Artículo
Ricardo Paiva · 4 nov, 2021 Lectura de 5 min

Open Exchange

Pruebas unitarias y Cobertura de pruebas en ObjectScript Package Manager

En este artículo describiré los procesos para ejecutar pruebas unitarias mediante ObjectScript Package Manager (consulta https://openexchange.intersystems.com/package/ObjectScript-Package-Manager-2), incluyendo el cálculo de la Cobertura de pruebas (mediante https://openexchange.intersystems.com/package/Test-Coverage-Tool).

Pruebas unitarias en ObjectScript

Ya hay mucha documentación sobre cómo escribir pruebas unitarias en ObjectScript, por lo que no repetiré nada de eso. Puedes consultar el Tutorial de Pruebas Unitarias aquí: https://docs.intersystems.com/irislatest/csp/docbook/Doc.View.cls?KEY=TUNTpreface

La práctica recomendada es incluir las pruebas Unitarias en algún lugar/carpeta separada en la estructura de fuentes, ya sea simplemente "/pruebas" o algo más sofisticado. Dentro de InterSystems, terminamos usando /internal/testing/unittests/ como nuestro estándar *de facto*, lo que tiene sentido porque las pruebas son internas/no distribuibles y hay otros tipos de pruebas además de las unitarias, pero esto podría ser un poco complejo para proyectos sencillos de código abierto. Puedes ver esta estructura en algunos de nuestros repositorios de GitHub.

Desde el punto de vista del flujo de trabajo, esto es súper fácil en VSCode: solo hay que crear el directorio y colocar las clases allí. Con enfoques más antiguos centrados en el servidor para el control de la fuente (los utilizados en Studio), tendrás que mapear este paquete de manera apropiada, y el enfoque para eso varía según la extensión del control de la fuente.

Desde la perspectiva de los nombres de clases para las pruebas unitarias, mi preferencia personal (y la práctica recomendada de mi grupo) es:

UnitTest.<package/class being tested>[.<method/feature being tested>]

Por ejemplo, si las pruebas unitarias son para el Método Foo en la clase MyApplication.SomeClass, la clase de la prueba unitaria se llamaría UnitTest.MyApplication.SomeClass.Foo; si las pruebas fueran para la clase en su totalidad, simplemente sería UnitTest.MyApplication.SomeClass.

Pruebas unitarias en ObjectScript Package Manager

¡Hacer que ObjectScript Package Manager esté informado de tus pruebas unitarias es sencillo! Basta con añadir una línea como la siguiente a module.xml (tomada de https://github.com/timleavitt/ObjectScript-Math/blob/master/module.xml, una bifurcación del excelente paquete matemático de @Peter Steiwer de Open Exchange, el cual utilizo como un simple ejemplo inspirador):

<Module>
 ...
 <UnitTest Name="tests" Package="UnitTest.Math" Phase="test"/>
</Module>

Lo que todo esto significa es:

- Las pruebas unitarias están en el directorio "tests" debajo de la raíz del módulo.
- Las pruebas unitarias están en el paquete "UnitTest.Math". Esto tiene sentido, porque las clases que se están probando están en el paquete "Math".
- Las pruebas unitarias se ejecutan en la fase "test" en el ciclo de vida del paquete. (También hay una fase de "verificación" en la que podrían ejecutarse, pero esa es una historia para otro día).

Cómo ejecutar pruebas unitarias

Con las pruebas unitarias definidas como se explicó anteriormente, el administrador de paquetes ofrece algunas herramientas realmente útiles para ejecutarlas. Todavía puedes configurar ^UnitTestRoot, como lo harías normalmente con %UnitTest.Manager, pero probablemente encontrarás las siguientes opciones mucho más fáciles, especialmente si estás trabajando en varios proyectos en el mismo entorno.

Puedes probar todos estas opciones clonando el repositorio objectscript-math enumerado anteriormente y luego cargarlo con zpm "load /path/to/cloned/repo/", o en tu propio paquete reemplazando "objectscript-math" con los nombres de tus paquetes (y nombres de prueba).

Para recargar el módulo y luego ejecutar todas las pruebas unitarias:

zpm "objectscript-math test"

Para simplemente ejecutar las pruebas unitarias (sin recargar):

zpm "objectscript-math test -only"

Para simplemente ejecutar las pruebas unitarias (sin recargar) y proporcionar una salida detallada:

zpm "objectscript-math test -only -verbose"

Para ejecutar un conjunto de pruebas en particular (es decir, un directorio de pruebas, en este caso, todas las pruebas en UnitTest/Math/Utils) sin recargar, y proporcionar una salida detallada:

zpm "objectscript-math test -only -verbose -DUnitTest.Suite=UnitTest.Math.Utils"

Para ejecutar un caso particular de prueba (en este caso, UnitTest.Math.Utils.TestValidateRange) sin recargar y proporcionar una salida detallada:

zpm "objectscript-math test -only -verbose -DUnitTest.Case=UnitTest.Math.Utils.TestValidateRange"

O, si solo estás resolviendo los problemas de un único método de prueba:

zpm "objectscript-math test -only -verbose -DUnitTest.Case=UnitTest.Math.Utils.TestValidateRange
-DUnitTest.Method=TestpValueNull"

Cálculo de la Cobertura de pruebas mediante ObjectScript Package Manager

Así que tienes algunas pruebas unitarias, pero ¿son buenas? Calcular la cobertura de pruebas no responderá completamente a esa pregunta, pero al menos ayuda. Presenté esto en la Convención anual (*Global Summit*) de InterSystems, allá por el año 2018 - aquí puedes ver el vídeo: https://youtu.be/nUSeGHwN5pc .

Lo primero que tendrás que hacer es instalar el paquete de cobertura de pruebas:

zpm "install testcoverage"

Ten en cuenta que esto no requiere la instalación/ejecución de ObjectScript Package Manager; puedes encontrar más información en Open Exchange: https://openexchange.intersystems.com/package/Test-Coverage-Tool

Dicho esto, puedes aprovechar al máximo la herramienta de cobertura de pruebas si también utilizas ObjectScript Package Manager.

Antes de ejecutar pruebas, debes especificar qué clases/rutinas esperas que cubran tus pruebas. Esto es importante porque, en las bases de código muy grandes (por ejemplo, HealthShare), calcular y recopilar la Cobertura de pruebas para todos los archivos del proyecto puede requerir más memoria de la que tiene tu sistema. (Específicamente, gmheap para un análisis por linea de código, si tienes curiosidad).

La lista de archivos se incluye en un archivo llamado cover.list, que está dentro de la raíz de la prueba unitaria. Diferentes subdirectorios (conjuntos) de pruebas unitarias pueden tener su propia copia de esto para anular las clases/rutinas que se rastrearán mientras se ejecuta el conjunto de pruebas.

Para ver un ejemplo sencillo con objectscript-math, consulta: https://github.com/timleavitt/ObjectScript-Math/blob/master/tests/UnitTest/coverage.list. La guía de usuario para la Herramienta de cobertura de pruebas incluye más detalles.

Para ejecutar las pruebas unitarias con el cálculo de la Cobertura de pruebas habilitado, solo hay que añadir un argumento más al comando, especificando que se debe utilizar TestCoverage.Manager en vez de %UnitTest.Manager para ejecutar las pruebas:

zpm "objectscript-math test -only -DUnitTest.ManagerClass=TestCoverage.Manager"

La salida (incluso en el modo resumido) incluirá una URL donde podrás ver qué líneas de tus clases/rutinas estaban cubiertas por las pruebas unitarias, así como algunas estadísticas agregadas.

Siguientes pasos

¿Qué sucede con la automatización de todo esto en CI? ¿Qué sucede con los reportes de resultados de las pruebas unitarias y las puntuaciones/diffs de cobertura? ¡También puedes hacer eso! Para ver un ejemplo sencillo usando Docker, Travis CI y codecov.io, consulta https://github.com/timleavitt/ObjectScript-Math. Estoy planeando escribir esto en un artículo futuro que analice algunos enfoques diferentes.

#Integración continua #InterSystems Package Manager (IPM) #InterSystems IRIS #InterSystems IRIS for Health
Ir a la aplicación en InterSystems Open Exchange