Pruebas unitarias en Java Data-Driven con TestNG

El enfoque de las pruebas basadas en Data-Driven busca la separación de la lógica de la prueba del escenario.

Durante las pruebas unitarias es común contar con casos de prueba similares dónde la única diferencia entre ellos son los datos de prueba utilizados. En este artículo presentamos el enfoque basado en *Data-Driven;* este enfoque permite separar la lógica de la prueba de los datos de la misma.
En el artículo anterior presentamos una nueva versión del método *merge* y un conjunto con 9 casos de prueba. Estos casos de prueba son el ejemplo en este artículo.

Enfoque Data-Driven

El enfoque de las pruebas basadas en *Data-Driven* busca la separación de la lógica de la prueba del escenario de la misma. De esta forma permite utilizar la misma lógica con diversos conjuntos de datos de prueba (los datos de entrada y resultados esperados). Estos datos se guardan (y luego al ejecutar las pruebas se obtienen) de una fuente de datos externa como puede ser un archivo XML o una planilla electrónica.
Al momento de la ejecución de las pruebas que utilizan *Data-Driven* se toma el primer dato del archivo de datos, se ejecuta la prueba con ese dato y se verifican los resultados obtenidos. Así sucesivamente para todos los datos del archivo.

Herramienta TestNG

La herramienta TestNG es un framework de soporte a las pruebas unitarias y de integración para Java. Fue creada basándose en JUnit y NUnit (para .NET), pero introduciendo nuevas funcionalidades que buscan hacerla más poderosa y fácil de usar.
Una de sus funcionalidades es brindar soporte para pruebas que utilizan *Data-Driven*. La herramienta requiere que se defina un método que será el contenedor de los datos de prueba, no siendo posible ingresar los mismos de forma directa desde una fuente de datos externa. Si bien esto es una limitante de la herramienta, es relativamente sencillo crear un método que traduzca los datos de un archivo externo al método requerido por TestNG.

Utilizando el enfoque Data-Driven con el método merge

A continuación se presenta la utilización de este enfoque con la herramienta TestNG para el método *merge*. Este método recibe dos arreglos de enteros (cada uno ordenado de menor a mayor) y devuelve un arreglo que es la unión ordenada de los elementos de ambos arreglos. La firma del método es la siguiente: `public int[] merge( int[ ] a, int [ ] b)`

Utilizando la anotación *@dataProvider* se señala que un método será proveedor de datos para un caso de prueba. El método debe retornar un *Object[ ][ ]* donde cada *Object[ ]* se puede asignar a una lista de parámetros del caso de prueba.
En la Figura 1 se muestra el contenedor de datos creado para los 9 casos de prueba definidos en el artículo anterior. Los 9 *Object[]* que se crean al final del método como parte del arreglo de arreglos de objetos son los 9 datos de entrada y resultados esperados para los casos de prueba.

```
@dataProvider
public Object [][] datos() {
      int[] a1= {1,2,3,4,5,6};
      int[] b1 = {7,8,9,10,11};
      int[] res1= {1,2,3,4,5,6,7,8,9,10,11};
      int[] a2= {1,2,5,11};
      int[] b2 = {3,4,8,10};
      int[] res2= {1,2,3,4,5,8,10,11};
      int[] a3= {};
      int[] b3 = {8};
      int[] res3= {8};
      int[] a4= {4,5};
      int[] b4 = {3};
      int[] res4= {3,4,5};
      int[] a5= {1,2,3};
      int[] b5 = {};
      int[] res5= {1,2,3};
      int[] a6= {1,3,4};
      int[] b6 = {2};
      int[] res6 = {1,2,3,4};
      int[] a7= {2};
      int[] b7 = {1,3,4};
      int[] res7 = {1,2,3,4};
      int[] a8= {3,4};
      int[] b8 = {1,2};
      int[] res8 = {1,2,3,4};
      int[] a9= {1,3};
      int[] b9 = {2};
      int[] res9 = {1,2,3};

      return new Object [][] {
            new Object[] { a1, b1, res1},
            new Object[] { a2, b2, res2},
            new Object[] { a3, b3, res3},
            new Object[] { a4, b4, res4},
            new Object[] { a5, b5, res5},
            new Object[] { a6, b6, res6},
            new Object[] { a7, b7, res7},
            new Object[] { a8, b8, res8},
            new Object[] { a9, b9, res9},
      };
}
```

Figura 1 – Contenedor de Datos

En la Figura 2 se presenta la lógica del caso de prueba. En la primera línea del mismo, mediante el uso de una anotación de TestNG, se declara que los datos de la prueba van a ser proporcionados por el DataProvider de nombre datos. En definitiva, en nuestro ejemplo, se ejecutará este mismo caso con 9 escenarios diferentes que los provee el DataProvider definido.

```
@Test(dataProvider="datos")
public void Test_Merge(int [] a, int [] b, int [] res) {
      int[] resObtenido = Merge.merge(a, b);
      assertArrayEquals(res,resObtenido);
}
```

Figura 2 – Caso de Prueba

Como se puede apreciar, al definir los casos de prueba utilizando el enfoque *Data-Driven*, se logra una mayor prolijidad en los casos de prueba, facilitando de esta forma el mantenimiento de los mismos. Ahora la ejecución de los casos se encuentra en un único lugar y separado de los datos de entrada y de los resultados esperados.

Adriana Ávila
Lucía Camilloni
Diego Vallespir

Grupo de Ingeniería de Software, UdelaR
gris@fing.edu.uy