Что интересного в kotlinx.fuzz?

Инструментальное тестирование

kotlinx.fuzz

Соколинский Александр
Центр Научного Программирования, МФТИ

Новая библиотека (2025 год)

Новая библиотека (2025 год)

Начало разработки

Первый коммит

Новая библиотека (2025 год)

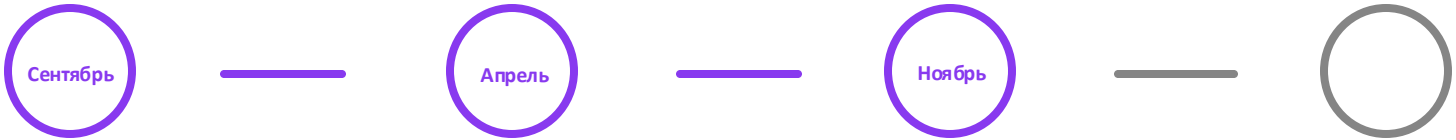
Начало разработки

Первый коммит

Публикация блог-поста

Де-факто релиз

Новая библиотека (2025 год)



Начало разработки

Первый коммит

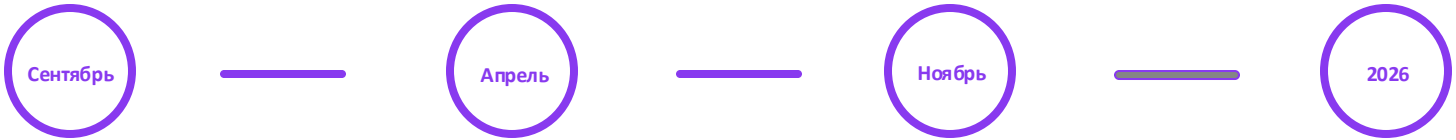
Публикация блог-поста

Де-факто релиз

Мобиус

> Вы здесь <

Новая библиотека (2025 год)



Начало разработки

Первый коммит

Публикация блог-поста

Де-факто релиз

Мобиус

> Вы здесь <

Публикация записей

Или здесь

Приготовьтесь к новому подходу

Приготовьтесь к новому подходу

Приготовьтесь к новому подходу

Инструментальное тестирование

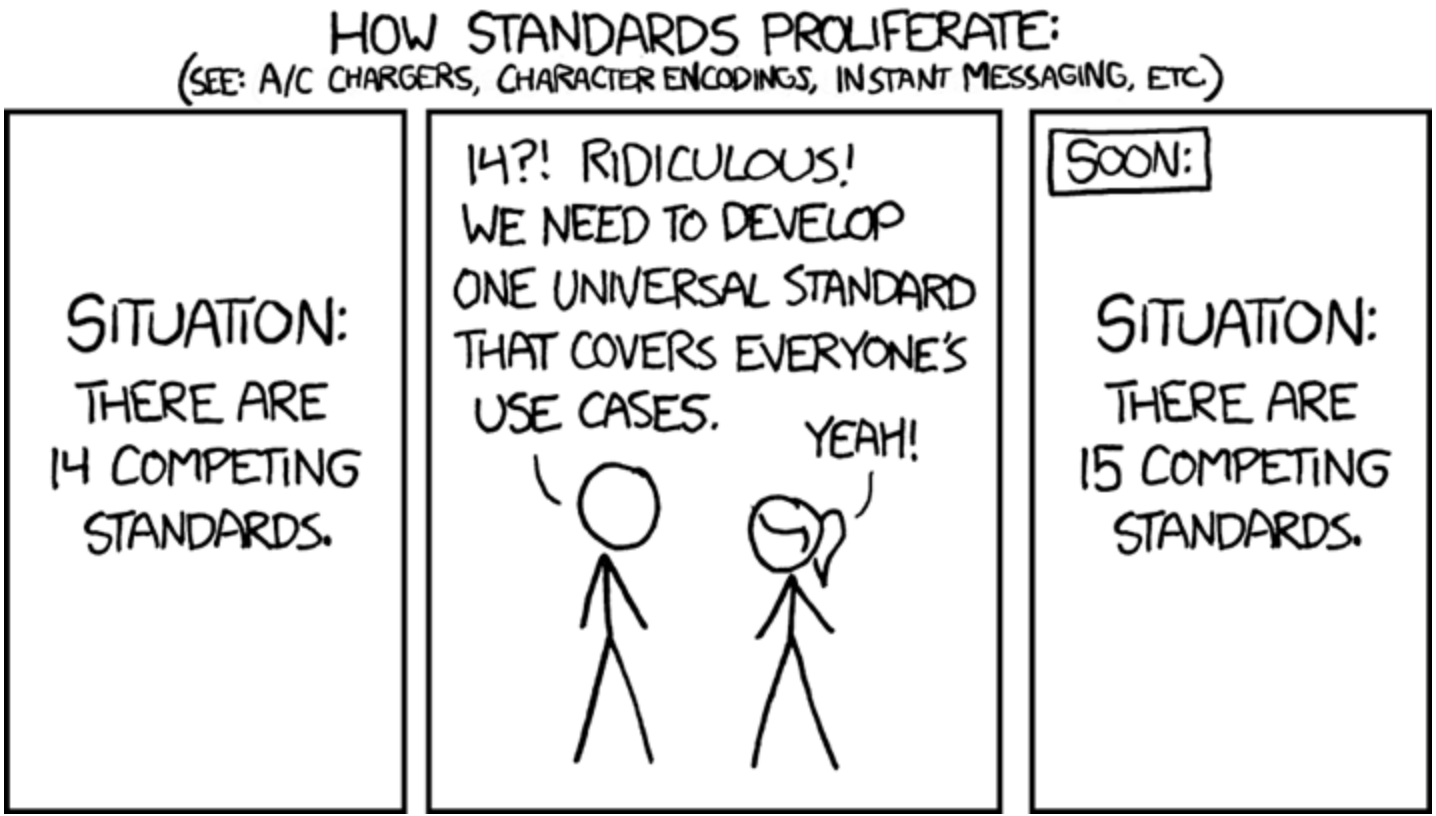
Приготовьтесь к новому подходу

Инструментальное тестирование

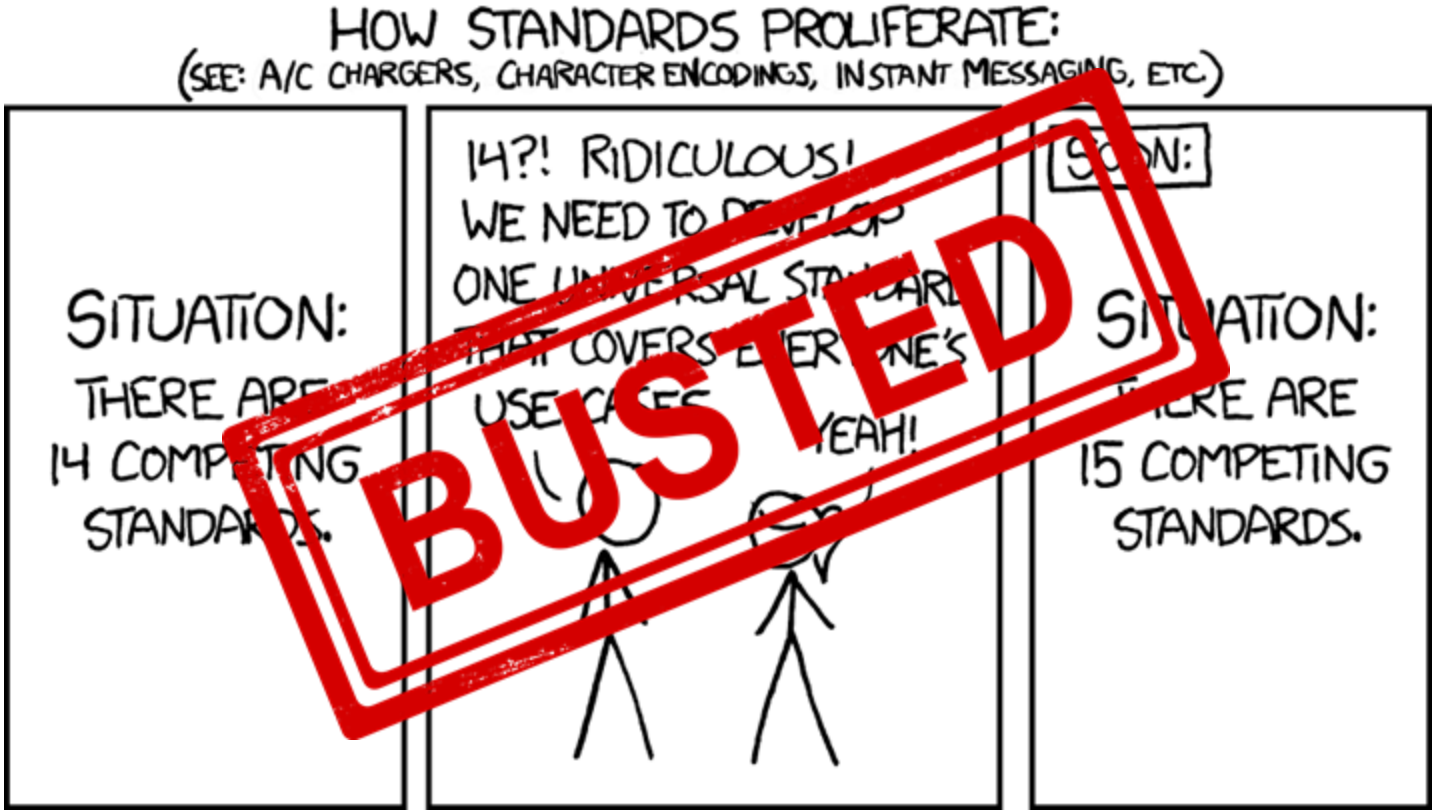
Приготовьтесь к новому подходу

Инструментальное тестирование

kotlinx.fuzz


HOW STANDARDS PROLIFERATE:
(SEE: A/C CHARGERS, CHARACTER ENCODINGS, INSTANT MESSAGING, ETC.)
SITUATION:
THERE ARE
14 COMPETING
STANDARDS.
14?! RIDICULOUS!
WE NEED TO DEVELOP
ONE UNIVERSAL STANDARD
THAT COVERS EVERYONE'S
USE CASES.
YEAH!
SOON:
SITUATION:
THERE ARE
15 COMPETING
STANDARDS.

xkcd.com/927


HOW STANDARDS PROLIFERATE:
(SEE: A/C CHARGERS, CHARACTER ENCODINGS, INSTANT MESSAGING, ETC.)
SITUATION:
14?! RIDICULOUS!
SOON:
SITUATION:
15 COMPETING
STANDARDS.
BUSTED

xkcd.com/927

Существующая проблема юнит-тестирования

```
class ExampleTest {
    @Test
    fun `you can add many chat members`() {
        val chat = Chat.create("Test Chat")
        val y9san9 = User.create("y9san9@gmail.com")
        val y9vad9 = User.create("y9vad9@gmail.com")
        val y9kap = User.create("y9kap@gmail.com")
        chat.addMember(y9san9)
        chat.addMember(y9vad9)
        chat.addMember(y9kap)
        assertTrue(chat.isMember(y9san9))
        assertTrue(chat.isMember(y9vad9))
        assertTrue(chat.isMember(y9kap))
    }
}
```

Существующая проблема юнит-тестирования

example % ./gradlew test

> Task :test

ExampleTest > you can add many chat members PASSED

BUILD SUCCESSFUL in 50ms

3 actionable tasks: 2 executed, 1 up-to-date

Существующая проблема юнит-тестирования

```
class ExampleTest {
    @Test
    fun `you can add many chat members`() {
        val chat = Chat.create("Test Chat")
        val y9san9 = User.create("y9san9@gmail.com")
        val y9vad9 = User.create("y9vad9@gmail.com")
        val y9kap = User.create("y9kap@gmail.com")
        chat.addMember(y9san9)
        chat.addMember(y9vad9)
        chat.addMember(y9kap)
        assertTrue(chat.isMember(y9san9))
        assertTrue(chat.isMember(y9vad9))
        assertTrue(chat.isMember(y9kap))
    }
}
```

<- Очень конкретный тест

Существующая проблема юнит-тестирования

```
class ExampleTest {
    @Test
    fun `you can add many chat members`() {
        val chat = Chat.create("Test Chat") <-
        val y9san9 = User.create("y9san9@gmail.com")
        val y9vad9 = User.create("y9vad9@gmail.com")
        val y9kap = User.create("y9kap@gmail.com")
        chat.addMember(y9san9)
        chat.addMember(y9vad9)
        chat.addMember(y9kap)
        assertTrue(chat.isMember(y9san9))
        assertTrue(chat.isMember(y9vad9))
        assertTrue(chat.isMember(y9kap))
    }
}
```

<- Очень конкретный тест

Существующая проблема юнит-тестирования

```
class ExampleTest {
    @Test
    fun `you can add many chat members`() {
        val chat = Chat.create("Test Chat")
        val y9san9 = User.create("y9san9@gmail.com")
        val y9vad9 = User.create("y9vad9@gmail.com")
        val y9kap = User.create("y9kap@gmail.com")
        chat.addMember(y9san9)
        chat.addMember(y9vad9)
        chat.addMember(y9kap)
        assertTrue(chat.isMember(y9san9))
        assertTrue(chat.isMember(y9vad9))
        assertTrue(chat.isMember(y9kap))
    }
}
```

<- Очень конкретный тест

<-

Существующая проблема юнит-тестирования

```
class ExampleTest {
    @Test
    fun `you can add many chat members`() {
        val chat = Chat.create("Test Chat")
        val y9san9 = User.create("y9san9@gmail.com")
        val y9vad9 = User.create("y9vad9@gmail.com")
        val y9kap = User.create("y9kap@gmail.com")
        chat.addMember(y9san9)
        chat.addMember(y9vad9)
        chat.addMember(y9kap)
        assertTrue(chat.isMember(y9san9))
        assertTrue(chat.isMember(y9vad9))
        assertTrue(chat.isMember(y9kap))
    }
}
```

<- Очень конкретный тест

Конкретные тесты не выполняют своих обещаний

Существующая проблема юнит-тестирования

```
class ExampleTest {
    @Test
    fun `you can add y9san9 y9vad9 y9kap to chat named test chat`() {
        val chat = Chat.create("Test Chat")
        val y9san9 = User.create("y9san9@gmail.com")
        val y9vad9 = User.create("y9vad9@gmail.com")
        val y9kap = User.create("y9kap@gmail.com")
        chat.addMember(y9san9)
        chat.addMember(y9vad9)
        chat.addMember(y9kap)
        assertTrue(chat.isMember(y9san9))
        assertTrue(chat.isMember(y9vad9))
        assertTrue(chat.isMember(y9kap))
    }
}
```

Конкретные тесты не выполняют своих обещаний


Мы можем улучшить тесты
Нужно всего лишь...
Писать больше тестов!
Писать больше тестов!

Конкретные тесты -> Конкретные выводы

```
class ExampleTest {
    @Test
    fun `you can add many chat members`() {
        val chat = Chat.create("Test Chat")
        val y9san9 = User.create("y9san9@gmail.com")
        val y9vad9 = User.create("y9vad9@gmail.com")
        val y9kap = User.create("y9kap@gmail.com")
        chat.addMember(y9san9)
        chat.addMember(y9vad9)
        chat.addMember(y9kap)
        assertTrue(chat.isMember(y9san9))
        assertTrue(chat.isMember(y9vad9))
        assertTrue(chat.isMember(y9kap))
    }
}
```

Общие тесты -> Общие выводы

```
class ExampleTest {
    @Test
    fun `you can add many chat members`() {
        val chat = chat()
        val users = users()
        for (user in users) {
            chat.addMember(user)
        }
        for (user in users) {
            assertTrue(chat.isMember(user))
        }
    }
}
```

Общие тесты -> Общие выводы

```
class ExampleTest {
    @Test
    fun `you can add many chat members`() {
        val chat = chat()
        val users = users()
        for (user in users) {
            chat.addMember(user)
        }
        for (user in users) {
            assertTrue(chat.isMember(user))
        }
    }
}
```

kotlinx.fuzz

Работа с библиотекой kotlinx.fuzz

```
class ExampleTest {
    @Test
    fun `you can add many chat members`() {
        val chat = chat()
        val users = users()
        for (user in users) {
            chat.addMember(user)
        }
        for (user in users) {
            assertTrue(chat.isMember(user))
        }
    }
}
```

Работа с библиотекой kotlinx.fuzz

```
class ExampleTest {
    @KFuzzTest
    fun `you can add many chat members`() {
        val chat = chat()
        val users = users()
        for (user in users) {
            chat.addMember(user)
        }
        for (user in users) {
            assertTrue(chat.isMember(user))
        }
    }
}
```

Работа с библиотекой kotlinx.fuzz

```
class ExampleTest {
    @KFuzzTest
    fun `you can add many chat members`(fuzzer: KFuzzer) {
        val chat = chat()
        val users = users()
        for (user in users) {
            chat.addMember(user)
        }
        for (user in users) {
            assertTrue(chat.isMember(user))
        }
    }
}
```

Работа с библиотекой kotlinx.fuzz

```
class ExampleTest {
    @KFuzzTest
    fun `you can add many chat members`(fuzzer: KFuzzer) {
        val chat = chat()
        val users = users()
        for (user in users) {
            chat.addMember(user)
        }
        for (user in users) {
            assertTrue(chat.isMember(user))
        }
    }
}
```

Что такое KFuzzer?

Работа с библиотекой kotlinx.fuzz

```kotlin
interface KFuzzer {
    fun boolean(): Boolean
    fun byte(range: IntRange): Byte
    fun short(range: IntRange): Short
    fun int(range: IntRange): Int
    fun long(range: LongRange): Long
    fun float(range: FloatRange): Float
    fun double(range: DoubleRange): Double
    fun char(range: CharRange): Char
    fun string(maxLength: Int): String
}
```

Работа с библиотекой kotlinx.fuzz

```
interface KFuzzer {
    fun boolean(): Boolean
    fun byte(range: IntRange): Byte
    fun short(range: IntRange): Short
    fun int(range: IntRange): Int
    fun long(range: LongRange): Long
    fun float(range: FloatRange): Float
    fun double(range: DoubleRange): Double
    fun char(range: CharRange): Char
    fun string(maxLength: Int): String
}
```

kotlin.Random для тестирования

Работа с библиотекой kotlinx.fuzz

```
class ExampleTest {
    @KFuzzTest
    fun `you can add many chat members`(fuzzer: KFuzzer) {
        val chat = chat()
        val users = users()
        for (user in users) {
            chat.addMember(user)
        }
        for (user in users) {
           assertTrue(chat.isMember(user))
        }
    }
}
```

Работа с библиотекой kotlinx.fuzz

```
val title =
```

Работа с библиотекой kotlinx.fuzz

```
val title = fuzzer.string()
```

Работа с библиотекой kotlinx.fuzz

```
fun KFuzzer.chat(): Chat {
    val title = this.string()
    return Chat.create(title)
}
```

Работа с библиотекой kotlinx.fuzz

```
fun KFuzzer.chat(): Chat {
    val title = this.string()
    return Chat.create(title)
}

fun KFuzzer.users(): List<User> {
    return List(int()) {
        val email = string()
        User.create(email)
    }
}
```

Работа с библиотекой kotlinx.fuzz

```kotlin
fun KFuzzer.chat(): Chat {
    val title = this.string()
    return Chat.create(title)
}

fun KFuzzer.users(): List<User> {
    return List(int()) {
        val email = string()
        User.create(email)
    }
}
```

Можно создавать пользовательские расширения

Работа с библиотекой kotlinx.fuzz

```
class ExampleTest {
    @KFuzzTest
    fun `you can add many chat members`(fuzzer: KFuzzer) {
        val chat = chat()
        val users = users()
        for (user in users) {
            chat.addMember(user)
        }
        for (user in users) {
            assertTrue(chat.isMember(user))
        }
    }
}
```

Работа с библиотекой kotlinx.fuzz

```
class ExampleTest {
    @KFuzzTest
    fun `you can add many chat members`(fuzzer: KFuzzer) {
        val chat = fuzzer.chat()
        val users = fuzzer.users()
        for (user in users) {
            chat.addMember(user)
        }
        for (user in users) {
            assertTrue(chat.isMember(user))
        }
    }
}
```

Работа с библиотекой kotlinx.fuzz

Как подключить?

Работа с библиотекой kotlinx.fuzz

```
plugins {
    id(...)
}

fuzzConfig {
    keepGoing = 3
    maxFuzzTimePerTarget = 10.seconds
    coverage {
        reportTypes = setOf(
            CoverageReportType.HTML,
            CoverageReportType.CSV
        )
    }
}
```

Как подключить?

Работа с библиотекой kotlinx.fuzz

```
example % ./gradlew fuzz

> Task :fuzz
```

Как запустить?

Работа с библиотекой kotlinx.fuzz

example % ./gradlew fuzz

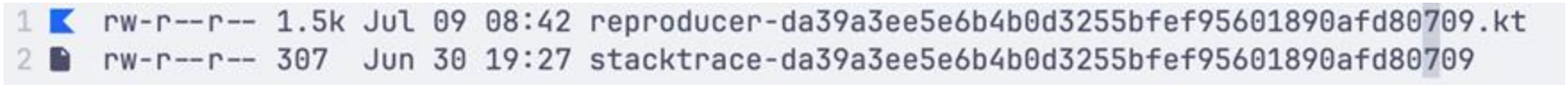
> Task :fuzz

Example > public final void Example.you can add many chat members(kotlinx.fuzz.KFuzzer) FAILED

Как запустить?

Работа с библиотекой kotlinx.fuzz

build/fuzz/reproducers



Что пошло не так?

Работа с библиотекой kotlinx.fuzz

build/fuzz/reproducers/stacktrace-da39a3ee5e6b4b0d3255bfef95601890afd80709

java.lang.IllegalStateException: Chat name can not be longer than a 100 chars

at Example.you can add many chat members(Example.kt:42)

at Chat.create(Chat.kt:17)

Что пошло не так?

Работа с библиотекой kotlinx.fuzz

build/fuzz/reproducers/reproducer-da39a3ee5e6b4b0d3255bfef95601890afd80709

```
class ExampleTest_da39a3ee5e6b4b0d3255bfef95601890afd80709 {
    @Test
    fun `you can add many chat members`() {
        val fuzzer = ValueReproducer("L8{I0'£Kl-;_V"xj=Z'%5o1gX5I(02FQ.l...")
        val chat = fuzzer.chat()
        val users = fuzzer.users()
        for (user in users) {
            chat.addMember(user)
        }
        for (user in users) {
            assertTrue(chat.isMember(user))
        }
    }
}
```

Что пошло не так?

Работа с библиотекой kotlinx.fuzz

build/fuzz/reproducers/stacktrace-da39a3ee5e6b4b0d3255bfef95601890afd80709

java.lang.IllegalStateException: Chat name can not be longer than a 100 chars

at Example.you can add many chat members(Example.kt:42)

at Chat.create(Chat.kt:17)

Что пошло не так?

Работа с библиотекой kotlinx.fuzz

build/fuzz/reproducers/stacktrace-da39a3ee5e6b4b0d3255bfef95601890afd80709

java.lang.IllegalStateException: Chat name can not be longer than a 100 chars

at Example.you can add many chat members(Example.kt:42)

at Chat.create(Chat.kt:17)

```
fun KFuzzer.chat(): Chat {
    val title = this.string()
    return Chat.create(title)
}
```

Что пошло не так?

Работа с библиотекой kotlinx.fuzz

build/fuzz/reproducers/stacktrace-da39a3ee5e6b4b0d3255bfef95601890afd80709

java.lang.IllegalStateException: Chat name can not be longer than a 100 chars

at Example.you can add many chat members(Example.kt:42)

at Chat.create(Chat.kt:17)

```
fun KFuzzer.chat(): Chat {
    val title = this.string(maxLength = 100)
    return Chat.create(title)
}
```

Можно указать границу с длиной строки

Работа с библиотекой kotlinx.fuzz

example % ./gradlew fuzz

> Task :fuzz

Работа с библиотекой kotlinx.fuzz

example % ./gradlew fuzz

> Task :fuzz

Example > public final void Example.you can add many chat members(kotlinx.fuzz.KFuzzer) FAILED

Опять ошибка – слишком много участников

Работа с библиотекой kotlinx.fuzz

example % ./gradlew fuzz

> Task :fuzz

Example > public final void Example.you can add many chat members(kotlinx.fuzz.KFuzzer) FAILED

java.lang.IllegalStateException: Max amount of members is reached: 100

at Example.you can add many chat members(Example.kt:42)

at Chat.addMember(Chat.kt:24)

Работа с библиотекой kotlinx.fuzz

```
fun KFuzzer.users(): List<User> {
    return List(int()) {
        val email = string()
        User.create(email)
    }
}
```

Работа с библиотекой kotlinx.fuzz

```
fun KFuzzer.users(): List<User> {
    return List(int()) {
        val email = string()
        User.create(email)
    }
}
```

java.lang.IllegalStateException: Max amount of members is reached: 100

at Example.you can add many chat members(Example.kt:42)

at Chat.addMember(Chat.kt:24)

Работа с библиотекой kotlinx.fuzz

```
fun KFuzzer.users(): List<User> {
    return List(int(0..100)) {
        val email = string()
        User.create(email)
    }
}
```

java.lang.IllegalStateException: Max amount of members is reached: 100

at Example.you can add many chat members(Example.kt:42)

at Chat.addMember(Chat.kt:24)

Работа с библиотекой kotlinx.fuzz

example % ./gradlew fuzz

> Task :fuzz

Работа с библиотекой kotlinx.fuzz

example % ./gradlew fuzz

> Task :fuzz

Example > public final void Example.you can add many chat members(kotlinx.fuzz.KFuzzer) FAILED

Работа с библиотекой kotlinx.fuzz

example % ./gradlew fuzz

> Task :fuzz

Example > public final void Example.you can add many chat members(kotlinx.fuzz.KFuzzer) FAILED

java.lang.IllegalStateException: Invalid user email

at Example.you can add many chat members(Example.kt:42)

at User.create(User.kt:24)

Работа с библиотекой kotlinx.fuzz

java.lang.IllegalStateException: Invalid user email

at Example.you can add many chat members(Example.kt:42)

at User.create(User.kt:24)

Работа с библиотекой kotlinx.fuzz

```
fun KFuzzer.users(): List<User> {
    return List(int(0..100)) {
        val email = string()
        User.create(email)
    }
}
```

java.lang.IllegalStateException: Invalid user email

at Example.you can add many chat members(Example.kt:42)

at User.create(User.kt:24)

Работа с библиотекой kotlinx.fuzz

```
fun KFuzzer.users(): List<User> {
    return List(int(0..100)) {
        val email = buildString {
            append(string(maxLength = 20))
            append('@')
            append(string(maxLength = 20))
            append('.')
            append(string(maxLength = 20))
        }
        User.create(email)
    }
}
```

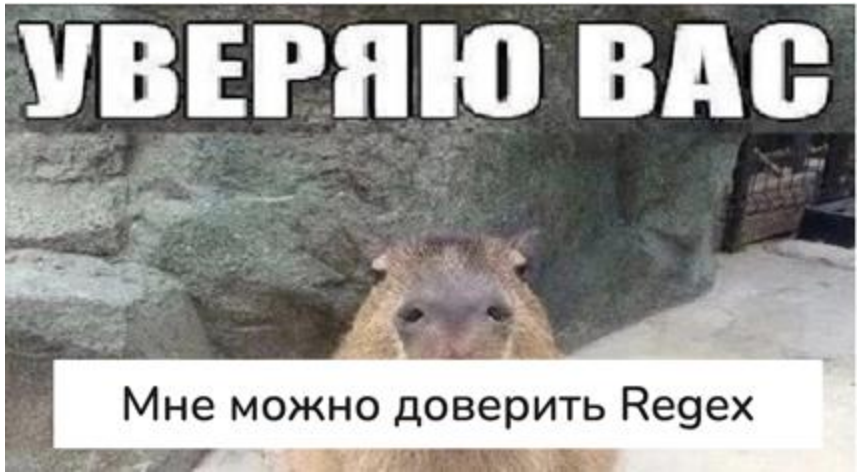
java.lang.IllegalStateException: Invalid user email

at Example.you can add many chat members(Example.kt:42)

at User.create(User.kt:24)

Работа с библиотекой kotlinx.fuzz

```
fun KFuzzer.users(): List<User> {
    return List(int(0..100)) {
        val email = buildString {
            append(string(maxLength = 20))
            append('@')
            append(string(maxLength = 20))
            append('.')
            append(string(maxLength = 20))
        }
        User.create(email)
    }
}
```



java.lang.IllegalStateException: Invalid user email

at Example.you can add many chat members(Example.kt:42)

at User.create(User.kt:24)

Работа с библиотекой kotlinx.fuzz

```
fun KFuzzer.users(): List<User> {
    return List(int(0..100)) {
        val email = string(Regex("[A-Za-z]+@[A-Za-z]+\.[A-Za-z]+"))
        User.create(email)
    }
}
```

java.lang.IllegalStateException: Invalid user email

at Example.you can add many chat members(Example.kt:42)

at User.create(User.kt:24)

Работа с библиотекой kotlinx.fuzz

example % ./gradlew fuzz

> Task :fuzz

Работа с библиотекой kotlinx.fuzz

example % ./gradlew fuzz

> Task :fuzz

Example > public final void Example.you can add many chat members(kotlinx.fuzz.KFuzzer) PASSED

BUILD SUCCESSFUL

3 actionable tasks: 2 executed, 1 up-to-date

А нельзя красивее?

```
fun KFuzzer.chat(): Chat {
    …
}

fun KFuzzer.users(): List<User> {
    …
}
```

А нельзя красивее?

```
fun KFuzzer.chat(): Chat {
    …
}

fun KFuzzer.users(): List<User> {
    …
}
```

en.wikipedia.org/wiki/The_Thinker

А нельзя красивее?

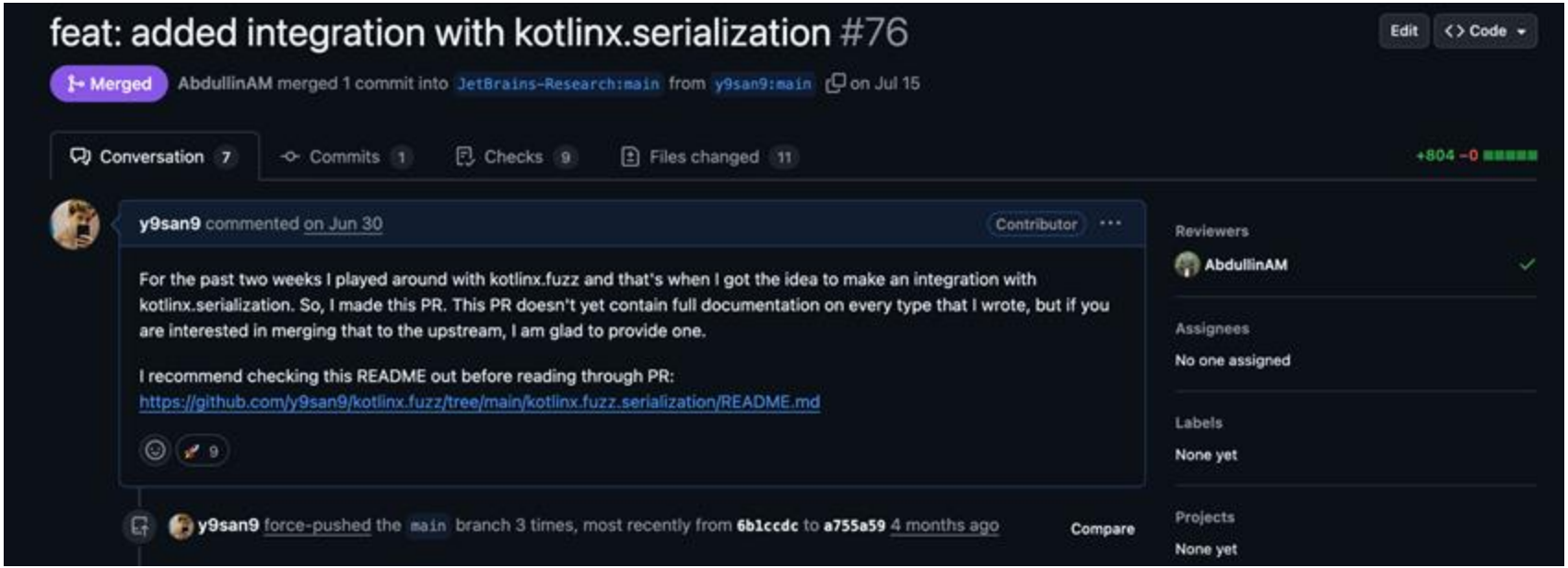
```
fun KFuzzer.chat(): Chat {
    …
}

fun KFuzzer.users(): List<User> {
    …
}
```

en.wikipedia.org/wiki/The_Thinker

```
@Serializable
data class ChatTest(
    val title: String,
    val users: List<User>,
) {
    @Serializable
    data class User(
        val email: String,
    )
}
```

Можно!


feat: added integration with kotlinx.serialization #76
Edit
Code
Merged
AbdullinAM merged 1 commit into JetBrains-Research:main from y9san9:main on Jul 15
Conversation 7
Commits 1
Checks 9
Files changed 11
+804 −0
y9san9 commented on Jun 30
Contributor
For the past two weeks I played around with kotlinx.fuzz and that's when I got the idea to make an integration with kotlinx.serialization. So, I made this PR. This PR doesn't yet contain full documentation on every type that I wrote, but if you are interested in merging that to the upstream, I am glad to provide one.
I recommend checking this README out before reading through PR:
https://github.com/y9san9/kotlinx.fuzz/tree/main/kotlinx.fuzz.serialization/README.md
9
y9san9 force-pushed the main branch 3 times, most recently from 6b1ccdc to a755a59 4 months ago
Compare
Reviewers
AbdullinAM
Assignees
No one assigned
Labels
None yet
Projects
None yet

Натягиваем сову на глобус

```
interface KFuzzer {
    fun boolean(): Boolean
    fun byte(range: IntRange): Byte
    fun short(range: IntRange): Short
    fun int(range: IntRange): Int
    fun long(range: LongRange): Long
    fun float(range: FloatRange): Float
    fun double(range: DoubleRange): Double
    fun char(range: CharRange): Char
    fun string(maxLength: Int): String
}
```

Натягиваем сову на глобус... или нет?

```
interface Decoder {
    fun decodeBoolean(): Boolean
    fun decodeByte(): Byte
    fun decodeShort(): Short
    fun decodeChar(): Char
    fun decodeInt(): Int
    fun decodeLong(): Long
    fun decodeFloat(): Float
    fun decodeDouble(): Double
    fun decodeString(): String
}
```

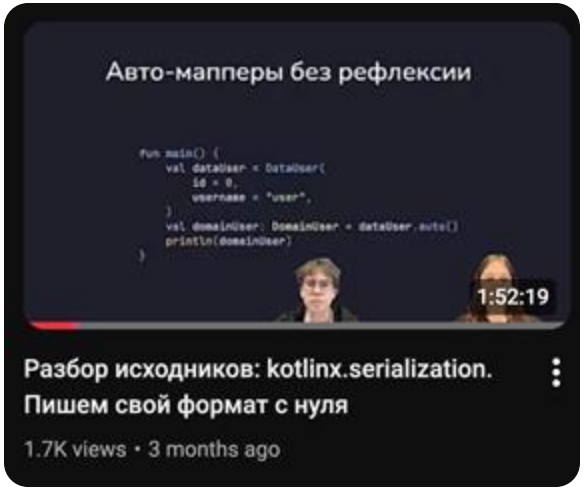
Натягиваем сову на глобус... или нет?

```
interface Decoder {
    fun decodeBoolean(): Boolean
    fun decodeByte(): Byte
    fun decodeShort(): Short
    fun decodeChar(): Char
    fun decodeInt(): Int
    fun decodeLong(): Long
    fun decodeFloat(): Float
    fun decodeDouble(): Double
    fun decodeString(): String
}
```

Написать свой формат для kotlinx.serialization может каждый

Натягиваем сову на глобус... или нет?

```
interface Decoder {
    fun decodeBoolean(): Boolean
    fun decodeByte(): Byte
    fun decodeShort(): Short
    fun decodeChar(): Char
    fun decodeInt(): Int
    fun decodeLong(): Long
    fun decodeFloat(): Float
    fun decodeDouble(): Double
    fun decodeString(): String
}
```



youtu.be/FlBkE4V7VGI

Написать свой формат для kotlinx.serialization может каждый

Теперь использовать kotlinx.fuzz просто

```
@Serializable
data class ChatTest(
    val title: String,
    val users: List<User>,
) {
    @Serializable
    data class User(
        val email: String,
    )
}
```

Теперь использовать kotlinx.fuzz просто

```
class ExampleTest {
    @KFuzzTest
    fun `you can add many chat members`(fuzzer: KFuzzer) {
        val test = fuzzer.serialization.fuzz<ChatTest>()
        for (user in users) {
            chat.addMember(user)
        }
        for (user in users) {
            assertTrue(chat.isMember(user))
        }
    }
}
```

Теперь использовать kotlinx.fuzz просто

```
class ExampleTest {
    @KFuzzTest
    fun `you can add many chat members`(fuzzer: KFuzzer) {
        val test = fuzzer.serialization.fuzz<ChatTest> {
            collectionSize { fuzzer -> fuzzer.int(0..100) }
        }
        for (user in users) {
            chat.addMember(user)
        }
        for (user in users) {
            assertTrue(chat.isMember(user))
        }
    }
}
```

Мы получили то, что хотели

Общие тесты -> Общие выводы

Общие тесты -> Общие выводы

```
class ExampleTest {
    @KFuzzTest
    fun `you can add many chat members`(fuzzer: KFuzzer) {
        val chat = fuzzer.chat()
        val users = fuzzer.users()
        for (user in users) {
            chat.addMember(user)
        }
        for (user in users) {
            assertTrue(chat.isMember(user))
        }
    }
}
```

Общие тесты -> Общие выводы

```
class ExampleTest {
    @KFuzzTest
    fun `you can up to 100 members to chat`(fuzzer: KFuzzer) {
        val chat = fuzzer.chat()
        val users = fuzzer.users()
        for (user in users) {
            chat.addMember(user)
        }
        for (user in users) {
            assertTrue(chat.isMember(user))
        }
    }
}
```

Конкретные тесты не выполняют своих обещаний

Конкретные тесты не выполняют своих обещаний

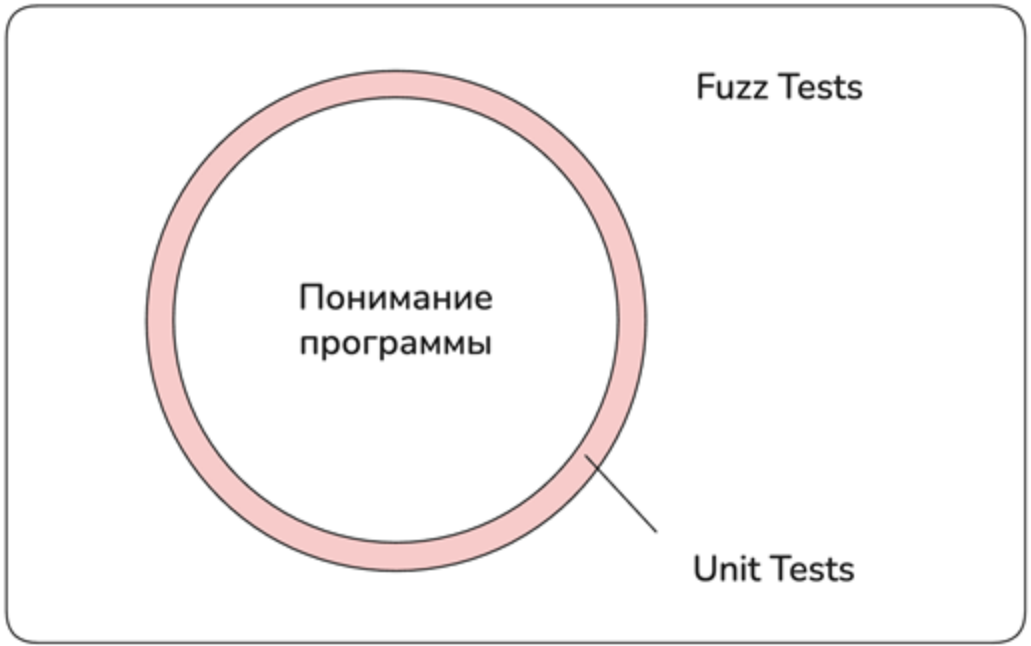
Фазз-тесты показывают, что для определённого класса данных

выполняются общие правила.

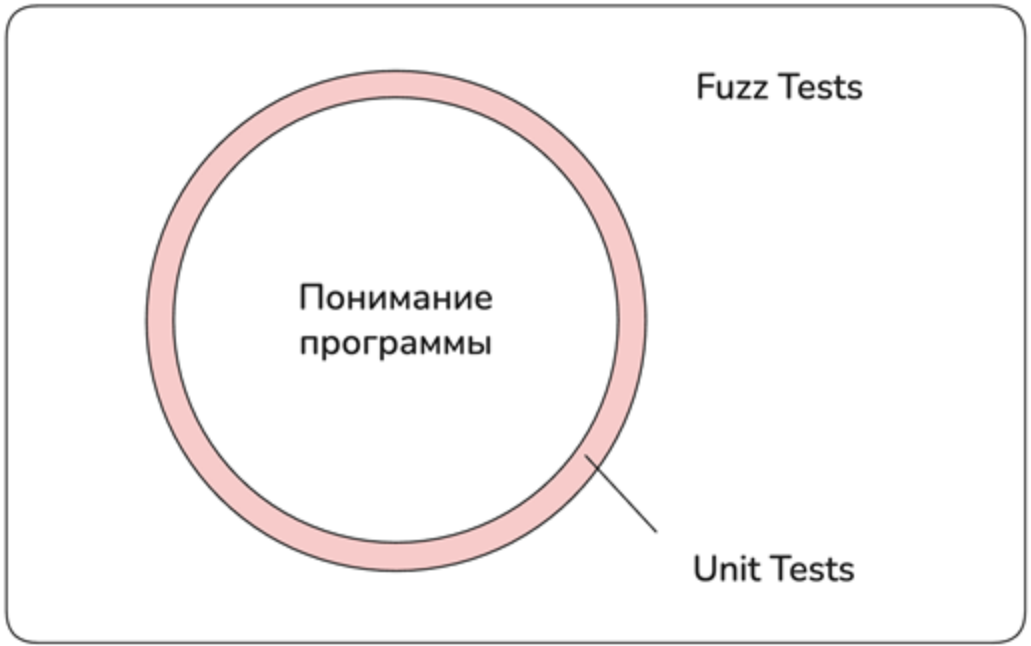
Немного философии

Немного философии

Немного философии



Почему тогда мы не используем Фазз-тестирование везде?



Фазз-тестирование имеет конкретное применение

- Нахождение скрытых ошибок

Фазз-тестирование имеет конкретное применение

- Нахождение скрытых ошибок
- Нахождение скрытых уязвимостей

Google OSS-Fuzz

Google OSS-Fuzz

- 50_000+ ошибок

Google OSS-Fuzz

- 50_000+ ошибок
- 10_000+ уязвимостей

Google OSS-Fuzz

- 50_000+ ошибок
- 10_000+ уязвимостей

500+ Проектов, включая:



Это не просто Random

Это не просто Random

- Выборки

Это не просто Random

- Выборки (SQL Injections, XSS Injections, Regex Injections, Os Command Injections, etc.)

Это не просто Random

- Выборки (SQL Injections, XSS Injections, Regex Injections, Os Command Injections, etc.)
- Упавшие тесты сохраняются и запускаются первыми

Это не просто Random

- Выборки (SQL Injections, XSS Injections, Regex Injections, Os Command Injections, etc.)
- Упавшие тесты сохраняются и запускаются первыми
- Фаззинг, основанный на покрытии кода

Фаззинг, основанный на покрытии кода

```
fun coverageTest(int: Int) {
    if (int % 2 == 0) {
        // если чётное
    } else {
        // если нечётное
        if (int < 0) {
            // если отрицательное
        } else {
            // если неотрицательное
        }
    }
}
```

Фаззинг, основанный на покрытии кода

```
fun coverageTest(int: Int = 0) {
    if (int % 2 == 0) {
        // если чётное
    } else {
        // если нечётное
        if (int < 0) {
            // если отрицательное
        } else {
            // если неотрицательное
        }
    }
}
```

Фаззинг, основанный на покрытии кода

```
fun coverageTest(int: Int = 1) {
    if (int % 2 == 0) {
        // если чётное
    } else {
        // если нечётное
        if (int < 0) {
            // если отрицательное
        } else {
            // если неотрицательное
        }
    }
}
```

Фаззинг, основанный на покрытии кода

```
fun coverageTest(int: Int = -1) {
    if (int % 2 == 0) {
        // если чётное
    } else {
        // если нечётное
        if (int < 0) {
            // если отрицательное
        } else {
            // если неотрицательное
        }
    }
}
```

Список Трофеев

22 ошибки найдено в kotlinx-библиотеках

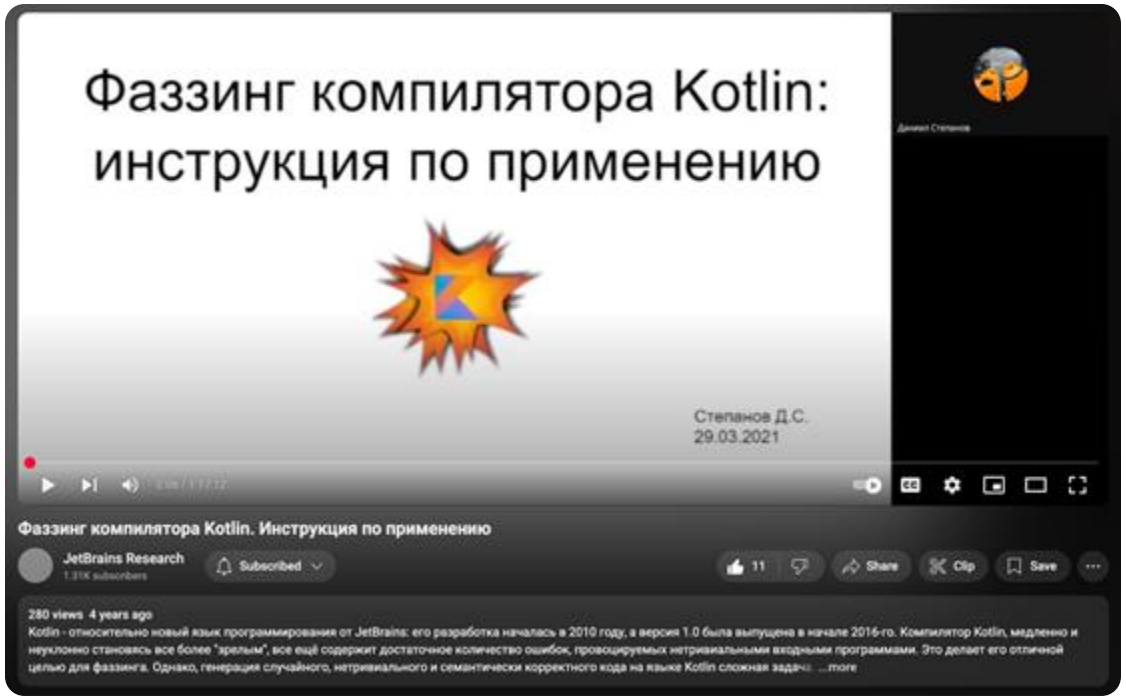
- `kotlinx.serialization`
  - JSON
    - Invalid serialization due to unhandled conflict of `classDesctiptor` and `JsonNames`
    - Undocumented behaviour of `decodeToSequence` with `allowTrailingComma` option
    - Wrong error message in combination of `decodeToSequence` and `allowTrailingComma=false`
  - CBOR
    - Unhandled `IllegalStateException` when `decodeFromByteArray` reaches the end of the byte array
    - Unhandled `NegativeArraySizeException` when `decodeFromByteArray` encounters invalid data
    - Infinite recursion and `StackOverflowError` in `decodeFromByteArray` due to not handling reaching the end of the buffer
    - Unhandled `ArrayIndexOutOfBounds` when trying to skip an unknown element in the byte array
  - Properties
    - Undocumented behaviour with empty primitive arrays: they are not present in the resulting string
  - ProtoBuf
    - Some non-empty inputs are parsed as empty message, which breaks `a == deserialize(serialize(a))` invariant
    - Equal messages are encoded differently depending on the generic type
    - `a == deserialize(serialize(a))` invariant breaks for some not empty messages
    - Exception when trying to serialize a field with default value `null`

Список Трофеев

22 ошибки найдено в kotlinx-библиотеках

- `kotlinx.collections.immutable`
  - Equal PersistentOrderedSets are not equal
- `kotlinx.cli`
  - Incorrect handling of "--" in GNU mode
- `kotlinx-datetime`
  - A number of bugs and inconsistencies with Java
    - `kotlinx.datetime.LocalDate.parse(s)` vs `LocalDate.Formats.ISO.parse(s)` have different behaviour with leading zeroes
    - `DatePeriod.parse` and `DateTimePeriod.parse` parsing "P" behaviour is inconsistent with ISO 8601
    - `kotlinx.datetime.DatePeriod` is always normalized unlike `java.time.Duration` (Intentional)
    - `kotlin.time.Duration.parseIsoString` with too many digits returns an infinite duration
    - `kotlin.time.Duration.parse("PT+-2H")` successfully parses -2 hours: not valid ISO 8601
    - Different periods between two dates in Java and Kotlin: bug in Java
- `kotlinx.io`
  - {Source,Buffer}.indexOf(ByteString, Int) should return start index for empty ByteString
  - Buffer.indexOf(ByteString) accepts negative indices

Kotlin Compiler



Бонус...


litlrekamor:
y9kapnowar:
y9kapnowar: Так
y9kapnowar: Эхо
litlrekamor: а твич вроде банит, если кинуть репорт, давайте добавим ютуб коммы
y9kapnowar: XBXbIXbIXXB
litlrekamor: ура нету эхо
zer: KFuzzer) {
r.int() % 2 == 0) {
zer.int() % 3 = 2) {
uzzer.in % 31 == 11) {
ow Ille gumentException
0:01 / 1:45:55 • Читаем бло...
kotlinx.fuzz: новая библиотека для тестирования от JetBrains
Kotlin Meta
47 subscribers
Analytics
Edit video
3
Share

Бонус... Можно ли применять AI?

Бонус... Можно ли применять AI?

Бонус... Можно ли применять AI?

Оказывается, можно!

Бонус... Можно ли применять AI?

Оказывается, можно!

Large Language Model assisted Hybrid Fuzzing

Ruijie Meng
National University of Singapore Singapore
ruijie.meng@u.nus.edu

Gregory J. Duck
National University of Singapore Singapore
gregory@comp.nus.edu.sg

Abhik Roychoudhury
National University of Singapore Singapore
abhik@comp.nus.edu.sg

Abstract.

Greybox fuzzing is one of the most popular methods for detecting software vulnerabilities, which conducts a biased random search within the program input space. To enhance its effectiveness in achieving deep coverage of program behaviors, greybox fuzzing is often combined with concolic execution, which performs a path-sensitive search over the domain of program inputs. In hybrid fuzzing, conventional greybox fuzzing is followed by concolic execution in an iterative loop,

Фаззинг, основанный на покрытии кода

```
fun coverageTest(int: Int) {
    if (int % 2 == 0) {
        // if even
    } else {
        // if odd
        if (int < 0) {
            // if negative
        } else {
            // if positive
        }
    }
}
```

Конец

@y9san9 - main contact
y9san9@gmail.com
@y9san9
@y9san9
@y9san9
@y9san9
@y9san9

@kotlinmeta