性能测试

--性能测试工具LoadRunner使用—关联、事务和检查点

内容回顾

- 函数和变量
  - 四类函数：
  - 基于协议（如web开头、通用lr开头、lr.开头：Java和.net中使用、自定义函数）
  - 两种变量
    - 全局变量：在globals.h中定义的变量是全局变量
    - 局部变量：在init、action、end中定义的变量

内容回顾

- 什么是参数化
  - 用参数替代常量，可更加真实的模拟实际用户操作并简化脚本
- 为什么进行参数化
  - 每次使用的数据不同，更方便脚本执行
- 怎样进行参数化
  - 右键替换并设置参数属性
  - 参数列表、类型、读取下一行数据的方式，参数值更新方式

内容回顾

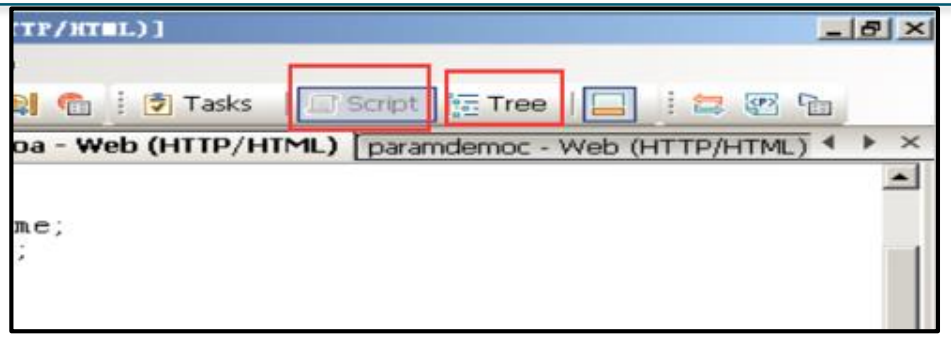
- 读取下一行数据和参数值更新方式
  - 顺序+迭代（参数多余，正好等于，大于）
    - 注意：只有run上设置的迭代才更新参数值
  - 顺序+每次取值更新
  - 顺序+ Once(不变)
  - 随机 + 迭代
  - 唯一 + 迭代

目录

- 脚本查看方式（不同视图）
- 运行结果查看
- 关联
- 事务
- 检查点

脚本查看方式

- 脚本视图（Script View）

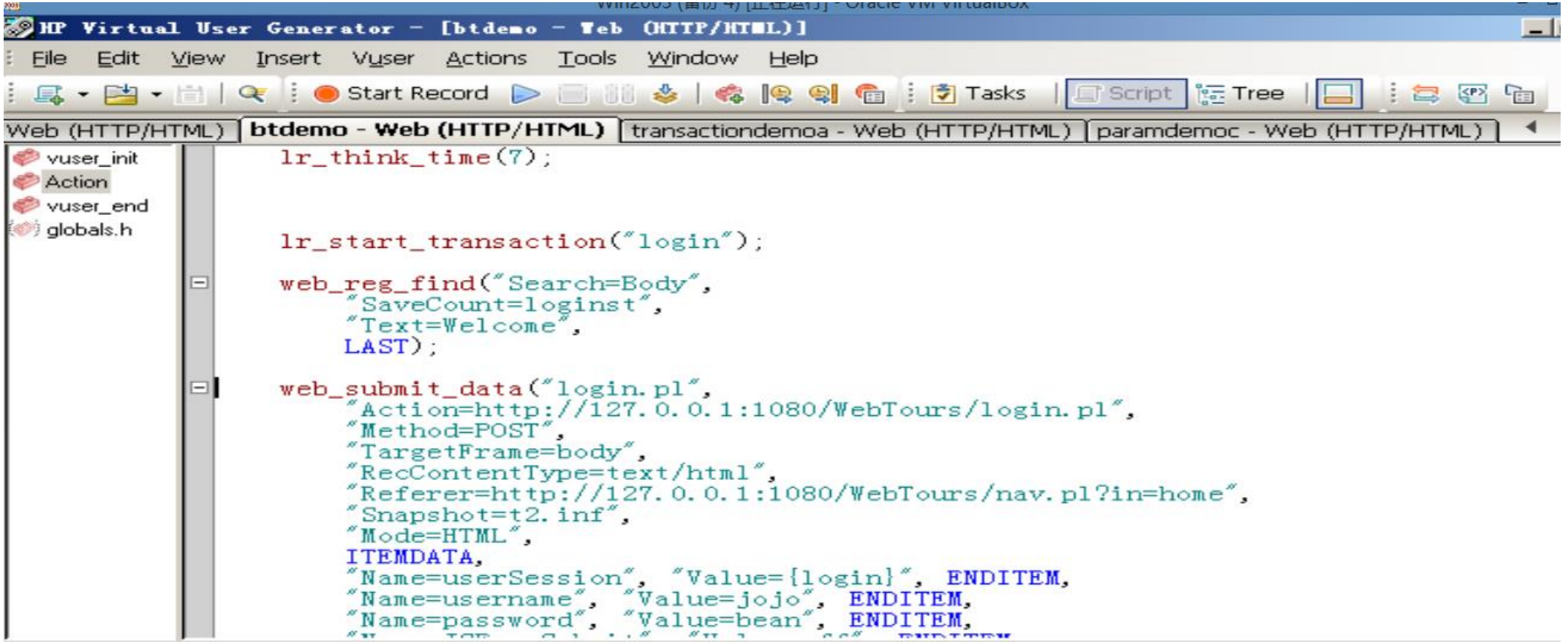


  - 基于文本视图
  - VuGen将在编辑器中显示带有颜色编码的函数及其变量值的脚本

- 树视图（Tree View）
  - 基于图标的视图
  - VuGen将在测试树中为录制期间所执行的每一步骤生成一个图标和标题
  - 大多步骤有快照

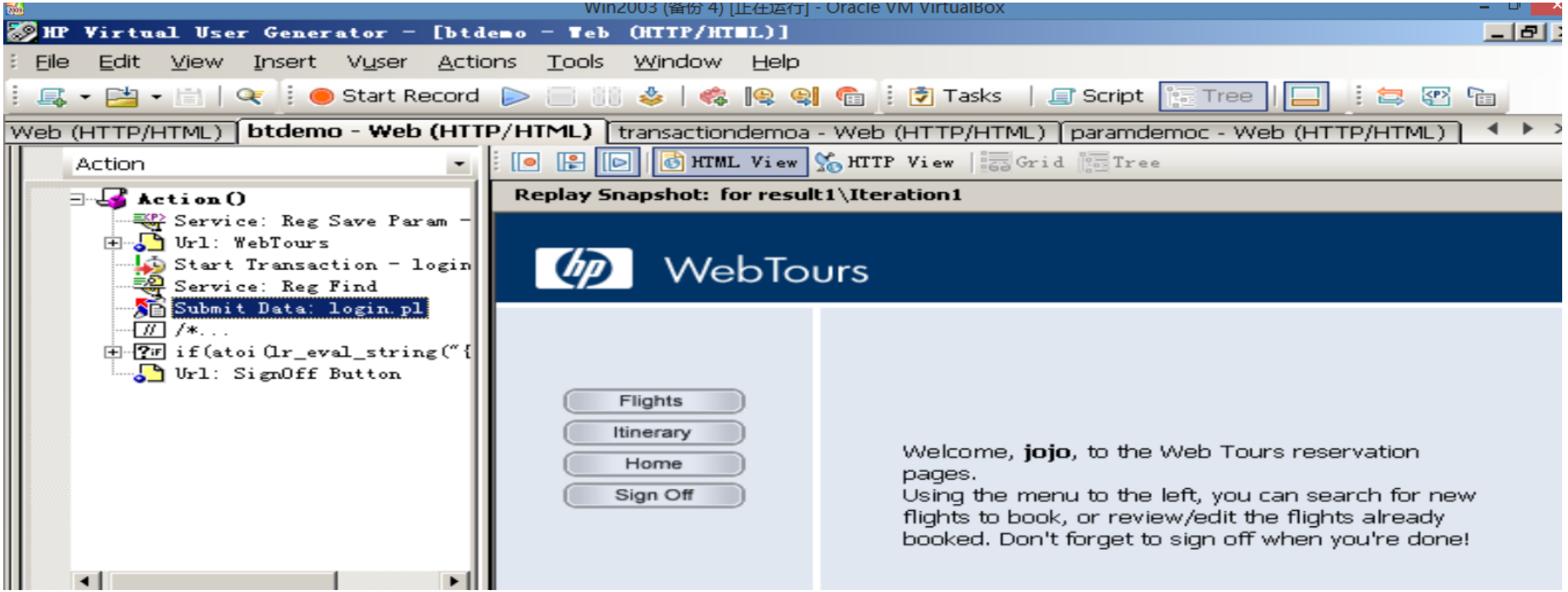
脚本查看方式

脚本视图


HP Virtual User Generator - [btdemo - Web (HTTP/HTML)]
File Edit View Insert Vuser Actions Tools Window Help
Start Record
Tasks
Script
Tree
Web (HTTP/HTML)
btdemo - Web (HTTP/HTML)
transactiondemoa - Web (HTTP/HTML)
paramdemoc - Web (HTTP/HTML)
vuser_init
Action
vuser_end
globals.h
lr_think_time(7);
lr_start_transaction("login");
web_reg_find("Search=Body",
"SaveCount=loginst",
"Text=Welcome",
LAST);
web_submit_data("login.pl",
"Action=http://127.0.0.1:1080/WebTours/login.pl",
"Method=POST",
"TargetFrame=body",
"RecContentType=text/html",
"Referer=http://127.0.0.1:1080/WebTours/nav.pl?in=home",
"Snapshot=t2.inf",
"Mode=HTML",
ITEMDATA,
"Name=userSession", "Value={login}", ENDITEM,
"Name=username", "Value=jojo", ENDITEM,
"Name=password", "Value=bean", ENDITEM,

脚本查看方式

- 树视图


Win2003 (备份 4) [正在运行] - Oracle VM VirtualBox
HP Virtual User Generator - [btdemo - Web (HTTP/HTML)]
File Edit View Insert Vuser Actions Tools Window Help
Start Record
Tasks
Script
Tree
Web (HTTP/HTML)
btdemo - Web (HTTP/HTML)
transactiondemoa - Web (HTTP/HTML)
paramdemoc - Web (HTTP/HTML)
Action
Action()
Service: Reg Save Param -
Url: WebTours
Start Transaction - login
Service: Reg Find
Submit Data: login.pl
//*...
if(atoi(lr_eval_string("{
Url: SignOff Button
HTML View
HTTP View
Grid
Tree
Replay Snapshot: for result1\Iteration1
WebTours
Flights
Itinerary
Home
Sign Off
Welcome, jojo, to the Web Tours reservation pages.
Using the menu to the left, you can search for new flights to book, or review/edit the flights already booked. Don't forget to sign off when you're done!

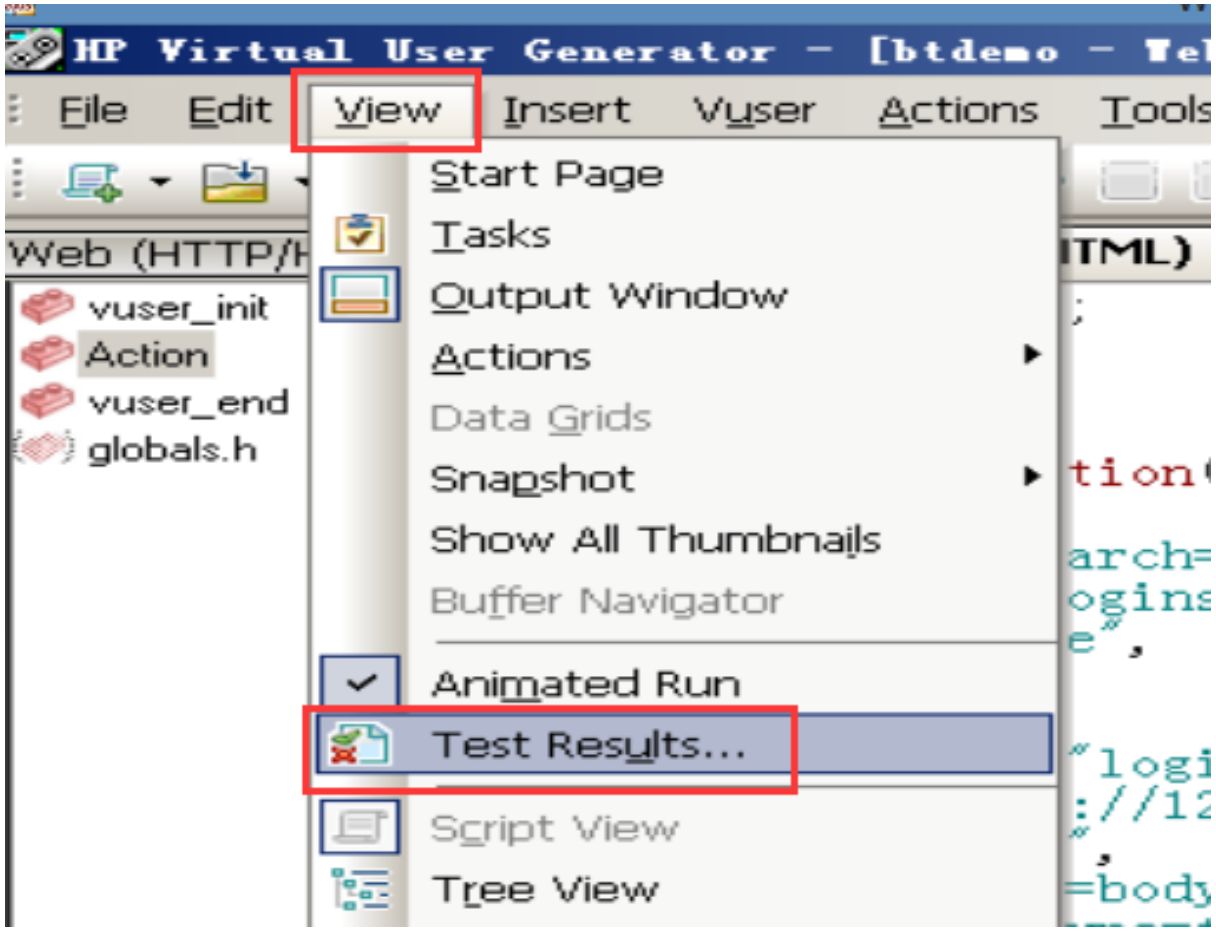
目录

- 脚本查看方式（不同视图）
- 运行结果查看
- 关联
- 事务
- 检查点

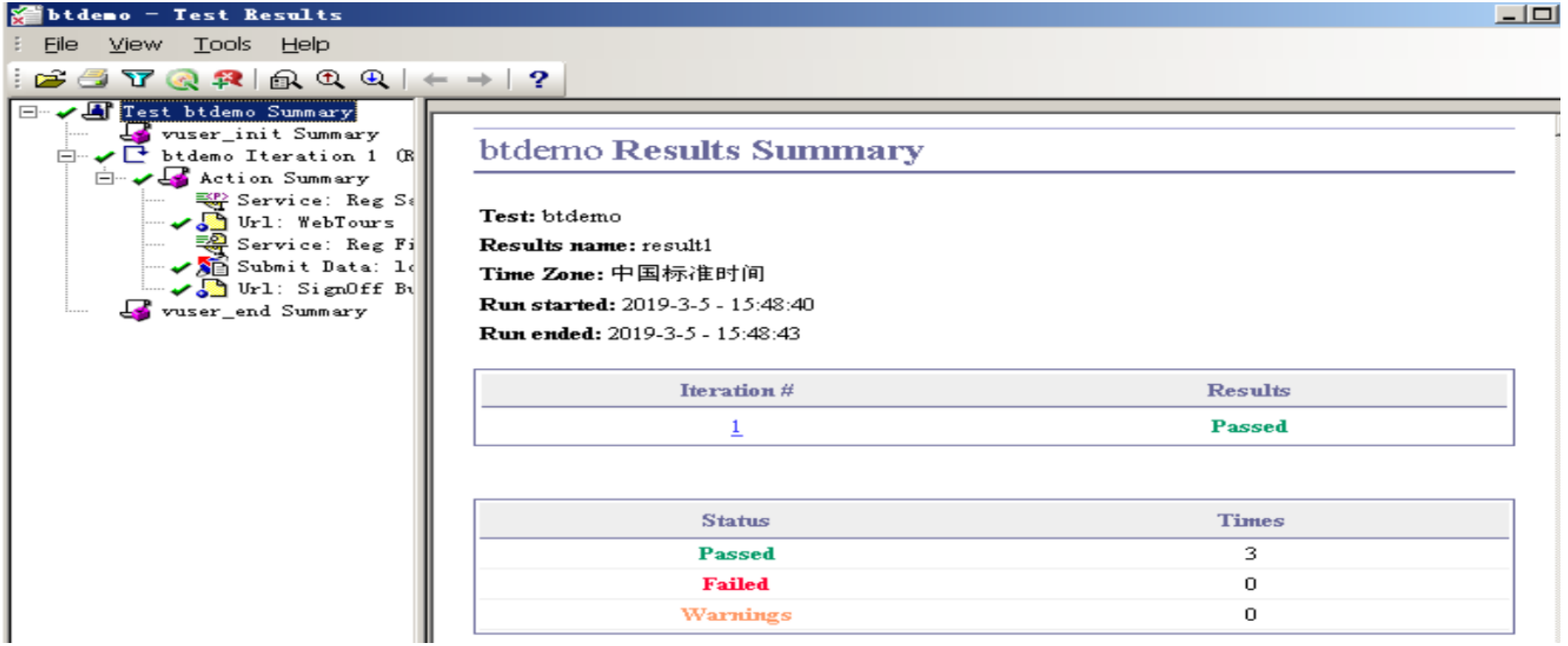
运行结果查看

- 脚本运行后，点击

View—Test Results



运行结果查看


btdemo - Test Results
File View Tools Help
Test btdemo Summary
vuser_init Summary
btdemo Iteration 1
Action Summary
Service: Reg
Url: WebTours
Service: Reg
Submit Data:
Url: SignOff
vuser_end Summary
btdemo Results Summary
Test: btdemo
Results name: result1
Time Zone: 中国标准时间
Run started: 2019-3-5 - 15:48:40
Run ended: 2019-3-5 - 15:48:43
Iteration # Results
1 Passed
Status Times
Passed 3
Failed 0
Warnings 0

思考

- 录制飞机订票系统中，使用正确的用户名和密码登录，为什么回放时，结果不成功
  - 服务器有动态生成的数据返回给客户端
  - 动态数据怎样拿到
  - 关联

目录

- 脚本查看方式（不同视图）
- 运行结果查看
- 关联
- 事务
- 检查点

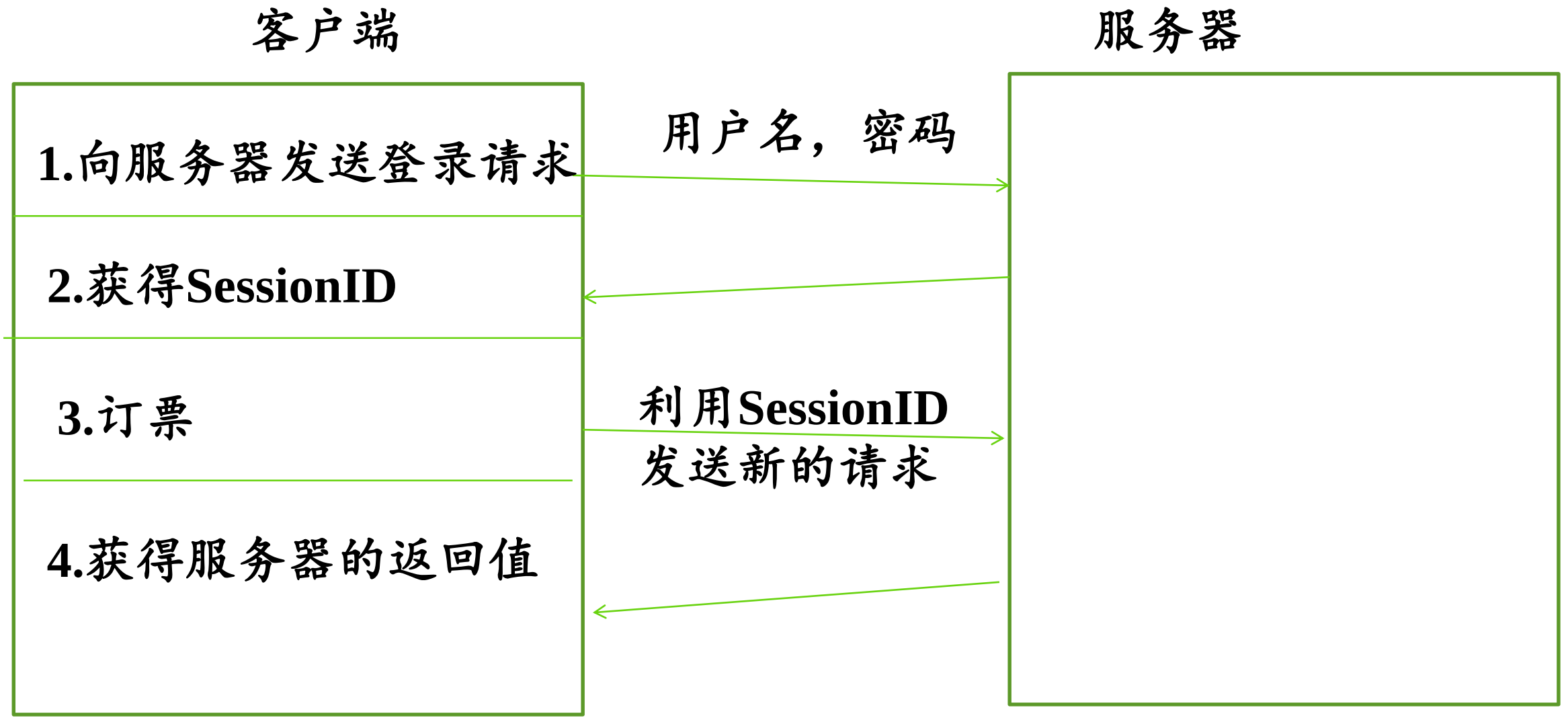
关联

- 什么是关联
  - 脚本回放过程中，客户端发出请求，通过一些规则，在服务器所响应的内容中查找，得到相应的值，以参数的形式替换录制时的静态值，从而向服务器发出正确的请求，这种动态获得服务器响应内容的方法被称作关联

关联

- 什么情况下需要关联
  - 当脚本中的数据每次回放都发生变化时，并且这个动态数据在后面的请求中需要发送给服务器，那么这个内容需要通过关联来询问服务器，获得该数据的变化结果。例如：
    - 登录字符串。带有会话 ID 或时间戳等动态数据的登录字符串
    - 日期/时间戳。使用日期或时间戳或者其他用户凭据的任意字符串
    - 常见前缀：如 SessionID 或 CustomerID

关联

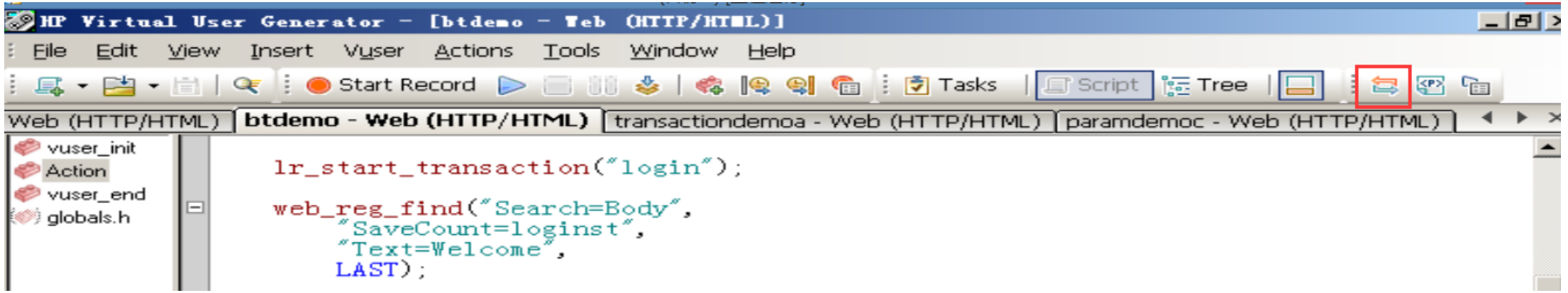

客户端
服务器
1.向服务器发送登录请求
用户名，密码
2.获得SessionID
3.订票
利用SessionID
发送新的请求
4.获得服务器的返回值

关联—怎样进行关联

- 自动关联
  - 最简单，有局限性。常用于在非常标准的动态数据处理中
  - 注意：使用自动关联前，脚本必须要先运行一次
- 一边录制一边关联
  - 无需操作，系统自带常见应用需要做的关联规则
  - 【Recording Options】—【Correlation】启用选项
- 手动关联
  - 最有效手段，能处理特殊的动态数据
  - 典型实例：论坛中置顶帖子和非置顶帖子中的顶端帖子ID

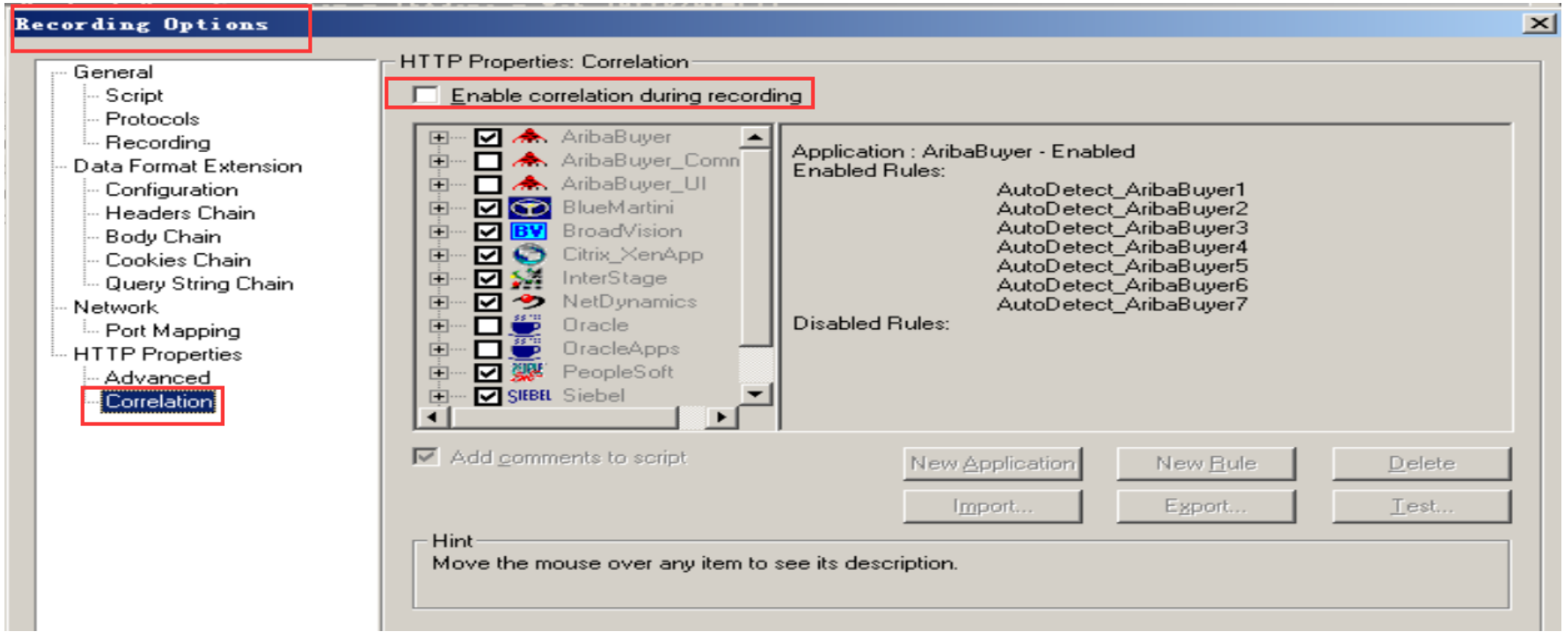
关联—怎样进行关联

- 自动关联
  - 是VuGen提供的自动扫描关联处理策略，它的原理是对同一个脚本运行和录制时的服务器的返回进行比较，来自动查找变化部分，并且提示是否生成关联



关联—怎样进行关联

- 边录制边关联
  - 适合于提前将关联规则定义好，然后依据其进行关联



关联—怎样进行关联

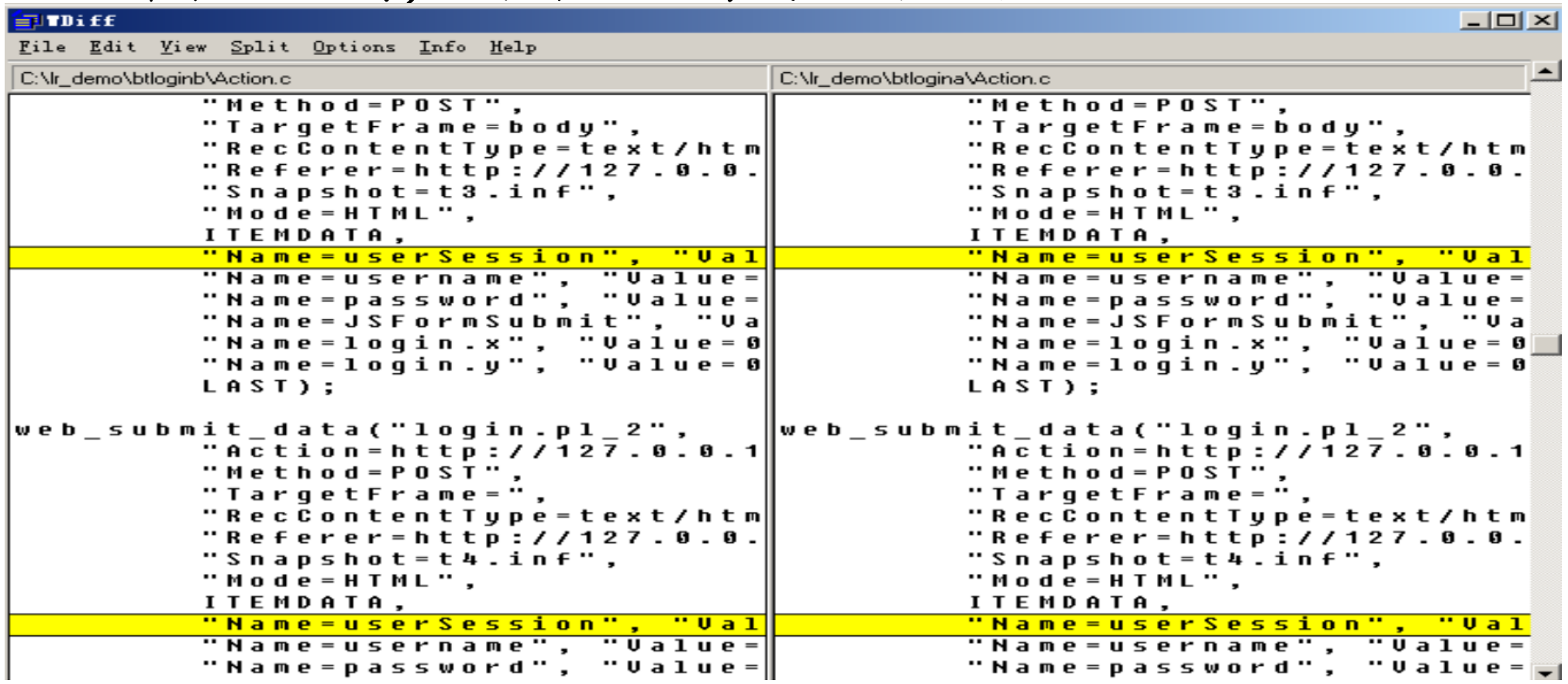
- 手动关联
  - 通过手动关联函数 web_reg_save_param()将想要的字符串保存到一个参数中。 手动关联是关联应用中的最有效手段

关联—怎样进行关联

- 手动关联的主要步骤：
  - 第一步：录制测试脚本，录制两遍（两遍操作内容须一致）

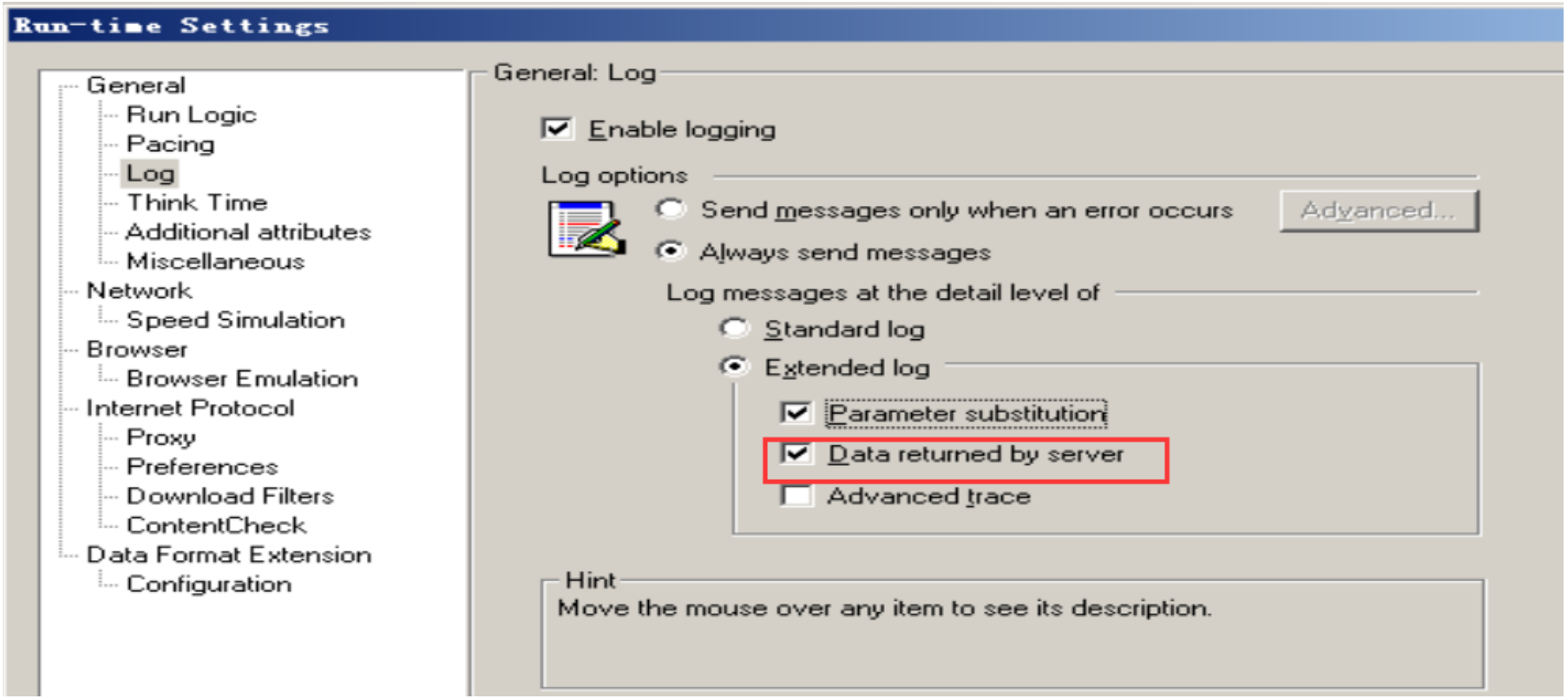
关联—怎样进行关联

- 第二步：使用脚本比较工具（Tools—compare with script）找出两次脚本的不同，判断是否需要进行关联



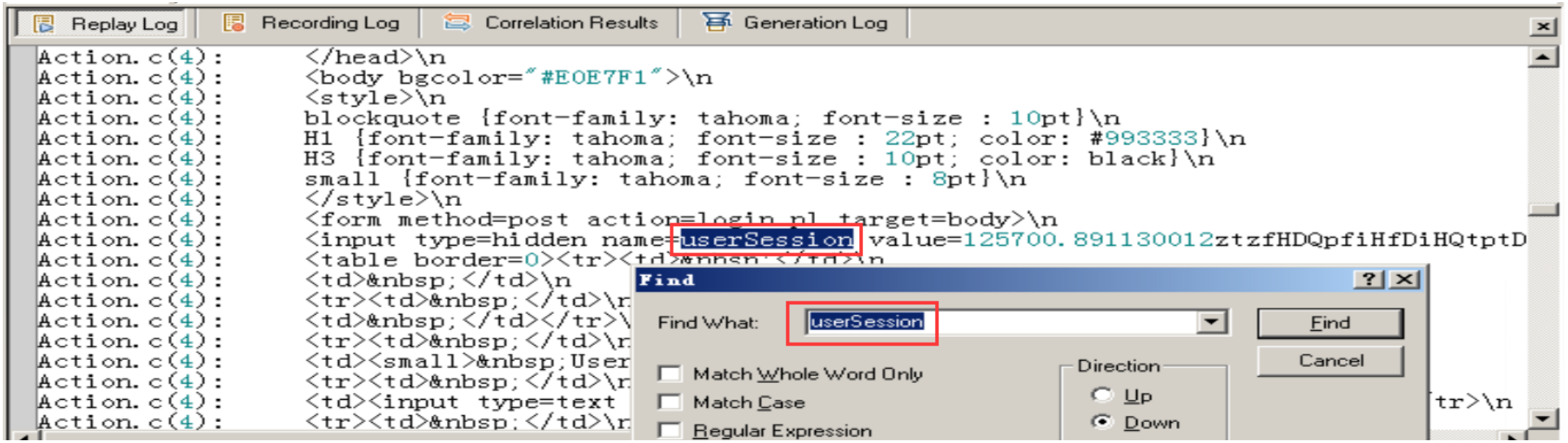
关联—怎样进行关联

- 第三步：在日志（打开扩展日志中服务器返回日志）中查找脚本不同的内容（判断是否是服务器返回内容）



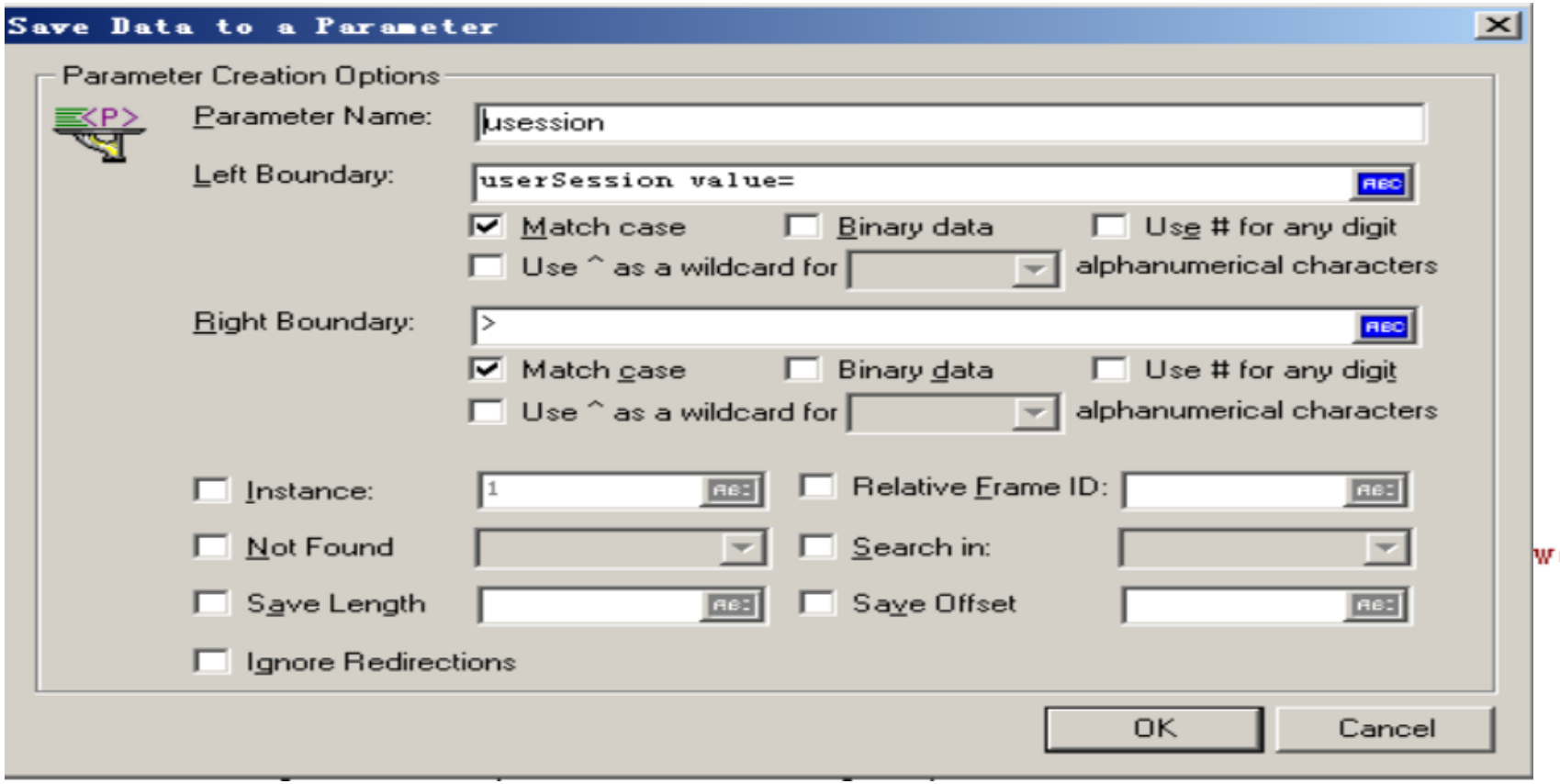
关联—怎样进行关联

- 第四步：在日志中找到查找内容的地方点击鼠标右键，选择“跳至原行”，跳到脚本位置，在该请求前插入关联函数



关联—怎样进行关联

- 插入关联函数需要参数名称和左右边界，在日志中找到其左右边界填入参数设置框中




```
web_reg_save_param("usession",
    "LB=userSession value=",
    "RB=>",
    LAST);
```

关联—怎样进行关联

- 强调：关联函数必须写在被关联数据所在的请求前（或所有请求前）
- 原因：
  - 发出请求前，申明要捕获的数据边界，这样在数据流返回时，直接过滤
  - 否则需要将上一个请求的所有内容保存后，再去单独读取遍历，效率降低

关联—怎样进行关联

- 第五步：将动态数据替换成变量名

```
web_submit_data("login.pl",
    "Action=http://127.0.0.1:1080/WebTours/login.pl",
    "Method=POST",
    "TargetFrame=body",
    "RecContentType=text/html",
    "Referer=http://127.0.0.1:1080/WebTours/nav.pl?in=home",
    "Snapshot=t3.inf",
    "Mode=HTML",
    ITEMDATA,
    "Name=userSession", "Value={usession}", ENDITEM,
    "Name=username", "Value=jojo", ENDITEM,
    "Name=password", "Value=bean", ENDITEM,
    "Name=JSFormSubmit", "Value=off", ENDITEM,
    "Name=login.x", "Value=0", ENDITEM,
    "Name=login.y", "Value=0", ENDITEM,
    LAST);
```

关联—怎样进行关联

- 第六步：验证
  - 查看日志
  - 查看运行结果

目录

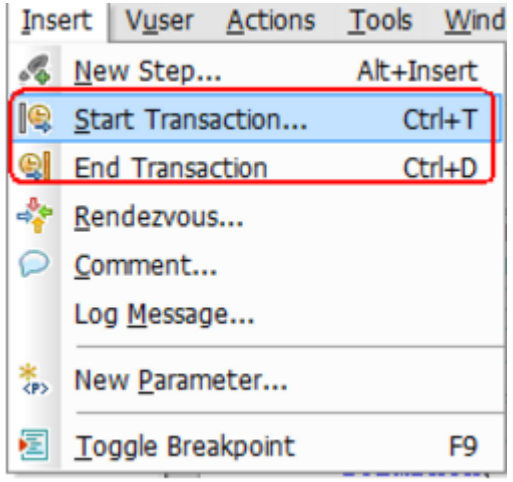
- 脚本查看方式（不同视图）
- 运行结果查看
- 关联
- 事务
- 检查点

事务—基本概念

- 什么是事务
  - 模拟用户的一个相对完整的、有意义的业务操作过程
- 什么情况下使用事务
  - 查看某系列操作的使用时间

事务—插入事务

- 怎样使用事务
  - 手动输入函数
  - 使用菜单插入
  - 录制过程中插入



- lr_start_transaction("事务名称")：事务开始
- lr_end_transaction("事务名称","事务状态")：事务结束，结束状态

事务注意事项

注意：1、开始与结束函数必须成对出现

2、事务的名称必须一样

```
lr_start_transaction("login");

web_submit_form("login.pl",
    "Snapshot=t2.inf",
    ITEMDATA,
    "Name=username", "Value={username1}", ENDITEM,
    "Name=password", "Value={password}", ENDITEM,
    "Name=login.x", "Value=44", ENDITEM,
    "Name=login.y", "Value=12", ENDITEM,
    LAST);

lr_end_transaction("login",LR_AUTO);
```

事务—事务函数

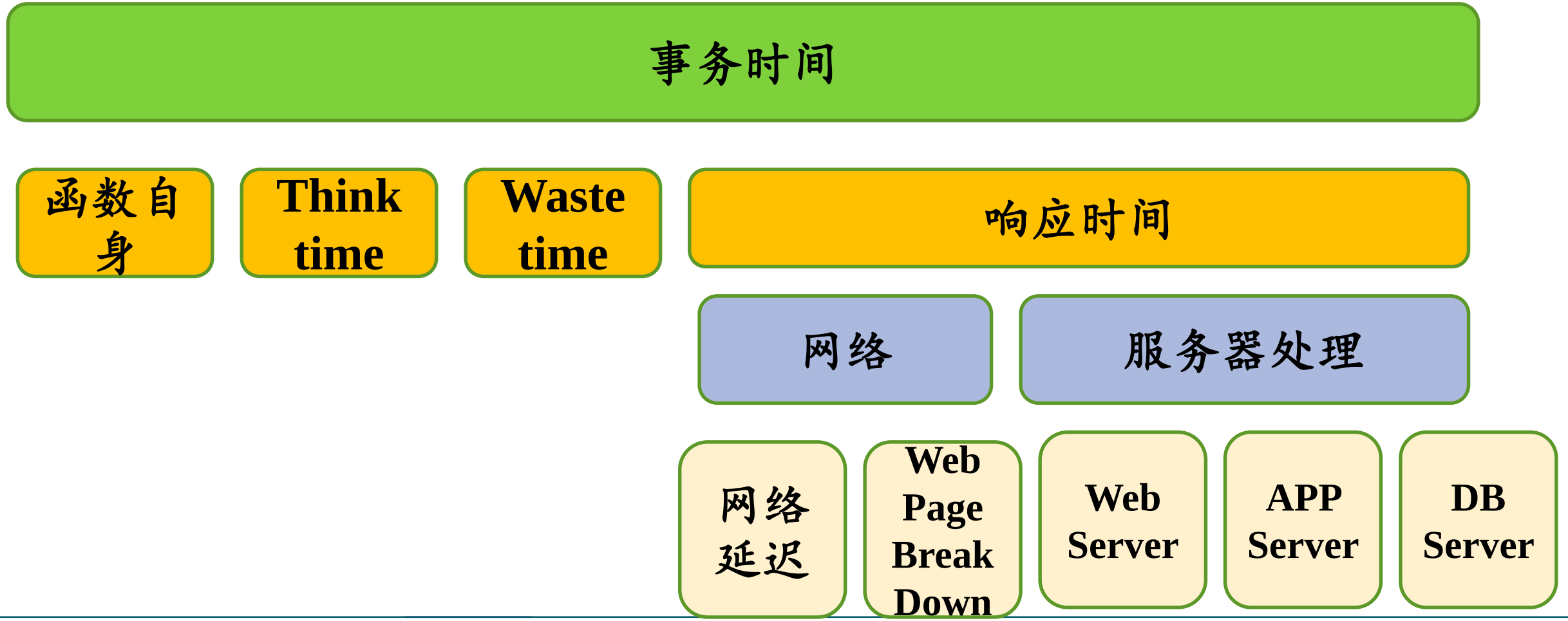
- LR_AUTO是指事务的状态有系统自动根据默认规则来判断，结果为PASS/FAIL
- LR_PASS是指事务是以PASS状态通过的，说明该事务正确的完成了，并且记录下对应的时间，这个时间就是指做这件事情所需要消耗的时间

事务— 事务函数

- LR_FAIL是指事务以FAIL状态结束，该事务是一个失败的事务，没有完成事务中脚本应该达到的效果，得到的时间不是正确操作的时间，这个时间在后期的统计中将被独立统计
- LR_STOP将事务以STOP状态停止。事务的PASS和FAIL状态会在场景的对应计数器中记录，包括通过的次数和事务的响应时间，方便后期分析该事务的吞吐量以及响应时间的变化情况

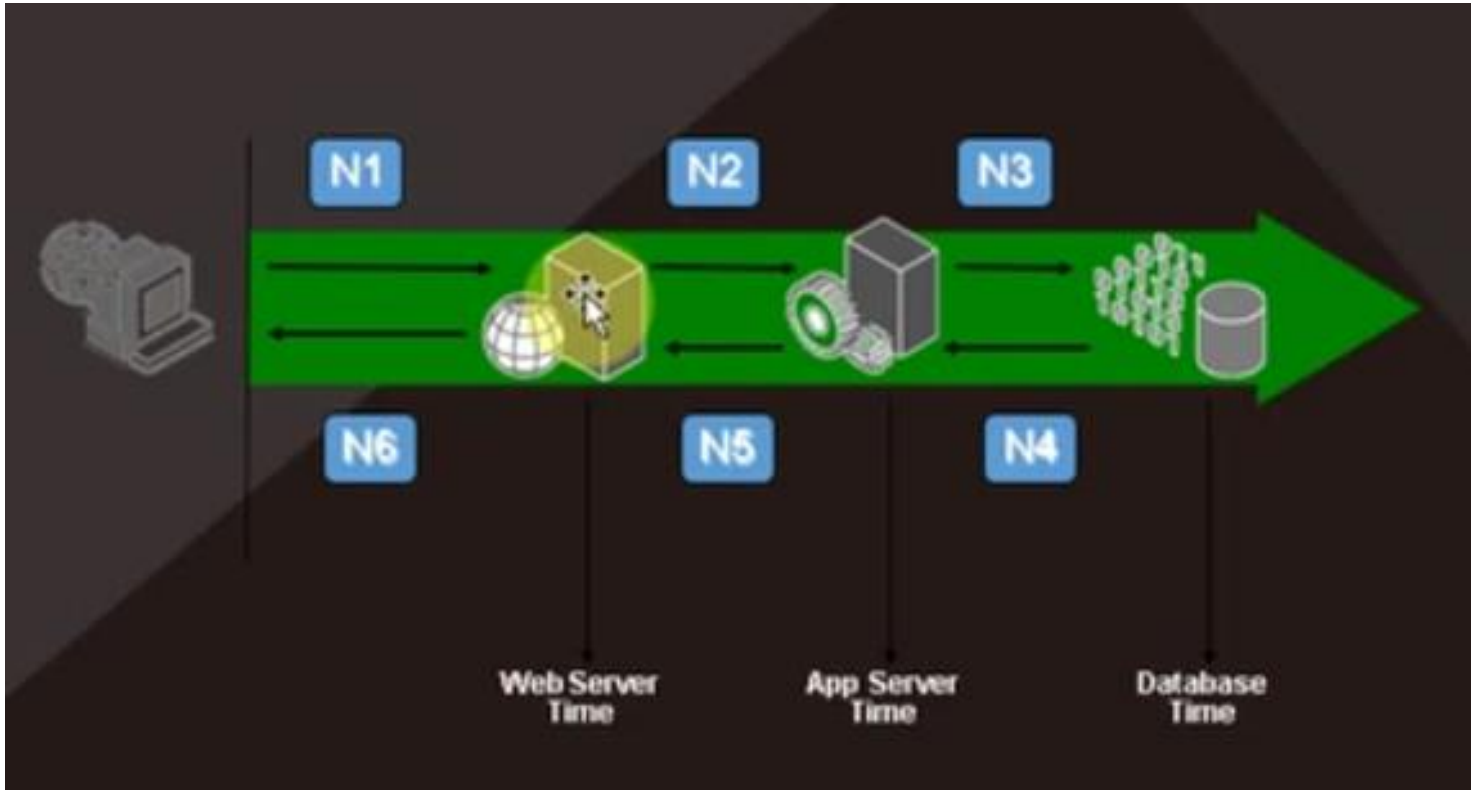
事务时间

- 事务的时间是响应时间吗？



事务—消耗时间分析

- 思考：发一个请求，得到回应需要8S,从哪里找问题？
  - 网络？
  - Java？
  - PHP?
  - 数据库



事务—时间分析

- 实践：分别使用正确的用户名、密码和错误的用户名、密码回放登录WebTours的脚本，查看其事务时间分别是多少
  - 分析：回放脚本都是正确的
  - 原因：服务器判断只要是200系列的状态码，都是pass（只要不是404，返回都是正确的）
  - 不成功的事务比成功的事务时间短
  - 怎样判断服务器到底有没有登录成功？

目录

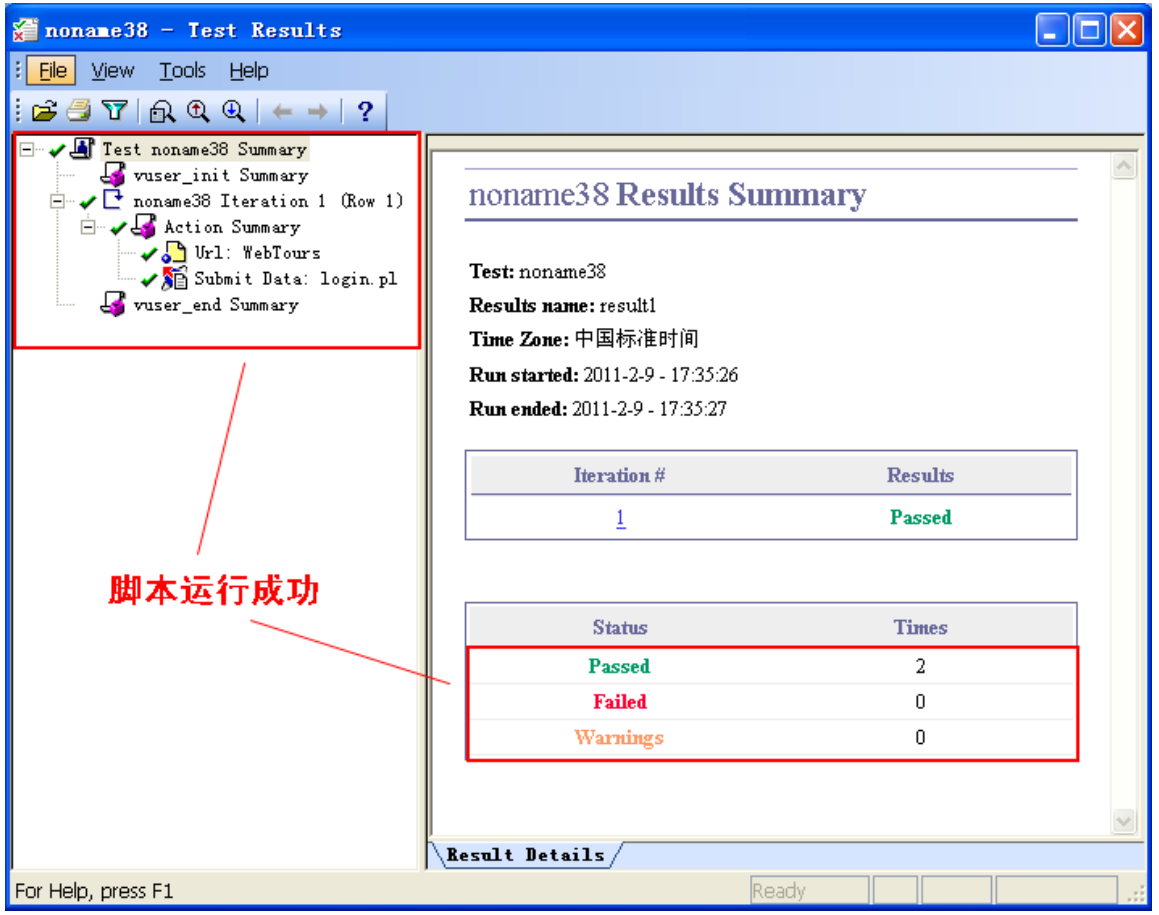
- 脚本查看方式（不同视图）
- 运行结果查看
- 关联
- 事务
- 检查点

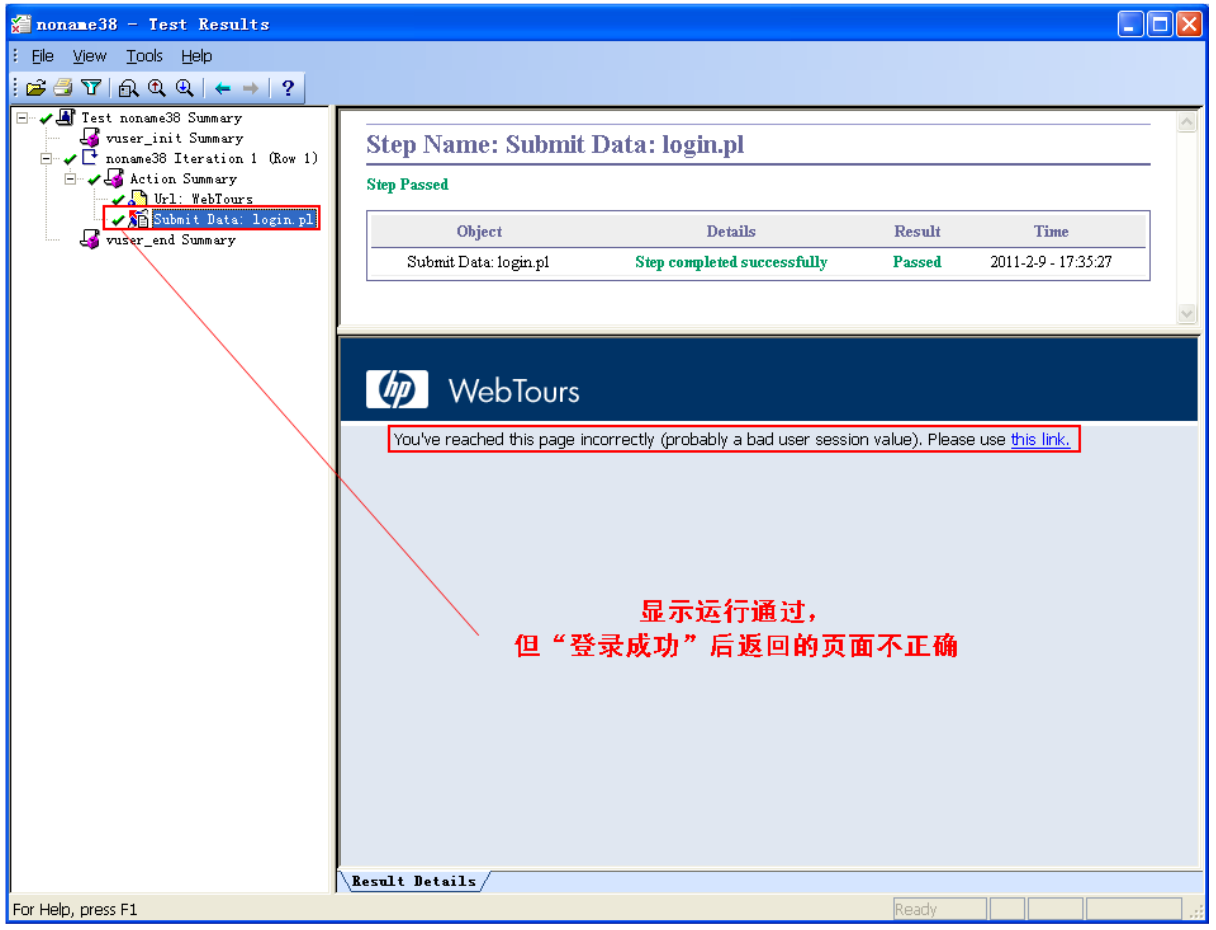
检查点—基本概念

- 什么是检查点
  - 检查服务器返回的页面是否正确

检查点—基本概念

- 为什么使用检查点



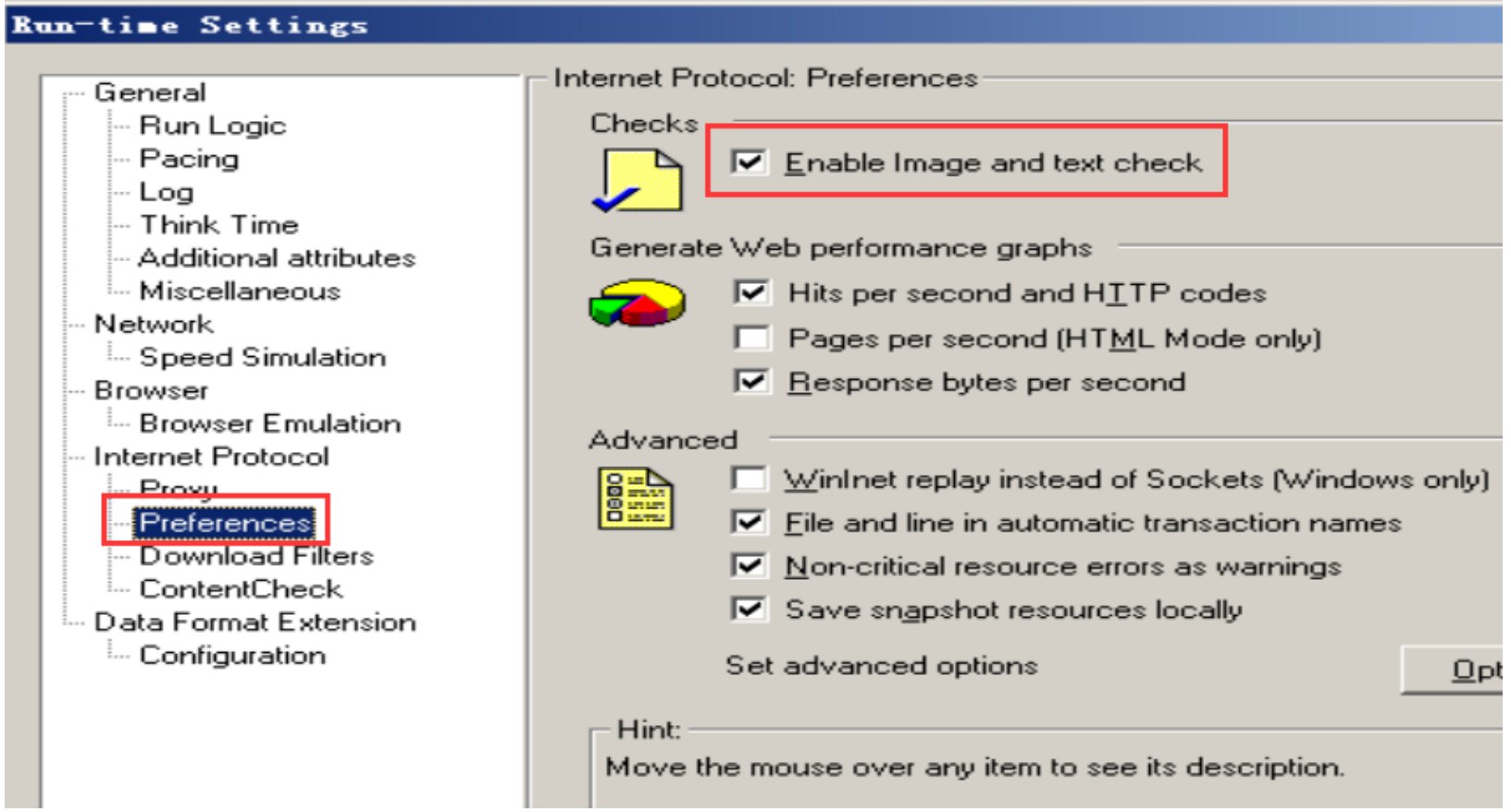


检查点—使用方式

- 插入检查点函数
  - **web_find()**
  - **web_reg_find()**

检查点—使用方式

```
web_find("web_find",
         "What=Welcome",
         LAST);
```



检查点—使用方式

- web_find()函数
  - 函数作用：
    - 在HTML页面中查找相应的内容
  - 函数用法：
    - web_find函数用于查找HTML页面中的内容，故须放在待查找内容的后面

检查点—使用方式

```
web_submit_data("login.pl",
    "Action=http://127.0.0.1:1080/WebTours/login.pl",
    "Method=POST",
    "TargetFrame=body",
    "RecContentType=text/html",
    "Referer=http://127.0.0.1:1080/WebTours/nav.pl?in=home",
    "Snapshot=t2.inf",
    "Mode=HTML",
    ITEMDATA,
    "Name=userSession", "Value={login}", ENDITEM,
    "Name=username", "Value=jojo", ENDITEM,
    "Name=password", "Value=bean", ENDITEM,
    "Name=JSFormSubmit", "Value=off", ENDITEM,
    "Name=login.x", "Value=54", ENDITEM,
    "Name=login.y", "Value=9", ENDITEM,
    LAST);

web_find("web_find",
    "What=Welcome",
    LAST);
```

检查点—使用方式

```
web_reg_find("Search=Body",
             "SaveCount=loginst",
             "Text=Welcome",
             LAST);
```

```
if(atoi(lr_eval_string("{loginst}"))>0)
        lr_end_transaction("login", LR_PASS);
else
        lr_end_transaction( “login", LR_FAIL);
```

检查点—使用方式

- web_reg_find（）函数
  - 函数作用：
    - 在HTML的源文件中查找所需内容。较web_find()方式，web_reg_find（）方式查找的更加精确
  - 函数用法：
    - web_reg_find（）函数是在HTML的源文件中查找相应的内容，故需插入在待查找内容之前
  - 参数举例：web_reg_find("Search=Welcome","SaveCount=name",LAST);

检查点—注意事项

- 注意事项
  - 尽量使用web_reg_find函数
  - 检查的字符尽量不要是中文

关联、事务、检查点实践练习

- 录制飞机订票系统登录操作
- 使用关联
- 插入login的事务
- 插入检查点，检查是否登录成功

内容总结

- 脚本查看方式（不同视图）
- 运行结果查看
- 关联
- 事务
- 检查点

Question