Existing Test Suite

There is a lot of fun to look through the test cases of our project and here're three interesting test cases we found.

1. CombinedBitSetTests (org.apache.lucene.util.CombinedBitSetTests)

This test case tests a data structure class named CombinedBitSet, which is used in ElasticSearch for improving the speed of the calculation of bitsets. The combined bitset class is a BitSet implementation that combines two instances of BitSet and Bits to provide a single merged view. In this test, It contains four different tests: testEmpty, testSparse, testDense, and testRandom. These tests different types of combined bitset and verify whether the result of the combination is correct.

From this test suite, we found that BitSet is a very important data structure in our project(by reading the tech blog of ElasticSearch[1]). ElasticSearch uses it for simple exclusion tasks because of the high speed of doing bit calculations.

Here's the code of this test.

```
public class CombinedBitSetTests extends ESTestCase {
    public void testEmpty() {
        for (float percent : new float[] {0f, 0.1f, 0.5f, 0.9f, 1f}) {
            testCase(randomIntBetween(1, 10000), 0f, percent);
            testCase(randomIntBetween(1, 10000), percent, 0f);
        }
    }

    public void testSparse() {
        for (float percent : new float[] {0f, 0.1f, 0.5f, 0.9f, 1f}) {
            testCase(randomIntBetween(1, 10000), 0.1f, percent);
            testCase(randomIntBetween(1, 10000), percent, 0.1f);
        }
    }

    public void testDense() {
        for (float percent : new float[] {0f, 0.1f, 0.5f, 0.9f, 1f}) {
            testCase(randomIntBetween(1, 10000), 0.9f, percent);
            testCase(randomIntBetween(1, 10000), percent, 0.9f);
```

```
        }
    }

    public void testRandom() {
        int iterations = atLeast(10);
        for (int i = 0; i < iterations; i++) {
            testCase(randomIntBetween(1, 10000), randomFloat(),
randomFloat());
        }
    }

    private void testCase(int numBits, float percent1, float percent2) {
        BitSet first = randomSet(numBits, percent1);
        BitSet second = randomSet(numBits, percent2);
        CombinedBitSet actual = new CombinedBitSet(first, second);
        FixedBitSet expected = new FixedBitSet(numBits);
        or(expected, first);
        and(expected, second);
        assertEquals(expected.cardinality(), actual.cardinality());
        assertEquals(expected, actual, numBits);
        for (int i = 0; i < numBits; ++i) {
            assertEquals(expected.nextSetBit(i), actual.nextSetBit(i));
            assertEquals(Integer.toString(i), expected.prevSetBit(i),
actual.prevSetBit(i));
        }
    }
```

2. DeleteRequestTests (org.elasticsearch.action.delete.DeleteRequestTests)

This test suite tests the validation of the delete request used in ElasticSearch. A delete request means a type of request to delete a document from an index by its id and type. We found this test case in the module *action,* which contains a series of operations in our project. And we think the way the project tests the validation of the request is interesting because it uses a method named *validate()* to return an exception class, and then check whether it is null in the assert method.

In this test, the testValidatoin method first tests the request with validated parameters, then compares it with the one that is not. If one of the parameters is missing, the validate() method

will return an error. This test case helps us understand how to use the delete request class properly.

Here's the code of this test.

```
public class DeleteRequestTests extends ESTestCase {

    public void testValidation() {
        {
            final DeleteRequest request = new DeleteRequest("index4", "0");
            final ActionRequestValidationException validate =
request.validate();

            assertThat(validate, nullValue());
        }

        {
            final DeleteRequest request = new DeleteRequest("index4",
null);
            final ActionRequestValidationException validate =
request.validate();

            assertThat(validate, not(nullValue()));
            assertThat(validate.validationErrors(), hasItems("id is
missing"));
        }

    }
}
```

3. MessageDigestsTests (org.elasticsearch.common.hash.MessageDigestsTests)

This test suite tests the function of the class MessageDigests, which is used for providing different string message digest algorithms like SHA-1, MD5, etc. These algorithms are secure one-way hash functions, it takes an arbitrary sized string as input and returns a fixed-sized value as output. This test suite checks that whether the outputs of different digest algorithms provided by the MessageDigests class are correct. We think it is interesting because the way how this test work is to use hard-coded string and company it with the output of the digest algorithm implemented in this project. It is just easy and necessary.

```
public class MessageDigestsTests extends ESTestCase {
    private void assertHash(String expected, String test, MessageDigest
messageDigest) {
        String actual =
MessageDigests.toHexString(messageDigest.digest(test.getBytes(StandardCharsets.UT
F_8)));
        assertEquals(expected, actual);
    }

    public void testMd5() throws Exception {
        assertHash("d41d8cd98f00b204e9800998ecf8427e", "", MessageDigests.md5());
        assertHash("900150983cd24fb0d6963f7d28e17f72", "abc",
MessageDigests.md5());
        assertHash("8215ef0796a20bcaaae116d3876c664a",
            "abcdbcdecdefdefgefghfghighijhijkijkljklmklmnlmnomnopnopq",
MessageDigests.md5());
        assertHash("7707d6ae4e027c70eea2a935c2296f21",
            new String(new char[1000000]).replace("\0", "a"),
MessageDigests.md5());
        assertHash("9e107d9d372bb6826bd81d3542a419d6", "The quick brown fox jumps
over the lazy dog", MessageDigests.md5());
        assertHash("1055d3e698d289f2af8663725127bd4b", "The quick brown fox jumps
over the lazy cog", MessageDigests.md5());
    }

    public void testSha1() throws Exception {
        assertHash("da39a3ee5e6b4b0d3255bfef95601890afd80709", "",
MessageDigests.sha1());
        assertHash("a9993e364706816aba3e25717850c26c9cd0d89d", "abc",
MessageDigests.sha1());
        assertHash("84983e441c3bd26ebaae4aa1f95129e5e54670f1",
            "abcdbcdecdefdefgefghfghighijhijkijkljklmklmnlmnomnopnopq",
MessageDigests.sha1());
        assertHash("34aa973cd4c4daa4f61eeb2bdbad27316534016f",
            new String(new char[1000000]).replace("\0", "a"),
MessageDigests.sha1());
        assertHash("2fd4e1c67a2d28fced849ee1bb76e7391b93eb12", "The quick brown
fox jumps over the lazy dog", MessageDigests.sha1());
        assertHash("de9f2c7fd25e1b3afad3e85a0bd17d9b100db4b3", "The quick brown
fox jumps over the lazy cog", MessageDigests.sha1());
```

```
    }

    public void testSha256() throws Exception {

assertHash("e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855",
"", MessageDigests.sha256());

assertHash("ba7816bf8f01cfea414140de5dae2223b00361a396177a9cb410ff61f20015ad",
"abc", MessageDigests.sha256());

assertHash("248d6a61d20638b8e5c026930c3e6039a33ce45964ff2167f6ecedd419db06c1",
            "abcdbcdecdefdefgefghfghighijhijkijkljklmklmnlmnomnopnopq",
MessageDigests.sha256());

assertHash("cdc76e5c9914fb9281a1c7e284d73e67f1809a48a497200e046d39ccc7112cd0",
            new String(new char[1000000]).replace("\0", "a"),
MessageDigests.sha256());

assertHash("d7a8fbb307d7809469ca9abcb0082e4f8d5651e46d3cdb762d02d0bf37c9e592",
            "The quick brown fox jumps over the lazy dog",
MessageDigests.sha256());

assertHash("e4c4d8f3bf76b692de791a173e05321150f7a345b46484fe427f6acc7ecc81be",
            "The quick brown fox jumps over the lazy cog",
MessageDigests.sha256());
    }
```

Reference

[1]https://www.elastic.co/blog/all-about-elasticsearch-filter-bitsets
[2]https://github.com/elastic/elasticsearch