SpringOne Platform

Developing a Geospatial WebService with Kotlin and Spring Boot

by Sébastien Deleuze

@sdeleuze

Pivotal

Unless otherwise indicated, these slides are © 2013-2016 Pivotal Software, Inc. and licensed under a Creative Commons Attribution-NonCommercial license: http://creativecommons.org/licenses/by-nc/3.0/

Who am I?

- Sébastien Deleuze
- Live in Lyon, France
- Work at Pivotal
-  Spring Framework and  Reactor commiter
- Actual focus on Spring Framework 5 Reactive support
- Co-worker at la cordée 
- Staff member of mix-it conference

Unless otherwise indicated, these slides are © 2013-2016 Pivotal Software, Inc. and licensed under a Creative Commons Attribution-NonCommercial license: http://creativecommons.org/licenses/by-nc/3.0/

SpringOne Platform

Introducing Kotlin

Pivotal

Unless otherwise indicated, these slides are © 2013-2016 Pivotal Software, Inc. and licensed under a Creative Commons Attribution-NonCommercial license: http://creativecommons.org/licenses/by-nc/3.0/

Why Kotlin?

Unless otherwise indicated, these slides are © 2013-2016 Pivotal Software, Inc. and licensed under a Creative Commons Attribution-NonCommercial license: http://creativecommons.org/licenses/by-nc/3.0/

Limits of Java

- Verbose
- Limited type inference
- No properties
- Checked exception
- NullPointerException
- Extensibility
- End of lines with ;
- Java Puzzlers
- We deserve a better solution than Lombok ...

Unless otherwise indicated, these slides are © 2013-2016 Pivotal Software, Inc. and licensed under a Creative Commons Attribution-NonCommercial license: http://creativecommons.org/licenses/by-nc/3.0/

Keep what makes Java great

- Fast
- Optimized bytecode
- Static typing
- Simple to learn
- Amazing ecosystem

Unless otherwise indicated, these slides are © 2013-2016 Pivotal Software, Inc. and licensed under a Creative Commons Attribution-NonCommercial license: http://creativecommons.org/licenses/by-nc/3.0/

Kotlin

- Elegant and pragmatic language
  - Concise code
  - Fast to write
- Easy to read
- Awesome type inference with static typing
- Simple and easy to learn
- Very good Java interoperability
- Null safety, extensions, DSL, etc.
- No more ; at the end of lines :-)

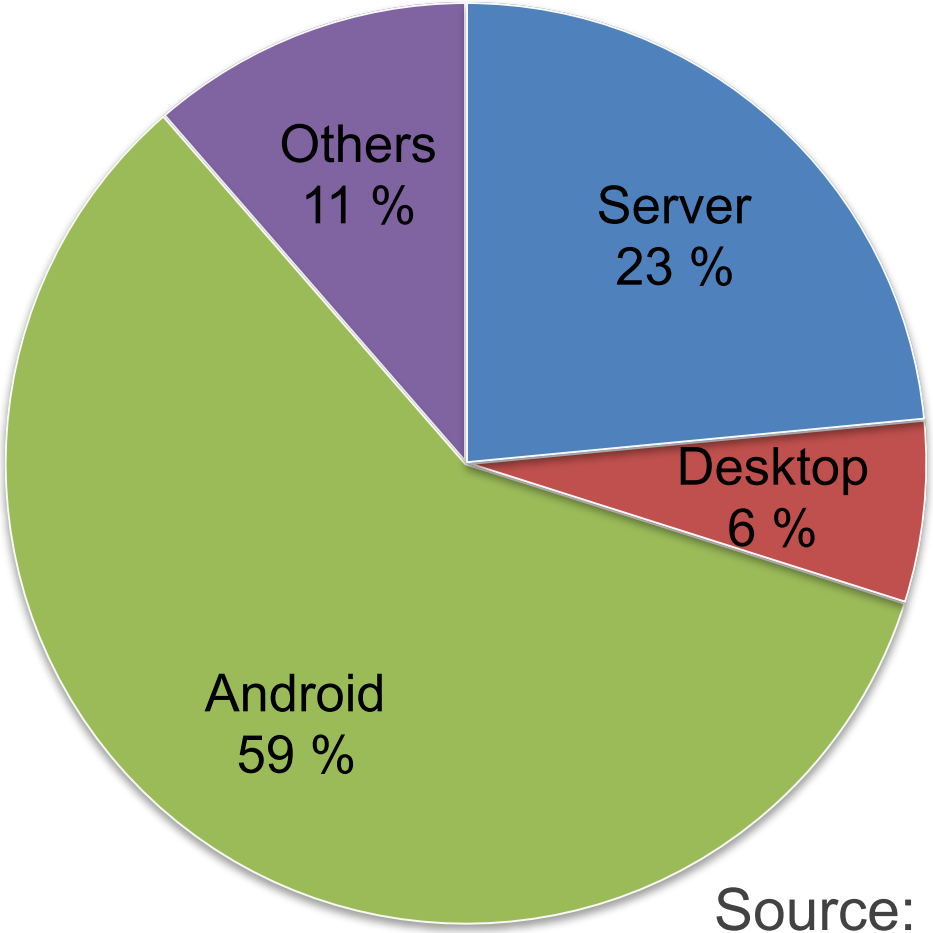
Unless otherwise indicated, these slides are © 2013-2016 Pivotal Software, Inc. and licensed under a Creative Commons Attribution-NonCommercial license: http://creativecommons.org/licenses/by-nc/3.0/

Status

- February 2016: Kotlin 1.0.0
- Latest stable version is 1.0.3
- First Kotlin 1.1 preview available

Unless otherwise indicated, these slides are © 2013-2016 Pivotal Software, Inc. and licensed under a Creative Commons Attribution-NonCommercial license: http://creativecommons.org/licenses/by-nc/3.0/

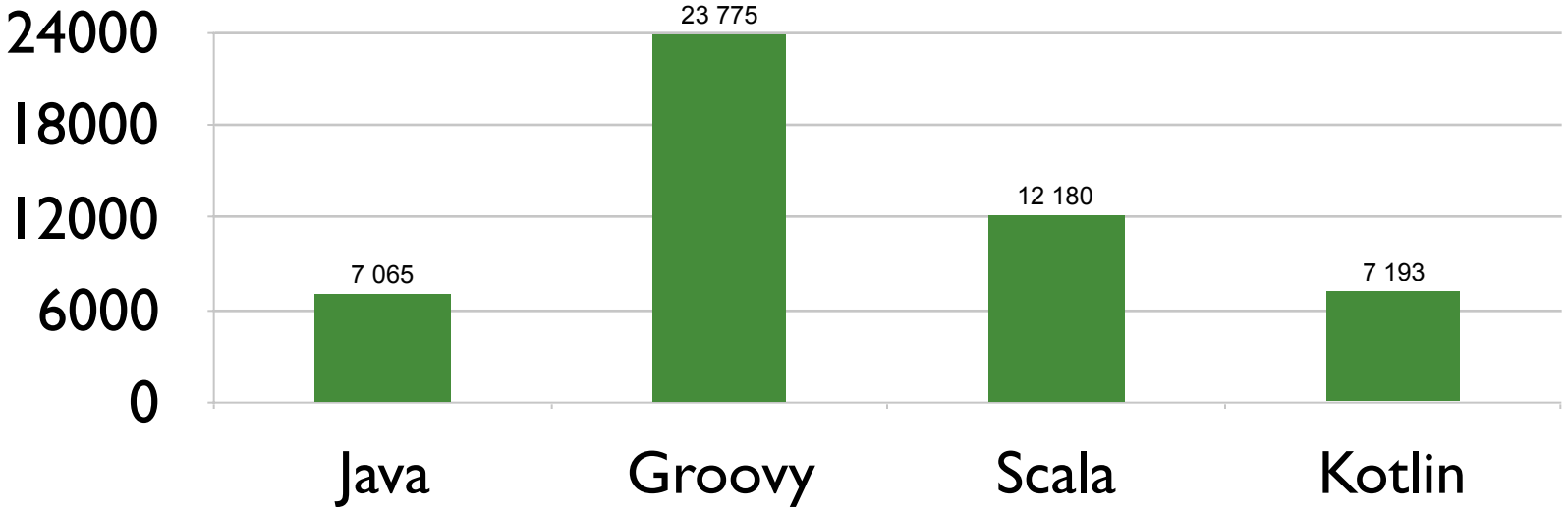
Use cases



Source: poll on Kotlin Slack

Unless otherwise indicated, these slides are © 2013-2016 Pivotal Software, Inc. and licensed under a Creative Commons Attribution-NonCommercial license: http://creativecommons.org/licenses/by-nc/3.0/

Methods count on Android app*



* https://github.com/SidneyXu/AndroidDemoIn4Languages

Unless otherwise indicated, these slides are © 2013-2016 Pivotal Software, Inc. and licensed under a Creative Commons Attribution-NonCommercial license: http://creativecommons.org/licenses/by-nc/3.0/

How does it compare with ...

Same conciseness and expressive code, but Kotlin static typing and null-safety make a big difference.

"Some people say Kotlin has 80% the power of Scala, with 20% of the features" *
"Kotlin is a software engineering language in contrast to Scala which is a computing science language." *

Swift and Kotlin are VERY similar. Swift is LLVM based and has C interop while Kotlin is JVM based and has Java interop.

* Quotes from Kotlin: The Ying and Yang of Programming Languages

Unless otherwise indicated, these slides are © 2013-2016 Pivotal Software, Inc. and licensed under a Creative Commons Attribution-NonCommercial license: http://creativecommons.org/licenses/by-nc/3.0/

Classes

```
class User(
        var userName:  String,
        var firstName: String,
        var lastName:  String,
        var location:  Point? = null
)
```

```
public class User {

    private String userName;
    private String firstName;
    private String lastName;
    private Point  location;

    public User(String userName, String firstName, String lastName) {
        this(userName,firstName, lastName, null);
    }

    public User(String userName, String firstName,
                String lastName, Point location) {
        Assert.notNull(userName);
        Assert.notNull(firstName);
        Assert.notNull(lastName);
        this.userName = userName;
        this.firstName = firstName;
        this.lastName = lastName;
        this.location = location;
    }

    public String getUserName() {
        return userName;
    }

    public void setUserName(String userName) {
        this.userName = userName;
    }

    public String getFirstName() {
        return firstName;
    }

    public void setFirstName(String firstName) {
        this.firstName = firstName;
    }

    public String getLastName() {
        return lastName;
    }

    public void setLastName(String lastName) {
        this.lastName = lastName;
    }

    public Point getLocation() {
        return location;
    }

    public void setLocation(Point location) {
        this.location = location;
    }
}
```


Unless otherwise indicated, these slides are © 2013-2016 Pivotal Software, Inc. and licensed under a Creative Commons Attribution-NonCommercial license: http://creativecommons.org/licenses/by-nc/3.0/

Data classes

```
data class User(
        var userName:  String,
        var firstName: String,
        var lastName:  String,
        var location:  Point? = null
)
```

```
public class User {

    private String userName;
    private String firstName;
    private String lastName;
    private Point  location;

    public User(String userName, String firstName, String lastName) {
        this(userName,firstName, lastName, null);
    }

    public User(String userName, String firstName, String lastName, Point location) {
        Assert.notNull(userName);
        Assert.notNull(firstName);
        Assert.notNull(lastName);
        this.userName = userName;
        this.firstName = firstName;
        this.lastName = lastName;
        this.location = location;
    }

    public String getUserName() {
        return userName;
    }

    public void setUserName(String userName) {
        this.userName = userName;
    }

    public String getFirstName() {
        return firstName;
    }

    public void setFirstName(String firstName) {
        this.firstName = firstName;
    }

    public String getLastName() {
        return lastName;
    }

    public void setLastName(String lastName) {
        this.lastName = lastName;
    }

    public Point getLocation() {
        return location;
    }

    public void setLocation(Point location) {
        this.location = location;
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) {
            return true;
        }
        if (o == null || getClass() != o.getClass()) {
            return false;
        }

        UserJ j = (UserJ) o;

        if (!userName.equals(j.userName)) {
            return false;
        }
        if (!firstName.equals(j.firstName)) {
            return false;
        }
        if (!lastName.equals(j.lastName)) {
            return false;
        }
        return location != null ? location.equals(j.location) : j.location == null;

    }

    @Override
    public int hashCode() {
        int result = userName.hashCode();
        result = 31 * result + firstName.hashCode();
        result = 31 * result + lastName.hashCode();
        result = 31 * result + (location != null ? location.hashCode() : 0);
        return result;
    }

    @Override
    public String toString() {
        return "User{" +
                "userName='" + userName + '\'' +
                ", firstName='" + firstName + '\'' +
                ", lastName='" + lastName + '\'' +
                ", location=" + location +
                '}';
    }
}
```


Unless otherwise indicated, these slides are © 2013-2016 Pivotal Software, Inc. and licensed under a Creative Commons Attribution-NonCommercial license: http://creativecommons.org/licenses/by-nc/3.0/

Optional and named parameter

```
// Given the following function
fun reformat(str: String,
             normalizeCase: Boolean = true,
             upperCaseFirstLetter: Boolean = true,
             divideByCamelHumps: Boolean = false,
             wordSeparator: Char = ' ') {

}

// You could call
reformat("foo bar")
```


Unless otherwise indicated, these slides are © 2013-2016 Pivotal Software, Inc. and licensed under a Creative Commons Attribution-NonCommercial license: http://creativecommons.org/licenses/by-nc/3.0/

Optional and named parameter

```
// Given the following function
fun reformat(str: String,
             normalizeCase: Boolean = true,
             upperCaseFirstLetter: Boolean = true,
             divideByCamelHumps: Boolean = false,
             wordSeparator: Char = ' ') {

}

// You could call
reformat("foo bar", true, true, false, '_')
```


Unless otherwise indicated, these slides are © 2013-2016 Pivotal Software, Inc. and licensed under a Creative Commons Attribution-NonCommercial license: http://creativecommons.org/licenses/by-nc/3.0/

Optional and named parameter

```
// Given the following function
fun reformat(str: String,
             normalizeCase: Boolean = true,
             upperCaseFirstLetter: Boolean = true,
             divideByCamelHumps: Boolean = false,
             wordSeparator: Char = ' ') {

}

// You could call
reformat("foo bar", wordSeparator = '_')
```


Unless otherwise indicated, these slides are © 2013-2016 Pivotal Software, Inc. and licensed under a Creative Commons Attribution-NonCommercial license: http://creativecommons.org/licenses/by-nc/3.0/

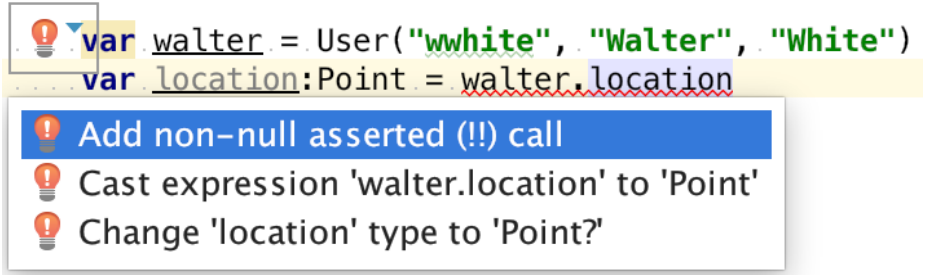
Optional and named parameter

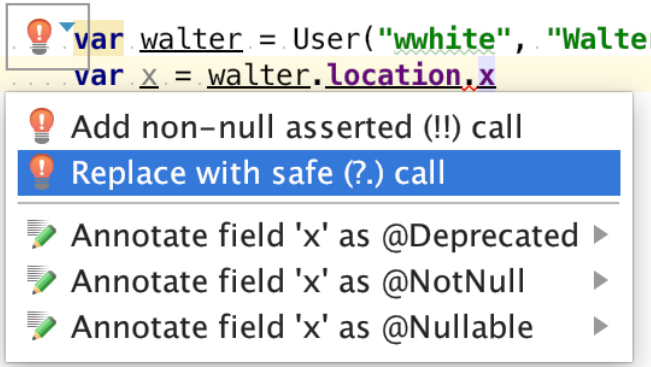
```
// Given the following function
fun reformat(str: String,
             normalizeCase: Boolean = true,
             upperCaseFirstLetter: Boolean = true,
             divideByCamelHumps: Boolean = false,
             wordSeparator: Char = ' ') {

}

// You could call
reformat("foo bar",
    normalizeCase = true,
    upperCaseFirstLetter = true,
    divideByCamelHumps = false,
    wordSeparator = '_'
)
```


Unless otherwise indicated, these slides are © 2013-2016 Pivotal Software, Inc. and licensed under a Creative Commons Attribution-NonCommercial license: http://creativecommons.org/licenses/by-nc/3.0/

Null safety

```
class User(
        var userName:  String,
        var firstName: String,
        var lastName:  String,
        var location:  Point? = null
)
```





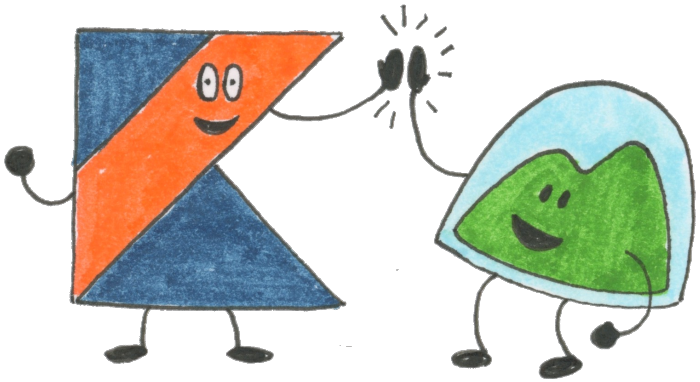
Unless otherwise indicated, these slides are © 2013-2016 Pivotal Software, Inc. and licensed under a Creative Commons Attribution-NonCommercial license: http://creativecommons.org/licenses/by-nc/3.0/

Null safety

```
class User(
        var userName:  String,
        var firstName: String,
        var lastName:  String,
        var location:  Point? = null
)

var walter = User("wwhite", "Walter", "White")

var location:Point? = walter.location
var location2       = walter.location
var location3:Point = walter.location ?: Point(0.0,0.0,0.0)

var x = walter.location?.x
```


Unless otherwise indicated, these slides are © 2013-2016 Pivotal Software, Inc. and licensed under a Creative Commons Attribution-NonCommercial license: http://creativecommons.org/licenses/by-nc/3.0/

How I fell in love with a programming language?



« For many years my perspective was simple—I didn’t have to love Java (or whatever programming language) to do my work well.That all changed a few months ago. » Dan Kim, Basecamp

Unless otherwise indicated, these slides are © 2013-2016 Pivotal Software, Inc. and licensed under a Creative Commons Attribution-NonCommercial license: http://creativecommons.org/licenses/by-nc/3.0/

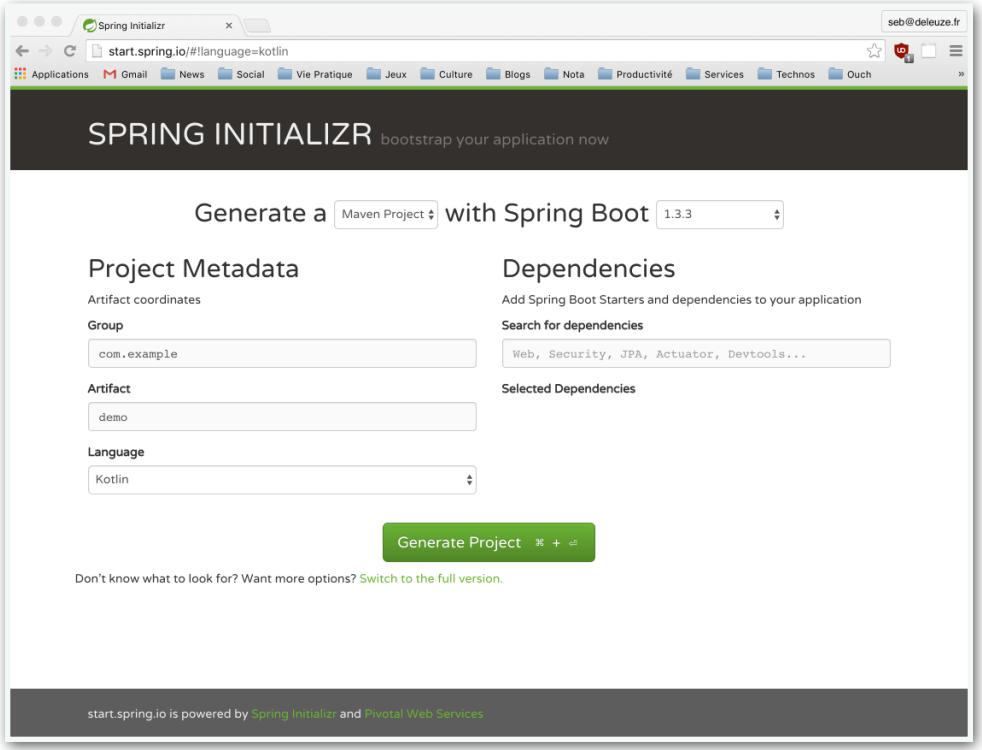
SpringOne Platform

Spring + Kotlin

Pivotal

Unless otherwise indicated, these slides are © 2013-2016 Pivotal Software, Inc. and licensed under a Creative Commons Attribution-NonCommercial license: http://creativecommons.org/licenses/by-nc/3.0/

http://start.spring.io/#!language=kotlin



Unless otherwise indicated, these slides are © 2013-2016 Pivotal Software, Inc. and licensed under a Creative Commons Attribution-NonCommercial license: http://creativecommons.org/licenses/by-nc/3.0/

Spring Boot + Kotlin

+

Same pragmatic and innovative mindset

Unless otherwise indicated, these slides are © 2013-2016 Pivotal Software, Inc. and licensed under a Creative Commons Attribution-NonCommercial license: http://creativecommons.org/licenses/by-nc/3.0/

Friction points (will be fixed in 1.1)

- **`@RequestMapping(method=arrayOf(GET))`**
- **`@GetMapping`** with Spring 4.3+

- Kotlin's « final by default » behavior is not a good fit with CGLIB proxies
  - For **`@Service`** and **`@Repository`**, use JDK dynamic proxies when possible (default with class + interface)
  - For other cases, **`@Configuration`** classes and **`@Bean`** methods, explicit **`open`** modifier is required

Unless otherwise indicated, these slides are © 2013-2016 Pivotal Software, Inc. and licensed under a Creative Commons Attribution-NonCommercial license: http://creativecommons.org/licenses/by-nc/3.0/

https://github.com/sdeleuze/spring-kotlin

Spring + Kotlin FAQ

What is the simplest way to start a Spring Boot + Kotlin application?

Go to http://start.spring.io/#!language=kotlin, add your dependencies, choose Gradle or Maven and click on "Generate Project" ;-)

Where this "Configuration problem: @Configuration class 'Foo' may not be final" error message come from?

Spring applications are using proxies to deal with most `@Component` beans like `@Configuration`, `@Service` or `@Repository`. JDK dynamic proxies are used when possible (your bean should implement an interface, and `proxyTargetClass` should be set to `false`, which is the default) while CGLIB proxies are used with classes that do not expose methods with interfaces, and/or when `proxyTargetClass` is set to `true`.

Unlike Java, Kotlin have made the choice to have a `final` by default behavior, with `open` keyword used at class or method level to allow extending or overriding them. As a consequence, with CGLIB proxies you need to explicitely specify `open` at class and method level since they require to extend/override the class/methods for technical reasons. There is some discussions on issue KT-12149 and here to see if we can find a solution but there is no perfect one yet. JDK dynamic proxy works perfectly well with Kotlin and are considered as best practice, but can't be used everywhere.

So current best practice is:

- Add `open` modifier at class level for `@Configuration` class and at `@Bean` methods level since they require CGLIB proxies
- Use JDK dynamic proxies for other `@Component` beans like `@Service` and `@Repository` by implementing interfaces and making sure `proxyTargetClass=false`. Be aware that Spring Boot set `proxyTargetClass=true` by default when JDBC is used, to use JDK dynamic proxies instead you just have to declare a `PersistenceExceptionTranslationPostProcessor` `@Bean` like here.

Unless otherwise indicated, these slides are © 2013-2016 Pivotal Software, Inc. and licensed under a Creative Commons Attribution-NonCommercial license: http://creativecommons.org/licenses/by-nc/3.0/

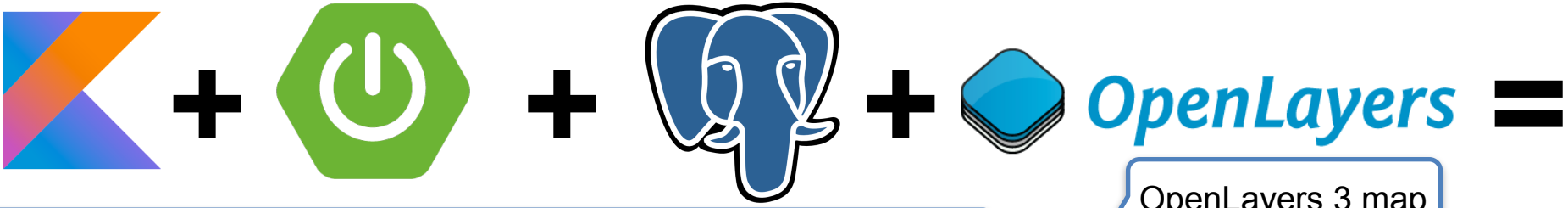
SpringOne Platform

Geospatial Messenger case study

Pivotal

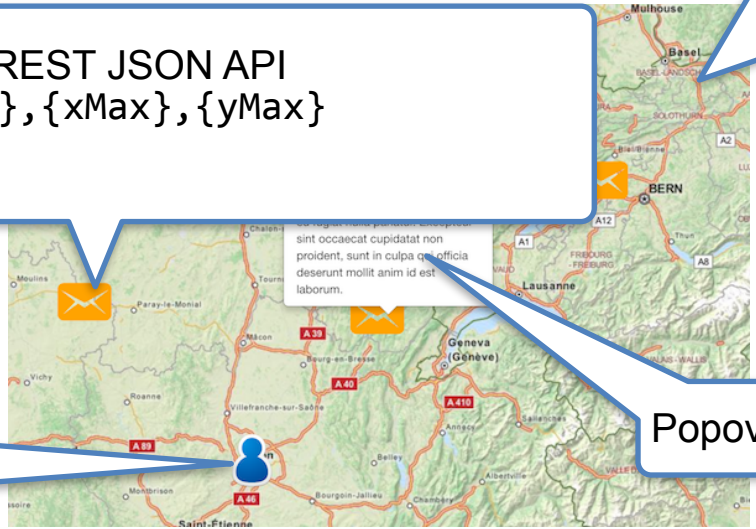
Unless otherwise indicated, these slides are © 2013-2016 Pivotal Software, Inc. and licensed under a Creative Commons Attribution-NonCommercial license: http://creativecommons.org/licenses/by-nc/3.0/

Geospatial Messenger



Layer of points retrieved through REST JSON API
`/messages/bbox/{xMin},{yMin},{xMax},{yMax}`
or pushed via Server-Sent Events

OpenLayers 3 map

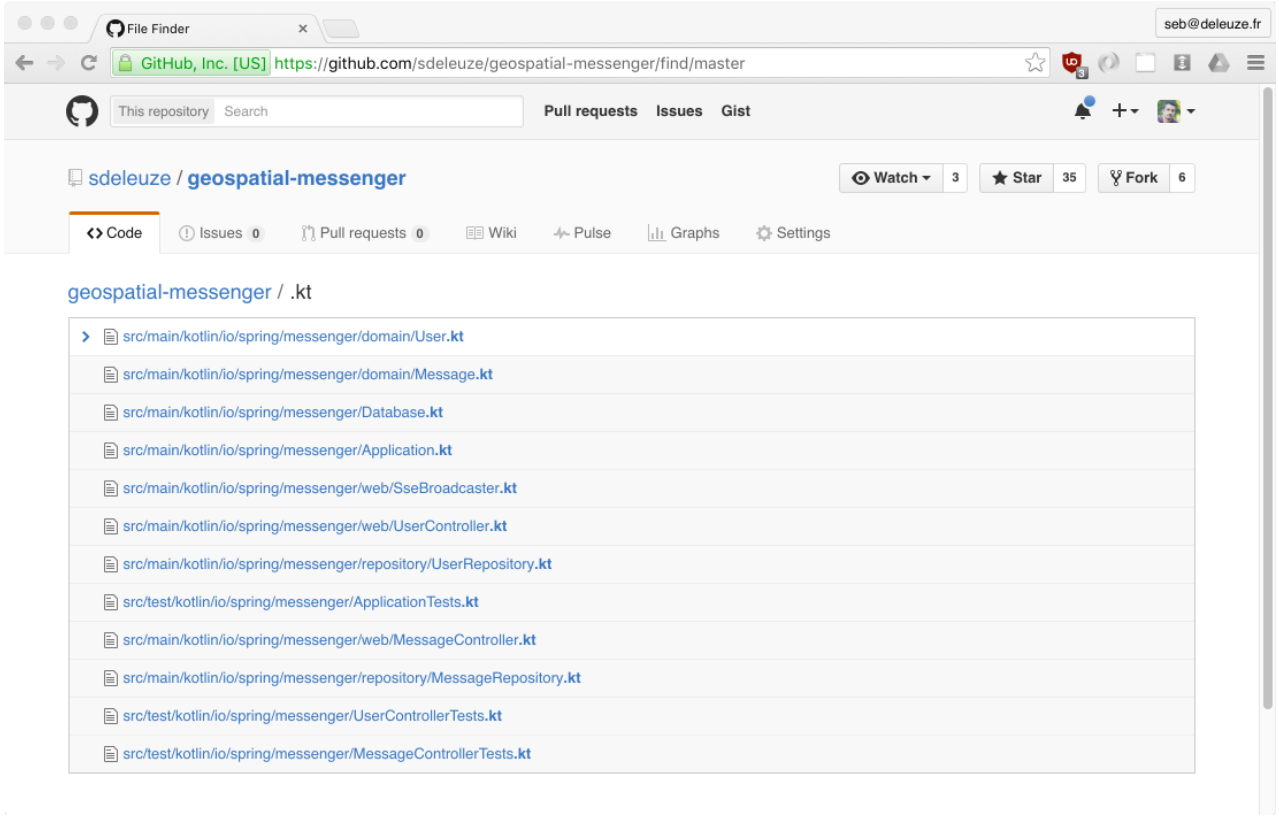


User location with
HTML5 geolocation

Popover Bootstrap (CSS + JS)

Unless otherwise indicated, these slides are © 2013-2016 Pivotal Software, Inc. and licensed under a Creative Commons Attribution-NonCommercial license: http://creativecommons.org/licenses/by-nc/3.0/

Code available on GitHub



Unless otherwise indicated, these slides are © 2013-2016 Pivotal Software, Inc. and licensed under a Creative Commons Attribution-NonCommercial license: http://creativecommons.org/licenses/by-nc/3.0/

Spring MVC Controller

```
@RestController
@RequestMapping("/user")
class UserController(val repository: UserRepository) {

    @PostMapping
    @ResponseStatus(CREATED)
    fun create(@RequestBody u: User) { repository.create(u) }

    @GetMapping
    fun list() = repository.findAll()

    @GetMappin
    fun findBy                                              e yMin:Double,
                                                            e yMax:Double)
            = repository.findByBoundingBox(PGbox2d(Point(xMin, yMin), Point(xMax, yMax)))

    @PutMapping("/{userName}/location/{x},{y}")
    @ResponseStatus(NO_CONTENT)
    fun updateLocation(@PathVariable userName:String,
                       @PathVariable x: Double, @PathVariable y: Double)
            = repository.updateLocation(userName, Point(x, y))
}
```

Classes are public by default

New in 4.3: constructor injection without @Autowired if single constructor

New in 4.3: method specific aliases for @RequestMapping

Type inference + shorter syntax for one line functions

Methods are public by default

Unless otherwise indicated, these slides are © 2013-2016 Pivotal Software, Inc. and licensed under a Creative Commons Attribution-NonCommercial license: http://creativecommons.org/licenses/by-nc/3.0/

Repository

Choosing Spring Data JPA is perfectly fine with Kotlin* but why not trying using a **type-safe Kotlin SQL DSL like**

Exposed

* See https://github.com/sdeleuze/spring-boot-kotlin-demo

Unless otherwise indicated, these slides are © 2013-2016 Pivotal Software, Inc. and licensed under a Creative Commons Attribution-NonCommercial license: http://creativecommons.org/licenses/by-nc/3.0/

Why using SQL without JPA?

- More control on SQL queries and joins
- Lighter technology stack
- Take advantage of native database functionalities
- We rarely need to change databases

Unless otherwise indicated, these slides are © 2013-2016 Pivotal Software, Inc. and licensed under a Creative Commons Attribution-NonCommercial license: http://creativecommons.org/licenses/by-nc/3.0/

PostgreSQL native JSON support

```
CREATE TABLE "users" (
    user_name text PRIMARY KEY,
    first_name text,
    last_name text,
    metadatas jsonb
);
CREATE INDEX users_metadatas_gin on users USING GIN(metadatas jsonb_path_ops);
INSERT INTO users VALUES ("wwhite", "Walter", "White",
    '{
    "address":{
        "line1":"308 Negra Arroyo Lane",
        "postcode":"87104"
        "city":"Albuquerque"
        "state":"New Mexico"
        },
    "foo":"bar"
    }');
SELECT * FROM users WHERE doc @> '{ "address": { "postcode":"87104" } }';
```


Unless otherwise indicated, these slides are © 2013-2016 Pivotal Software, Inc. and licensed under a Creative Commons Attribution-NonCommercial license: http://creativecommons.org/licenses/by-nc/3.0/

Regular schema.sql

```
CREATE TABLE IF NOT EXISTS "users" (
    user_name text PRIMARY KEY,
    first_name text,
    last_name text,
    location GEOMETRY(Point, 4326)
);
CREATE INDEX IF NOT EXISTS users_gix
    ON "users"
    USING GIST (location);


CREATE TABLE IF NOT EXISTS messages (
    id SERIAL PRIMARY KEY,
    content text NOT NULL,
    author text REFERENCES users(user_name),
    location GEOMETRY(Point, 4326)
);
CREATE INDEX IF NOT EXISTS messages_gix
    ON messages
    USING GIST (location);
```


Unless otherwise indicated, these slides are © 2013-2016 Pivotal Software, Inc. and licensed under a Creative Commons Attribution-NonCommercial license: http://creativecommons.org/licenses/by-nc/3.0/

SQL schema with Kotlin + Exposed

```
object Messages : Table() {
    val id       = integer("id").autoIncrement().primaryKey()
    val content  = text("content")
    val author   = reference("author", Users.userName)
    val location = point("location").nullable()
}

object Users : Table() {
    val userName  = text("user_name").primaryKey()
    val firstName = text("first_name")
    val lastName  = text("last_name")
    val location  = point("location").nullable()
}
```


Unless otherwise indicated, these slides are © 2013-2016 Pivotal Software, Inc. and licensed under a Creative Commons Attribution-NonCommercial license: http://creativecommons.org/licenses/by-nc/3.0/

Repository interface

```
interface CrudRepository<T, K> {
    fun createTable()
    fun create(m: T): T
    fun findAll(): Iterable<T>
    fun deleteAll(): Int
    fun findByBoundingBox(box: PGbox2d): Iterable<T>
    fun updateLocation(userName:K, location: Point)
}

interface MessageRepository: CrudRepository<Message, Int>

interface UserRepository:    CrudRepository<User, String>
```


Unless otherwise indicated, these slides are © 2013-2016 Pivotal Software, Inc. and licensed under a Creative Commons Attribution-NonCommercial license: http://creativecommons.org/licenses/by-nc/3.0/

Repository implementation

```
@Repository
@Transactional
class DefaultUserRepository : UserRepository {

    override fun createTable() = SchemaUtils.create(Users)

    override fun create(user: User): User {
        Users.insert(toRow(user))
        return use
    }

    override fun updateLocati userName:String, location: Point) {
        location.srid = 4326
        Users.update({Users.userName eq userName}) { it[Users.location] = location}
    }

    override fun findAll() = Users.selectA

    override fun findByBoundingBox(box: PGbox2d) =
        Users.select { Users.location within box }.map { fromRow(it) }

    override fun deleteAll() = Users.deleteAll()

    private fun toRow(u: User): Users.(U ateBuilder<*>) -> Unit = {
        it[userName] = u.userName
        it[firstName] = u.firstName
        it[lastName] = u.lastName
        it[location] = u.location
    }

    private fun fromRow(r: ResultRow) =
        User(r[Users.userName], r[Users.firstName], r[Users.lastName], r[Users.location])

}
```

Exposed now supports Spring @Transactional (should be on @Service in real apps)

Write your SQL queries with a Kotlin type-safe DSL

infix notation equivalent to Users.userName.eq(userName)

Geospatial extensions are not supported by Exposed, so how can I write that ???

Unless otherwise indicated, these slides are © 2013-2016 Pivotal Software, Inc. and licensed under a Creative Commons Attribution-NonCommercial license: http://creativecommons.org/licenses/by-nc/3.0/

Kotlin extensions

```
class PointColumnType(val srid: Int = 4326): ColumnType() {…}

fun Table.point(name: String, srid: Int = 4326): Column<Point>
  = registerColumn(name, PointColumnType())

// Usage
object FooTable : Table() {
   val location  = point("location")
}
```

```
class WithinOp(val e: Expression<*>, val box: PGbox2d) : Op<Boolean>() {…}

infix fun ExpressionWithColumnType<*>.within(box: PGbox2d) : Op<Boolean>
  = WithinOp(this, box)

// Usage
var box = PGbox2d(Point(0.0, 0.0),Point(0.0, 0.0))
FooTable.select { FooTable.location within box }
```


Unless otherwise indicated, these slides are © 2013-2016 Pivotal Software, Inc. and licensed under a Creative Commons Attribution-NonCommercial license: http://creativecommons.org/licenses/by-nc/3.0/

Configuration

```
@SpringBootApplication
@EnableTransactionM
open class Application {

    @Bean
    open fun objectMapper(): ObjectMapper {
        val mapper:ObjectMapper = Jackson2ObjectMapperBuilder().modulesToInstall(PostGISModule()).build()
        mapper.setSerializationInclusion(Include.NON_NULL)
        return mapper
    }

    @Bean
    open fun transactionManager(dataSource: DataSource) = SpringTransactionManager(dataSource)

    @Bean
    open fun persistenceExceptionTranslationPostProcessor() = PersistenceExceptionTranslationPostProcessor()

    @Bean
    open fun init(ur: UserRepository, mr: MessageRepository) = CommandLineRunner {
        ur.createTable()
        // ...
        ur.create(User("swhite", "Skyler", "White"))
        // ...
   }
}

fun main(args: Array<String>) {
    SpringApplication.run(Application::class.java, *args)
}
```

open needed because of CGLIB

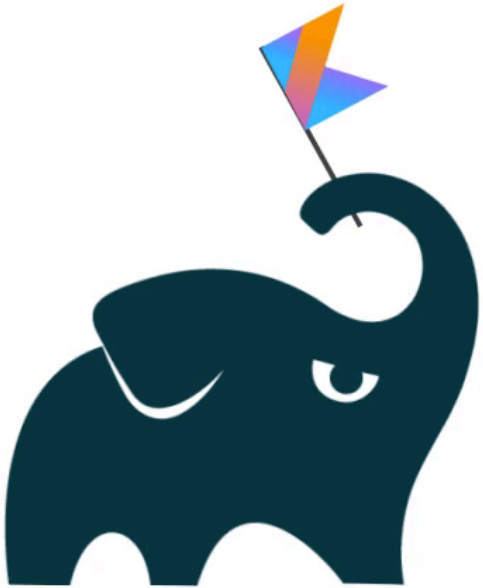
New in 4.3: Kotlin Jackson module automatically registered

open needed because of CGLIB

Top level function

Unless otherwise indicated, these slides are © 2013-2016 Pivotal Software, Inc. and licensed under a Creative Commons Attribution-NonCommercial license: http://creativecommons.org/licenses/by-nc/3.0/

Gradle goes Kotlin !!!



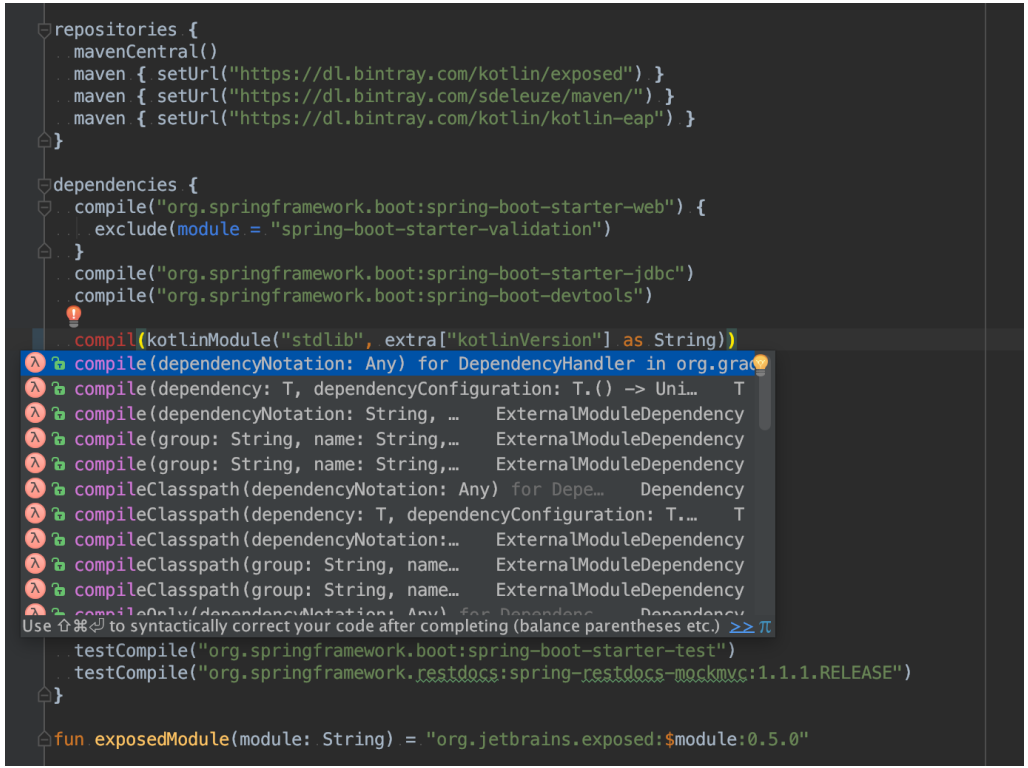
Unless otherwise indicated, these slides are © 2013-2016 Pivotal Software, Inc. and licensed under a Creative Commons Attribution-NonCommercial license: http://creativecommons.org/licenses/by-nc/3.0/

Build Gradle written in Kotlin (build.gradle.kts)

```
buildscript {
  repositories {
    mavenCentral()
    gradleScriptKotlin()
  }
  dependencies {
    classpath(kotlinModule("gradle-plugin"))
    classpath("org.springframework.boot:spring-boot-gradle-plugin:1.4.0.RELEASE")
  }
}
apply {
  plugin("kotlin")
  plugin("spring-boot")
}
repositories {
  mavenCentral()
  maven { setUrl("https://dl.bintray.com/kotlin/exposed") }
  maven { setUrl("https://dl.bintray.com/sdeleuze/maven/") }
}
dependencies {
  compile("org.springframework.boot:spring-boot-starter-web") {
    exclude(module = "spring-boot-starter-validation")
  }
  compile("org.springframework.boot:spring-boot-starter-jdbc")
  compile("org.springframework.boot:spring-boot-devtools")

  compile(kotlinModule("stdlib"))
  compile(kotlinModule("reflect"))
  compile("com.fasterxml.jackson.module:jackson-module-kotlin:2.7.5")

  compile(exposedModule("exposed"))
  compile(exposedModule("spring-transaction"))
  ...
}
fun exposedModule(val module: String) = "org.jetbrains.exposed:$module:0.5.0"
```


Unless otherwise indicated, these slides are © 2013-2016 Pivotal Software, Inc. and licensed under a Creative Commons Attribution-NonCommercial license: http://creativecommons.org/licenses/by-nc/3.0/

Game changer : autocomplete and validation



Unless otherwise indicated, these slides are © 2013-2016 Pivotal Software, Inc. and licensed under a Creative Commons Attribution-NonCommercial license: http://creativecommons.org/licenses/by-nc/3.0/

Deployment

- Regular Spring Boot Runnable JAR
- 18 MBytes
- Start within 2.8 seconds on my laptop
- Run with -Xmx32m !!!

Unless otherwise indicated, these slides are © 2013-2016 Pivotal Software, Inc. and licensed under a Creative Commons Attribution-NonCommercial license: http://creativecommons.org/licenses/by-nc/3.0/

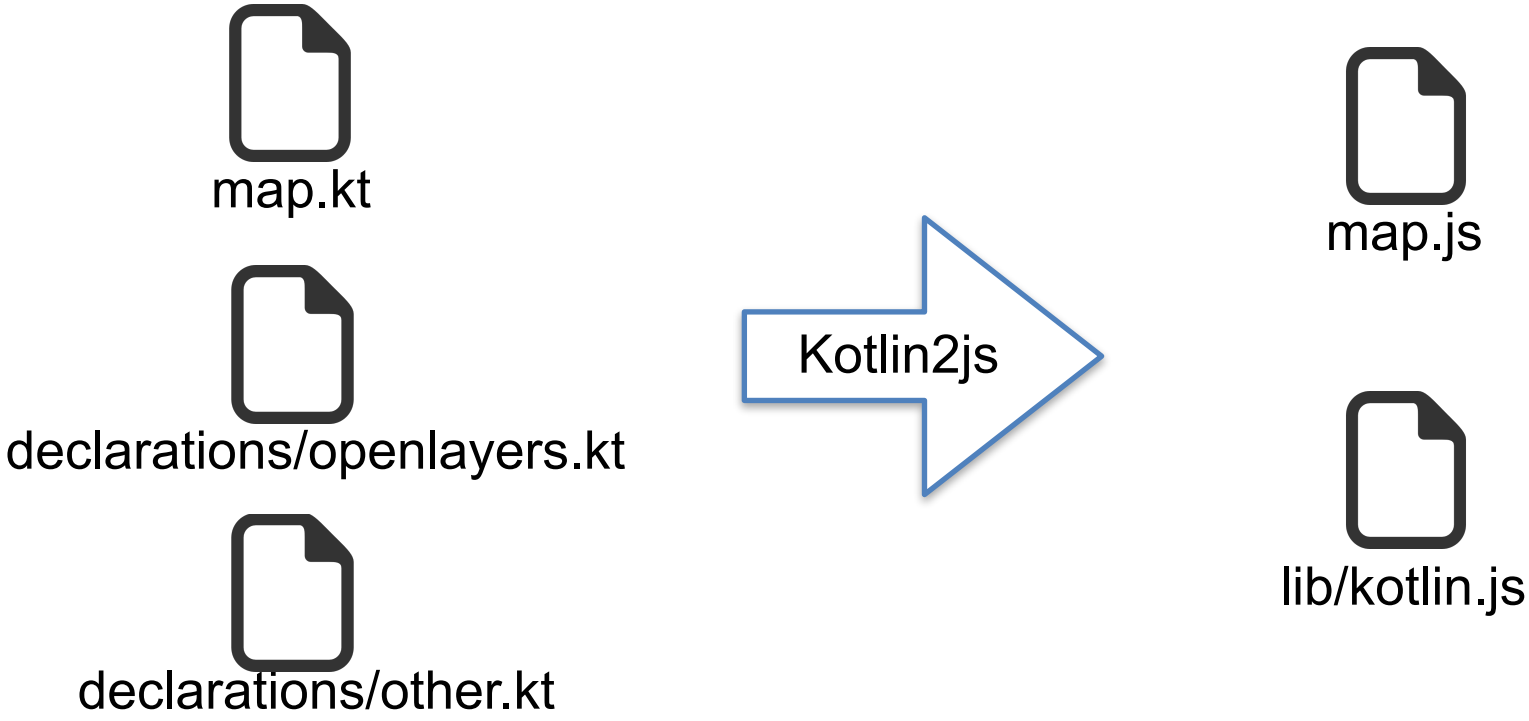
SpringOne Platform

Kotlin for client side development

Pivotal

Unless otherwise indicated, these slides are © 2013-2016 Pivotal Software, Inc. and licensed under a Creative Commons Attribution-NonCommercial license: http://creativecommons.org/licenses/by-nc/3.0/

Kotlin2js



See https://github.com/sdeleuze/geospatial-messenger/tree/kotlin-js

Unless otherwise indicated, these slides are © 2013-2016 Pivotal Software, Inc. and licensed under a Creative Commons Attribution-NonCommercial license: http://creativecommons.org/licenses/by-nc/3.0/

declarations/other.kt

```
package declarations

import jquery.JQuery
import org.w3c.dom.Element

@native("$")
val j: dynamic = noImpl

@native
fun JQuery.append(element: Element): JQuery = noImpl

@native
fun alert(a: Any) {}
```


Unless otherwise indicated, these slides are © 2013-2016 Pivotal Software, Inc. and licensed under a Creative Commons Attribution-NonCommercial license: http://creativecommons.org/licenses/by-nc/3.0/

declarations/openlayers.kt

```
package openlayers

import org.w3c.dom.Element
import org.w3c.dom.events.Event

@native
object olx {
   interface MapOptions {
        var controls: Any?  get() = noImpl; set(value) = noImpl
        var layers: Array<Any>?  get() = noImpl; set(value) = noImpl
        var target: Any?  get() = noImpl; set(value) = noImpl
        var view: ol.View?  get() = noImpl; set(value) = noImpl
    }
    interface OverlayOptions {
        var element: Element?  get() = noImpl; set(value) = noImpl
        var positioning: ol.OverlayPositioning?  get() = noImpl; set(value) = noImpl
    }
    interface ViewOptions {
        var zoom: Number?  get() = noImpl; set(value) = noImpl
    }
    interface Projection {
    }
    @native
    object control {
        interface DefaultsOptions {
            var attributionOptions: Any?  get() = noImpl; set(value) = noImpl
        }
    }
    // ...
```


Unless otherwise indicated, these slides are © 2013-2016 Pivotal Software, Inc. and licensed under a Creative Commons Attribution-NonCommercial license: http://creativecommons.org/licenses/by-nc/3.0/

Kotlin on client-side

- Auto-complete + compile time checks

```
package openlayers

object options : olx.MapOptions {

}

var map = ol.Map(options).
```

```
addControl(control: ol.control.Control)                    Unit
addInteraction(interaction: ol.interaction.Interaction)    Unit
addLayer(layer: ol.layer.Base)                             Unit
addOverlay(overlay: ol.Overlay)                            Unit
beforeRender(var_args: ol.PreRenderFunction)               Unit
forEachFeatureAtPixel(pixel: ol.Pixel, callback: (ol.Feat… Unit
forEachLayerAtPixel(pixel: ol.Pixel, callback: (ol.layer.… Unit
getControls()                  ol.Collection<ol.control.Control>
get(key: String)                                           Any
getCoordinateFromPixel(pixel: ol.Pixel)          ol.Coordinate
getEventCoordinate(event: [ERROR : Event])       ol.Coordinate
Did you know that Quick Documentation View (^J) works in completion lookups as well? >>
```

- Sharing code between client-side and server-side (domain model, validation)
- Source map

Unless otherwise indicated, these slides are © 2013-2016 Pivotal Software, Inc. and licensed under a Creative Commons Attribution-NonCommercial license: http://creativecommons.org/licenses/by-nc/3.0/

Kotlin 1.1 Javascript support

- TypeScript API definition -> Kotlin API definition
- Module support
- Better documentation
- Better IDE integration
- More ES6 support

Unless otherwise indicated, these slides are © 2013-2016 Pivotal Software, Inc. and licensed under a Creative Commons Attribution-NonCommercial license: http://creativecommons.org/licenses/by-nc/3.0/

Web Assembly?

- WebAssembly (WASM) = Web bytecode
- Supported by major browser vendors
- ES6 or TypeScript for the JavaScript ecosystem
- In 2017, JavaScript won't be the web bytecode anymore
- WASM is a huge opportunity for Kotlin
- Avoid the compiling to JS hack
- Faster, less memory consumption
- A new ecosystem to create

Unless otherwise indicated, these slides are © 2013-2016 Pivotal Software, Inc. and licensed under a Creative Commons Attribution-NonCommercial license: http://creativecommons.org/licenses/by-nc/3.0/

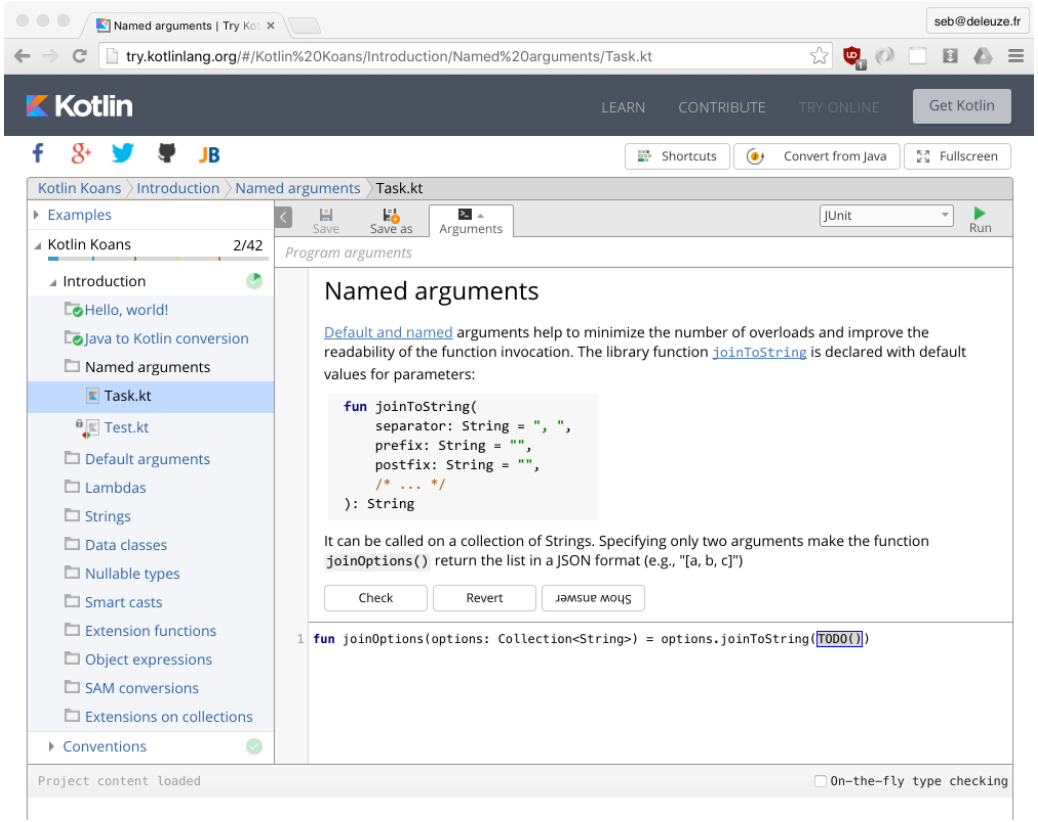
SpringOne Platform

Learn Kotlin

Pivotal

Unless otherwise indicated, these slides are © 2013-2016 Pivotal Software, Inc. and licensed under a Creative Commons Attribution-NonCommercial license: http://creativecommons.org/licenses/by-nc/3.0/

Kotlin Koans



try.kotlinlang.org

Unless otherwise indicated, these slides are © 2013-2016 Pivotal Software, Inc. and licensed under a Creative Commons Attribution-NonCommercial license: http://creativecommons.org/licenses/by-nc/3.0/

Community

Kotlin

The Kotlin community is growing day by day. Join in. Take part!

Forums

You can post technical questions on

- Kotlin forum
- Slack: *get invite here*
- StackOverflow. Make sure you tag questions with **kotlin**
- Reddit

Other mediums

- Twitter
- LinkedIn. Mostly for job related posts.
- Google+
- #kotlin channel on the freenode.net IRC network

kotlinlang.org/community.html

Unless otherwise indicated, these slides are © 2013-2016 Pivotal Software, Inc. and licensed under a Creative Commons Attribution-NonCommercial license: http://creativecommons.org/licenses/by-nc/3.0/

SpringOne Platform

What's next?

Pivotal

Unless otherwise indicated, these slides are © 2013-2016 Pivotal Software, Inc. and licensed under a Creative Commons Attribution-NonCommercial license: http://creativecommons.org/licenses/by-nc/3.0/

Kotlin support in Spring

- **spring-kotlin**: Kotlin extensions for Spring
- **reactor-kotlin**: Kotlin extensions for Reactor
- Kotlin nullable first class support in Spring Framework
- Improve Kotlin integration in Spring Boot

Unless otherwise indicated, these slides are © 2013-2016 Pivotal Software, Inc. and licensed under a Creative Commons Attribution-NonCommercial license: http://creativecommons.org/licenses/by-nc/3.0/

Kotlin 1.1 (M01 preview released)

- Coroutines
- Type aliases
- Java 8 bytecode generation
- Jigsaw support
- CGLIB style proxies without open
- Kotlin to Javascript compiler

Unless otherwise indicated, these slides are © 2013-2016 Pivotal Software, Inc. and licensed under a Creative Commons Attribution-NonCommercial license: http://creativecommons.org/licenses/by-nc/3.0/

Kotlin native?

- LLVM backend
- AOT compiler
- Lightweight runtime (think 128K RAM)
- Targets: embedded, IoT, iOS
- Bridge to Web Assembly?

Unless otherwise indicated, these slides are © 2013-2016 Pivotal Software, Inc. and licensed under a Creative Commons Attribution-NonCommercial license: http://creativecommons.org/licenses/by-nc/3.0/


The End