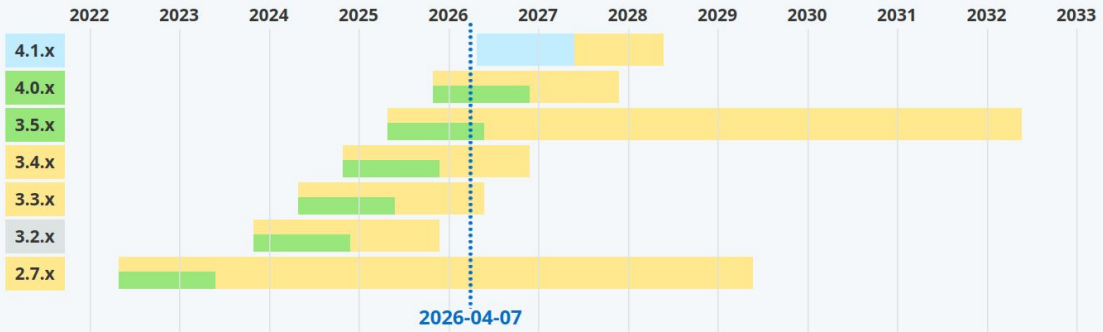
Inside Spring Boot 4: Restructuring for the Future

Moritz Halbritter
Spring Engineering Team @ Broadcom
2026-04-15

Branch	Initial Release	End of OSS Support	End Enterprise Support *
4.1.x	2026-05	2027-06	2028-06
4.0.x	2025-11	2026-12	2027-12
3.5.x	2025-05	2026-06	2032-06
3.4.x	2024-11	2025-12	2026-12
3.3.x	2024-05	2025-06	2026-06
3.2.x	2023-11	2024-12	2025-12
2.7.x	2022-05	2023-06	2029-06

More



spring

Requirements

Requirements

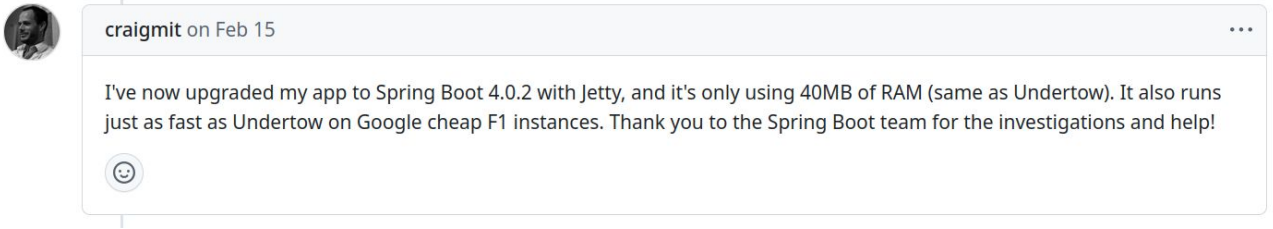
- Baseline: Java 17 (unchanged), Java 25 recommended

- Spring Framework 7.0
  - Jakarta EE 11
  - Servlet 6.1
  - Netty 4.2
  - JPA 3.2 (Hibernate 7.1 / 7.2)
  - Bean Validation 3.1 (Hibernate Validator 9.0)
  - JUnit 6
  - Kotlin 2.2
  - GraalVM 25

Removed Features

Undertow support is gone

- Undertow has no Servlet 6.1 compatible release yet
- We removed support for Undertow
- Migration path: Use Tomcat or Jetty



spring

Removed Features

- Pulsar Reactive
- Embedded Executable Uber Jar Launch Scripts
- Spock
  - Welcome back in 4.1!
- Transferred
  - Spring Session Hazelcast
  - Spring Session MongoDB

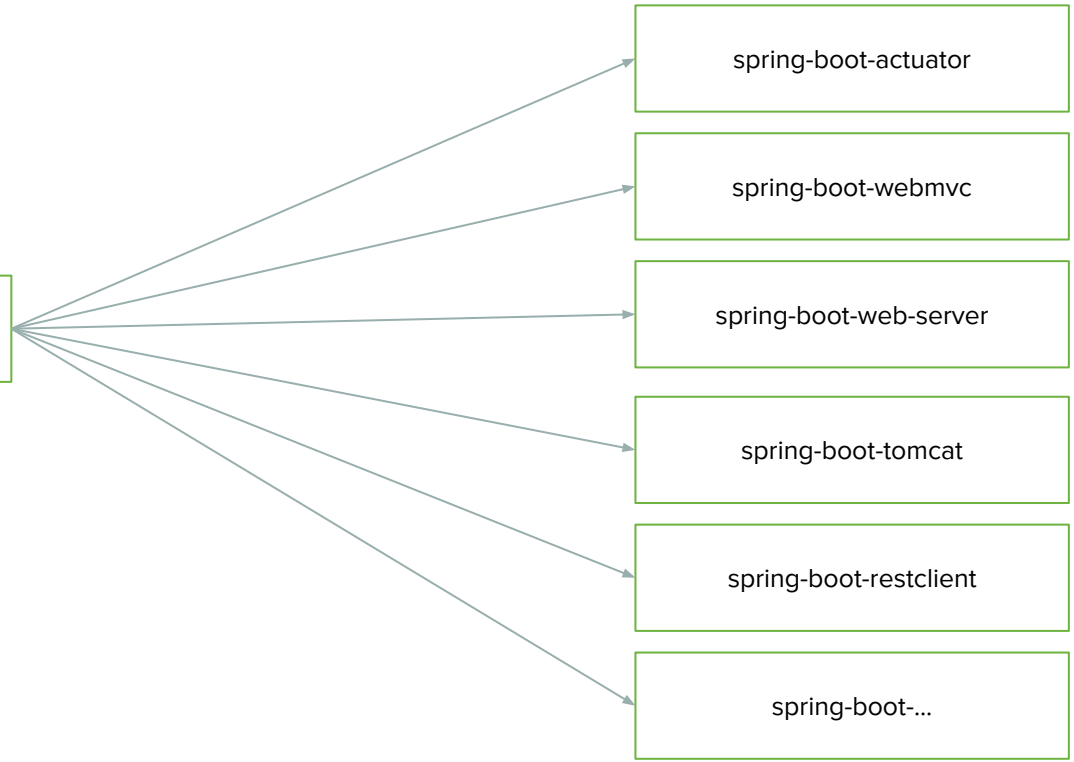
Modularization

Spring Boot 3.x

spring-boot-autoconfigure

Contains all auto-configurations
for all supported libraries

Spring Boot 4.x



One auto-configuration library for
every supported technology

And we also split up spring-boot-test-autoconfigure

Why?

- Better signal for auto-configuration, e.g.:
  - having webflux on the classpath just for webclient doesn't configure a reactive web server
- New features
  - E.g. metrics and tracing without the need for spring-boot-actuator
- Better developer experience
  - Less classes and properties in auto-complete
- Slightly smaller JARs for you
  - Only include what's needed

PetClinic 3.5.13 (65.2 MiB)

```
2,1M spring-boot-autoconfigure-3.5.13.jar
1,9M spring-boot-3.5.13.jar
842k spring-boot-actuator-autoconfigure-3.5.13.jar
707k spring-boot-actuator-3.5.13.jar
 53k spring-boot-jarmode-tools-3.5.13.jar
```

PetClinic 4.0.5 (66.6 MiB)

```
1,4M spring-boot-4.0.5.jar
371k spring-boot-autoconfigure-4.0.5.jar
357k spring-boot-actuator-4.0.5.jar
241k spring-boot-micrometer-metrics-4.0.5.jar
201k spring-boot-jdbc-4.0.5.jar
174k spring-boot-webmvc-4.0.5.jar
169k spring-boot-web-server-4.0.5.jar
154k spring-boot-health-4.0.5.jar
141k spring-boot-actuator-autoconfigure-4.0.5.jar
124k spring-boot-tomcat-4.0.5.jar
 84k spring-boot-cache-4.0.5.jar
 68k spring-boot-jackson-4.0.5.jar
 54k spring-boot-jarmode-tools-4.0.5.jar
 54k spring-boot-http-converter-4.0.5.jar
 47k spring-boot-servlet-4.0.5.jar
 36k spring-boot-sql-4.0.5.jar
 31k spring-boot-thymeleaf-4.0.5.jar
 30k spring-boot-data-commons-4.0.5.jar
 29k spring-boot-hibernate-4.0.5.jar
 29k spring-boot-micrometer-observation-4.0.5.jar
 24k spring-boot-jpa-4.0.5.jar
 21k spring-boot-transaction-4.0.5.jar
 16k spring-boot-data-jpa-4.0.5.jar
 16k spring-boot-persistence-4.0.5.jar
 15k spring-boot-validation-4.0.5.jar
```

Why?

- Improved maintainability of the Spring Boot codebase
  - More isolated changes
  - Architecture validated through build system
  - Faster build

Faster builds

Before the modularization:

```
BUILD SUCCESSFUL in 7m 39s
3625 actionable tasks: 1755 executed, 1870 up-to-date
```

After the modularization:

```
BUILD SUCCESSFUL in 33s
8263 actionable tasks: 425 executed, 7838 up-to-date
```

Improved starters arrangement

- Every technology now has their own starter
  - `spring-boot-starter-<technology>`
  - Package `org.springframework.boot.<technology>`
- And every starter has a test starter
  - `spring-boot-starter-<technology>-test`
  - Package `org.springframework.boot.<technology>.test`
  - Contains code useful for testing, e.g. `@WebMvcTest`
  - Even if that test starter is “empty”
  - You can remove your dependency to `spring-boot-starter-test`

Some examples

- WebMVC
  - `spring-boot-starter-webmvc`
  - `spring-boot-starter-webmvc-test`
- Webflux
  - `spring-boot-starter-webflux`
  - `spring-boot-starter-webflux-test`
- Security
  - `spring-boot-starter-security`
  - `spring-boot-starter-security-test`

All technologies now have a starter

- Some dependencies didn’t have a starter in 3.5.x
  - E.g. Liquibase, Flyway, Kafka, Zipkin, LDAP, gson, Hazelcast, OpenTelemetry, …
- Now they have:
  - E.g. `spring-boot-starter-flyway`, `spring-boot-starter-kafka`, …
- The full list can be found here
- start.spring.io is aware of that

Library Authors

- Do not try to support Spring Boot 3 and Spring Boot 4 with the same artifact
- We moved internal classes too, e.g.:
  - `BootstrapRegistry` is now in the `org.springframework.boot.bootstrap` package
  - `EnvironmentPostProcessor` is now in the `org.springframework.boot` package
  - Don’t forget `spring.factories` keys!

Migration Guide

- Add `spring-boot-starter-classic` and `spring-boot-starter-test-classic`
- Get it to compile again by fixing all the imports
  - We changed the packages, but not the names
- Use `spring-boot-properties-migrator` to fix the properties
- Remove the `spring-boot-starter-classic` and `spring-boot-starter-test-classic`
- Add technology specific starters and test starters
  - Look at the imports if you're lost!
- `spring-boot-starter-aop` has been renamed to `spring-boot-starter-aspectj`

Deprecated Starter	Replacement
`spring-boot-starter-oauth2-authorization-server`	`spring-boot-starter-security-oauth2-authorization-server`
`spring-boot-starter-oauth2-client`	`spring-boot-starter-security-oauth2-client`
`spring-boot-starter-oauth2-resource-server`	`spring-boot-starter-security-oauth2-resource-server`
`spring-boot-starter-web`	`spring-boot-starter-webmvc`
`spring-boot-starter-web-services`	`spring-boot-starter-webservices`

spring

Beware: RestTemplate / RestClient / WebClient

- `RestTemplate` is now deprecated for removal, use `RestClient` instead

- `spring-boot-starter-webmvc` no longer pulls in auto-configuration for http clients
- If you need a `RestClient`, add:
  - `spring-boot-starter-restclient`
- If you need a reactive `WebClient`, add:
  - `spring-boot-starter-webclient`

Migrate Petclinic

Ways to migrate

- Enterprise customers: Use Spring Application Advisor
  - Uses custom OpenRewrite recipes
  - Seamless CI/CD integration
  - Built by Tanzu
- OpenRewrite Recipes
- Use your favorite LLM
  - There’s an MCP server coming for Spring Application Advisor, see Raquel’s talk
- By hand

Upgrade Spring Boot version

```
id 'org.springframework.boot' version '3.5.13'
```

```
id 'org.springframework.boot' version '4.0.5'
```

spring

Upgrade to Testcontainers 2

```
* What went wrong:
Could not determine the dependencies of task ':checkstyleAotTest'.
> Could not resolve all dependencies for configuration ':aotTestCompileClasspath'.
   > Could not find org.testcontainers:junit-jupiter:.
     Required by:
         root project :
   > Could not find org.testcontainers:mysql:.
     Required by:
         root project :
```

```
testImplementation 'org.testcontainers:junit-jupiter'
testImplementation 'org.testcontainers:mysql'
```



```
testImplementation 'org.testcontainers:testcontainers-junit-jupiter'
testImplementation 'org.testcontainers:testcontainers-mysql'
```

spring

Add starter-classic and starter-test-classic

```
implementation 'org.springframework.boot:spring-boot-starter-classic'

testImplementation 'org.springframework.boot:spring-boot-starter-test'
```



```
testImplementation 'org.springframework.boot:spring-boot-starter-test-classic'
```

spring

Fix the imports

```
import org.springframework.boot.test.autoconfigure.web.servlet.WebMvcTest;

import org.springframework.boot.webmvc.test.autoconfigure.WebMvcTest;
```


spring

1000 imports later ...

Where's TestRestTemplate?!

```
private TestRestTemplate rest;
        ^
symbol:   class TestRestTemplate
location: class CrashControllerIntegrationTests
```

Bug #50006 in `spring-boot-starter-test-classic`, fixed in 4.0.6

`TestRestTemplate` can be pulled in with `spring-boot-starter-webmvc-test`

```
testImplementation 'org.springframework.boot:spring-boot-starter-webmvc-test'
```

Check configuration properties

At this point, we can do `./gradlew bootRun` (but the tests are failing)

```
runtimeOnly("org.springframework.boot:spring-boot-properties-migrator")
```

And do a `./gradlew bootRun`, watching for log messages

Doesn't detect any deprecated properties, so remove it again

Run the tests

```
No qualifying bean of type 'org.springframework.boot.resttestclient.TestRestTemplate'
org.springframework.beans.factory.NoSuchBeanDefinitionException Create breakpoint : No qua
```

```
@AutoConfigureTestRestTemplate
class CrashControllerIntegrationTests {

    java.lang.AssertionError:
    Expecting actual:
      "<!DOCTYPE html>
```

Fix configuration properties in tests

```
@SpringBootTest(webEnvironment = RANDOM_PORT,
        properties = { "server.error.include-message=ALWAYS", "management.endpoints.enabled-by-default=false" })
```

spring

Tests are now green

```
@SpringBootTest(webEnvironment = RANDOM_PORT,
        properties = { "spring.web.error.include-message=ALWAYS", "management.endpoints.access.default=none" })
@AutoConfigureTestRestTemplate
class CrashControllerIntegrationTests {
```

Now all the tests pass and we can run the application!

Clean up: Remove `spring-boot-starter-classic` and `spring-boot-starter-test-classic`

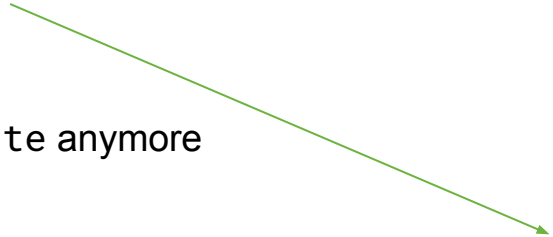
```
: error: package org.springframework.boot.data.jpa.test.autoconfigure does not exist
```

Add test starters for every starter you have

```
testImplementation 'org.springframework.boot:spring-boot-starter-cache-test'
testImplementation 'org.springframework.boot:spring-boot-starter-data-jpa-test'
testImplementation 'org.springframework.boot:spring-boot-starter-thymeleaf-test'
testImplementation 'org.springframework.boot:spring-boot-starter-validation-test'
testImplementation 'org.springframework.boot:spring-boot-starter-actuator-test'
testImplementation 'org.springframework.boot:spring-boot-starter-webmvc-test'
```

RestTemplateBuilder is missing

```
error: package org.springframework.boot.restclient does not exist
```



`spring-boot-starter-web` doesn't include `RestTemplate` anymore

```
implementation 'org.springframework.boot:spring-boot-starter-restclient'

testImplementation 'org.springframework.boot:spring-boot-starter-restclient-test'
```

spring

Build successful

```
BUILD SUCCESSFUL in 40s
21 actionable tasks: 16 executed, 1 from cache, 4 up-to-date
```

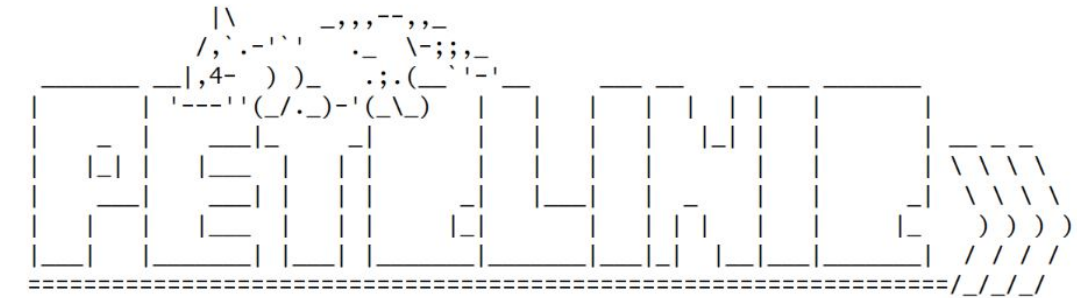
Replace deprecated starters

```
implementation 'org.springframework.boot:spring-boot-starter-web'
```



```
implementation 'org.springframework.boot:spring-boot-starter-webmvc'
```

spring



```
:: Built with Spring Boot :: 4.0.5
```

spring

Jackson 3

Spring support for Jackson 3

As of Spring Boot 4 and Spring Framework 7, the Spring portfolio is:

- Introducing Jackson 3 support
- Deprecating the Jackson 2 support
- Switching the default Jackson version enabled (classpath detection) to Jackson 3

Jackson 3

- New package `tools.jackson`
  - except for annotations
- Mutable `ObjectMapper` → Immutable `JsonMapper` with related builder
- Unchecked Exceptions
- Updated default settings
  - `SORT_PROPERTIES_ALPHABETICALLY` is true by default
  - `WRITE_DATES_AS_TIMESTAMPS` is false by default
  - FAIL_ON_TRAILING_TOKENS is true by default
- Auto-detection of modules
  - Control with `spring.jackson.find-and-add-modules` property
- More in the Jackson 3 Migration Guide

How to use Jackson 3

- Inject `JsonMapper` instead of `ObjectMapper`
- Use `JsonMapperBuilderCustomizer` to customize the `JsonMapper`
- Properties have moved:
  - `spring.jackson.json.read.*` and `spring.jackson.json.write.*`

How to use Jackson 2

- `spring.jackson.use-jackson2-defaults` property
  - Configures `JsonMapper` to behave like `ObjectMapper`
  - Works without Jackson 2 on the classpath
- Dependency management for Jackson 2 is still there
  - Jackson 2's `ObjectMapper` can be used alongside the new `JsonMapper`
- If you want Jackson 2 auto-configuration: use spring-boot-jackson2
  - **NOTE: This is deprecated and will go away!**
  - Configures an `ObjectMapper` bean
  - Use `spring.jackson2` properties to configure it
  - Use `Jackson2ObjectMapperBuilderCustomizer` for more customization
  - Use properties to indicate that you want to use Jackson 2:
    - Web MVC: `spring.http.converters.preferred-json-mapper`
    - Webflux: `spring.http.codecs.preferred-json-mapper`
    - GraphQL: `spring.graphql.rsocket.preferred-json-mapper`
    - RSocket: `spring.rsocket.preferred-mapper`
    - Websocket: `spring.websocket.messaging.preferred-json-mapper`

Spring Retry

> ⚠️ **Project Status: Maintenance Only**
> This project has been superseded by **Spring Framework 7** and is no longer accepting enhancements.
> Please migrate to Spring Framework's resilience features.

```
@Configuration(proxyBeanMethods = false)
@EnableResilientMethods
class RetryConfiguration {

}
```

```
@Service
public class MyService {
    private final RestClient restClient;

    MyService(RestClient.Builder restClientBuilder) {
        this.restClient = restClientBuilder.build();
    }

    @Retryable(maxRetries = 5, timeoutString = "10s", delayString = "2s", jitterString = "200ms")
    public String fetchData() {
        return this.restClient
                .get().uri( uri: "http://some-flaky-ws/data")
                .retrieve().body(String.class);
    }
}
```

JSpecify

“I call it my billion-dollar mistake. It was the invention of the null reference in 1965.”

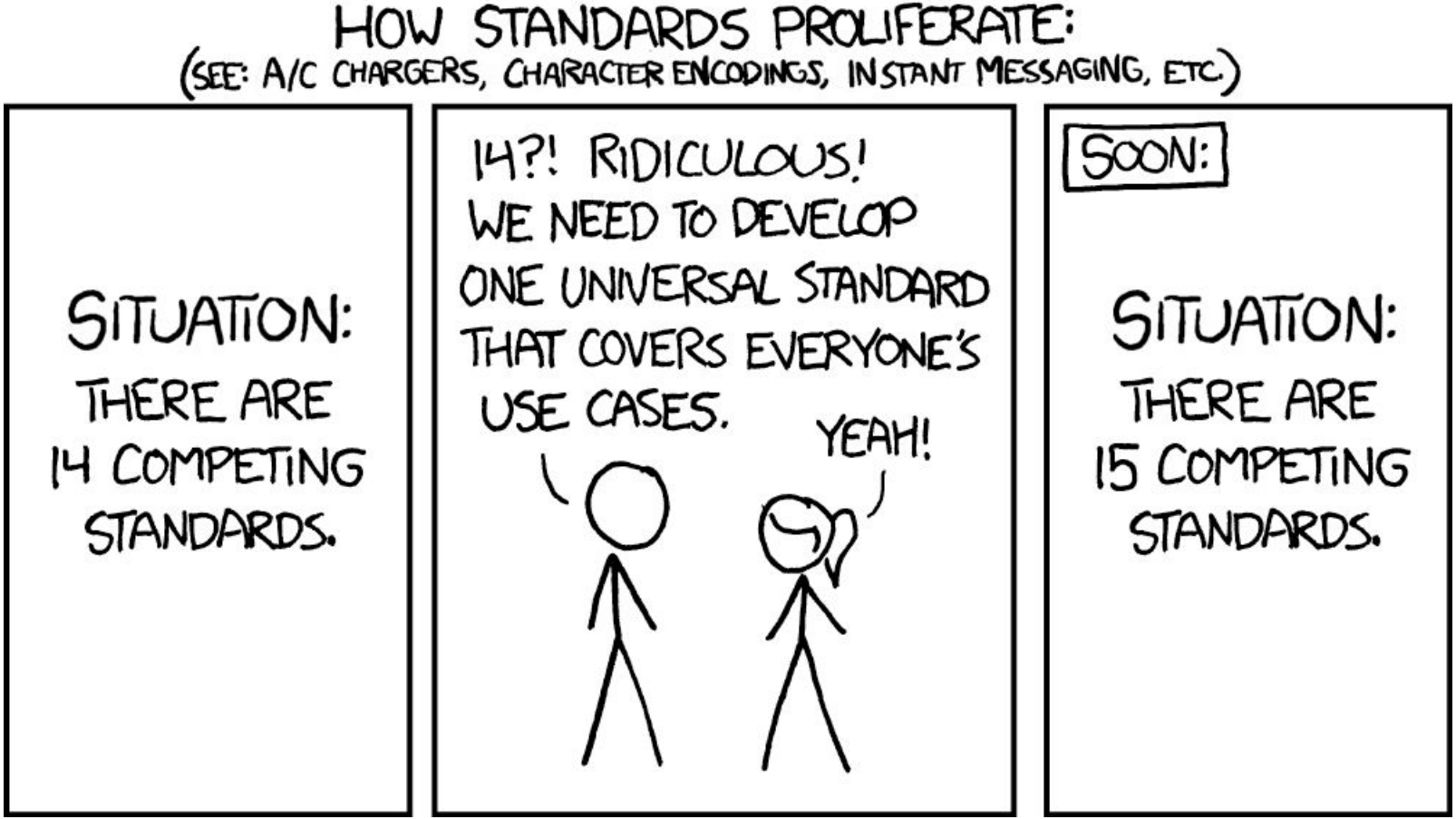
- *Tony Hoare, 2009*

JSpecify

- Spring Framework and various other Spring projects had `@Nullable` annotations
- This was `org.springframework.lang.Nullable`
- JSpecify is a industry collaboration to create a(nother) standard for nullness

Organization	Projects
EISOP Team	EISOP
Google	Android, Error Prone, Guava
JetBrains	Kotlin, IntelliJ IDEA
Meta	Infer
Microsoft	Azure SDK for Java
Oracle	OpenJDK
PMD Team	PMD
Sonar	SonarQube, SonarCloud, SonarLint
Square	(various)
Uber	NullAway
Broadcom	Spring

spring



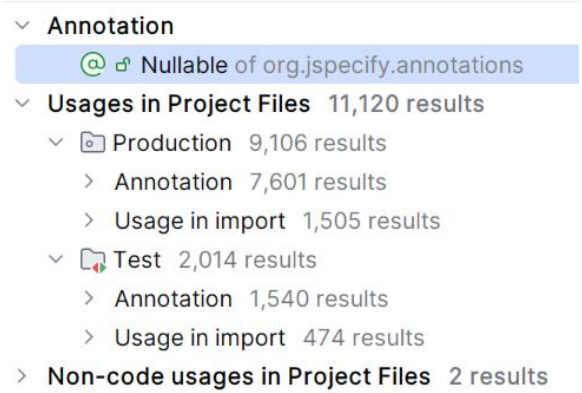
https://xkcd.com/927/

JSpecify

- The main work on JSpecify is [the specification]
- The new annotations are type-use
  - e.g. `List<@Nullable String>` denotes a `List` which can contain `null`
- 4 annotations:
  - `@Nullable`
  - `@NonNull`
  - `@NullMarked` (usually on a package)
  - `@NullUnmarked`

JSpecify and Spring Boot

- Spring Boot didn't have any `@Nullable` in 3.5.x
- Spring Boot has full JSpecify support in 4.0
  - `3481 files changed, 25027 insertions(+), 14107 deletions(-)`



- Kotlin automatically understands JSpecify and creates nullable and non-nullable types
  - No platform-types (`String!`) anymore when using the Spring Boot API

Spring Boot 3.5.x

```java
public class MySpringApplicationRunListener implements SpringApplicationRunListener {
    private static final Logger LOGGER = LoggerFactory.getLogger(MySpringApplicationRunListener.class);

    @Override
    public void failed(ConfigurableApplicationContext context, Throwable exception) {
        String name = context.getEnvironment().getProperty("spring.application.name");
        LOGGER.info("Oh no, {} failed to start :(", name);
    }

}
```

Params: `context` – the application context or `null` if a failure occurred before the context was created
`exception` – the failure

spring

Spring Boot 4.0.x

```
public interface SpringApplicationRunListener {

    Called when a failure occurs when running the application.

    Params: context – the application context or null if a failure occurred before the context was
                created
            exception – the failure

    Since:  2.0.0

    default void failed(@Nullable ConfigurableApplicationContext context, Throwable exception) {
    }
```

```
@Override
public void failed(ConfigurableApplicationContext context, Throwable exception) {
    String name = context.getEnvironment().getProperty("spring.application.name");
    LOGGER.info("Oh no, {} failed
}
```

Method invocation 'getEnvironment' may produce 'NullPointerException'

Assert 'context != null' Alt+Shift+Enter More actions... Alt+Enter

spring

Spring Boot 4.0.x and Kotlin

```
public interface SpringApplicationRunListener {

    Called when a failure occurs when running the application.

    Params: context – the application context or null if a failure occurred before the context was
                created
            exception – the failure

    Since:  2.0.0

    default void failed(@Nullable ConfigurableApplicationContext context, Throwable exception) {
    }
```

Nullable type!

```
class MySpringApplicationRunListener: SpringApplicationRunListener {
    val logger: Logger = LoggerFactory.getLogger( clazz = javaClass)

    override fun failed(context: ConfigurableApplicationContext?, exception: Throwable) {
        val name = context.environment.getProperty("spring.application.name")
        logger.info("Oh no
    }
}
```

Only safe (?.) or non-null asserted (!!.) calls are allowed on a nullable receiver of type 'ConfigurableApplicationContext?'.

Replace with safe (?.) call Alt+Shift+Enter More actions... Alt+Enter

spring

Null Safety portfolio-wide!

Spring Boot

Spring Framework

Spring Data

Spring Security

Spring AI (2.0)

and more!

spring

HTTP Service Client

HTTP Service Clients

```java
public interface HelloClient {

    @GetExchange("/hello/{name}")
    String hello(@PathVariable String name);

}
```

```java
@Configuration(proxyBeanMethods = false)
@ImportHttpServices(group = "hello", types = HelloClient.class)
class HelloClientConfiguration {

}
```

```
spring.http.client.service.factory=jdk
spring.http.client.service.connect-timeout=1s
spring.http.client.service.read-timeout=1s
spring.http.client.service.group.hello.base-url=http://localhost:8080/
```

spring

HTTP Service Clients

```
@Component
class CallHello implements CommandLineRunner {
    private final HelloClient helloClient;

    CallHello(HelloClient helloClient) {
        this.helloClient = helloClient;
    }

    @Override
    public void run(String... args) {
        String greeting = this.helloClient.hello( name: "Moritz")
        System.out.println(greeting);
    }
}
```

spring

API Versioning

API Versioning

```java
@RestController
@RequestMapping("/hello")
class HelloController {

    @GetMapping(version = "1.0")
    String hello10() {
        return "Hello World 1.0";
    }

    @GetMapping(version = "2.0")
    String hello20() {
        return "Hello World 2.0";
    }

    @GetMapping(version = "3.0+")
    String hello30andAbove() {
        return "Hello World 3.0+";
    }

}
```

spring

API Versioning

```
spring.mvc.apiversion.default=1.0
spring.mvc.apiversion.use.header=X-Version
spring.mvc.apiversion.detect-supported=false
spring.mvc.apiversion.supported=1.0,2.0,3.0,4.0
```

API Versioning

RestTestClient

```java
@SpringBootTest
@AutoConfigureRestTestClient
class RestTestClientTestApplicationTests {

    @Test
    void contextLoads(@Autowired RestTestClient restTestClient) {
        restTestClient.get().uri( uri: "/") capture of ?
                .exchangeSuccessfully() ResponseSpec
                .expectBody(String.class) BodySpec<String, capture of ?>
                .isEqualTo( expected: "hello");
    }

}
```

```
@SpringBootTest
@AutoConfigureRestTestClient
class RestTestClientTestApplicationTests {

    @Test
    void assertJ(@Autowired RestTestClient restTestClient) {
        RestTestClient.ResponseSpec spec = restTestClient.get().uri( uri: "/").exchange();
        RestTestClientResponse response = RestTestClientResponse.from(spec);
        assertThat(response).hasStatusOk().bodyText().isEqualTo( expected: "hello");
    }

}
```

Miscellaneous changes

Miscellaneous changes

- Liveness and readiness probes are now enabled by default
- `@MockBean` and `@SpyBean` are gone (deprecated since 3.4)
- Use `@MockitoBean` and `@MockitoSpyBean` instead

```java
@SpringBootTest
class MyCLRTest {

    @MockitoBean
    private MyService myService;

    @BeforeEach
    void setUp() {
        when(this.myService.fetchData()).thenReturn( value: "mock-data");
    }

    @Test
    void test() {
        // ...
    }
}
```

spring

Miscellaneous changes

- Optional Maven dependencies are no longer included in the uber jar by default
- Including requires opt-in:

```
<plugin>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-maven-plugin</artifactId>
    <configuration>
        <includeOptional>true</includeOptional>
    </configuration>
</plugin>
```

- `MockitoTestExecutionListener` is gone, use `@ExtendWith(MockitoExtension.class)` instead

Thank you

Contact me at mhalbritter.github.io

Literature & References

- https://spring.io/blog/2025/10/28/modularizing-spring-boot
- https://github.com/spring-projects/spring-boot/wiki/Spring-Boot-4.0-Migration-Guide
- https://github.com/spring-projects/spring-boot/wiki/Spring-Boot-4.0-Release-Notes
- https://docs.spring.io/spring-boot/reference/features/json.html#features.json.jackson2
- https://docs.spring.io/spring-framework/reference/core/null-safety.html#null-safety-guidelines-jspecify
- https://github.com/FasterXML/jackson/blob/main/jackson3/MIGRATING_TO_JACKSON_3.md
- https://spring.io/blog/2025/09/02/road_to_ga_introduction

Topics not covered

Topics not covered

- JmsClient
- Task Decoration
- OpenTelemetry starter
- Configuration Properties Metadata for External Types
- SSL Info / SSL Health
- MongoDB Health Indicators
- Kotlin Serialization
- Redis Static Master/Replica
- Redis Observability
- Public members (aside from constants) have been removed from auto-configuration classes. Auto-configurations never have been public API and now this is enforced through Java mechanisms.
- You can now set Spring-Boot-Jar-Type to development-tool in your MANIFEST.MF to exclude dependencies from uber jars.

Topics not covered

- The properties for Spring Session Data Redis have been renamed to reflect the dependencies on Spring Data Redis. Properties that previously began with spring.session.redis now begin with spring.session.data.redis. Similarly, properties that previously began with spring.session.mongodb now begin with spring.session.data.mongodb
- HttpMessageConverters Deprecation: Instead, your application can declare one or more ClientHttpMessageConvertersCustomizer and ServerHttpMessageConvertersCustomizer that will let you customize converters in a flexible way.
- If you're deploying a war file to Tomcat, you need to switch the dependency spring-boot-starter-tomcat to spring-boot-starter-tomcat-runtime.

Topics not covered

- Some properties for configuring MongoDB have been updated so that their names reflect whether or not Spring Data MongoDB is required. Many properties whose names previously began with spring.data.mongodb now begin with spring.mongodb
- Spring Batch can now operate without a database (i.e. in memory), and the regular spring-boot-starter-batch uses this simplified mode. On upgrade, Spring Batch will no longer store metadata in your existing database. You can either simplify your configuration and use this new mode, or restore the previous arrangement. To go back to using a database, you need to change to spring-boot-starter-batch-jdbc.

Topics not covered

- Using the @SpringBootTest annotation will no longer provide any MockMVC support. If you want to use MockMVC in your tests you should now add an @AutoConfigureMockMvc annotation to the test class.
- Using the @SpringBootTest annotation will no longer provide any WebClient or TestRestTemplate beans. If you want to use a TestRestTemplate you should add an @AutoConfigureTestRestTemplate annotation to the test class. Dependencies on org.springframework.boot:spring-boot-resttestclient and org.springframework.boot:spring-boot-restclient are also required.
- In addition, you might want to consider replacing any use of TestRestTemplate with the new RestTestClient class. To configure this, add an @AutoConfigureRestTestClient annotation to the test class.