Upgrade to Spring Boot 3? Spring Tools to the Rescue

Martin Lippert, @martinlippert

September 2023

Where are we?

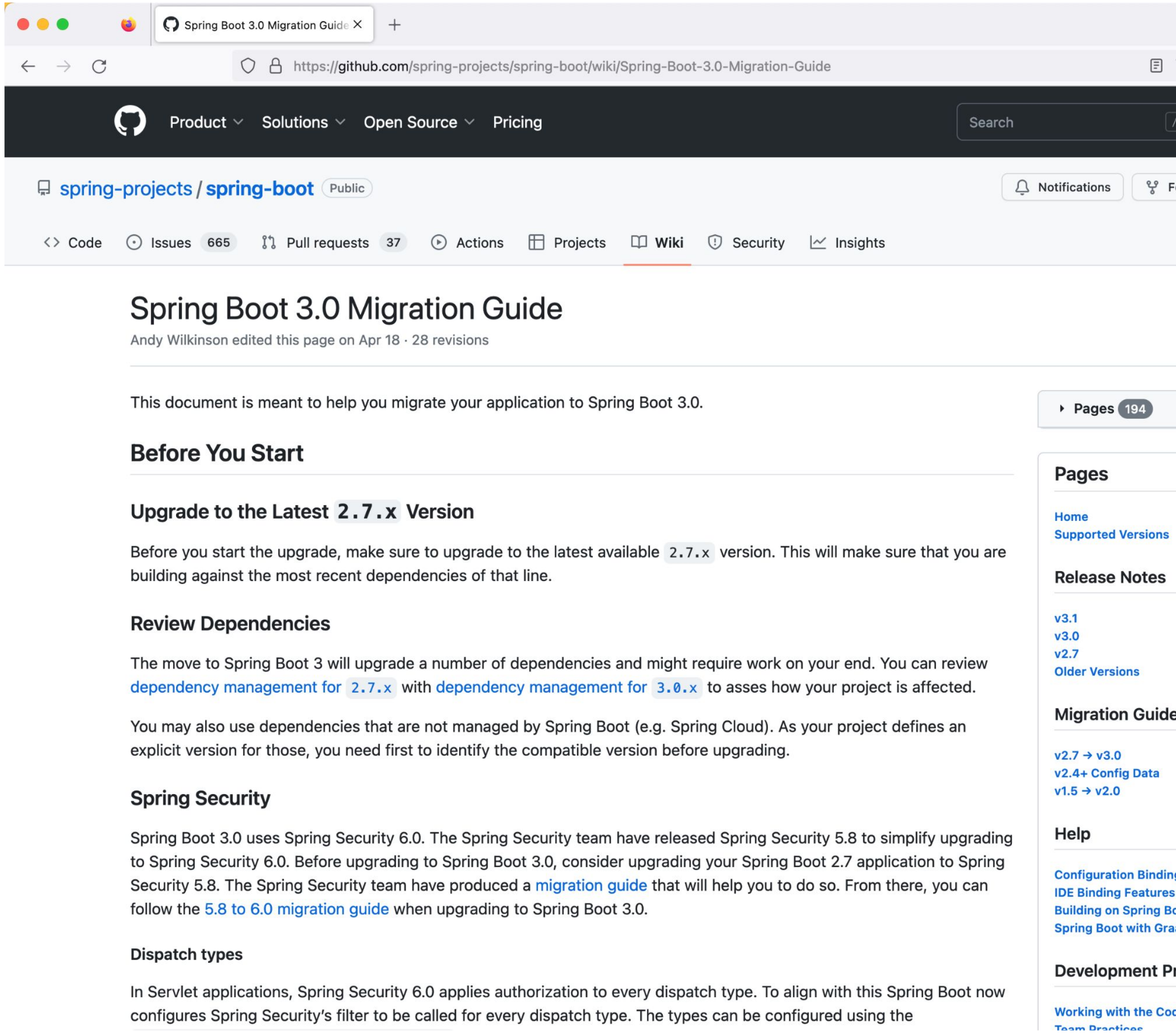
New Spring Boot releases all the time

- Many patch releases all the time
- New minor release every 6 month, sometimes new major releases
- It is super important to stay up-to-date
- But it is sometimes hard to always stay up-to-date

How to upgrade?

Release Notes + Migration Guides

- You have to read everything carefully
- You need to find out what needs to be changed for your project
- You need to apply all those changes manually



Spring Boot 3.0 Migration Guide
https://github.com/spring-projects/spring-boot/wiki/Spring-Boot-3.0-Migration-Guide
Product
Solutions
Open Source
Pricing
Search
spring-projects / spring-boot
Public
Notifications
Code
Issues 665
Pull requests 37
Actions
Projects
Wiki
Security
Insights
Pages 194

Spring Boot 3.0 Migration Guide

Andy Wilkinson edited this page on Apr 18 · 28 revisions

This document is meant to help you migrate your application to Spring Boot 3.0.

Before You Start

Upgrade to the Latest `2.7.x` Version

Before you start the upgrade, make sure to upgrade to the latest available `2.7.x` version. This will make sure that you are building against the most recent dependencies of that line.

Review Dependencies

The move to Spring Boot 3 will upgrade a number of dependencies and might require work on your end. You can review dependency management for `2.7.x` with dependency management for `3.0.x` to asses how your project is affected.

You may also use dependencies that are not managed by Spring Boot (e.g. Spring Cloud). As your project defines an explicit version for those, you need first to identify the compatible version before upgrading.

Spring Security

Spring Boot 3.0 uses Spring Security 6.0. The Spring Security team have released Spring Security 5.8 to simplify upgrading to Spring Security 6.0. Before upgrading to Spring Boot 3.0, consider upgrading your Spring Boot 2.7 application to Spring Security 5.8. The Spring Security team have produced a migration guide that will help you to do so. From there, you can follow the 5.8 to 6.0 migration guide when upgrading to Spring Boot 3.0.

Dispatch types

In Servlet applications, Spring Security 6.0 applies authorization to every dispatch type. To align with this Spring Boot now configures Spring Security's filter to be called for every dispatch type. The types can be configured using the

Pages

Home
Supported Versions

Release Notes

v3.1
v3.0
v2.7
Older Versions

Migration Guide

v2.7 → v3.0
v2.4+ Config Data
v1.5 → v2.0

Help

Configuration Bindin
IDE Binding Features
Building on Spring Bo
Spring Boot with Gra

Development Pr

Working with the Cod
Team Practices

Let’s do something about this

(Spring Tools to the Rescue)

What is new to Spring Tools?

Let the user know

- Automatically check the versions that you use
- Show information about new versions and support ranges

Help the user to upgrade

- Migration guides written in “code”
- Looks at your project and applies necessary changes - AUTOMATICALLY
- Some limitations apply

Limitations

No silver bullet

- The tools apply many changes, but not all of them
- The goal is to automate as much as possible
- There is no guarantee that you are done with the upgrade afterwards - probably additional manual steps needed
  - But this will improve with every tools release - of course... 😉

Looking for feedback

Reminder: Everything that you will see is early days

- We are looking for feedback and suggestions
- If you want to get involved here, let us know

Live Demo

(Spring Version Validation & Upgrade Support)

Under the hood

What is OpenRewrite?

- *“Open-source, semantic type aware search and transformation framework.”*
- *“OpenRewrite enables large-scale distributed source code refactoring for framework migrations, vulnerability patches, and API migrations”*
- Automatically transform source code (for various purposes)
- https://docs.openrewrite.org/
- https://github.com/openrewrite

Based on initial work at Netflix to keep source code up-to-date. Sponsored now by Moderne.io. The Moderne SaaS allows organizations to run search and transformations across hundreds of repositories (millions of lines of code) simultaneously and offer a free service for the OSS community at https://public.moderne.io/

Purpose

What can OpenRewrite be used for?

- Patching CVEs
- Migrate from Java 8 to Java 11 to Java 17…
- Migrate between framework versions
- Automatically adapt code to changed APIs
- …

- Works across various source file types (like Java Source Code, property files, YAML, other languages, etc.)

spring

The internals

How does OpenRewrite work internally?

- Step 1: Parse source files into AST
  - Type resolution
  - Keep formatting intact
- Step 2: Run visitors on ASTs to transform them
  - Visitors contain the logic what exactly to do for the refactoring, the migration, the code fix, etc.
- Step 3: Generate source changes

Recipes

Recipes aggregate visitors

- Users deal with recipes
  - The AST visitors are an implementation detail

- Recipes are either
  - defined using YAML, or
  - implemented in Java

The power behind it

Recipes can be written by anyone

- OpenRewrite comes with a huge set of basic transformation recipes pre-packaged and ready-to-use
    - https://docs.openrewrite.org/reference/recipes

- It is easy to use them and write custom recipes
- Community around recipes
- Packages could bring their own recipes
    - E.g. a library contains recipes to migrate client code to a new version of the library

spring

Transforming the code

Running recipes via Maven or Gradle

- `./mvnw rewrite:discover` – Lists all the available recipes
- `./mvnw rewrite:run` – Runs the recipes configured as active (in the build config)
- `./mvnw rewrite:dryRun` – Runs the recipes, but creates a patch file instead of changing the files directly

What we do inside the Spring Tools

List and run recipes from the UI

- Show the recipes that are available
- Let the user select the recipes
- Execute the recipes within the IDE

Authoring recipes

You can write your own recipes and try them

- A preference allows you to add your own recipes to the language server
- Write them in one workspace, test them in another
- No need to restart the IDE, just press “Refresh”

Live Demo

(writing your own recipes)

Another use case

Validations and Quick Fixes

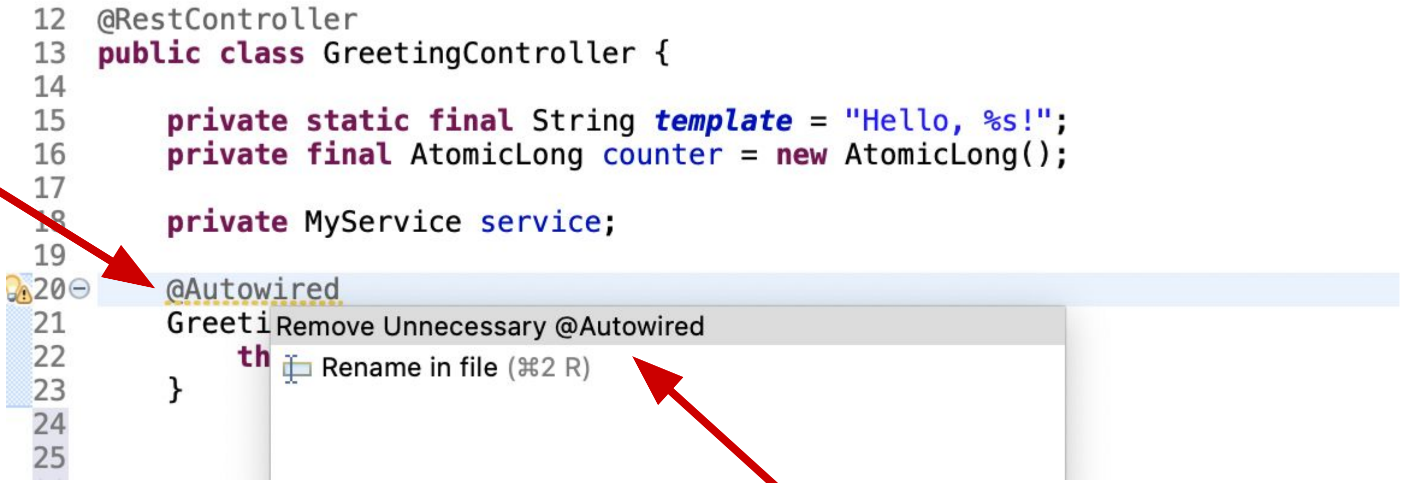
- Let’s now push this beyond running recipes on projects
- Let’s combine this with validations/markers and code actions/quick fixes

- *This goes beyond what OpenRewrite supports out-of-the-box, but it can be added on top*

Validations and Quick Fixes

Something that looks like this

validation



```
@RestController
public class GreetingController {

    private static final String template = "Hello, %s!";
    private final AtomicLong counter = new AtomicLong();

    private MyService service;

    @Autowired
    Greeti
        th
    }
```

code action / quick fix
(implemented as a recipe)

spring

Live Demo

(additional cool new things)

Resources

OpenRewrite

- https://docs.openrewrite.org/
- https://github.com/openrewrite

IDE Integration

- Started as part of the Spring Tools: https://github.com/spring-projects/sts4/
- Independent of Spring Tooling in the future?
- Contact us on Twitter: http://twitter.com/springtools4/

spring

Thank you

@martinlippert

(special thanks to Tyler van Gorder and Alex Boyko for their support and work on this)