**Hamcrest Assertation**

<!-- https://mvnrepository.com/artifact/org.hamcrest/hamcrest-all -->

<dependency>

<groupId>org.hamcrest</groupId>

<artifactId>hamcrest-all</artifactId>

<version>1.3</version>

<scope>test</scope>

</dependency>

**Hamcrest Text Matchers**

isEmpty

String favoriteCereal = "";

assertThat(favoriteCereal, isEmptyOrNullString());

isEmptyOrNullString

String favoriteCereal = null;

assertThat(favoriteCereal, isEmptyOrNullString());

EqualTo

String favoriteCereal = "cinnamon life";

assertThat(favoriteCereal, equalTo("cinnamon life"));

EqualToIgnoreingWithSpace

String favoriteCereal = "CINNAMON LIFE ";

assertThat(favoriteCereal, equalToIgnoringWhiteSpace("cinnamon life"));

containsString

String cereal = "mini wheats";

assertThat(cereal, containsString("mini"));

EndsWith

String cereal = "corn flakes";

assertThat(cereal, endsWith("s"));

startsWith

String cereal = "honey smacks";

assertThat(cereal, startsWith("honey"));

stringContainsInOrder

String cereal = "apple jacks";

assertThat(cereal, stringContainsInOrder(Lists.newArrayList("apple", "jacks")));

Hamcrest Number Matchers

Is closeTo

assertThat(200.24, is(closeTo(200, 1)));

greaterThan

assertThat(20, greaterThan(18));

every Item greaterThan

List<Integer> ages = Lists.newArrayList(21, 25, 30);

assertThat(ages, everyItem(greaterThan(18)));

every Item greaterThanOrEqualTo

List<Integer> ages = Lists.newArrayList(21, 25, 30, 18);

assertThat(ages, everyItem(greaterThanOrEqualTo(18)));

every Item lessThan

List<Integer> ages = Lists.newArrayList(21, 25, 30);

assertThat(ages, everyItem(lessThan(31)));

every Item lessThanOrEqualTo

List<Integer> ages = Lists.newArrayList(21, 25, 30, 18);

assertThat(ages, everyItem(lessThanOrEqualTo(30)));

**Hamcrest Core Matchers**

allOf

String microBrew = "Lake Louie Brewery Company";

assertThat(microBrew, allOf(startsWith("Lake"), containsString("Brew")));

anyOf

String microBrew = "Grumpy Troll Brewery";

assertThat(microBrew, anyOf(startsWith("brew"), containsString("Troll")));

anything

assertThat("", anything());

combine matchers

String isLite = "Miller Lite";

assertThat(isLite, both(containsString("Miller")).and(containsString("Lite")));

containString

String brewery = "Pabst Brewing Company";

assertThat(brewery, containsString("Brew"));

everyItem

List<Integer> ages = Lists.newArrayList(21, 25, 30, 18);

assertThat(ages, everyItem(greaterThanOrEqualTo(18)));

hasItems

List<String> regionalBreweries = Lists.newArrayList( "Capital Brewery", "City Brewing Company ","Jacob Leinenkugel Brewing Company", "Lakefront Brewery, Inc.", "New Glarus Brewing Company", "Stevens Point Brewery");

assertThat(regionalBreweries, hasItems("Capital Brewery", "City Brewing Company ", "Jacob Leinenkugel Brewing Company", "Lakefront Brewery, Inc.", "New Glarus Brewing Company", "Stevens Point Brewery"));

hasItems w/ matchers

List<String> regionalBreweries = Lists.newArrayList( "Capital Brewery","City Brewing Company ",

"Jacob Leinenkugel Brewing Company","Lakefront Brewery","New Glarus Brewing Company",

"Stevens Point Brewery");

assertThat(regionalBreweries, hasItems( containsString("Brew"),endsWith("y")));

is equalTo

String isLite = "Miller Brewing Company";

assertThat("Miller Brewing Company", is(equalTo(isLite)));

is notNullValue

Set<String> daNull = new HashSet<String>();

assertThat(daNull, is(notNullValue()));

is nullValue

Set<String> daNull = null;

assertThat(daNull, is(nullValue()));

isSameInstance

List<String> someList = new ArrayList<String>();

assertThat(someList, IsSame.<List<String>>sameInstance(someList));

isA

Map<Integer, String> map = new HashMap<Integer, String>();

assertThat(map, isA(Map.class));

equalTo

String spottedCreator = "New Glarus Brewing Company";

assertThat(spottedCreator, equalTo("New Glarus Brewing Company"));

instanceOf

Calendar cal = Calendar.getInstance();

assertThat(cal, instanceOf(Calendar.class));

is sameInstance

String wiBrewery = "Capital Brewery";

String wiRegionalBrewery = "Capital Brewery";

assertThat(wiRegionalBrewery, IsSame.<String>sameInstance(wiBrewery));

endsWith

String baseBallTeam = "Milwaukee brewers";

assertThat(baseBallTeam, endsWith("brewers"));

startsWith

String highSchool = "Tarpon spring spongers";

assertThat(highSchool, startsWith("Tarpon"));

**Hamcrest Collection Matchers**

Iterables

With Size

List<String> cloths = Lists.newArrayList("shirts", "shoes", "pants", "socks");

assertThat(cloths, IsIterableWithSize.<String>iterableWithSize(4));

Has Order

List<String> cloths = Lists.newArrayList("shirts", "shoes", "pants", "socks");

assertThat(cloths, IsIterableContainingInAnyOrder.<String>containsInAnyOrder("shoes", "pants", "shirts", "socks"));

Matchers Order

List<String> cloths = Lists.newArrayList("shirts", "shoes", "pants", "socks");

assertThat(cloths, IsIterableContainingInOrder.<String>contains("shirts", "shoes", "pants", "socks"));

**Array**

With Size

Integer[] numbers = new Integer[] {10, 15, 20};

assertThat(numbers, arrayWithSize(3));

Contains all

Integer[] numbers = new Integer[] {10, 15, 20};

assertThat(numbers, arrayContaining(10, 15, 20));

Contains in any order

Integer[] numbers = new Integer[] {10, 15, 20};

assertThat(numbers, arrayContainingInAnyOrder(20, 10, 15));

Elements equal

Integer[] numbers = new Integer[] {10, 15, 20};

assertThat(numbers,is(array(equalTo(10), equalTo(15), equalTo(20))));

**Collections**

Has size

List<String> fruit = Lists.newArrayList("apple", "banana", "pear", "blackberry", "grape");

assertThat(fruit, hasSize(5));

In order

List<String> fruit = Lists.newArrayList("apple", "banana", "pear", "blackberry", "grape");

assertThat(fruit, contains("apple", "banana", "pear", "blackberry", "grape"));

In any order

List<String> fruit = Lists.newArrayList("apple", "banana", "pear", "blackberry", "grape");

assertThat(fruit, containsInAnyOrder("banana", "apple", "blackberry", "grape", "pear"));

Has item

List<String> fruit = Lists.newArrayList("apple", "banana", "pear", "blackberry", "grape");

assertThat(fruit, hasItem("apple"));

Has items

List<String> fruit = Lists.newArrayList("apple", "banana", "pear", "blackberry", "grape");

assertThat(fruit, hasItems("apple", "pear"));

Element in

List<String> fruit = Lists.newArrayList("apple", "banana", "pear", "blackberry", "grape");

assertThat("apple", isIn(fruit));

Elements is one of

List<String> fruit = Lists.newArrayList("apple", "banana", "pear", "blackberry", "grape");

assertThat("apple", isOneOf(fruit.toArray()));

is empty

List<String> fruit = Lists.newArrayList();

assertThat(fruit, empty());

is not empty

List<String> fruit = Lists.newArrayList("apple", "banana", "pear", "blackberry", "grape");

assertThat(fruit, not(empty()));

Each elements ends with

List<String> cereal = Lists.newArrayList("mini wheats", "corn flakes","honey smacks", "apple jacks",

"lucky charms");

assertThat(cereal, hasItem(endsWith("s")));

**Map**

Has entry

Map<String, String> breeds = Maps.newHashMap();breeds.put("labrador", "buzz"); breeds.put("dachshund", "gypsy"); breeds.put("boxer", "buddy");

assertThat(breeds, hasEntry("labrador", "buzz"));

Has key

Map<String, String> breeds = Maps.newHashMap();breeds.put("labrador", "buzz");

breeds.put("dachshund", "gypsy");breeds.put("boxer", "buddy");

assertThat(breeds, hasKey("labrador"));

Has Value

Map<String, String> breeds = Maps.newHashMap();breeds.put("labrador", "buzz");

breeds.put("dachshund", "gypsy");breeds.put("boxer", "buddy");

assertThat(breeds, hasValue("gypsy"));