AssertUtils

How to Use?

AssertUtils is designed to simplify and streamline **"if-then-throw"** validation patterns. It helps you write **cleaner, more concise, and maintainable** code.

Before: Traditional Validation

```
public void validate(ActivatePhysicalCardRequestModel model) throws
CardException {
    if (ValidatorUtils.isEmpty(model)){
        throw new CardException(CardErrorEnum.INVALID_REQUEST);
    }

    if (ValidatorUtils.isBlank(model.getAccountNo())){
        throw new CardException(CardErrorEnum.ACCOUNT_NO_NOT_FOUND);
    }

    if (ValidatorUtils.isBlank(model.getCardId())){
        throw new CardException(CardErrorEnum.CARD_ID_NOT_FOUND);
    }

    if(ValidatorUtils.isBlank(model.getCardPin())){
        throw new CardException(CardErrorEnum.CARD_PIN_NOT_FOUND);
    }
}
```

After: Using AssertUtils

```
public void validate(ActivatePhysicalCardRequestModel model) throws
CardException {
    AssertUtils.isNotEmpty(model, () -> new
CardException(CardErrorEnum.INVALID_REQUEST));
    AssertUtils.isNotBlank(model.getAccountNo(), () -> new
CardException(CardErrorEnum.ACCOUNT_NO_NOT_FOUND));
    AssertUtils.isNotBlank(model.getCardId(), () -> new
CardException(CardErrorEnum.CARD_ID_NOT_FOUND));
    AssertUtils.isNotBlank(model.getCardPin(), () -> new
CardException(CardErrorEnum.CARD_PIN_NOT_FOUND));
}
```

What If There's No Suitable Assertion Method?

If your validation logic is complex or domain-specific, use: **AssertUtils.isTrue.** For example:

Before:

```
if(conditionA() && conditionB() || (conditionC() && conditionD())) {
    throw new CardException(CardErrorEnum.INVALID_REQUEST);
}
```

After:

```
boolean isValidRequest = conditionA() && conditionB() || (conditionC() &&
conditionD());
AssertUtils.isTrue(isValidRequest, () -> new
CardException(CardErrorEnum.INVALID_REQUEST));
```

If the validation is reusable, first add a method in **ValidatorUtils**, then call it via **AssertUtils**.

In **ValidatorUtils.java**

```
public static boolean someReusableValidation() {
    return false;
}
```

In **AssertUtils.java**

```
public static <X extends IfastPayException> void
someReusableAssertions(Supplier<X> ex) {
    isTrue(true, ex);
}
```

Follow the following signature when you are trying to add a new method in AssertUtils

```
public static <X extends IfastPayException> void yourMethodName(args,
Supplier<X> ex) {
    isTrue(ValidatorUtils.yourMethodName(args), ex);
}
```

Testing Your Assertions

It is always encouraged to write tests for new validation logic. Use **JUnit 5** and **AssertJ** for writing clear and structured tests.

- Use @Nested classes to group related test cases.
- Use @ParameterizedTest with @CsvSource for multiple inputs.
- Follow the naming convention: test_ prefix for all test methods.

```
@Nested
class YourClassTests {
    @ParameterizedTest
    @CsvSource("arg1, arg2")
    void test_yourTestMustStartWithTest(Integer value, boolean expected) {
        assertThat(ValidatorUtils.isZero(value)).isEqualTo(expected);
    }
}
```

References:

1. https://junit.org/
2. https://assertj.github.io/doc/