CATCHING NEW TRICKS

Martin Hořeňovský

CATCHING NEW TRICKS

Martin Hořeňovský

DATA GENERATORS

```
bool is_success_status(int value);

TEST_CASE("Simple generator") {
    auto [input, expected] = GENERATE(table<int, bool>({
        {101, false},
        {200, true},
        {403, false},
        {500, false},
    }));

    REQUIRE(is_success_status(input) == expected);
}
```

```
TEST_CASE("Chaining generators") {
    auto val = GENERATE(take(10, random(0., 10.)));

    REQUIRE(0. <= val);
    REQUIRE(val < 10.);
}
```

```
TEST_CASE("Chaining generators") {
    auto val = GENERATE(take(10, random(0., 10.)));

    REQUIRE(0. <= val);
    REQUIRE(val < 10.);
}
```

There are more generic generators built-in, e.g.

```
TEST_CASE("Chaining generators") {
    auto val = GENERATE(take(10, random(0., 10.)));

    REQUIRE(0. <= val);
    REQUIRE(val < 10.);
}
```

There are more generic generators built-in, e.g. filter

```
TEST_CASE("Chaining generators") {
    auto val = GENERATE(take(10, random(0., 10.)));

    REQUIRE(0. <= val);
    REQUIRE(val < 10.);
}
```

There are more generic generators built-in, e.g. filter, map

```
TEST_CASE("Chaining generators") {
    auto val = GENERATE(take(10, random(0., 10.)));

    REQUIRE(0. <= val);
    REQUIRE(val < 10.);
}
```

There are more generic generators built-in, e.g. filter, map, and repeat.

You can also write your own generators.

TEMPLATED TEST CASES

```
TEMPLATE_TEST_CASE("You can have a test across multiple types", "",
                   int, float) {

    std::vector<TestType> vec;
    vec.reserve(5);

    REQUIRE(vec.size() == 0);
    REQUIRE(vec.capacity() >= 5);
}
```

```
using MyTypes = std::tuple<int, float>;

TEMPLATE_LIST_TEST_CASE("You can use existing type lists", "",
                        MyTypes) {

    std::vector<TestType> vec;
    vec.reserve(5);

    REQUIRE(vec.size() == 0);
    REQUIRE(vec.capacity() >= 5);
}
```

```
TEMPLATE_PRODUCT_TEST_CASE("Products of type lists", "",
                           (std::vector, std::list),
                           (int, float)) {

    TestType x;
    REQUIRE(x.size() == 0);
}
```

There are also macro variants for NTTPs

BENCHMARKS

```
TEST_CASE("Test and benchmark") {
    REQUIRE(fib(20) == 6765);

    BENCHMARK("fib 20") {
        return fib(20);
    };

    REQUIRE(fib(0) == 0);
}
```

RECAP

RECAP

Catch2 contains many cool features

RECAP

Catch2 contains many cool features

You can find them through the documentation.

RECAP

Catch2 contains many cool features

You can find them through the documentation.

Or through reading Catch2's test project(s).

THE END

All snippets in this talk can be found at https://github.com/horenmar/catch-lightning-talk-snippets