INSTALACIÓN

1. Dentro del repositorio del proyecto, navegar al siguiente directorio:
   `\04 Project\02 Booking\04 Test Plan\package-booking-service\libraries-EXCEL`

2. Este directorio contiene los siguientes ficheros, que debemos copiar:

   ```
   poi-3.15.jar
   poi-examples-3.15.jar
   poi-excelant-3.15.jar
   poi-ooxml-3.15.jar
   poi-ooxml-schemas-3.15.jar
   poi-scratchpad-3.15.jar
   ```

3. Navegar hasta el directorio de instalación de SoapUI (por defecto, en Windows: `C:\Program files\SmartBear\SoapUI-x.y.z`, donde xyz es el número de versión).

4. Una vez ahí entramos al directorio `\bin\ext` y copiamos los archivos anteriores.

5. También será necesario copiar los scripts de la librería dentro del proyecto de SoapUI si todavía no lo tiene. Estos ficheros se pueden encontrar en otros proyectos bajo el test suite "Library" y el test case "Utils".

FICHERO EXCEL

Estos archivos forman parte de una librería que se usa para leer los datos del archivo de Excel, que contiene las siguientes pestañas:

CONFIG

SUITE	CASE	DATA_USED
Bateria Basica	102 - (2PAX)	PROPERTIES

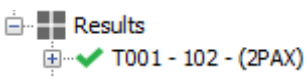
- La columna **SUITE** y **CASE** se usan para seleccionar el **Test Case** a usar.
- La columna **DATA_USED** contiene el nombre de la **pestaña** con el **conjunto de datos** que queremos usar.

DATOS

PROPERTY_SET	currency	market	userType	tariffClass	typeOfPayment	deposit	promocode
T001	EUR	FR	CUSTOMER	Regular0	0	FULL	HOTEL2
T004	EUR	IT	CUSTOMER	Regular0	1	DEPOSIT	HOTEL2

- La **primera columna** de la pestaña (la que está más hacia la izquierda) contiene el **identificador** del set de datos. Este se pondrá en el nombre del Test Case para saber con qué datos se ha ejecutado. Ejemplo:

  Ejecutamos el Test Case **102 – (2PAX)** con el conjunto de datos **T0001**:

- El **resto de columnas** son los **datos** que podemos añadir o no. Están definidos por nosotros

mismos y por las necesidades del proyecto. A estos datos se podrá acceder en el código, por lo que conviene que el encabezado sea **fácil de escribir** e **identificativo**.

En las columnas de datos podemos usar referencias a otras tablas que nos permiten **reutilizar datos** fácilmente o **mantenerlos organizados**. Un ejemplo sería reutilizar los datos de login de un usuario para distintas pruebas en lugar de repetirlos constantemente. A continuación, unas capturas que ilustran este ejemplo. Nótese que la referencia funciona de la siguiente manera:

#(nombre de la pestaña)#(identificador de la fila)

Del campo "**loginData**" en la primera pestaña se interpreta que tenemos que ir a la tabla "**LOGIN**" e insertar ahí la fila "**admin**" o "**invalid_user**" según cada caso.

TC_ID	url	loginData
0001	login.php	#LOGIN#admin
0002	login.php	#LOGIN#invalid_user

DATA

Para el primer caso (TC_ID=0001), en esta pestaña se leen los campos "**username**" y "**password**" y se insertan dentro del campo "**loginData**" de la tabla anterior.

USER	username	password
admin	root	root
invalid_user	fake_user	fake_pa$$word

LOGIN

Esto nos dejará con un array de datos (la primera fila) que contendrá a otro array de datos, que puede contener a otro... y así hasta el infinito. Es una estructura de árbol desde la que podremos acceder cómoda e intuitivamente a los datos que definamos en el Excel. Debemos vigilar, no obstante, que no incluyamos un elemento dentro de sí mismo directa o indirectamente para evitar un bucle

CÓDIGO

Dentro de SoapUI, sólo será necesario conocer dos ficheros de código:

CLEAN RESULTS

El objetivo de este fichero es el de limpiar el test suite de "Results" después de cada ejecución. Esto se le deja hacer al usuario para que pueda inspeccionar con calma los resultados obtenidos.

RUN

Este script se encarga de ejecutar el test runner. Para esto, es necesario pasarle dos funciones como parámetros, que se definen en el mismo script Run. Esas funciones son las siguientes:

Apply Data To Test Case:

Esta función se aplica **después de clonar** el Test Case y justo **antes de ejecutarlo**. Su función es la de **aplicar** los **datos** que se han **leído** en el **fichero Excel** sobre el caso y/o proyecto a ejecutar.

Los **parámetros** que recibe la función son:

- **testCase:** el objeto del test case que se va a ejecutar. Desde él podemos acceder al proyecto, a cada step individual o incluso a otros test cases.
- **data:** un objeto Map con estructura de árbol que contiene los datos leídos del excel. En el ejemplo del Login anterior, podríamos acceder al nombre de usuario de esta manera:

```
data.loginData.username
```

Un ejemplo de asignación de datos podría ser el siguiente, en el que se asignan los datos del ejemplo anterior a unas propiedades del proyecto:

```
utils.setProjectProperty("username", data.loginData.username, testRunner)
utils.setProjectProperty("password", data.loginData.password, testRunner)
```

Get Step Data:

El objetivo de esta función es **generar** los **datos de salida** del **reporte**. En un momento dado puede interesarnos mostrar un valor que no se está mostrando actualmente. No debería ser necesario cambiar el código del generador del reporte para ello, por lo que se ofrece esta función como solución.

Los **parámetros** de esta función son:

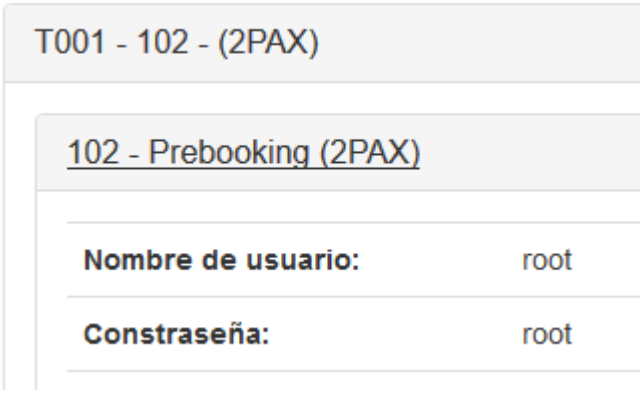
- **testCase:** el objeto del test case, mediante el cual podemos acceder a sus steps, al proyecto o incluso a otros test cases.
- **testStep:** el objeto del step, que permite el acceso a otros objetos de pruebas y
- **stepOK:** 'true' si el step se ha ejecutado sin problemas, 'false' si ha habido algún error.
- **isRequest:** 'true' si el step es una request, 'false' si no.
- **messages:** una lista con los mensajes que ha lanzado el step al ejecutarse (por ejemplo, mensajes de error)

Esta función retorna un objeto del tipo **Map** con las parejas (Nombre del **atributo** / **Valor**) como (Llave / Valor). Un ejemplo de cómo usar esta función sería el siguiente, en el que recogemos usuario y password de la respuesta del step si este es una request.

```
def dataMap = [:]

if (isRequest) {
	def response = testStep.getPropertyValue("response")
	def json = new JsonSlurper().parseText(response)
	dataMap['Nombre de usuario'] = json.usr
	dataMap['Constraseña'] = json.pwd
}

dataMap['Messages'] = messages.join('<br>')

return dataMap
```

Nótese también que los distintos mensajes se ponen en una sola string y se unen con **'
'**. Esto se debe, en primer lugar, a el generador sólo acepta una string, por lo que si le pasamos una lista dará un error. En segundo lugar, no hay que olvidar que el reporte es **código HTML**, por lo que se pueden insertar elementos y, para hacer un salto de línea, hace falta un **
**.

El resultado, en este caso y con los datos de ejemplos anteriores, sería el siguiente:



Editors

Estos scripts facilitan realizar algunas acciones sobre el proyecto y hasta permiten hacerlo de forma masiva sobre todos los test cases del proyecto.

Update json values

Este script permite cambiar valores de los JSON de las request de forma masiva. Toma 3 **parámetros**:

- **Nombre del suite:** este campo es una string que actua como filtro. Sólo se modificarán los steps dentro de las suites con este nombre. Para hacer que se modifiquen de todas las suites independientemente del nombre se puede introducir un asterisco `"*"`.
- **Nombre del case:** este campo es una string que actua como filtro. Sólo se modificarán los steps dentro de los cases con este nombre. Para hacer que se modifiquen de todos los cases independientemente del nombre se puede introducir un asterisco `"*"`.
- **Nombre del step:** este campo es una string que actua como filtro. Sólo se modificarán los steps con este nombre. Para hacer que se modifiquen todas los steps independientemente del nombre se puede introducir un asterisco `"*"`.
- **Ruta del valor a modificar:** esta string define la ruta a seguir para llegar hasta el valor que se debe modificar. Esta ruta viene definida con el nombre de las claves de cada elemento separadas por un punto. Un ejemplo, para el siguiente JSON:

```
{
    "paxes": [
        {
            "dateOfBirth": "1975-06-28",
            "middleName": "MA",
            "lastName": "${#Project#lastname}",
            "travelDocumentData": {
```

  Si usamos la ruta `".paxes.lastName"` llegaremos hasta el campo `"lastName"` y cambiaremos su valor actual, `"${#Project#lastname}"`.
- **Nuevo valor:** el nuevo valor del campo seleccionado.