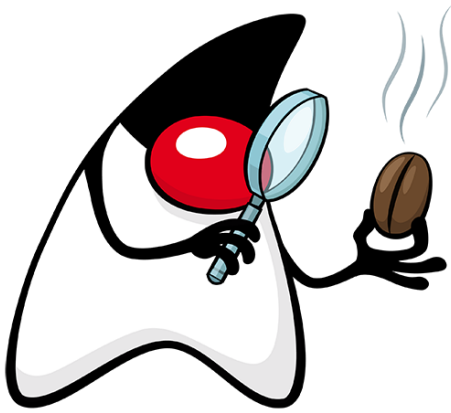
The TCK Process of Bean Validation 2.0 (JSR 380)



Overview

A TCK, or Technology Compatibility Kit, is one of the three required pieces for any JSR (the other two being the specification document and the reference implementation). The TCK is a set of tools and tests to verify that an implementation of the technology conforms to the specification.

The Bean Validation TCK is

- A **comprehensive** test suite of the Bean Validation API that each implementor must pass
- Fully **open-source** (under Apache Software License 2.0)
- Hosted on GitHub: https://github.com/beanvalidation/beanvalidation-tck
- Thoroughly **documented**: http://docs.jboss.org/hibernate/beanvalidation/tck/2.0/reference/html_single/

Structure

The Bean Validation TCK comprises:

- > 950 Arquillian-based **unit tests**
    - Must be **run in Java EE container** for certification
    - Alternative Java-SE-based set-up for development purposes
- A **signature file** for SigTest (http://wiki.netbeans.org/SigTest) to assert 100% API compatibility of API JARs with the specified API

Linking of TCK Tests to Specification

When developing the TCK, it is vital to have an understanding of which parts of the specification text are covered by corresponding TCK tests and where more tests are needed. For this purpose:

- Phrases and statements in the spec that are relevant for the TCK are **marked with the "tck-testable" label**:

```
[[constraintdeclarationvalidationprocess-inheritance]]

=== Inheritance (interface and superclass)

[tck-testable]#A constraint declaration can be placed on an interface.# [tck-testable]#For a given class, constraint dec

[tck-testable]#The effect of constraint declarations is cumulative. Constraints declared on a superclass getter will be

[[constraintdeclarationvalidationprocess-groupsequence]]

=== Group and group sequence

A group defines a subset of constraints. Instead of validating all constraints for a given object graph, only a subset i

[tck-testable]#Groups are represented by interfaces.#
```

- All "tck-testable" phrases are **extracted** into the *tck-audit.xml* file:

```
    <section id="constraintdeclarationvalidationprocess-inheritance" title="Inheritance (interface and superclass)" leve
        <!-- 5.3 - CONSTRAINTDECLARATIONVALIDATIONPROCESS_INHERITANCE -->
        <assertion id="a">
            <text>A constraint declaration can be placed on an interface.</text>
        </assertion>
        <assertion id="b">
            <text>For a given class, constraint declarations held on superclasses as well as interfaces are evaluated by
        </assertion>
        <assertion id="c">
            <text>The effect of constraint declarations is cumulative. Constraints declared on a superclass getter will
        </assertion>
    </section>
    <section id="constraintdeclarationvalidationprocess-groupsequence" title="Group and group sequence" level="2">
        <!-- 5.4 - CONSTRAINTDECLARATIONVALIDATIONPROCESS_GROUPSEQUENCE -->
        <assertion id="a">
            <text>If no group is explicitly declared, a constraint belongs to the Default group.</text>
        </assertion>
        <assertion id="b">
            <text>Groups are represented by interfaces.</text>
        </assertion>
```

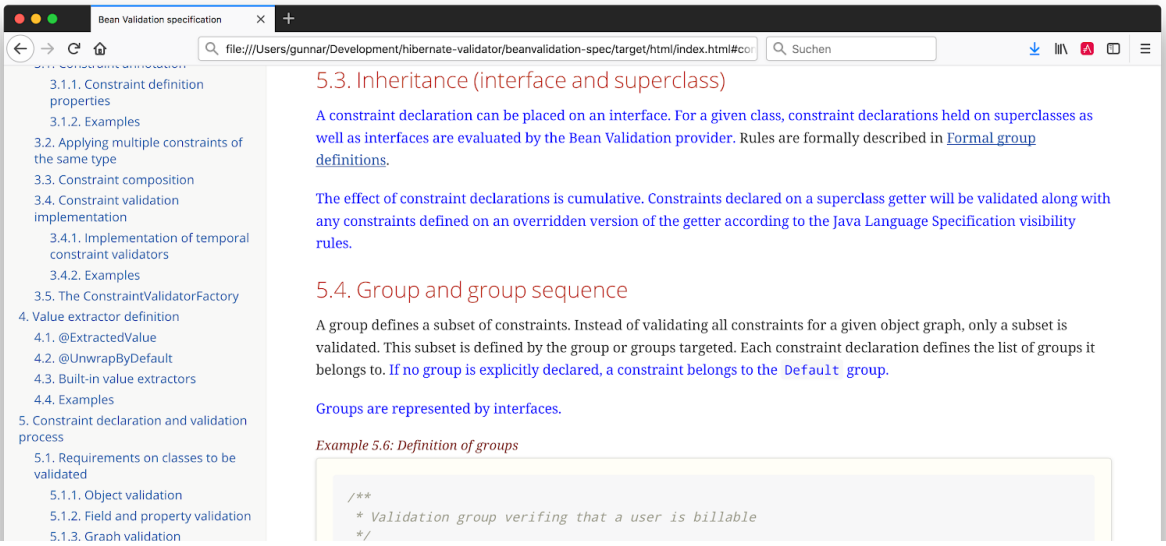
- TCK **tests are linked to "tck-testable"** spec statements through the *@SpecAssertion* annotation:

```
/**
 * @author Hardy Ferentschik
 */
@SpecVersion(spec = "beanvalidation", version = "2.0.0")
public class ConstraintInheritanceTest extends AbstractTCKTest {

    @Deployment
    public static WebArchive createTestArchive() {
        return webArchiveBuilder()
                .withTestClassPackage( ConstraintInheritanceTest.class )
                .build();
    }

    @Test
    @SpecAssertion(section = Sections.CONSTRAINTDECLARATIONVALIDATIONPROCESS_INHERITANCE, id = "b")
    public void testConstraintsOnSuperClassAreInherited() {
        BeanDescriptor beanDescriptor = getValidator().getConstraintsForClass( Bar.class );

        String propertyName = "foo";
        assertTrue( beanDescriptor.getConstraintsForProperty( propertyName ) != null );
        PropertyDescriptor propDescriptor = beanDescriptor.getConstraintsForProperty( propertyName );

        Annotation constraintAnnotation = propDescriptor.getConstraintDescriptors()
                .iterator()
                .next().getAnnotation();
        assertTrue( constraintAnnotation.annotationType() == NotNull.class );
    }
```

- A **report is generated** that shows how many "tck-testable" statements have corresponding tests:

5 Constraint declaration and validation process [constraintdeclarationvalidationprocess]	0	0	0	0	0	
5.1 Requirements on classes to be validated [constraintdeclarationvalidationprocess-requirements]	5	4	4	0	4	100.00%
5.1.1 Object validation [constraintdeclarationvalidationprocess-requirements-objectvalidation]	2	2	2	0	2	100.00%
5.1.2 Field and property validation [constraintdeclarationvalidationprocess-requirements-propertyvalidation]	5	5	5	0	5	100.00%
5.1.3 Graph validation [constraintdeclarationvalidationprocess-requirements-graphvalidation]	16	14	14	0	14	100.00%
5.1.3.1 Examples [constraintdeclarationvalidationprocess-requirements-graphvalidation-examples]	0	0	0	0	0	
5.2 Constraint declaration [constraintdeclarationvalidationprocess-constraintdeclaration]	1	1	1	0	1	100.00%
5.3 Inheritance (interface and superclass) [constraintdeclarationvalidationprocess-inheritance]	3	3	3	0	3	100.00%
5.4 Group and group sequence [constraintdeclarationvalidationprocess-groupsequence]	4	4	4	0	4	100.00%
5.4.1 Group inheritance [constraintdeclarationvalidationprocess-groupsequence-groupinheritance]	2	2	2	0	2	100.00%
5.4.2 Group sequence [constraintdeclarationvalidationprocess-groupsequence-groupsequence]	10	7	7	0	7	100.00%
5.4.3 Redefining the Default group for a class [constraintdeclarationvalidationprocess-groupsequence-redefiningdefaultgroup]	4	3	3	0	3	100.00%
5.4.4 Implicit grouping [constraintdeclarationvalidationprocess-groupsequence-implicitgrouping]	1	1	1	0	1	100.00%

- Spec document can be rendered to highlight all "tck-testable" statements, allowing to visually identify sections with lacking coverage:



Certification

In order to certify a Bean Validation implementation as a fully-compliant implementation, the spec lead or their designate

- **runs the TCK** against the implementation in a Java EE container
- **runs the SigTest** check against the implementation's Bean Validation API JAR (if present)

Compatible implementations are listed on https://beanvalidation.org/. Once certified, implementations must not be re-certified after new releases.

TCK Appeals Process (official description)

- Any Bean Validation implementor may challenge any test case (i.e. @Test method), test case configuration (e.g. @Deployment, validation.xml), test entities, annotations and other resources
- Challenges are submitted by opening a JIRA issue at https://hibernate.atlassian.net/browse/BVTCK
    - if needed for security reasons, visibility can be constrained to private
    - Can send email to beanvalidation-tck@redhat.com upfront in case of uncertainties
- Challenges will be addressed in a timely fashion by the Bean Validation TCK Lead
- Discussion in JIRA to acknowledge the issue and possible solutions; once resolved, appellant should, within 30 days, either close the issue if they agree, or reopen the issue if they do not believe the issue to be resolved
- Resolved issues not addressed for 30 days will be closed by the TCK Lead. If the TCK Project Lead and appellant are unable to agree on the issue resolution, it will be referred to the JSR 380 specification lead
- A new TCK release with updated (or removed) tests will be done
- Some numbers:
    - **18 appeals** in Bean Validation 1.0 - 2.0 (17 Fixed, one rejected)
    - Existing appeals **resolved within ~6 days**