



# How to Keep your Spring Boot Projects Up-To-Date

Martin Lippert, Spring Tools Lead, Broadcom



# Where are we?

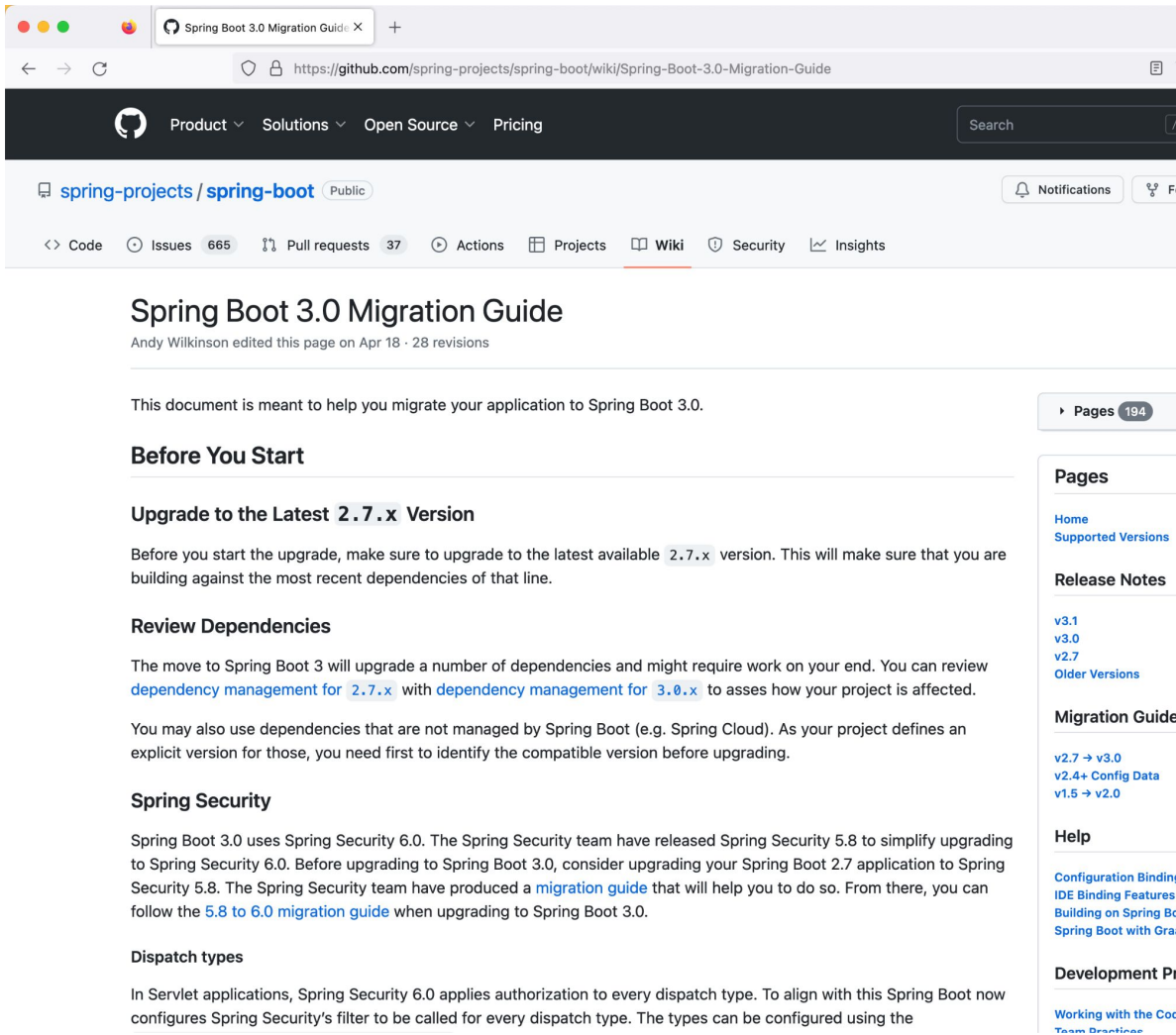
## New Spring Boot releases all the time

- 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

A screenshot of the GitHub page for the Spring Boot 3.0 Migration Guide. The browser address bar shows the URL https://github.com/spring-projects/spring-boot/wiki/Spring-Boot-3.0-Migration-Guide. The page header includes the GitHub logo, navigation links (Product, Solutions, Open Source, Pricing), a search bar, and repository information (spring-projects / spring-boot, Public). Below the header, there are tabs for Code, Issues (665), Pull requests (37), Actions, Projects, Wiki (selected), Security, and Insights. The main content area is titled 'Spring Boot 3.0 Migration Guide' and notes that Andy Wilkinson edited the page on Apr 18 with 28 revisions. The text states: 'This document is meant to help you migrate your application to Spring Boot 3.0.' The page is divided into sections: 'Before You Start', 'Upgrade to the Latest 2.7.x Version' (which advises upgrading to the latest 2.7.x version before upgrading to 3.0), 'Review Dependencies' (which discusses the impact of the move to Spring Boot 3 on dependencies), 'Spring Security' (which mentions the release of Spring Security 5.8 to simplify upgrading to Spring Boot 3.0), and 'Dispatch types' (which discusses authorization in Servlet applications). On the right side, there is a sidebar with links to 'Pages' (194), 'Release Notes' (listing v3.1, v3.0, v2.7, and Older Versions), 'Migration Guide' (listing v2.7 to v3.0, v2.4+ Config Data, and v1.5 to v2.0), 'Help' (with links to Configuration Binding, IDE Binding Features, Building on Spring Boot, and Spring Boot with GraalVM), and 'Development Process' (with a link to Working with the Core 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 to Java 21 ...
  - 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.)

# 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

# Another use case

## Validations and Quick Fixes

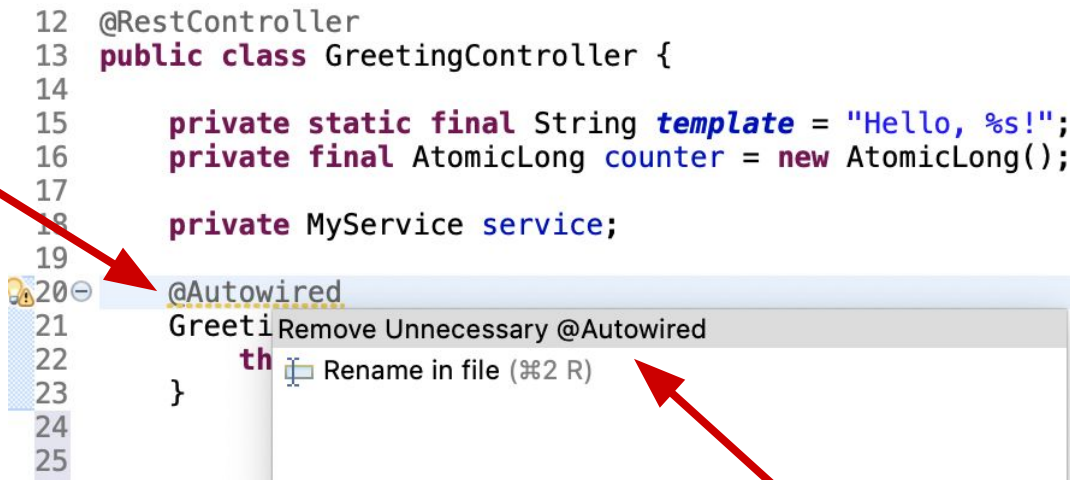
- Let's now push this beyond running recipes on projects
- Let's combine this with validations/markers and code actions/quick fixes
- *This goes beyond what OpenRewrite supports out-of-the-box, but it can be added on top*

# Validations and Quick Fixes

## Something that looks like this

validation

```
12 @RestController
13 public class GreetingController {
14
15     private static final String template = "Hello, %s!";
16     private final AtomicLong counter = new AtomicLong();
17
18     private MyService service;
19
20     @Autowired
21     GreetingController(MyService service) {
22         this.service = service;
23     }
24
25 }
```



code action / quick fix  
(implemented as a recipe)

# 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/>

# Thank you

@martinlippert



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