

1.3 Method Naming Conventions

/1.0 Conventions/1.3 Method Naming Conventions

Overview

The following coding standards were collected from various technologies, frameworks and articles that form a collection of best practices for method names and what the purpose and/or implementation is expected.

Each of the identifiers presented by this convention are clauses used in whole or as part of an identifier. For example, 'as' can be used as a stand alone function call or part of a method name:

```
// Method that converts parameter and returns integer
Integer i = r.as(Integer.class);

// Method that returns a value of a integer type.
Integer i = r.asInteger();
```

Glossary Of Identifiers

As

Return a representation of an instance as a different type.

This clause identifies a method that converts the object to a different type.

Example

```
var exam1 = Maybe.of(userRepo.findUserById(userId));
var user = exam1.as(UserDetail.class).getName(); // Cast exam1 to a
UserDetail and get the users name.
var str = userId.asString() // Return id value as a string.
```

Find

Return a subset of items matching criteria.

Method searches a collection or database for a specific instance(s) of a item.

The return type should be a Collection, Maybe or Optional.

The *find* clause should follow the Spring Data naming convention. See [Spring Data](#) .

Example

Prototype	Description
List findAll()	Return all instance of User
Optional findUserById(Long id)	Find a specific user with the given id
List findUserLikeLastName(String name)	Find all users with last name like name
List findUserByLastNameAndFirstName(..)	Find all users with last name and first name

From

Method that accept a reference to an Object as a parameter and returns a class with a value extracted from object.

Example

```
var comp = Maybe.from(anOptional); // Creates a Maybe instance from the
value of Optional.
```

Get

Getter method to retrieve the value of a property.

The implementation of a getter should be simple and short; preferably written to qualify for [Method Inlining in the JVM](#).

Of

Method that accepts a reference to an Object as a parameter and returns a wrapper class for the Object.

Example

```
var response = Maybe.of(resp); // Creates a Maybe instance containing
the value of resp.
```

When

If the predicate is true then return the reference to this, otherwise, return a default instance.

Bisects method chaining depending on the result of a Predicate lambda function.

The return value will be one of two possible values depending on the *truth* returned by the Predicate. The when clause is frequently coupled with the *then* clause.

Example

```
var someInt = Maybe.of(50); // Create a Maybe<Integer> with the value 50
var result1 = someInt.when(p -> p > 25).then(p -> p * 10); // result1
will be 500
```

```
var result2 = someInt.when(p -> p <=25).then(p -> p / 10); // result1  
will be None<Integer>
```

With

The *with* clause is used to make a call to a lambda that is a {@link Consumer} instance that will receive an instance of an object that the lambda function is expected to build. If the method name is *with* as a stand alone function then the object passed is usually the instance of the parent class.

This clause forms the DSL semantics is "with(x) do (y)".

Via

The *via* clause is used to make a call to a lambda that is a {@link Consumer} instance that will receive an instance of the object.