Kotlin meets Gadsu

Christoph Pickl

Kotlin Vienna Meetup – 2017-01-31

Agenda

1 **Introduction to Gadsu**

2 Kotlin in the wild

3 Code “*Schmankerln*”

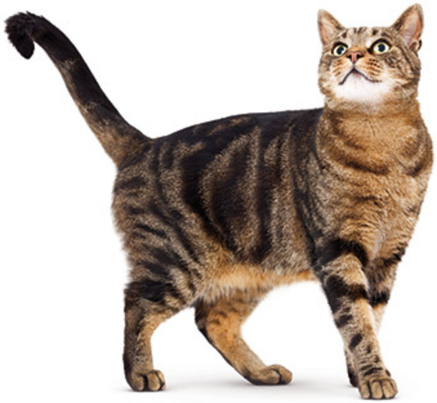
4 Lessons Learned

Agenda

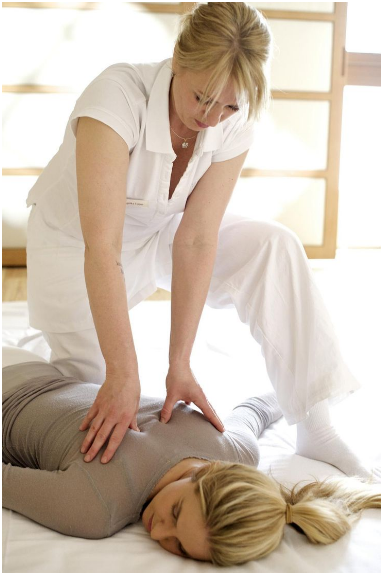
Agenda

Agenda

Introduction to Gadsu



This is a cat.



This is Shiatsu . . .

. . . so is this.

Gadsu is . . .

Gadsu is . . .

Gadse

Gadsu is . . .

Gadse

+ Shiat**su**

Gadsu is . . .

Gadse

+ Shiat**su**

= **Gadsu**

Shiatsu is . . .

. . . Japanese!

Shiatsu is . . .

. . . Japanese! Food!

Shiatsu is . . .

. . . Japanese! Food! Yam yam!

Shiatsu is . . .

- Massage, Physiotherapy, Bodywork
- Acupuncture, Meridiantherapy
- Nervous system stimulation

Shiatsu is . . .

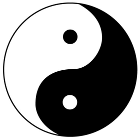
- Massage, Physiotherapy, Bodywork
- Acupuncture, Meridiantherapy
- Nervous system stimulation

- Based on the **T**raditional **C**hinese **M**edicine
  - Concept of **Qi** flowing through the body and everything
  - Body and mind seen as a unit, not separated from each other

Shiatsu is . . .

- Massage, Physiotherapy, Bodywork
- Acupuncture, Meridiantherapy
- Nervous system stimulation

- Based on the **T**raditional **C**hinese **M**edicine
  - Concept of **Qi** flowing through the body and everything
  - Body and mind seen as a unit, not separated from each other



Taiji Symbol, Theory of Yin and Yang

Gadsu can ...

Features:

- Client database
- Manage medical records
- Generate reports
- Google integration
- Auto update, auto backup

Gadsu can . . .

Features:

- Client database
- Manage medical records
- Generate reports
- Google integration
- Auto update, auto backup

Roadmap:

- Pain indicator, 5 Elements
- Statistics
- TCM intelligence
- Doodle integration
- Invoicing

Technology Stack

- Gradle

Technology Stack

- Gradle
- Swing

Technology Stack

- Gradle
- Swing
- Guice

Technology Stack

- Gradle
- Swing
- Guice
- Spring JDBC, HSQLDB, Flyway

Technology Stack

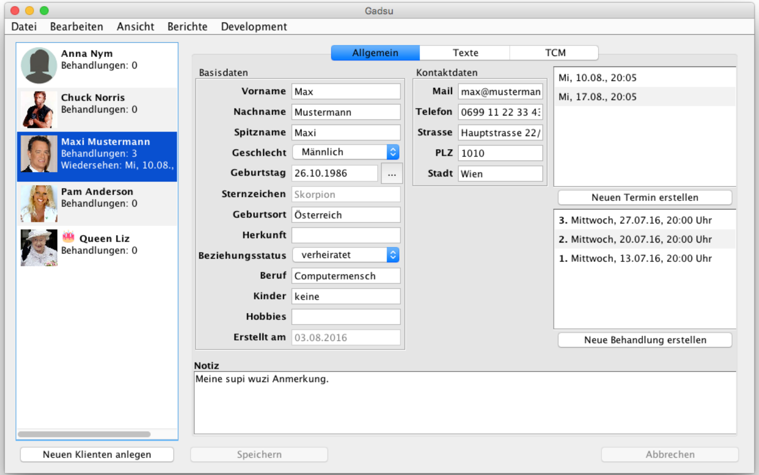
- Gradle
- Swing
- Guice
- Spring JDBC, HSQLDB, Flyway
- Jasper, Pdfbox, Freemarker

Technology Stack

- Gradle
- Swing
- Guice
- Spring JDBC, HSQLDB, Flyway
- Jasper, Pdfbox, Freemarker
- TestNG, Mockito, Hamcrest, UISpec4J

Technology Stack

- Gradle
- Swing
- Guice
- Spring JDBC, HSQLDB, Flyway
- Jasper, Pdfbox, Freemarker
- TestNG, Mockito, Hamcrest, UISpec4J
- *Initial implementation used Kotlin 0.6* ;)



Anna Nym
Behandlungen: 0

Chuck Norris
Behandlungen: 0

Maxi Mustermann
Behandlungen: 3
Wiedersehen: Mi, 10.08.,

Pam Anderson
Behandlungen: 0

Queen Liz
Behandlungen: 0

Basisdaten

Vorname	Max
Nachname	Mustermann
Spitzname	Maxi
Geschlecht	Männlich
Geburtstag	26.10.1986
Sternzeichen	Skorpion
Geburtsort	Österreich
Herkunft	
Beziehungsstatus	verheiratet
Beruf	Computermensch
Kinder	keine
Hobbies	
Erstellt am	03.08.2016

Kontaktdaten

Mail	max@musterman
Telefon	0699 11 22 33 4:
Strasse	Hauptstrasse 22/
PLZ	1010
Stadt	Wien

Mi, 10.08., 20:05
Mi, 17.08., 20:05

Neuen Termin erstellen

3. Mittwoch, 27.07.16, 20:00 Uhr
2. Mittwoch, 20.07.16, 20:00 Uhr
1. Mittwoch, 13.07.16, 20:00 Uhr

Neue Behandlung erstellen

Notiz
Meine supi wuzi Anmerkung.

Neuen Klienten anlegen
Speichern
Abbrechen

Gadsu got something like 30,000 LoC :)

Let's see some app . . .

Kotlin in the wild

build.gradle

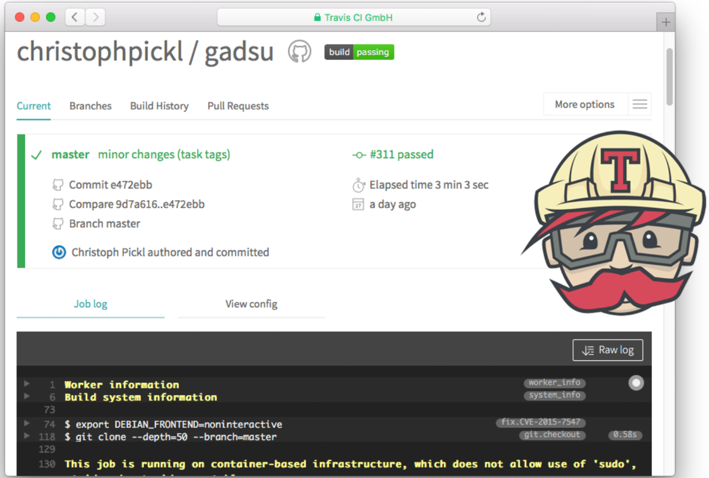
```
apply plugin: "kotlin"
```

build.gradle

```
apply plugin: "kotlin"
buildscript {
  ext.kotlin_version = '1.0.6'
  dependencies {
    classpath "org.jetbrains.kotlin:kotlin-
        gradle-plugin:$kotlin_version"
  }
}
```

build.gradle

```
apply plugin: "kotlin"
buildscript {
  ext.kotlin_version = '1.0.6'
  dependencies {
    classpath "org.jetbrains.kotlin:kotlin-
       gradle-plugin:$kotlin_version"
  }
}
dependencies {
  compile "org.jetbrains.kotlin:kotlin-
     stdlib:$kotlin_version"
  compile "org.jetbrains.kotlin:kotlin-
     reflect:$kotlin_version"
}
```



master minor changes (task tags)

#311 passed

Commit e472ebb

Compare 9d7a616..e472ebb

Branch master

Elapsed time 3 min 3 sec

a day ago

Christoph Pickl authored and committed

```
Worker information                                                   worker_info
Build system information                                             system_info

$ export DEBIAN_FRONTEND=noninteractive                             fix.CVE-2015-7547
$ git clone --depth=50 --branch=master                              git.checkout   0.58s

This job is running on container-based infrastructure, which does not allow use of 'sudo',
```

travis-ci.org

.travis.yml

```
language: kotlin
```

.travis.yml

```
language: kotlin
sudo: false
```

.travis.yml

```yaml
language: kotlin
sudo: false
jdk:
  - oraclejdk8
```

.travis.yml

```yaml
language: kotlin
sudo: false
jdk:
  - oraclejdk8
before_install:
  - "chmod +x gradlew"
```

.travis.yml

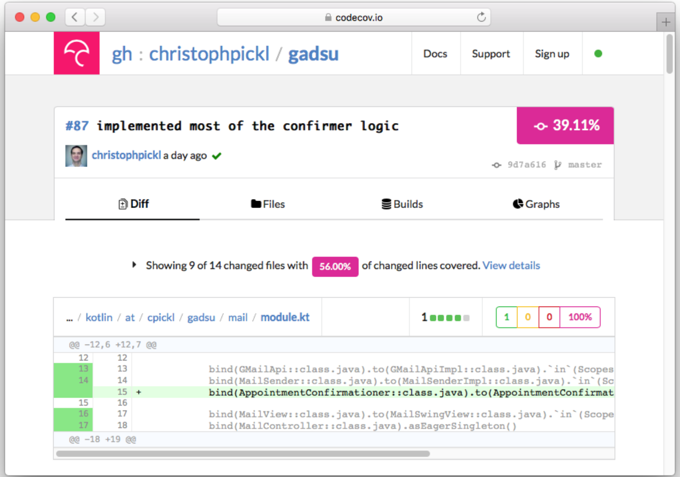
```
language: kotlin
sudo: false
jdk:
  - oraclejdk8
before_install:
  - "chmod +x gradlew"
  - "export DISPLAY=:99.0"
  - "sh -e /etc/init.d/xvfb start"
```

.travis.yml

```
language: kotlin
sudo: false
jdk:
  - oraclejdk8
before_install:
  - "chmod +x gradlew"
  - "export DISPLAY=:99.0"
  - "sh -e /etc/init.d/xvfb start"
script:
  - "./gradlew test ..."
```

.travis.yml

```yaml
language: kotlin
sudo: false
jdk:
  - oraclejdk8
before_install:
  - "chmod +x gradlew"
  - "export DISPLAY=:99.0"
  - "sh -e /etc/init.d/xvfb start"
script:
  - "./gradlew test ..."
notifications:
  email:
    - "MLtravis@gadsu.com"
```



#87 implemented most of the confirmer logic 39.11%

christophpickl a day ago ✓ 9d7a616 master

Showing 9 of 14 changed files with 56.00% of changed lines covered. View details

... / kotlin / at / cpickl / gadsu / mail / module.kt 1 1 0 0 100%

```
@@ -12,6 +12,7 @@
12 12
13 13        bind(GMailApi::class.java).to(GMailApiImpl::class.java).`in`(Scopes
14 14        bind(MailSender::class.java).to(MailSenderImpl::class.java).`in`(Sc
   15  +    bind(AppointmentConfirmationer::class.java).to(AppointmentConfirmat
15 16
16 17        bind(MailView::class.java).to(MailSwingView::class.java).`in`(Scope
17 18        bind(MailController::class.java).asEagerSingleton()
@@ -18 +19 @@
```

codecov.io

Codecov

Gradle Configuration:

```
plugins {
  id 'jacoco'
  id 'com.github.kt3k.coveralls'
}
```

Codecov

Gradle Configuration:

```
plugins {
  id 'jacoco'
  id 'com.github.kt3k.coveralls'
}
jacocoTestReport {
  reports {
    xml.enabled = true
}}
```

Codecov

Gradle Configuration:

```
plugins {
  id 'jacoco'
  id 'com.github.kt3k.coveralls'
}
jacocoTestReport {
  reports {
    xml.enabled = true
}}
```

Travis Configuration:

```
script:
  - "./gradlew ... jacocoTestReport ..."
```

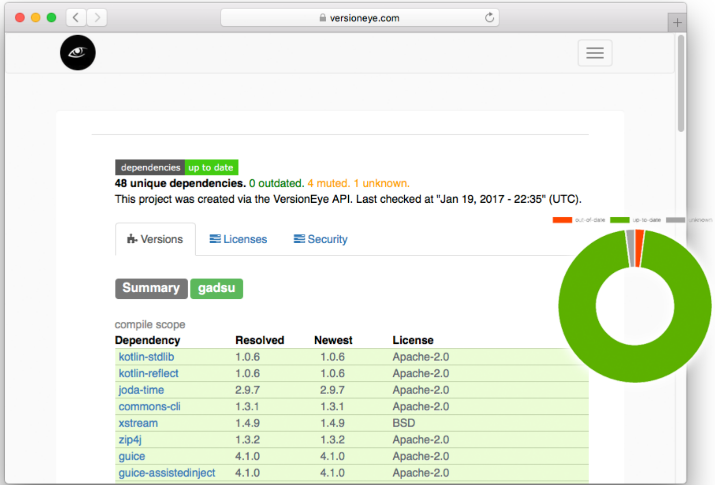
Codecov

Gradle Configuration:

```
plugins {
  id 'jacoco'
  id 'com.github.kt3k.coveralls'
}
jacocoTestReport {
  reports {
    xml.enabled = true
}}
```

Travis Configuration:

```
script:
  - "./gradlew ... jacocoTestReport ..."
after_success:
  - bash <(curl -s https://codecov.io/bash)
```



48 unique dependencies. 0 outdated. 4 muted. 1 unknown.
This project was created via the VersionEye API. Last checked at "Jan 19, 2017 - 22:35" (UTC).

compile scope

Dependency	Resolved	Newest	License
kotlin-stdlib	1.0.6	1.0.6	Apache-2.0
kotlin-reflect	1.0.6	1.0.6	Apache-2.0
joda-time	2.9.7	2.9.7	Apache-2.0
commons-cli	1.3.1	1.3.1	Apache-2.0
xstream	1.4.9	1.4.9	BSD
zip4j	1.3.2	1.3.2	Apache-2.0
guice	4.1.0	4.1.0	Apache-2.0
guice-assistedinject	4.1.0	4.1.0	Apache-2.0

versioneye.com

VersionEye

Gradle Configuration:

```
plugins {
  id "org.standardout.versioneye"
    version "1.4.0"
}
```

VersionEye

Gradle Configuration:

```
plugins {
  id "org.standardout.versioneye"
    version "1.4.0"
}
```

Gradle Properties:

```
versioneye.projectid=572880644a0f...00b78206
```

VersionEye

Gradle Configuration:

```
plugins {
  id "org.standardout.versioneye"
    version "1.4.0"
}
```

Gradle Properties:

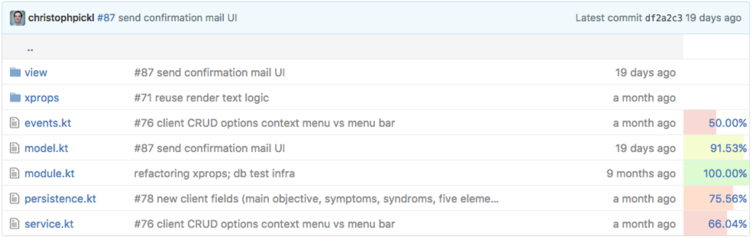
```
versioneye.projectid=572880644a0f...00b78206
```

Travis Configuration:

```
script:
  - "./gradlew ... versioneye-update ..."
```

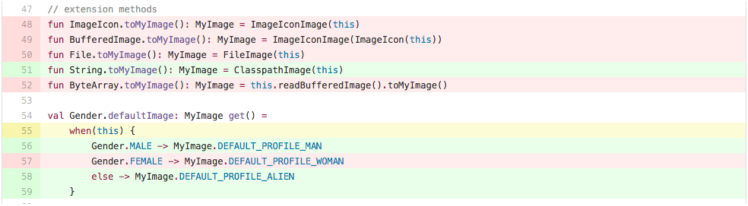
VersionEye GitHub Integration

Display coverage data via the Codecov Browser Extension:



christophpickl #87 send confirmation mail UI — Latest commit df2a2c3 19 days ago

..			
view	#87 send confirmation mail UI	19 days ago	
xprops	#71 reuse render text logic	a month ago	
events.kt	#76 client CRUD options context menu vs menu bar	a month ago	50.00%
model.kt	#87 send confirmation mail UI	19 days ago	91.53%
module.kt	refactoring xprops; db test infra	9 months ago	100.00%
persistence.kt	#78 new client fields (main objective, symptoms, syndroms, five eleme...	a month ago	75.56%
service.kt	#76 client CRUD options context menu vs menu bar	a month ago	66.04%



```
// extension methods
fun ImageIcon.toMyImage(): MyImage = ImageIconImage(this)
fun BufferedImage.toMyImage(): MyImage = ImageIconImage(ImageIcon(this))
fun File.toMyImage(): MyImage = FileImage(this)
fun String.toMyImage(): MyImage = ClasspathImage(this)
fun ByteArray.toMyImage(): MyImage = this.readBufferedImage().toMyImage()

val Gender.defaultImage: MyImage get() =
    when(this) {
        Gender.MALE -> MyImage.DEFAULT_PROFILE_MAN
        Gender.FEMALE -> MyImage.DEFAULT_PROFILE_WOMAN
        else -> MyImage.DEFAULT_PROFILE_ALIEN
    }
```

Code Schmankerln

Extension Methods

Domain Object

Implement a straight-forward, *clean* **domain object**:

```
package at.cpickl.gadsu.client

data class Client(
  val id: String,
  val name: String
)
```

Persistence Extensions

Persistence specific functionality:

```
package at.cpickl.gadsu.persistence

data class ClientDbo(
  val TXT_ID: String,
  val TXT_NAME: String
)
```

Persistence Extensions

Persistence specific functionality:

```
package at.cpickl.gadsu.persistence

data class ClientDbo(
  val TXT_ID: String,
  val TXT_NAME: String
)
fun Client.toDbo() =
        ClientDbo(id, name)
```

Persistence Extensions

Persistence specific functionality:

```
package at.cpickl.gadsu.persistence

data class ClientDbo(
  val TXT_ID: String,
  val TXT_NAME: String
)
fun Client.toDbo() =
        ClientDbo(id, name)

class ClientRepo {
    fun save(client: Client) {
        saveSomewhere(client.toDbo())
    }
}
```

Nullable Persistence Extensions

Or for those masochists out there who prefer **nullables**:

```
package at.cpickl.gadsu.persistence

fun Client?.toDbo() =
  if (this == null) null
  else ClientDbo(id, name)
```

Nullable Persistence Extensions

Or for those masochists out there who prefer **nullables**:

```
package at.cpickl.gadsu.persistence

fun Client?.toDbo() =
  if (this == null) null
  else ClientDbo(id, name)

val client: Client? = ....
val dbo = client?.toDbo()
  ?: ClientDbo.defaultInstance()
```

Extend Swing Components

Add a **fluent API** to an existing classes:

```
fun <T : JComponent> T.bold(): T {
  font = font.deriveFont(Font.BOLD)
  return this
}
```

Extend Swing Components

Add a **fluent API** to an existing classes:

```
fun <T : JComponent> T.bold(): T {
  font = font.deriveFont(Font.BOLD)
  return this
}

val myLabel = JLabel("text").bold().italic()
val myTextField = JTextField("text").bold()
val myTextArea = JTextArea("text").bold()

val panel = JPanel().transparent()
```

Extension Properties

Testee Properties

Possible replacement of common **test factories**:

```
package at.cpickl.gadsu.test

val Client.Companion.testee1: Client
  get() = Client(
    id = "",
    name = "Max Muster"
)
```

Testee Properties

Possible replacement of common **test factories**:

```
package at.cpickl.gadsu.test

val Client.Companion.testee1: Client
  get() = Client(
    id = "",
    name = "Max Muster"
)
```

Unfortunately Kotlin requires to have some *placeholder*:

```
package at.cpickl.gadsu.client

data class Client( ... ) {
    companion object {}
}
```

Integration Testee

Use those testees in your **tests**:

```
package at.cpickl.gadsu.test

@Test class ClientIT {

  @Inject lateinit var repo: ClientRepo

  fun ‘reference test scoped testee‘() {
    repo.save(
      Client.testee1.copy(name = "Otto")
    )
    // ... assertions ...
  }
}
```

λ

Functional done right!

Java 8 being veeery verbose as always:

```
List<Integer> numbers = asList(1, 2, 3);
List<String> numbers2 = numbers
   .stream()
   .filter(i -> i % 2 == 0)
   .map(Object::toString)
   .collect(toList());
```

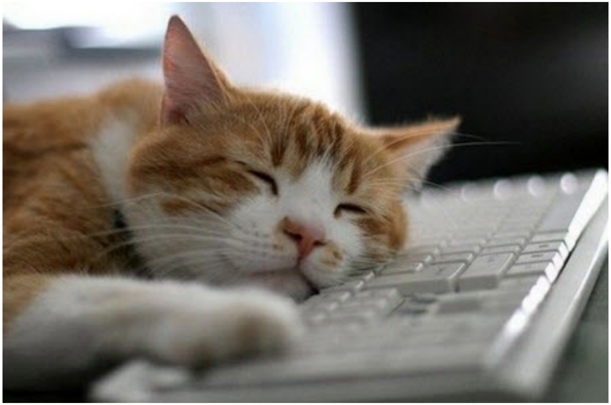
Kotlin (implicit `it` variable)

```
val numbers = listOf(1, 2, 3)
val numbers2 = numbers
   .filter { it % 2 == 0 }
   .map(Int::toString)
```

Java sometimes behaves like good old *Aunt Chatty*.

Kotlin Basic Functionals?!

- `fun <T, R> T.let(f: (T) -> R): R = f(this)`
- `fun <T> T.apply(f: T.() -> Unit): T { f(); return this }`
- `fun <T, R> with(receiver: T, f: T.() -> R): R = receiver.f()`
- `fun <T, R> T.run(f: T.() -> R): R = f()`



Sometimes I feel so lazy . . .

Lazy in Java

Given there is a *very expensive* `expensiveInit()` method:

Lazy in Java

Given there is a *very expensive* `expensiveInit()` method:

```
public class NaiveSingleton {
  private Object lazyField = null;

  public Object getLazyField() {
    if (lazyField == null) {
      lazyField = expensiveInit();
    }
    return lazyField;
  }
}
```

Lazy in Java8

```java
public class Java8 {
  private Supplier<Object> lazyField=() -> {
    Object value = expensiveInit();
    lazyField = () -> value;
    return value;
  };

  public Object getLazyField() {
    return lazyField.get();
  }
}
```

Lazy in Kotlin

```
class LazyKotlin {
  val lazyField by lazy {
    expensiveInit()
  }
}
```

Lazy in Kotlin

```
class LazyKotlin {
  val lazyField by lazy {
    expensiveInit()
  }
}

// part of stdlib:
fun <T> lazy(initializer: ()->T): Lazy<T> =
  SynchronizedLazyImpl(initializer)
```

Lazy in Kotlin

```
class LazyKotlin {
  val lazyField by lazy {
    expensiveInit()
  }
}

// part of stdlib:
fun <T> lazy(initializer: ()->T): Lazy<T> =
  SynchronizedLazyImpl(initializer)
```

Thanks to type inference, we don't need to specify types explicity.

Delegates

Class Delegation

Given the existing classes:

```
interface Step {
  fun take()
}
```

Class Delegation

Given the existing classes:

```
interface Step {
  fun take()
}

class StepImpl : Step {
  override fun take() {}
}
```

Class Delegation

Given the existing classes:

```
interface Step {
  fun take()
}

class StepImpl : Step {
  override fun take() {}
}
```

We now want some new service to implement this interface, but **delegate** all its methods to the StepImpl implementation.

Reimplement in Java

```
public class MyService implements Step {

  private final Step step;

  public MyService(Step step) {
    this.step = step;
  }

  @Override public void take() {
    step.take();
  }
}
```

Delegate by Kotlin

```kotlin
class MyService(step: Step) : Step by step
```

Delegate by Kotlin

```
class MyService(step: Step) : Step by step
```

Standard delegates in Kotlin:

- **lazy**
- observable
- map properties

Let's do some Swing!

Common Swing Component

```
class ClientTabMain(modifications, ...) {
```

Common Swing Component

```
class ClientTabMain(modifications, ...) {
  val fields = Fields<Client>(modifications)
```

Common Swing Component

```
class ClientTabMain(modifications, ...) {
  val fields = Fields<Client>(modifications)
  val inpNote = fields
    .newTextArea("Notiz", { it.note },
       ViewNames.Client.InputNote, bus)
```

Common Swing Component

```
class ClientTabMain(modifications, ...) {
  val fields = Fields<Client>(modifications)
  val inpNote = fields
    .newTextArea("Notiz", { it.note },
       ViewNames.Client.InputNote, bus)
  init {
    add(VFillFormPanel().apply {
      addFormInput(inpNote)
    })}
```

Common Swing Component

```kotlin
class ClientTabMain(modifications, ...) {
  val fields = Fields<Client>(modifications)
  val inpNote = fields
    .newTextArea("Notiz", { it.note },
       ViewNames.Client.InputNote, bus)
  init {
    add(VFillFormPanel().apply {
      addFormInput(inpNote)
    })}
  fun isModified(client: Client) =
    fields.isAnyModified(client)
```

Common Swing Component

```
class ClientTabMain(modifications, ...) {
  val fields = Fields<Client>(modifications)
  val inpNote = fields
    .newTextArea("Notiz", { it.note },
       ViewNames.Client.InputNote, bus)
  init {
    add(VFillFormPanel().apply {
      addFormInput(inpNote)
    })}
  fun isModified(client: Client) =
    fields.isAnyModified(client)
  fun updateFields(client: Client) {
    fields.updateAll(client)
  }
}
```

Spec4J UI Test

```
@Test(groups = arrayOf("uiTest"))
class ClientUiTest : UiTest() {
 fun ‘save client should change UI‘() {
```

Spec4J UI Test

```
@Test(groups = arrayOf("uiTest"))
class ClientUiTest : UiTest() {
 fun ‘save client should change UI‘() {
  val client = Client.unsavedValidInstance()
```

Spec4J UI Test

```
@Test(groups = arrayOf("uiTest"))
class ClientUiTest : UiTest() {
 fun ‘save client should change UI‘() {
  val client = Client.unsavedValidInstance()
  with(driver) {
```

Spec4J UI Test

```
@Test(groups = arrayOf("uiTest"))
class ClientUiTest : UiTest() {
 fun ‘save client should change UI‘() {
  val client = Client.unsavedValidInstance()
  with(driver) {
   assertSaveButtonTextEquals("Neu anlegen")
```

Spec4J UI Test

```
@Test(groups = arrayOf("uiTest"))
class ClientUiTest : UiTest() {
 fun ‘save client should change UI‘() {
  val client = Client.unsavedValidInstance()
  with(driver) {
   assertSaveButtonTextEquals("Neu anlegen")

   saveClient(client)
   assertSaveButtonTextEquals("Speichern")
```

Spec4J UI Test

```
@Test(groups = arrayOf("uiTest"))
class ClientUiTest : UiTest() {
 fun ‘save client should change UI‘() {
  val client = Client.unsavedValidInstance()
  with(driver) {
   assertSaveButtonTextEquals("Neu anlegen")

   saveClient(client)
   assertSaveButtonTextEquals("Speichern")

   assertListContains(client)
   assertListSelected(client)
   ...
```

Let's see some code . . .

Lessons Learned

Tooling infrastructure grows

- *Mostly* as good as for Java

Tooling infrastructure grows

- *Mostly* as good as for Java
- **IntelliJ** support feels already superb

Tooling infrastructure grows

- *Mostly* as good as for Java
- **IntelliJ** support feels already superb
- **Build** system support (Gradle Script Kotlin!)

Tooling infrastructure grows

- *Mostly* as good as for Java
- **IntelliJ** support feels already superb
- **Build** system support (Gradle Script Kotlin!)
- Static code analysis tools **missing**

Tooling infrastructure grows

- *Mostly* as good as for Java
- **IntelliJ** support feels already superb
- **Build** system support (Gradle Script Kotlin!)
- Static code analysis tools **missing**
- Syntax highlighting mostly **missing**

8 reasons to use Kotlin

1. **Null handling** is a MUST!

I DON'T ALWAYS CODE IN JAVA
BUT WHEN I DO, IT'S A NULLPOINTEREXCEPTION
memegenerator.net

Remember me?!

8 reasons to use Kotlin

1. **Null handling** is a MUST!

8 reasons to use Kotlin

1. **Null handling** is a MUST!
2. **Extension methods** for improved auto completion

8 reasons to use Kotlin

1. **Null handling** is a MUST!
2. **Extension methods** for improved auto completion
3. (Constructor) Properties

8 reasons to use Kotlin

1. **Null handling** is a MUST!
2. **Extension methods** for improved auto completion
3. (Constructor) Properties
4. **Lambdas** done properly

8 reasons to use Kotlin

1. **Null handling** is a MUST!
2. **Extension methods** for improved auto completion
3. (Constructor) Properties
4. **Lambdas** done properly
5. (Local) Type **inference**

8 reasons to use Kotlin

1. **Null handling** is a MUST!
2. **Extension methods** for improved auto completion
3. (Constructor) Properties
4. **Lambdas** done properly
5. (Local) Type **inference**
6. Named and default **arguments**

8 reasons to use Kotlin

1. **Null handling** is a MUST!
2. **Extension methods** for improved auto completion
3. (Constructor) Properties
4. **Lambdas** done properly
5. (Local) Type **inference**
6. Named and default **arguments**
7. Compact **syntax**: No semicolon, new

8 reasons to use Kotlin

1. **Null handling** is a MUST!
2. **Extension methods** for improved auto completion
3. (Constructor) Properties
4. **Lambdas** done properly
5. (Local) Type **inference**
6. Named and default **arguments**
7. Compact **syntax**: No semicolon, new
8. **Data** classes replaces Lombok

Kotlin is a great language

Kotlin is a great language . . . BUT!

Kotlin is a great language . . . BUT!

- **Data classes** are still somehow restricted in their usage (v1.1)
- Paradigm shift of **final**-by-default clashes with existing libs
- Requires young padawan to be more **disciplined**
    - Several classes in one (big) file gets common
    - Explicit type declaration for documentation
    - Overuse of single-expression functions
    - Overuse of functionals like `apply{}`

Apply – Who is this?

```
class MainPanel : JPanel() {
  init {
    val subPanel = JPanel()
    subPanel.background = Color.RED
    add(subPanel)
  }
}
```

Apply – Who is this?

```
class MainPanel : JPanel() {
  init {
    val subPanel = JPanel()
    subPanel.background = Color.RED
    add(subPanel)
  }
}
```

We can **refactor** this to get rid of the variable reference.

Apply – This is not this anymore!

```
class MainPanel : JPanel() {
  init {
    JPanel().apply {
      background = Color.RED
      add(this)
    }
  }
}
```

Apply – This is not this anymore!

```
class MainPanel : JPanel() {
  init {
    JPanel().apply {
      background = Color.RED
      add(this)
    }
  }
}
```

```
IllegalArgumentException: adding container's parent to itself
```

Apply – This is not this anymore!

```
class MainPanel : JPanel() {
  init {
    JPanel().apply {
      background = Color.RED
      add(this)
    }
  }
}
```

```
IllegalArgumentException: adding container's parent to itself
```

- The solution is to use: `this@MainPanel.add(this)`

Apply – This is not this anymore!

```
class MainPanel : JPanel() {
  init {
    JPanel().apply {
      background = Color.RED
      add(this)
    }
  }
}
```

```
IllegalArgumentException: adding container's parent to itself
```

- The solution is to use: `this@MainPanel.add(this)`
- But what happens if there are **two nested** `JPanels`?!

Apply – This is not this anymore!

```
class MainPanel : JPanel() {
  init {
    JPanel().apply {
      background = Color.RED
      add(this)
    }
  }
}
```

`IllegalArgumentException: adding container's parent to itself`

- The solution is to use: `this@MainPanel.add(this)`
- But what happens if there are **two nested** `JPanels`?!
- PS: Kotlin 1.1 will come with a new function called `also()`

```
JPanel().also {
  this.add(it)
}
```

11 Kotlin 1.1 News

1. Compiler plugins (open-by-default, no-arg ctor)

11 Kotlin 1.1 News

1. Compiler plugins (open-by-default, no-arg ctor)
2. Coroutines

11 Kotlin 1.1 News

1. Compiler plugins (open-by-default, no-arg ctor)
2. Coroutines
3. Type aliases

11 Kotlin 1.1 News

1. Compiler plugins (open-by-default, no-arg ctor)
2. Coroutines
3. Type aliases
4. Type inference for getters

11 Kotlin 1.1 News

1. Compiler plugins (open-by-default, no-arg ctor)
2. Coroutines
3. Type aliases
4. Type inference for getters
5. Bound callable references

11 Kotlin 1.1 News

1. Compiler plugins (open-by-default, no-arg ctor)
2. Coroutines
3. Type aliases
4. Type inference for getters
5. Bound callable references
6. Local delegated properties & Inline properties

11 Kotlin 1.1 News

1. Compiler plugins (open-by-default, no-arg ctor)
2. Coroutines
3. Type aliases
4. Type inference for getters
5. Bound callable references
6. Local delegated properties & Inline properties
7. Inheritance for data classes

11 Kotlin 1.1 News

1. Compiler plugins (open-by-default, no-arg ctor)
2. Coroutines
3. Type aliases
4. Type inference for getters
5. Bound callable references
6. Local delegated properties & Inline properties
7. Inheritance for data classes
8. Subclasses of sealed classes in the same file

11 Kotlin 1.1 News

1. Compiler plugins (open-by-default, no-arg ctor)
2. Coroutines
3. Type aliases
4. Type inference for getters
5. Bound callable references
6. Local delegated properties & Inline properties
7. Inheritance for data classes
8. Subclasses of sealed classes in the same file
9. Destructuring in lambdas

11 Kotlin 1.1 News

1. Compiler plugins (open-by-default, no-arg ctor)
2. Coroutines
3. Type aliases
4. Type inference for getters
5. Bound callable references
6. Local delegated properties & Inline properties
7. Inheritance for data classes
8. Subclasses of sealed classes in the same file
9. Destructuring in lambdas
10. Underscore for unused parameters

11 Kotlin 1.1 News

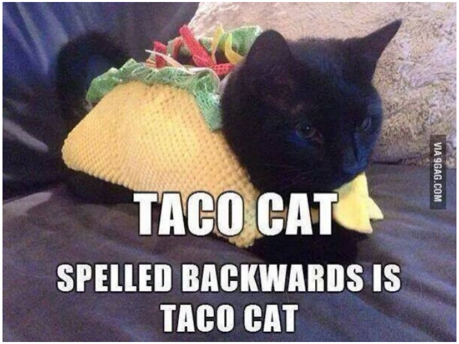
1. Compiler plugins (open-by-default, no-arg ctor)
2. Coroutines
3. Type aliases
4. Type inference for getters
5. Bound callable references
6. Local delegated properties & Inline properties
7. Inheritance for data classes
8. Subclasses of sealed classes in the same file
9. Destructuring in lambdas
10. Underscore for unused parameters
11. Underscore in numeric literals

Links

- Visit the website:
  https://github.com/christophpickl/gadsu
- LaTeX sources of the slides:
  https://github.com/christophpickl/gadsu_meetup
- Kotlin-Vienna Usergroup:
  https://www.meetup.com/Kotlin-Vienna

One more thing . . .

OBLIGATORY
MEME
memegenerator.net


TACO CAT
SPELLED BACKWARDS IS
TACO CAT
VIA 9GAG.COM

INVISIBLE BIKE

http://ifun.ru

I HATE
CAT MEMES