# 测试背景

1. 测试easyshell需要不段重启，测试时间长
2. 测试修改项目多，测试复杂

# 测试需求

1. 覆盖尽可能多的测试项目
2. 全程自动模拟人工操作
3. 全程自动检查结果
4. 全程测试log记录
5. 错误的测试点可以截屏
6. 通过邮件发送测试报告
7. 测试系统：Windows10, windows7

# 测试工具

1. Python, 编程语言
2. UI Automation, 编程扩展库

# 测试框架

1. 主要分为 测试对象、逻辑处理分析、测试数据、测试用例、测试报告

* Configuration: 存放对象库和邮件列表
* Library： 通用的python代码文件
* Misc. ：无法归类的数据、文件、工具
* Services： 测试需要的services安装文件
* Test\_Data：测试数据
* Test\_report：测试报告、log、截图
* Test\_scripts：测试脚本python代码
* Test\_suite：测试用例的xlsx文件
* Testing：正在测试的测试用例文件
* Autolist.xlsx：测试用例在QC上的映射关系
* Testset.xlsx：当前测试计划
* Start.py： 程序入口的代码文件
* Start.exe： 程序入口的执行文

1. 整个开发过程：

代码模块开发🡪测试用例制作🡪测试用例执行🡪报告收集

功能模块由多个子模块组成， 功能模块之间互相独立，一个测试用例由多个子模块组合而成

模块开发 用例制作 用例执行

用例1

模块1

模块1

模块3

用例2

模块2

用例3

模块2

模块3

被操的对象库

模拟手动操作对象

操作数据

操作结果检查

验证结果

输出报告

1. 测试过程：

在Admin用户下操作->重启机器->在User用户下验证结果->重启机器->Admin用户下恢复状态

# 功能模块

* User interface Settings
* User Settings
* Application
* Websites
* Storefront
* VMware view
* RDP
* Citrix
* Task Switcher
* Background
* Wi-Fi
* General test
* VDI Logon
* Common Lib

# 执行测试

1. 禁用待测机器HPWM
2. 修改testset.xlsx，添加需要测试的用例
3. 将框架目录复制到待测机器
4. 执行脚本preconfig.bat
5. 重新启动机，测试会自动执行
6. 测试结束后会收到相应邮件

# 错误处理

1. 每条用例的fail截屏会放在test report下面的screenshot目录下面
2. Test report目录下面会有test log
3. 根据测试log和fail截屏的详细信息验证bug