的数据都进行了表达式判断包括/index.php这些字段）。解析顺序依次为index、username、%{username}、password、%{password}。正常情况下解析应该在这里结束，但实际上当password的内容也为ognl如本例中%{1+2}时，还会进一步执行计算%{1+2}=3。

```
public static Object translateVariables(char open, String expression, ValueStack stack, Class asType, TextParseUtil.ParsedValueEvaluator evaluator) {
    Object result = expression;  result: "%{1+2}"

    while(true) {
        int start = expression.indexOf(open + "{");  start: 0  open: '%' 37
        int length = expression.length();  length: 6
        int x = start + 2;  x: 6
        int count = 1;  count: 0

        while(start != -1 && x < length && count != 0) {  length: 6
            char c = expression.charAt(x++);
            if (c == '{') {
                ++count;
            } else if (c == '}') {
                --count;
            }
        }

        int end = x - 1;  end: 5  x: 6
        if (start == -1 || end == -1 || count != 0) {  count: 0
            return XWorkConverter.getInstance().convertValue(stack.getContext(), result, asType);  result: "%{1+2}"
        }

        String var = expression.substring(start + 2, end);  var: "1+2"
        Object o = stack.findValue(var, asType);  o: "3"  stack: OgnlValueStack@3982  var: "1+2"  asType: "class java.lang.String"
        if (evaluator != null) {
            o = evaluator.evaluate(o);  evaluator: null
        }

        String left = expression.substring(0, start);  left: ""  start: 0
        String right = expression.substring(end + 1);  right: ""  expression: "%{1+2}"  end: 5
        if (o != null) {  o: "3"
            if (TextUtils.stringSet(left)) {
                result = left + o;
            } else {
                result = o;
            }

            if (TextUtils.stringSet(right)) {
                result = result + right;
            }

            expression = left + o + right;
        } else {
            result = left + right;
            expression = left + right;
        }
    }
}
```

确定表达式为完整的ognl表达式

循环的出口，当最后的结果不为ognl表达式即count!=0时就会跳出循环，再到下一个字段的判断。

计算ognl表达式的值，最后将表达式的值赋给result和expression