Next level Kotlin support in Spring Boot 4

Sébastien Deleuze
Spring Framework Core Committer
Tanzu Division, Broadcom
https://seb.deleuze.fr/

Introducing Kotlin support in Spring Framework 5.0

ENGINEERING | SÉBASTIEN DELEUZE | JANUARY 04, 2017 | 50 COMMENTS

Update: a comprehensive Spring Boot + Kotlin tutorial is now available.

Following the Kotlin support on start.spring.io we introduced a few months ago, we have continued to work to ensure that Spring and Kotlin play well together. One of the key strengths of Kotlin is that it provides a very good interoperability with libraries written in Java. But there are ways to go even further and allow writing fully idiomatic Kotlin code when developing your next Spring application. In addition to Spring Framework support for Java 8 that Kotlin applications can leverage like functional web or bean registration APIs, there are additional Kotlin dedicated features that should allow you to reach a new level of productivity.

That's why we are introducing a dedicated Kotlin support in Spring Framework 5.0, and I would like to summarize in this blog post the features that are designed to make your developer experience seamless when using these technologies together. You can use this link to find Kotlin related issues in Spring Framework bug tracker.

Today we are announcing a new major step

Strategic partnership between JetBrains and Spring

“For many years already, we have heavily embraced Kotlin as a first-class language for Spring applications. We are excited to turn the organic collaboration that has occurred until now into a strategic collaboration between JetBrains and the Spring team to make the Kotlin developer experience with Spring even better!”

Juergen Hoeller, Spring Framework Co-Founder and Project Lead.
Sébastien Deleuze, Core Committer and Kotlin Support Lead for Spring

New Spring major version to be released in November

Spring Framework 7

Spring Boot 4

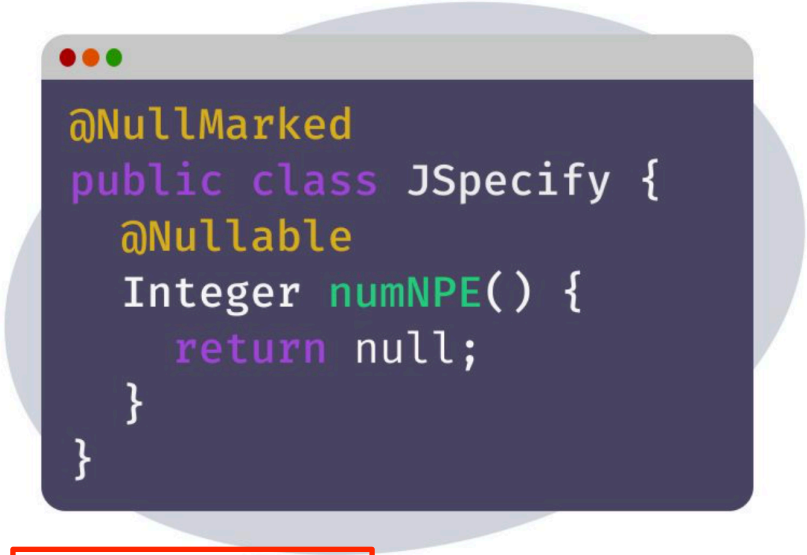
Kotlin 2.2+

Null Safety of Spring APIs

Sébastien Deleuze · You
Spring @ Broadcom
3mo ·

I have just merged into Spring Framework main branch the huge commit (3458 files changed) that migrates the codebase to JSpecify annotations. That will allow Spring Framework 7 and related portfolio projects to provide next-level null-safety support to avoid NullPointerException at runtime.

See related documentation at https://lnkd.in/d_uABV64.



Spencer Gibb and 418 others

39 comments · 24 reposts

Reactions

package-info.java

```java
/**
 * Spring's generic cache abstraction.
 * Concrete implementations are provided in the subpackages.
 */
@NullMarked
package org.springframework.cache;

import org.jspecify.annotations.NullMarked;
```

Cache.java

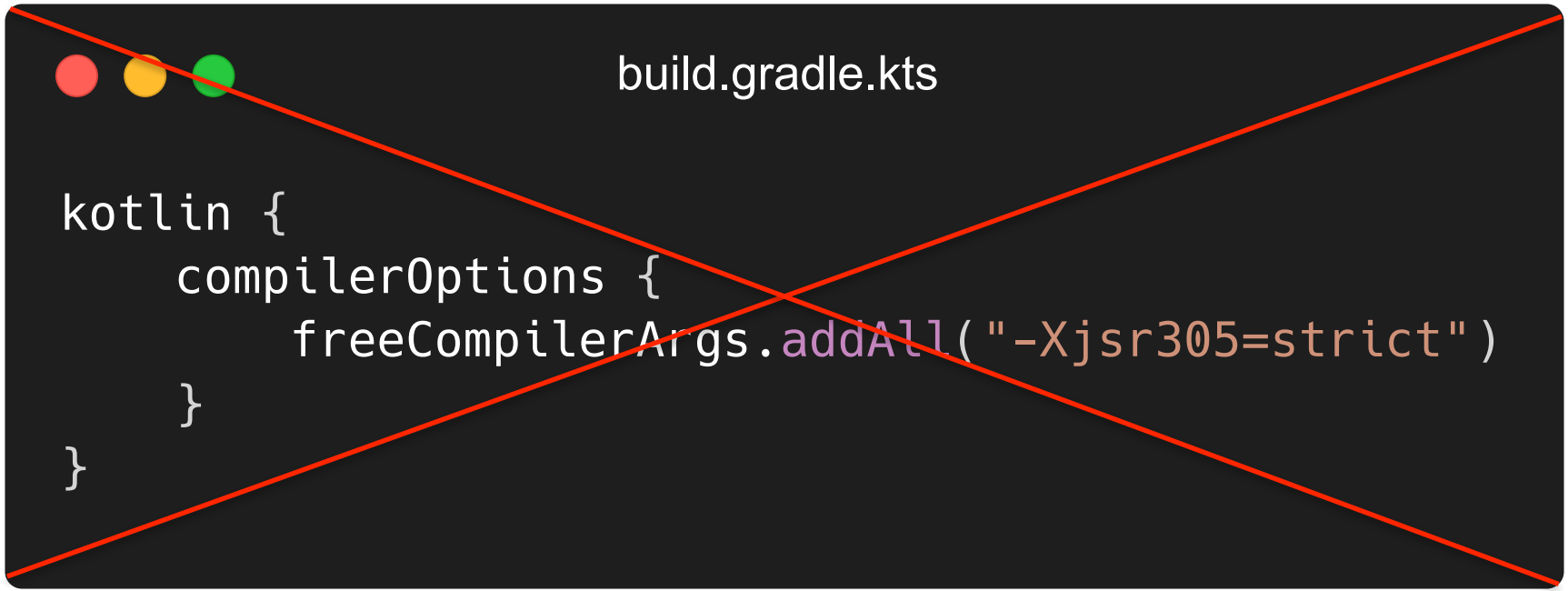
```java
package org.springframework.cache;

import org.jspecify.annotations.Nullable;
// ...

public interface Cache {

    String getName();

    @Nullable ValueWrapper get(Object key);

    <T> @Nullable T get(Object key, @Nullable Class<T> type);

    void put(Object key, @Nullable Object value);

    void evict(Object key);

    void clear();

    // ...
}
```

Cache from Kotlin without JSpecify

```
interface Cache {

    fun getName(): String!

    fun get(key: Any!): Cache.ValueWrapper!

    fun <T> get(key: Any!, type: Class<T!>!): T!

    fun put(key: Any!, value: Any!)

    fun evict(key: Any!)

    fun clear()

    // ...
}
```

Cache from Kotlin with JSpecify

```
interface Cache {

    fun getName(): String

    fun get(key: Any): Cache.ValueWrapper?

    fun <T> get(key: Any, type: Class<T>?): T?

    fun put(key: Any, value: Any?)

    fun evict(key: Any)

    fun clear()

    // ...
}
```


build.gradle.kts
kotlin {
compilerOptions {
freeCompilerArgs.addAll("-Xjsr305=strict")
}
}

Upcoming Null Safety portfolio-wide!

Spring Boot

Spring Framework

Spring Data

Spring Security

Spring AI

New Bean Registration DSL

Foundation for better programmatic bean definition

- ✅ Supersedes `BeanDefinitionDsl` and `beans { }`
- ✅ `BeanRegistrarDsl` extends Java `BeanRegistrar`
- ✅ `BeanRegistrar` allows to build custom configuration DSLs
- ✅ Full support for Spring AOT including with lambdas
- ✅ Seamless integration with Spring Boot and `@Configuration`
- ✅ Usable by libraries with automatic discoverability

BeanRegistrar.java

```java
package org.springframework.beans.factory;

import org.springframework.core.env.Environment;

@FunctionalInterface
public interface BeanRegistrar {

    /**
     * Register beans on the given {@link BeanRegistry} in a programmatic way.
     * @param registry the bean registry to operate on
     * @param env the environment that can be used to get the active profile or some properties
     */
    void register(BeanRegistry registry, Environment env);
}
```

Beans.java

```
class Beans implements BeanRegistrar {

    @Override
    public void register(BeanRegistry registry, Environment env) {
        registry.registerBean("foo", Foo.class);
        registry.registerBean("bar", Bar.class, spec -> spec
                .prototype()
                .lazyInit()
                .description("Custom description")
                .supplier(context -> new Bar(context.bean(Foo.class))));
        if (env.matchesProfiles("baz")) {
            registry.registerBean(Baz.class, spec -> spec
                    .supplier(context -> new Baz("Hello World!")));
        }
    }
}
```

Security.java

```
class Security extends SecurityRegistrar {

    @Override
    public void register(HttpSecurity registry, Environment env) {
        http.formLogin();
    }
}
```

SecurityRegistrar.java

```
// Could be provided by Spring Security
abstract class SecurityRegistrar implements BeanRegistrar {


    public abstract void register(HttpSecurity http, Environment env);

    @Override
    public void register(BeanRegistry registry, Environment env) {
        HttpSecurity http = new HttpSecurity();
        register(http, env);

        if (http.hasFormLogin()) {
            registry.registerBean(...);
        }
        // ...
    }
}
```

BeanRegistrarDsl.kt

```kotlin
package org.springframework.beans.factory

// ...

@BeanRegistrarDslMarker
open class BeanRegistrarDsl(private val init: BeanRegistrarDsl.() -> Unit): BeanRegistrar {

    // ...

}
```

Bean.kt

```
class Beans : BeanRegistrarDsl({
    registerBean<Foo>()
    registerBean(
        name = "bar",
        prototype = true,
        lazyInit = true,
        description = "Custom description") {
        Bar(bean<Foo>())
    }
    profile("baz") {
        registerBean { Baz("Hello World!") }
    }
    registerBean(::routerFunction)
})
```

Application.kt

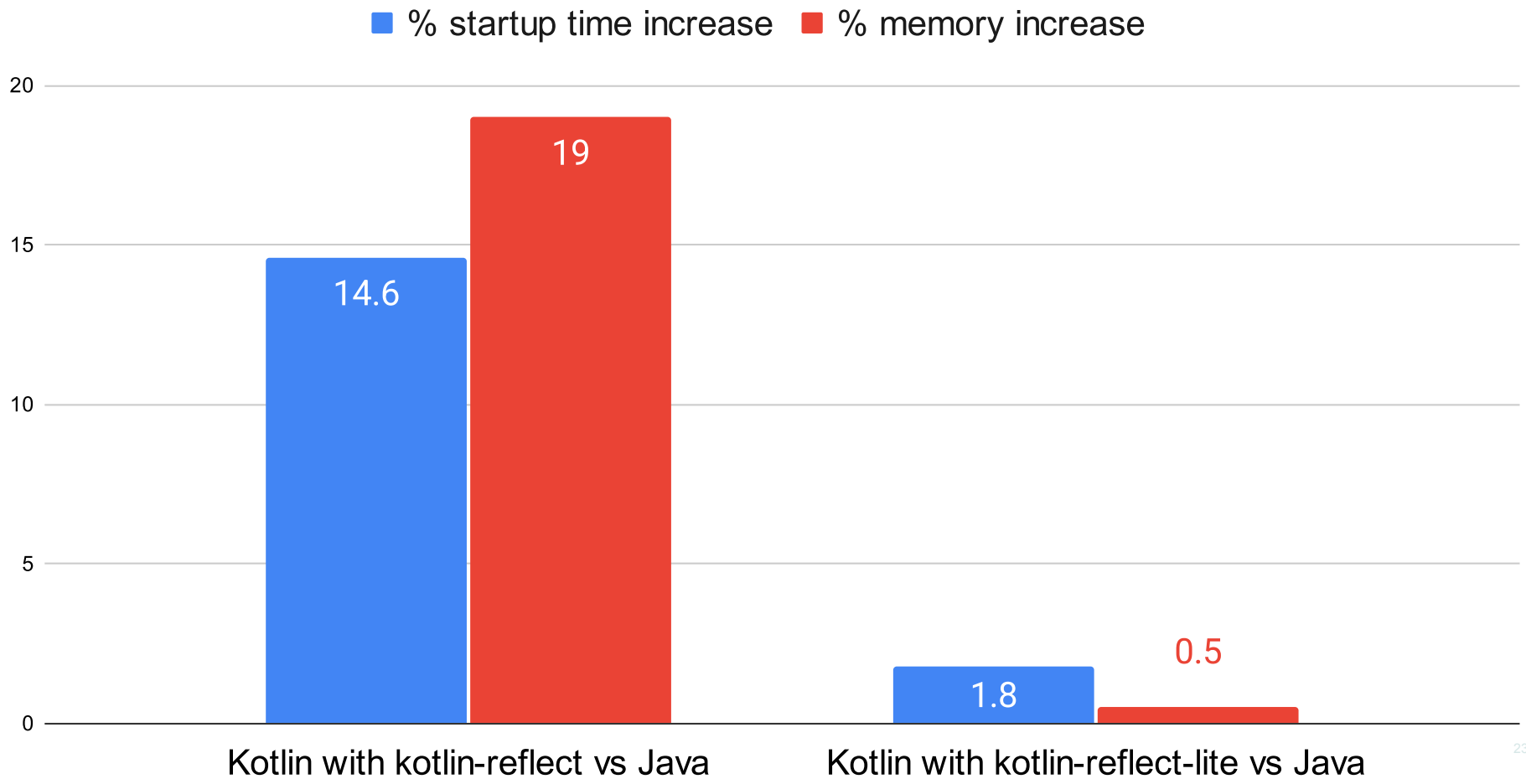
```
@EnableAutoConfiguration
@Import(Beans::class)
class Application

fun main(args: Array<String>) {
    runApplication<Application>(*args)
}
```

SampleConfiguration.kt

```kotlin
@Import(Beans::class)
@Configuration(proxyBeanMethods = false)
class SampleConfiguration {

    @Bean
    fun otherBean() = OtherBean()
}
```

More efficient Kotlin Reflection

kotlin-reflect impact on a simple Spring web app



Created by Alexander Udalov 3 months ago Updated by Alexander Udalov 21 days ago Visible to issue readers 3

Rewrite kotlin-reflect implementation to use kotlin-metadata-jvm instead of K1 compiler

We're going to rewrite the implementation of kotlin-reflect to avoid using obsolete K1 compiler code (descriptors) and to make it faster. The goal is to use kotlin-metadata-jvm to load information about Kotlin symbols, and Java reflection for Java symbols.

Current plan:

1. Bundle kotlin-metadata-jvm into kotlin-reflect (~~KT-75464~~)
2. Add `Km*` declarations to kotlin-reflect implementation by loading the parsed metadata from descriptors
3. Gradually rewrite implementations from descriptors to Km structures and/or Java reflection structures
4. Once everything is rewritten, remove dependency on the K1 compiler code, read and parse metadata via kotlin-metadata-jvm directly, and possibly unbundle kotlin-metadata-jvm to make it a proper dependency of kotlin-reflect

Relates to 3 **Parent for** 4

Relates to

- KT-32198 kotlin-reflect requires expensive first-time initialization to invoke property getter and setter
- KT-56358 KClasses.getMemberProperties takes too much time
- ~~KT-77183~~ Metadata: remove multi-field value class representation

Parent for

- KT-75505 Reflection: use kotlin-metadata-jvm to implement class visibility, modality, modifiers, etc
- KT-76521 Reflection: change KType representation to avoid dependency on K1
- ~~KT-75464~~ Bundle kotlin-metadata-jvm into kotlin-reflect
- ~~KT-76231~~ Dynamically added annotations are invisible to Kotlin Reflection

Project	Kotlin
Severity	Normal
Type	Meta Issue
Planned for	No planned for
Available in	No available in
State	Open
Assignee	Alexander Udalov
Subsystems	JVM. Reflection
Affected versions	No Affected versions

Created by Sébastien Deleuze 27 days ago Updated by Gavriil Maksyutenko 27 days ago Visible to issue readers

Provide a kotlin-reflect callable API avoiding boxing/unboxing in Java

Java frameworks like Spring have to recursively box parameters and unbox return values to support correctly inline value classes before invoking `kotlin.reflect.KCallable#callBy` and `kotlin.reflect.full.KCallables#callSuspendBy`. This is error prone and adds a performance overhead which defeats most of inline value classes purpose, as for this is a use case Kotlin with inline value classes should be faster than Java with regular classes but is in fact slower due to this additional processing.

Some would suggest Spring should invoke Kotlin functions with Java reflection, but this is what we did previously and that was a dead end as some Kotlin constructs like parameters with default values were not supported. Also we need Kotlin reflection to invoke suspending functions.

I understand that passing/returning boxed values makes sense from Kotlin perspective, but it would be extremely useful if kotlin-reflect could provide variants of those reflective function invocation capabilities that support unboxed parameters/return value for Java/Spring use case.

cc **@Alexander Udalov @Vsevolod Tolstopyatov**

Activity settings

Write a comment, @mention people

Project	Kotlin
Severity	Not specified
Type	Feature
Planned for	No planned for
Available in	No available in
State	Submitted
Assignee	Unassigned
Subsystems	JVM. Reflection
Affected versions	No Affected versions

Optimize your Spring Boot applications with CDS and AOT cache

Workflow

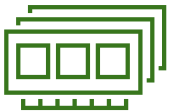
Training run

cache

Deployment run

Startup

Fast startup even on cheap servers or scaling to zero



Memory

Reduced memory consumption

Warmup

Peak performance available asap

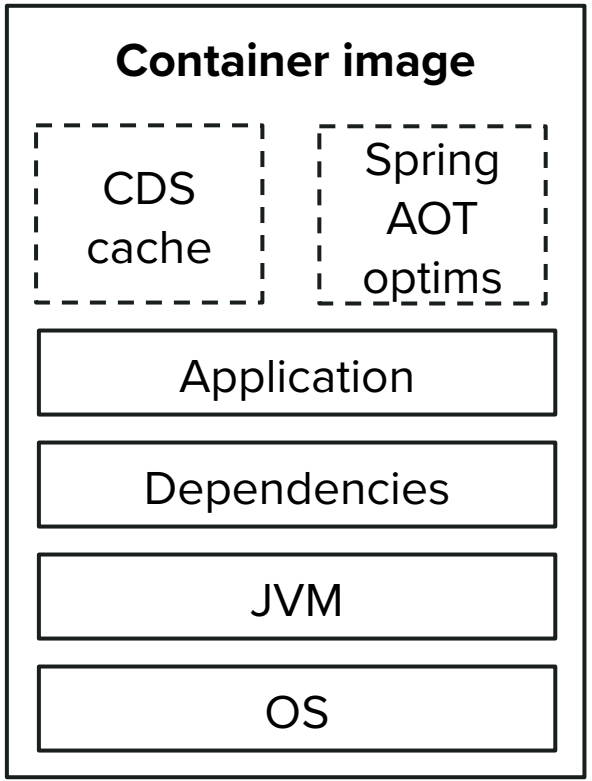
CDS support in Buildpacks

Spring Boot

\+

Buildpacks

=




```
mvn spring-boot:build-image

gradle bootBuildImage
```

The CDS training run is performed transparently when building the container image with the same JVM used at runtime!

JEP 483: Ahead-of-Time Class Loading & Linking

Authors	Ioi Lam, Dan Heidinga, & John Rose
Owner	Ioi Lam
Type	Feature
Scope	JDK
Status	Closed / Delivered
Release	24
Component	hotspot / runtime
Discussion	leyden dash dev at openjdk dot org
Reviewed by	Alex Buckley, Brian Goetz, Mark Reinhold, Vladimir Kozlov
Endorsed by	Vladimir Kozlov
Created	2023/09/06 04:07
Updated	2025/04/08 16:23
Issue	8315737

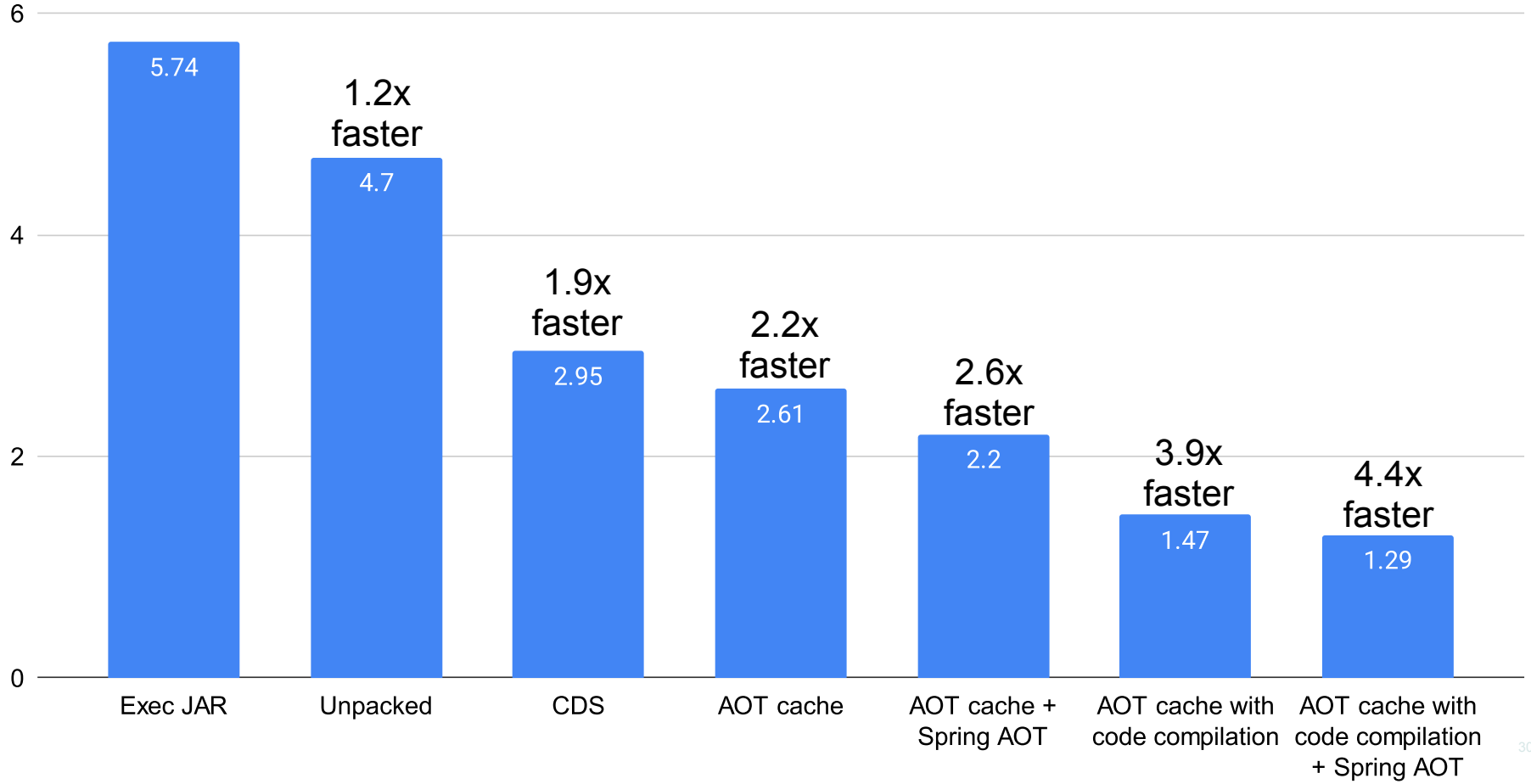
Summary

Improve startup time by making the classes of an application instantly available, in a loaded and linked state, when the HotSpot Java Virtual Machine starts. Achieve this by monitoring the application during one run and storing the loaded and linked forms of all classes in a cache for use in subsequent runs. Lay a foundation for future improvements to both startup and warmup time.

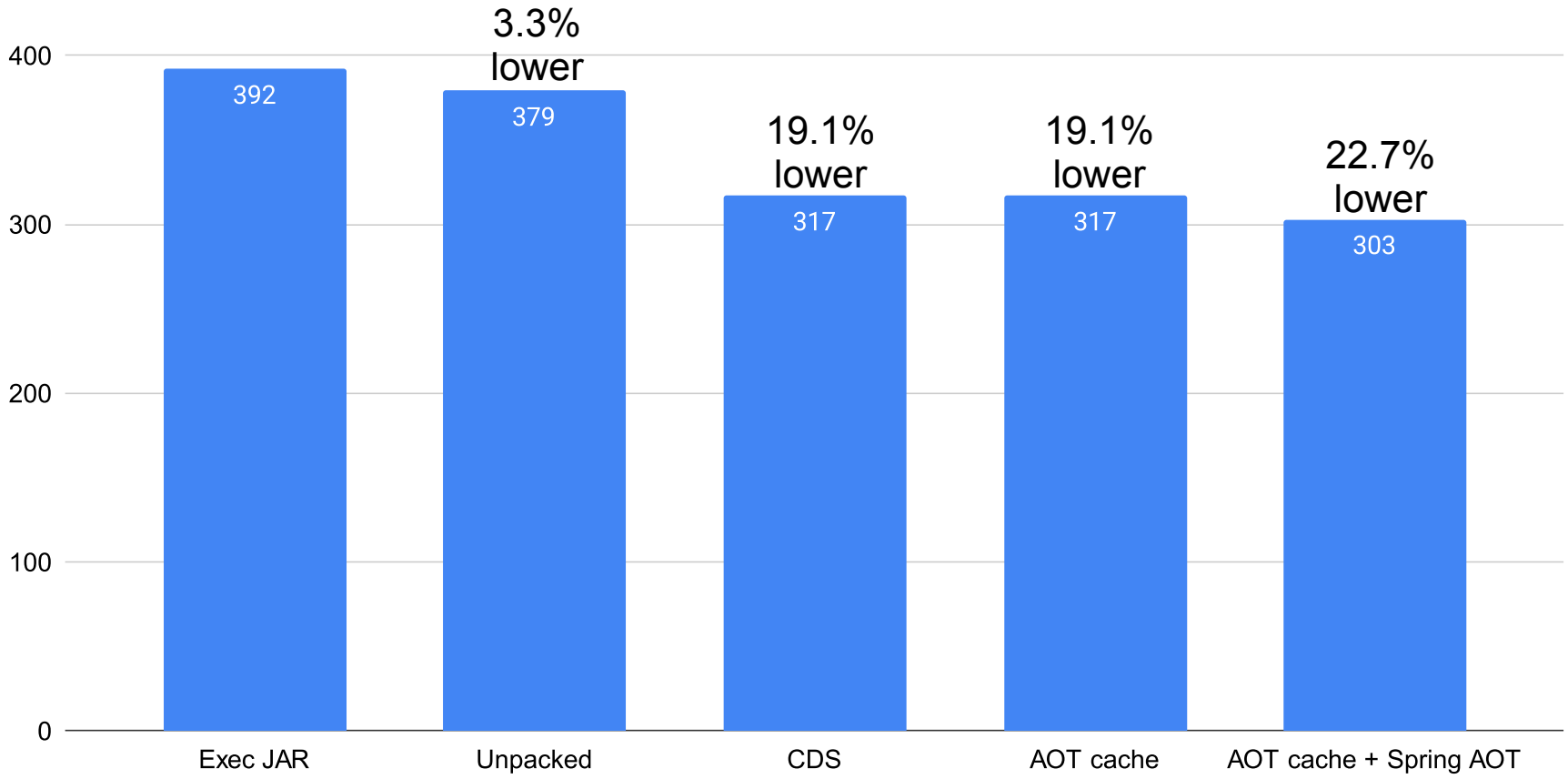
Goals

- Improve startup time by exploiting the fact that most applications start up in roughly the same way every time they run.

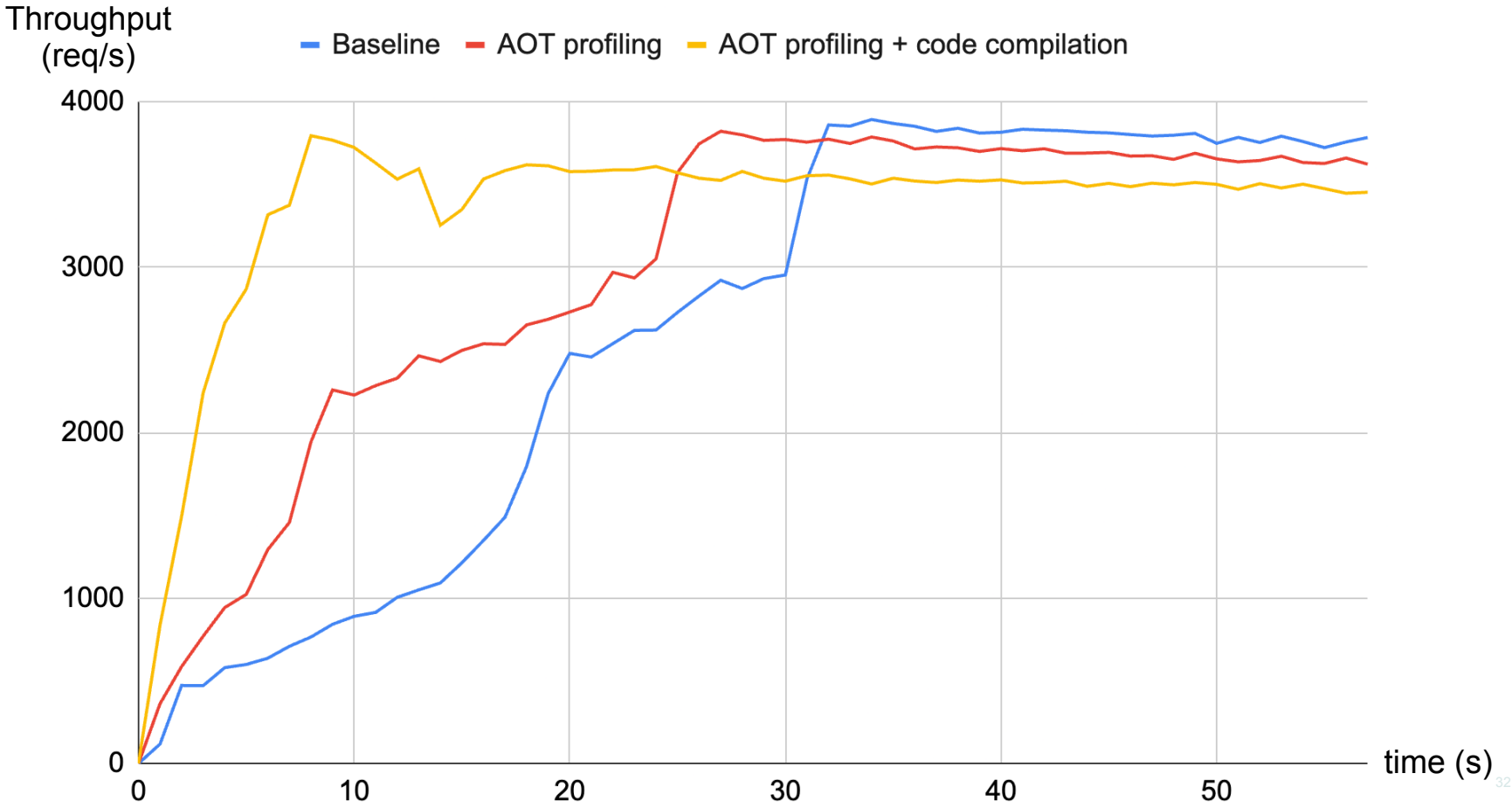
Spring Petclinic startup time (seconds)



Spring Petclinic RSS memory (MB)



Warmup with and without AOT cache



Spring AI Kotlin support

StructuredOutput.kt

```kotlin
fun structuredOutput(chatModel: OpenAiChatModel): MathReasoning {
    val outputConverter = BeanOutputConverter(MathReasoning::class.java)
    val jsonSchema = outputConverter.jsonSchema
    val prompt = Prompt(
        "how can I solve 8x + 7 = -23",
        OpenAiChatOptions.builder()
            .model(OpenAiApi.ChatModel.GPT_4_O_MINI)
            .responseFormat(ResponseFormat(ResponseFormat.Type.JSON_SCHEMA, jsonSchema))
            .build()
    )
    val response = chatModel.call(prompt)
    val content = response.result.output.text!!
    return outputConverter.convert(content)!!
}
```

MathReasoning.java

```
record MathReasoning(
    @JsonProperty(required = true) Steps steps,
    @JsonProperty(required = true, value = "final_answer") String finalAnswer) {

    record Steps(
        @JsonProperty(required = true) Items[] items) {

        record Items(
            @JsonProperty(required = true) String explanation,
            @JsonProperty(required = true) String output) {
        }
    }
}
```

MathReasoning.kt

```kotlin
// Kotlin 1.x
data class MathReasoning(
    val steps: Steps,
    @get:JsonProperty(value = "final_answer") val finalAnswer: String) {

    data class Steps(val items: Array<Items>) {

        data class Items(
            val explanation: String,
            val output: String)
    }
}
```

MathReasoning.kt

```kotlin
// Kotlin 2.2+ with -Xannotation-default-target=param-property
data class MathReasoning(
    val steps: Steps,
    @JsonProperty(value = "final_answer") val finalAnswer: String) {

    data class Steps(val items: Array<Items>) {

        data class Items(
            val explanation: String,
            val output: String)
    }
}
```

Build a Kotlin app that uses Spring AI to answer questions based on documents stored in Qdrant — tutorial

Edit page Last modified: 20 May 2025

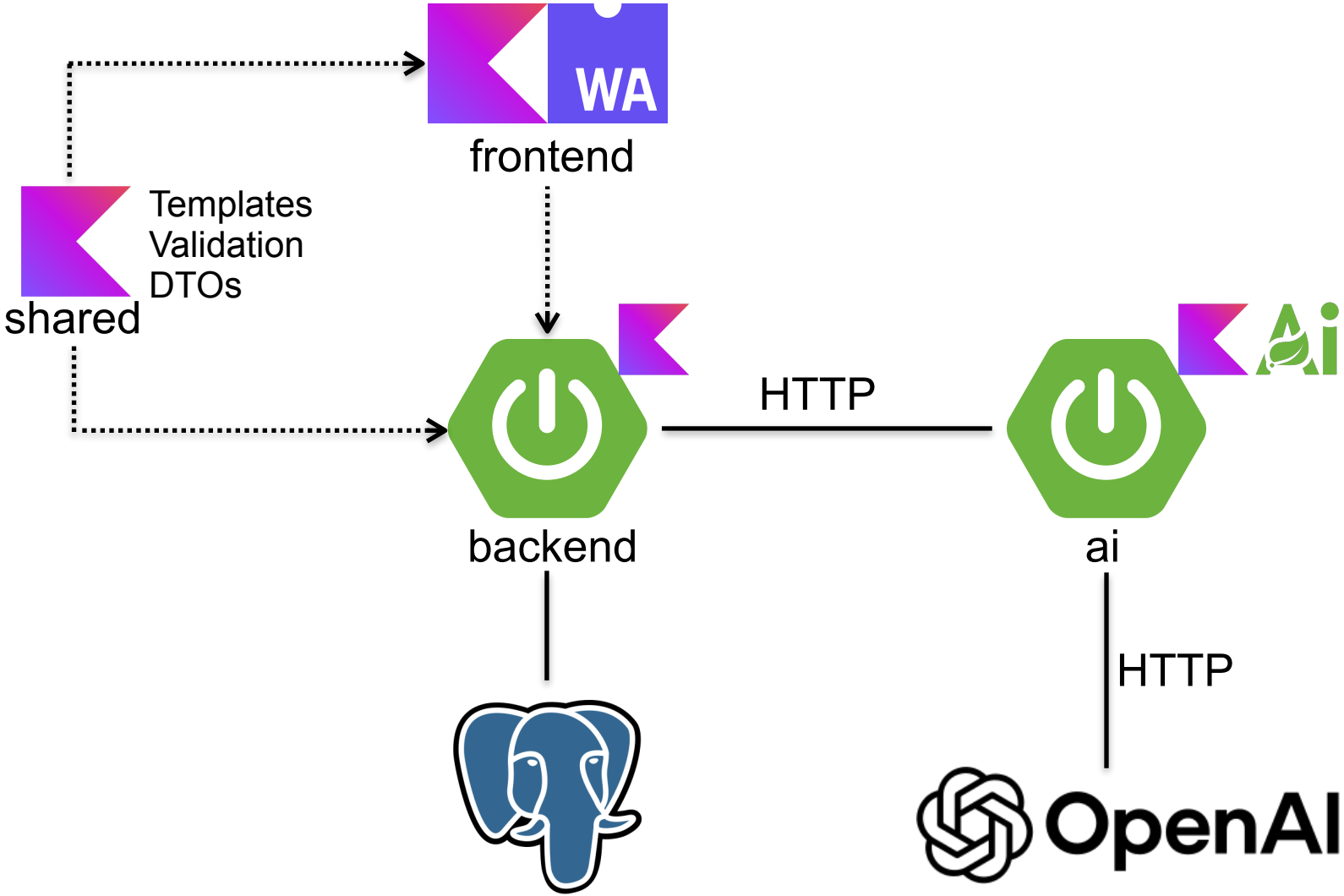
In this tutorial, you'll learn how to build a Kotlin app that uses Spring AI ↗ to connect to an LLM, store documents in a vector database, and answer questions using context from those documents.

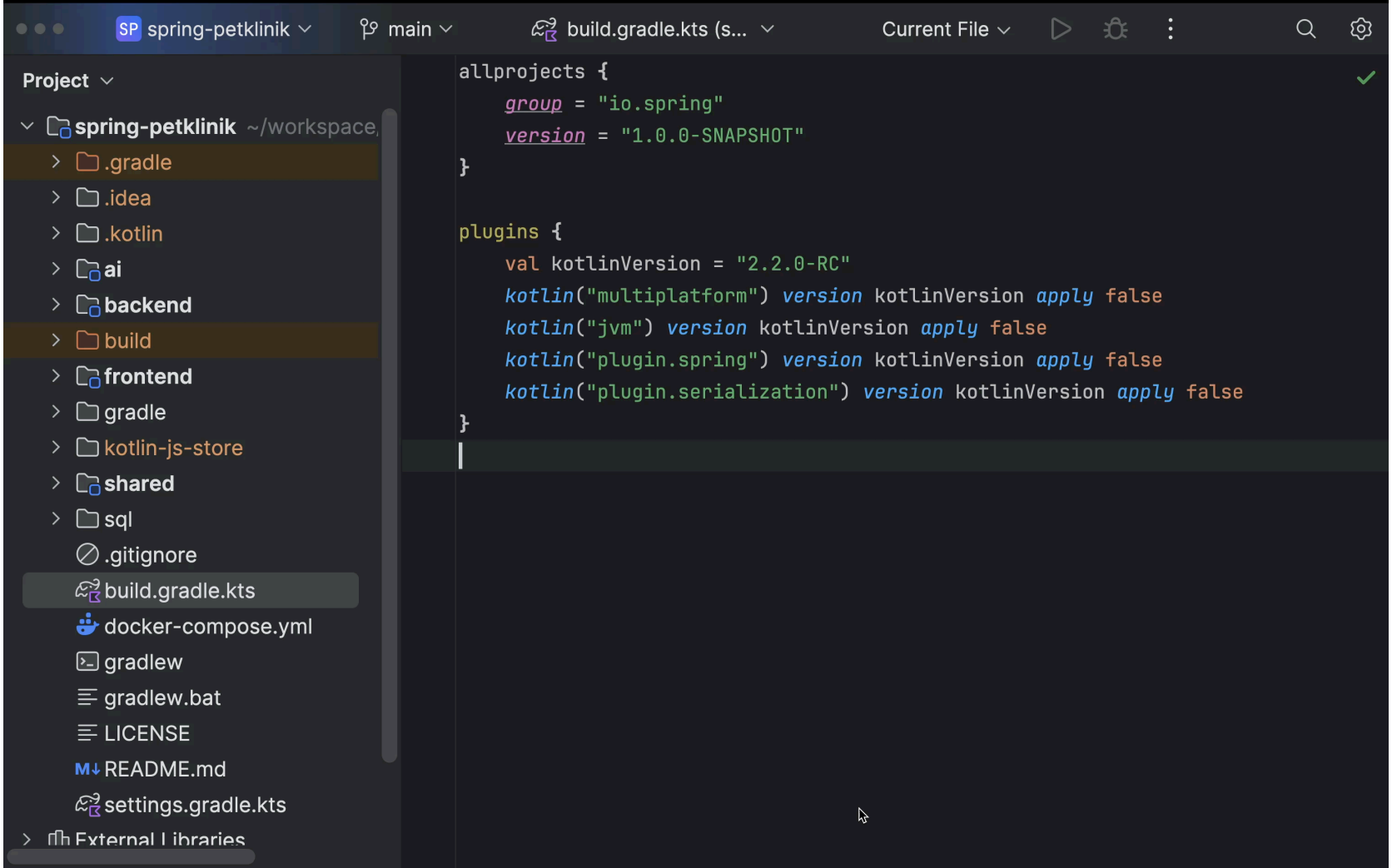
You will use the following tools during this tutorial:

- Spring Boot ↗ as the base to configure and run the web application.
- Spring AI ↗ to interact with the LLM and perform context-based retrieval.
- IntelliJ IDEA ↗ to generate the project and implement the application logic.
- Qdrant ↗ as the vector database for similarity search.
- Docker ↗ to run Qdrant locally.
- OpenAI ↗ as the LLM provider.

Introducing a new reference application

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


WA
frontend
Templates
Validation
DTOs
shared
HTTP
backend
ai
HTTP
OpenAI


spring-petklinik
main
build.gradle.kts (s...
Current File
Project
spring-petklinik ~/workspace
.gradle
.idea
.kotlin
ai
backend
build
frontend
gradle
kotlin-js-store
shared
sql
.gitignore
build.gradle.kts
docker-compose.yml
gradlew
gradlew.bat
LICENSE
README.md
settings.gradle.kts
External Libraries
allprojects {
group = "io.spring"
version = "1.0.0-SNAPSHOT"
}
plugins {
val kotlinVersion = "2.2.0-RC"
kotlin("multiplatform") version kotlinVersion apply false
kotlin("jvm") version kotlinVersion apply false
kotlin("plugin.spring") version kotlinVersion apply false
kotlin("plugin.serialization") version kotlinVersion apply false
}

Backend with Spring Boot 4 and shared code with Kotlin Multiplatform

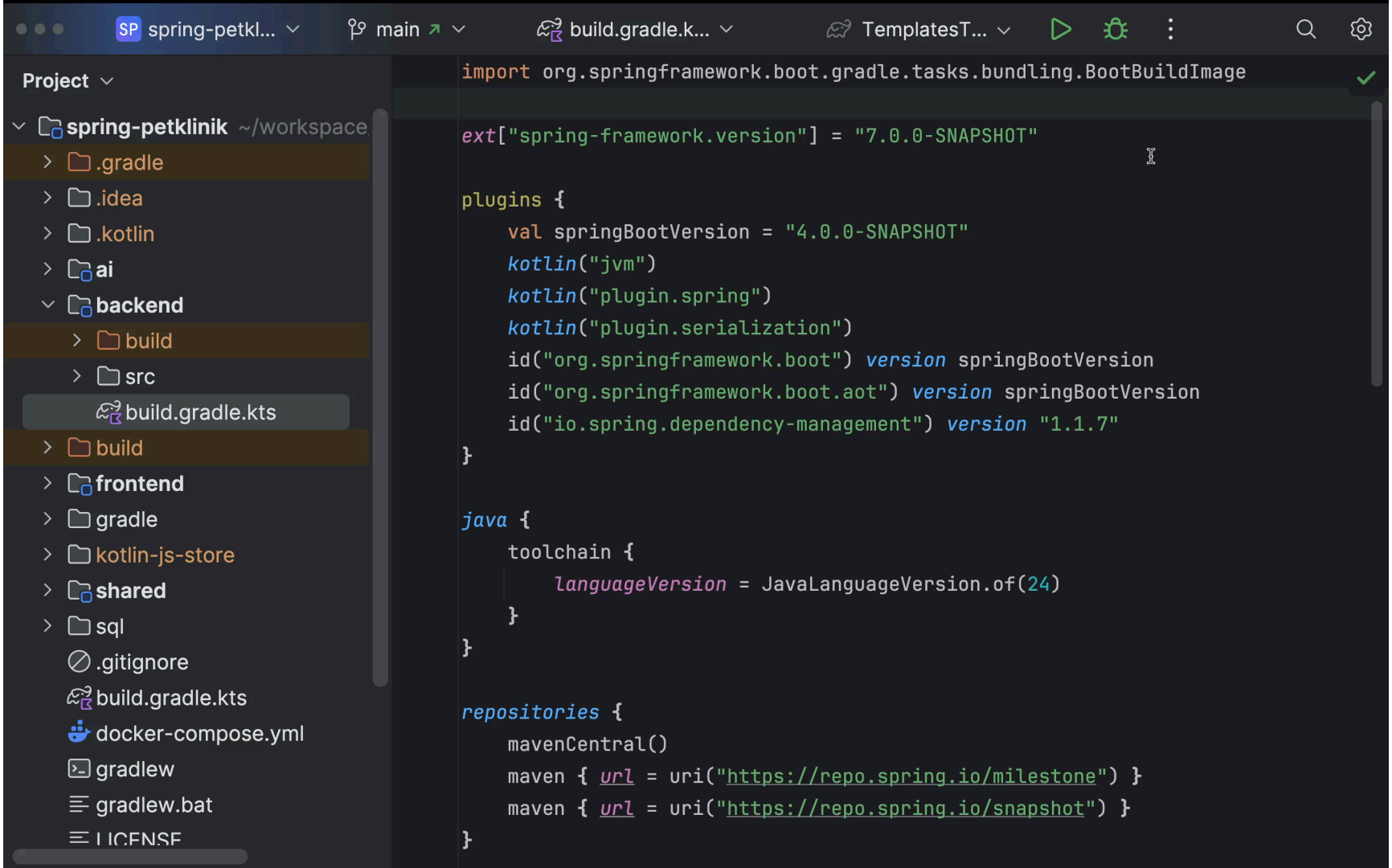
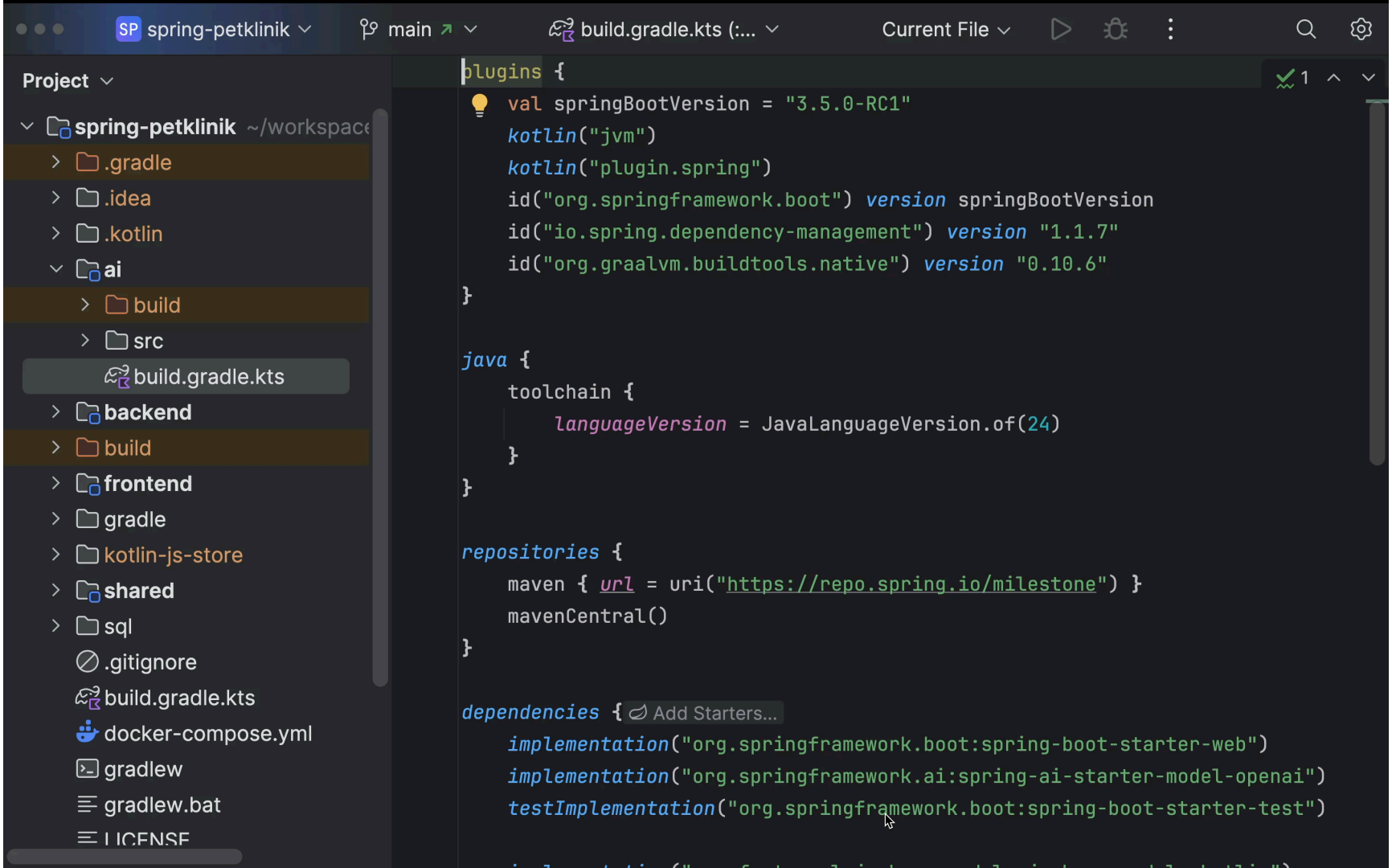
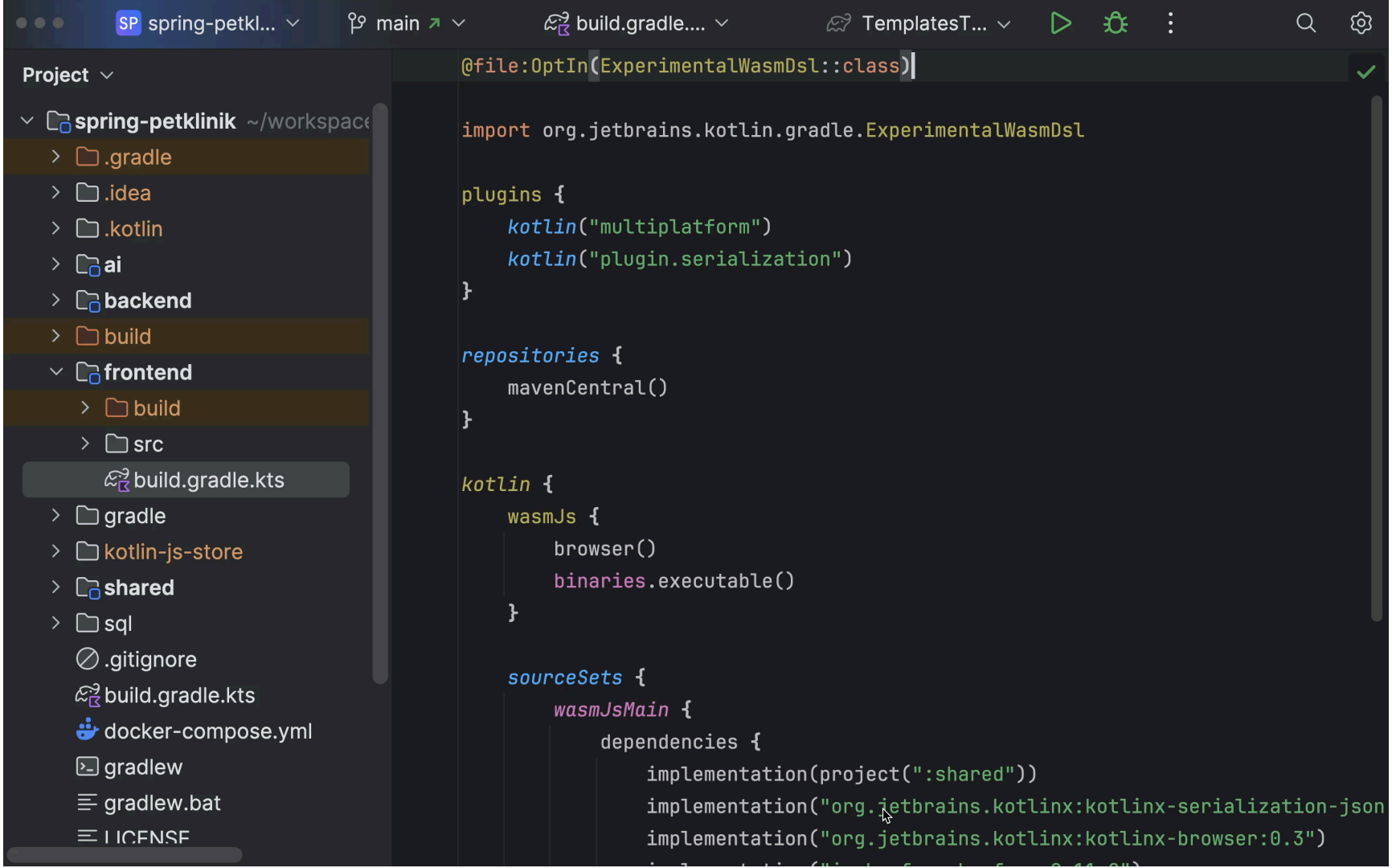

spring-petkl...
main
build.gradle.k...
TemplatesT...
Project
spring-petklinik ~/workspace
.gradle
.idea
.kotlin
ai
backend
build
src
build.gradle.kts
build
frontend
gradle
kotlin-js-store
shared
sql
.gitignore
build.gradle.kts
docker-compose.yml
gradlew
gradlew.bat
LICENSE
import org.springframework.boot.gradle.tasks.bundling.BootBuildImage
ext["spring-framework.version"] = "7.0.0-SNAPSHOT"
plugins {
val springBootVersion = "4.0.0-SNAPSHOT"
kotlin("jvm")
kotlin("plugin.spring")
kotlin("plugin.serialization")
id("org.springframework.boot") version springBootVersion
id("org.springframework.boot.aot") version springBootVersion
id("io.spring.dependency-management") version "1.1.7"
}
java {
toolchain {
languageVersion = JavaLanguageVersion.of(24)
}
}
repositories {
mavenCentral()
maven { url = uri("https://repo.spring.io/milestone") }
maven { url = uri("https://repo.spring.io/snapshot") }
}

Image generator with Spring AI


spring-petklinik
main
build.gradle.kts (:...
Current File
Project
spring-petklinik ~/workspace
.gradle
.idea
.kotlin
ai
build
src
build.gradle.kts
backend
build
frontend
gradle
kotlin-js-store
shared
sql
.gitignore
build.gradle.kts
docker-compose.yml
gradlew
gradlew.bat
LICENSE
plugins {
val springBootVersion = "3.5.0-RC1"
kotlin("jvm")
kotlin("plugin.spring")
id("org.springframework.boot") version springBootVersion
id("io.spring.dependency-management") version "1.1.7"
id("org.graalvm.buildtools.native") version "0.10.6"
}
java {
toolchain {
languageVersion = JavaLanguageVersion.of(24)
}
}
repositories {
maven { url = uri("https://repo.spring.io/milestone") }
mavenCentral()
}
dependencies { Add Starters...
implementation("org.springframework.boot:spring-boot-starter-web")
implementation("org.springframework.ai:spring-ai-starter-model-openai")
testImplementation("org.springframework.boot:spring-boot-starter-test")

Frontend with Kotlin/Wasm


spring-petkl...
main
build.gradle....
TemplatesT...
Project
spring-petklinik ~/workspace
.gradle
.idea
.kotlin
ai
backend
build
frontend
build
src
build.gradle.kts
gradle
kotlin-js-store
shared
sql
.gitignore
build.gradle.kts
docker-compose.yml
gradlew
gradlew.bat
LICENSE
@file:OptIn(ExperimentalWasmDsl::class)
import org.jetbrains.kotlin.gradle.ExperimentalWasmDsl
plugins {
kotlin("multiplatform")
kotlin("plugin.serialization")
}
repositories {
mavenCentral()
}
kotlin {
wasmJs {
browser()
binaries.executable()
}
sourceSets {
wasmJsMain {
dependencies {
implementation(project(":shared"))
implementation("org.jetbrains.kotlinx:kotlinx-serialization-json
implementation("org.jetbrains.kotlinx:kotlinx-browser:0.3")

Call for action: first class Compose HTML support

Kotlin/Wasm target

Client or Server rendering

Footprint optimization

Hot reloading

ComposeHTML.kt

```
fun main() {
    renderComposable(rootElementId = "root") {
        Button(
            attrs = {
                onClick { e ->
                    println("button clicked at ${e.movementX},${e.movementY}")
                }
            }
        ) {
            Text("Button")
        }
        val text = remember { mutableStateOf("") }
        TextArea(
            value = text.value,
            attrs = {
                onInput {
                    text.value = it.value
                }
            }
        )
        Span {
            Text("Typed text = ${text.value}")
        }
    }
}
```

Spring Boot 4.0
to be released in November

Thank you

https://seb.deleuze.fr/
https://github.com/sdeleuze/spring-petklinik

BROADCOM