(http://baeldung.com)

S ert S s

(/datadog)

Last modified: April 23, 2018

by baeldung (http://www.baeldung.com/author/baeldung/)

Spring (http://www.baeldung.com/category/spring/) +

I just announced the new *Spring 5* modules in REST With Spring:

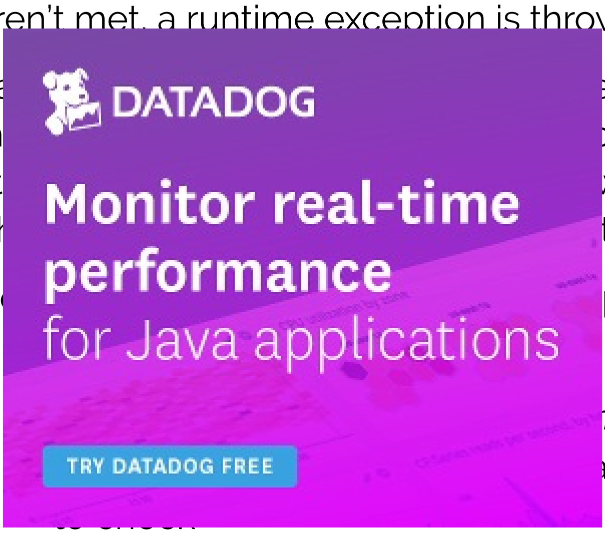
>> CHECK OUT THE COURSE (/rest-with-spring-course#new-modules)

1. Overview

In this tutorial, we'll focus on and describe the purpose of the Spring *Assert* (https://docs.spring.io/spring-framework/docs/current/javadoc-api/org/springframework/util/Assert.html) class and demonstrate how to use it.

2. Purpose of the *Assert* Class

The Spring *Assert* class helps us validate arguments. **By using methods of the *Assert* class, we can write assumptions which we expect to be true.** And if they aren't met, a runtime exception is thrown.



Ea ed with the Java *assert* (h ocs/technotes/guides/language/assert.html) st ws an *Error* at runtime if its condition fails. Th tions can be disabled.

H pring *Assert*'s methods:

ntException or *IllegalStateException*
rgument for validation or a logical condition to check

- The last parameter is usually an exception message which is displayed if the validation fails
- The message can be passed either as a *String* parameter or as a *Supplier<String>* parameter

Also note that despite the similar name, Spring assertions have nothing in common with the assertions of JUnit (https://junit.org/) and other testing frameworks. Spring assertions aren't for testing, but for debugging.

3. Example of Use

Let's define a *Car* class with a public method *drive()*:

```
public class Car {
    private String state = "stop";

    public void drive(int speed) {
        Assert.isTrue(speed > 0, "speed must be positive");
        this.state = "drive";
        // ...
    }
}
```

We can see how speed must be a positive number. The above row is a short way to check the condition and throw an exception if the condition fails:

```
if (!(speed > 0)) {
    throw new IllegalArgumentException("speed must be positive");
}
```

Each *Assert*'s public method contains roughly this code – a conditional block with a runtime exception from which the application is not expected to recover.

If a negative argument,

ar n will be thrown:

```
lang.IllegalArgumentException: speed must be
```

4

4.1. *isTrue()*

This assertion was discussed above. It accepts a *boolean* condition and throws an *IllegalArgumentException* when the condition is false.

4.2. *state()*

The *state()* method has the same signature as *isTrue()* but throws the *IllegalStateException*.

As the name suggests, it should be used when the method mustn't be continued because of an illegal state of the object.

Imagine that we can't call the *fuel()* method if the car is running. Let's use the *state()* assertion in this case:

```
public void fuel() {
    Assert.state(this.state.equals("stop"), "car must be stopped");
    // ...
}
```

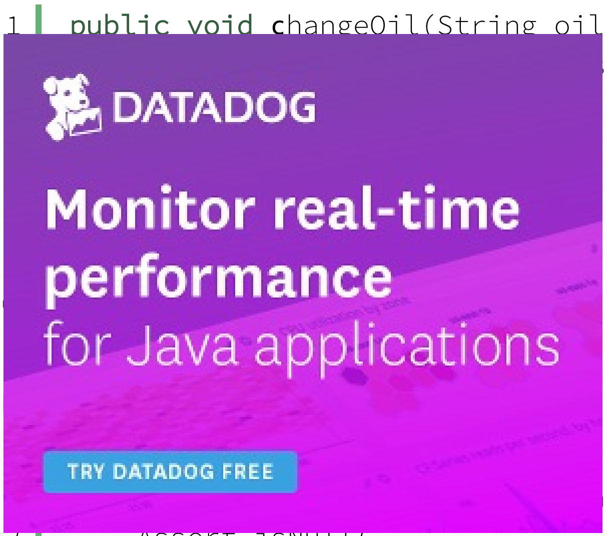
Of course, we can validate everything using logical assertions. But for better readability, we can use additional assertions which make our code more expressive.

5. Object and Type Assertions

5.1. *notNull()*

We can assume that an object is not *null* by using the *notNull()* method:

```
public void changeOil(String oil) {
                                  tn't be null");
```



(/datadog)

5

O object is *null* using the *isNull()* method:

```
                                   ttery carBattery) {
    Assert.isNull(
      carBattery.getCharge(),
      "to replace battery the charge must be null");
    // ...
}
```

5.3. *isInstanceOf()*

To check if an object is an instance of another object of the specific type we can use the *isInstanceOf()* method:

```
public void changeEngine(Engine engine) {
    Assert.isInstanceOf(ToyotaEngine.class, engine);
    // ...
}
```

In our example, the check passes successfully as *ToyotaEngine* is a subclass of *Engine.*

5.4. *isAssignable()*

To check types, we can use *Assert.isAssignable()*:

```
public void repairEngine(Engine engine) {
    Assert.isAssignable(Engine.class, ToyotaEngine.class);
    // ...
}
```

Two recent assertions represent an *is-a* relationship.

6. Text assertions

Te ecks on *String* arguments.

6

W eaning it contains at least one whitespace, by us

```
tring key) {
ust not be null and must not the empty");
    // ...
}
```

6.2. *hasText()*

We can strengthen the condition and check if a *String* contains at least one non-whitespace character, by using the *hasText()* method:

```
public void startWithHasText(String key) {
    Assert.hasText(
      key,
      "key must not be null and must contain at least one non-whitespace  cha
    // ...
}
```

6.3. *doesNotContain()*

We can determine if a *String* argument doesn't contain a specific substring by using the *doesNotContain()* method:

```
public void startWithNotContain(String key) {
    Assert.doesNotContain(key, "123", "key mustn't contain 123");
    // ...
}
```

7. Collection and Map Assertions

7.1. *notEmpty()* for collections

As the name says the notEmpty() method asserts that a collection is not empty m at least one element:

```
ring> repairParts) {
```

```
 mustn't be empty");
```

7.2. *notEmpty()* for maps

The same method is overloaded for maps, and we can check if a map is not empty and contains at least one entry:

```
public void repair(Map<String, String> repairParts) {
    Assert.notEmpty(
      repairParts,
      "map of repairParts mustn't be empty");
    // ...
}
```

8. Array Assertions

8.1. *notEmpty()* for arrays

Finally, we can check if an array is not empty and contains at least one element by using the *notEmpty()* method:

```
public void repair(String[] repairParts) {
    Assert.notEmpty(
      repairParts,
      "array of repairParts mustn't be empty");
    // ...
}
```

8.2. *noNullElements()*

We can verify that an array doesn't contain *null* elements by using the *noNullElements()* method:

```
ing[] repairParts) {

n't contain null elements");
```

N array is empty, as long as there are no *null* el

9. Conclusion

In this article, we explored the *Assert* class. This class is widely used within the Spring framework, but we could easily write more robust and expressive code taking advantage of it.

As always, the complete code for this article can be found in the GitHub project (https://github.com/eugenp/tutorials/tree/master/spring-5).

Which of these is closest to your current job/role?

Developer

Senior Developer

Lead Developer

Architect

Manager

Leave a Reply

2 Comments on "Spring Assert Statements"

Join the discussion

Subscribe

newest oldest most voted

ThingsDevelopersDo

tion("speed must be positive");

f(speed <0)

1 day ago

(/datadog)

(http://www.baeldung.com/author/loredana-crusoveanu/)

m/author/loredana-crusoveanu/)

Editor

ivalent to the code above it which has the condition "speed > 0". So this is negating that condition to throw an exception explicitly. But you're right, it does look jarring by itself.

+ 0 − Reply 1 day ago

CATEGORIES

SPRING (HTTP://WWW.BAELDUNG.COM/CATEGORY/SPRING/)

REST (HTTP://WWW.BAELDUNG.COM/CATEGORY/REST/)

JAVA (HTTP://WWW.BAELDUNG.COM/CATEGORY/JAVA/)

SECURITY (HTTP://WWW.BAELDUNG.COM/CATEGORY/SECURITY-2/)

PERSISTENCE (HTTP://WWW.BAELDUNG.COM/CATEGORY/PERSISTENCE/)

JACKSON (HTTP://WWW.BAELDUNG.COM/CATEGORY/JACKSON/)

HTTP CLIENT (HTTP://WWW.BAELDUNG.COM/CATEGORY/HTTP/)

KOTLIN (HTTP://WWW.BAELDUNG.COM/CATEGORY/KOTLIN/)

SERIES

JAVA "BACK TO BASICS" TUTORIAL (HTTP://WWW.BAELDUNG.COM/JAVA-TUTORIAL)

JACKSON JSON TUTORIAL (HTTP://WWW.BAELDUNG.COM/JACKSON)

HTTPCLIENT 4 TUTORIAL (HTTP://WWW.BAELDUNG.COM/HTTPCLIENT-GUIDE)

REST WITH SPRING TUTORIAL (HTTP://WWW.BAELDUNG.COM/REST-WITH-SPRING-SERIES/)

SPRING PERSISTENCE TUTORIAL (HTTP://WWW.BAELDUNG.COM/PERSISTENCE-WITH-SPRING-SERIES/)

SECURITY WITH SPRING (HTTP://WWW.BAELDUNG.COM/SECURITY-SPRING)

ABOUT

ABOUT BAELDUNG (HTTP://WWW.BAELDUNG.COM/ABOUT/)

THE COURSES (HTTP://COURSES.BAELDUNG.COM)

CONSULTING WORK (HTTP://WWW.BAELDUNG.COM/CONSULTING)

META BAELDUNG (HTTP://META.BAELDUNG.COM/)

THE FULL ARCHIVE (HTTP://WWW.BAELDUNG.COM/FULL_ARCHIVE)

WRITE FOR BAELDUNG (HTTP://WWW.BAELDUNG.COM/CONTRIBUTION-GUIDELINES)

CONTACT (HTTP://WWW.BAELDUNG.COM/CONTACT)

COMPANY INFO (HTTP://WWW.BAELDUNG.COM/BAELDUNG-COMPANY-INFO)

TERMS OF SERVICE (HTTP://WWW.BAELDUNG.COM/TERMS-OF-SERVICE)

PRIVACY POLICY (HTTP://WWW.BAELDUNG.COM/PRIVACY-POLICY)

EDITORS (HTTP://WWW.BAELDUNG.COM/EDITORS)

MEDIA KIT (PDF) (HTTPS://S3.AMAZONAWS.COM/BAELDUNG.COM/BAELDUNG+-+MEDIA+KIT.PDF)

Which of these is closest to your current job/role?

Developer

Senior Developer

Lead Developer

Architect

Manager