Configuring Factory Methods

Introduction

UTA can use the factory methods that are available in your code to create objects during test generation. This requires tagging individual methods as factory methods in the Javadoc (see Tagging a Factory Method) to enable UTA to discover them, and to use them to initialize objects when tests are created.

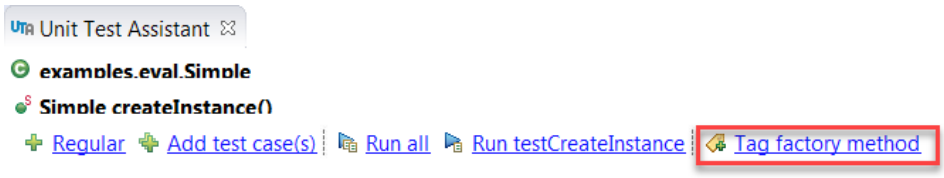
You can configure UTA to automatically discover the factory methods that you tagged, or manually instruct UTA to search your project(s) for these methods (see Scanning for Factory Methods).

Tagging a Factory Method

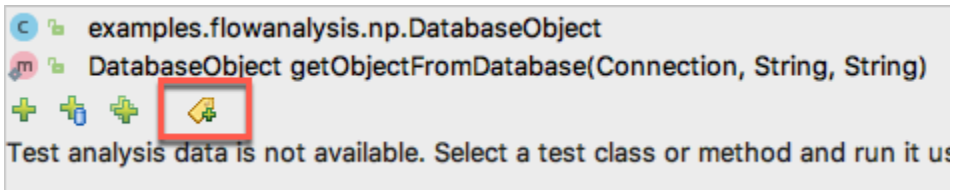
You can tag a public static method in a public non-inner class. The method must return a complex type that does not contain a bounded type parameter. You can tag a method in one of the following ways:

- manually add the **@jtest.factory** tag to the Javadoc comment for the method
- select a method in the editor, and use the **Tag factory method** action in the Unit Test Assistant interface; this will automatically add the **@jtest.factory** tag to the Javadoc comment for the selected method, and update the Factory Methods view with the tagged method.

Eclipse



IntelliJ



In the following example, the createMyObject method is tagged:

```
public class MyObjectFactory {
    /**
     * @jtest.factory
     */
    public static MyObject createMyObject(int param) {
        return new MyObject(param);
    }
}
```

If the method is not tagged, UTA will not use it during test creation, and myObject will be initialized to null:

```
@Test
public void testMyMethodUnderTest() throws Throwable {
    // When
    MyObject myObject = null;
    String results = myMethodUnderTest(myObject);

    // Then
    // assertEquals("", results);
}
```

If the method is tagged, UTA will use it to create the object in the test, and your code will look as follows:

```
@Test
public void testMyMethodUnderTest() throws Throwable {
    // When
    int param = 0;
    MyObject myObject = MyObjectFactory.createMyObject(param);
    String results = myMethodUnderTest(myObject);

    // Then
    // assertEquals("", results);
}
```

A tagged factory method must be in the same project as any tests that are going to use it.

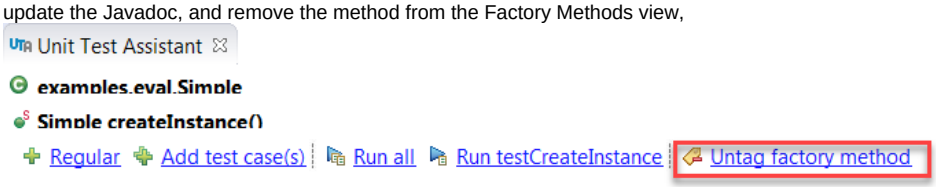
If you want UTA to use a factory method that was tagged in another project, create a new method in the project where the tests will be generated, tag it as a factory method, and point to the external factory method you want to use:

```
public class TestUtil {
    /**
     * @jtest.factory
    */
    public static MyObject createMyObject() {
        return MyObjectFactory.createMyObject(3);
    }
}
```

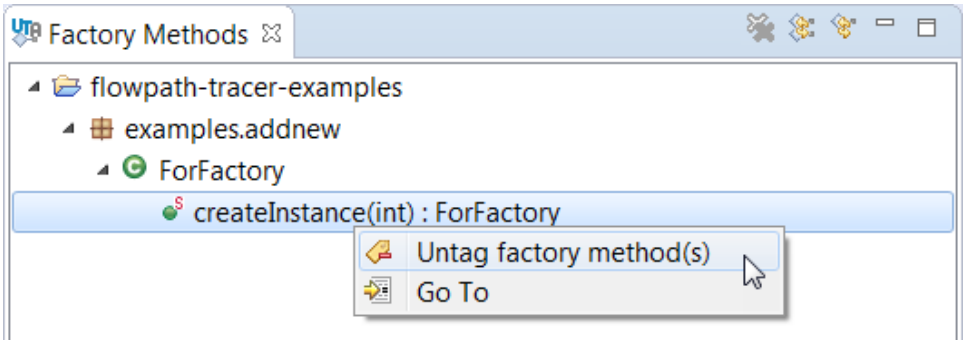
Untagging Factory Methods

You can remove the **@jtest.factory** tag from Javadoc in one of the following ways:

- manually remove the tag from the Javadoc,
- select a tagged method in the editor, and then use the **Untag factory method** action in the Unit Test Assistant interface; this will automatically update the Javadoc, and remove the method from the Factory Methods view,



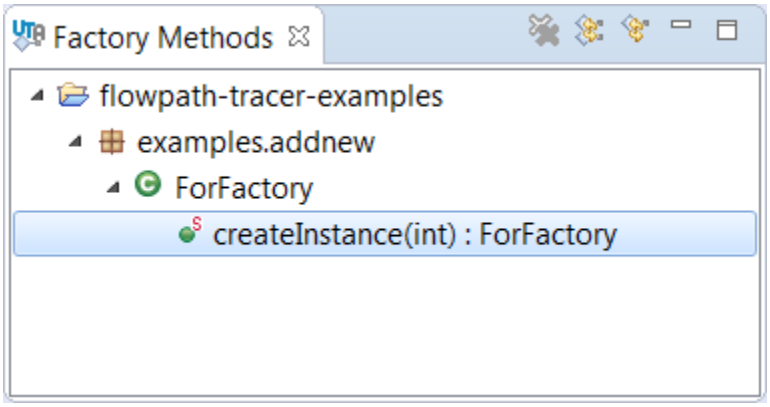
- right-click a method in the Factory Methods view, and choose the **Untag factory method(s)** option; this will automatically update the Javadoc, and remove the method from the Factory Methods view.



If you select one or more class, package, or project, choosing the **Untag factory method(s)** option will remove all @jtest.factory tags within the selection.

Scanning for Factory Methods

You can configure UTA to automatically scan all projects for tagged factory methods by enabling the **Automatically discover tagged factory methods** opt ion; see Configuring Preferences. With this option enabled, UTA will automatically scan your projects on the IDE startup, and display the discovered factory methods in the Factory Methods view:



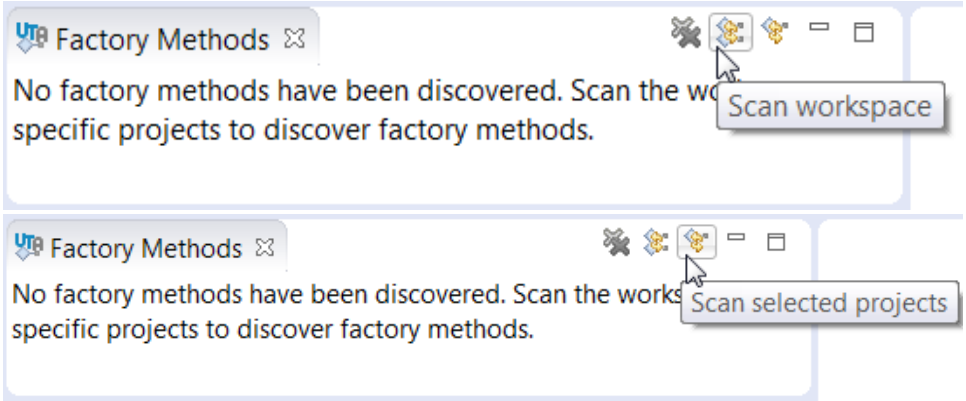
The view will be automatically updated when you modify your project(s) (for example, when you create or save a Java file, or create or delete a new project).

Alternatively, you can manually instruct UTA to scan your code to detect tagged factory methods.

1. If not already open, choose **Parasoft> Show View> Factory Methods** from the IDE menu bar.
2. Select a project(s) in the Package Explorer.

   > UTA will not scan projects selected in the Project Explorer or the Navigator.

3. Click the **Scan workspace** or **Scan selected projects** button in the Factory Methods menu to scan a specific project(s) or the entire workspace for factory methods:



Using Factory Methods

The factory methods that are discovered by UTA and displayed in the Factory Methods view will automatically be used during test generation when you use one of the following options:

- **Instantiate using factory** in the Unit Test Assitant view. The action will show up when a relevant factory method is available for the selected type.
- **Add test case(s)** in the Unit Test Assistant view, or in the context menu (see Creating Multiple Unit Tests) if a relevant factory method is available for the type that needs to be initialized. If more than one factory method is available for a specific type, UTA will prioritize factory methods whose return type is the closest match to the type of the variable being initialized. UTA will also prefer factory methods in the same package or in the closest parent package.