Externalizing Configuration with Properties and YAML Files



Federico Mestrone

Software Engineer and Training Consultant

@fedmest www.federicomestrone.com

Library Overview

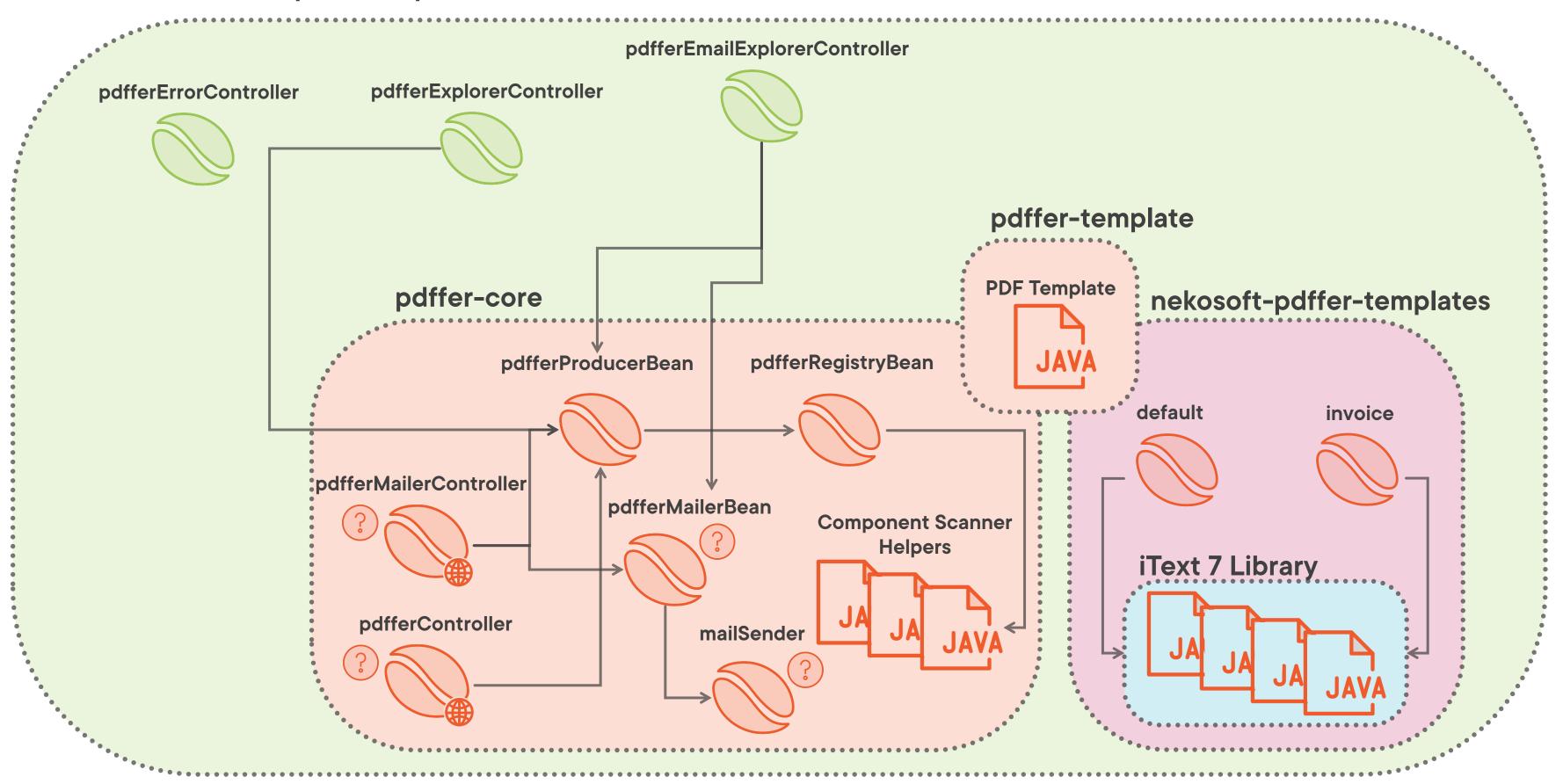


In this module

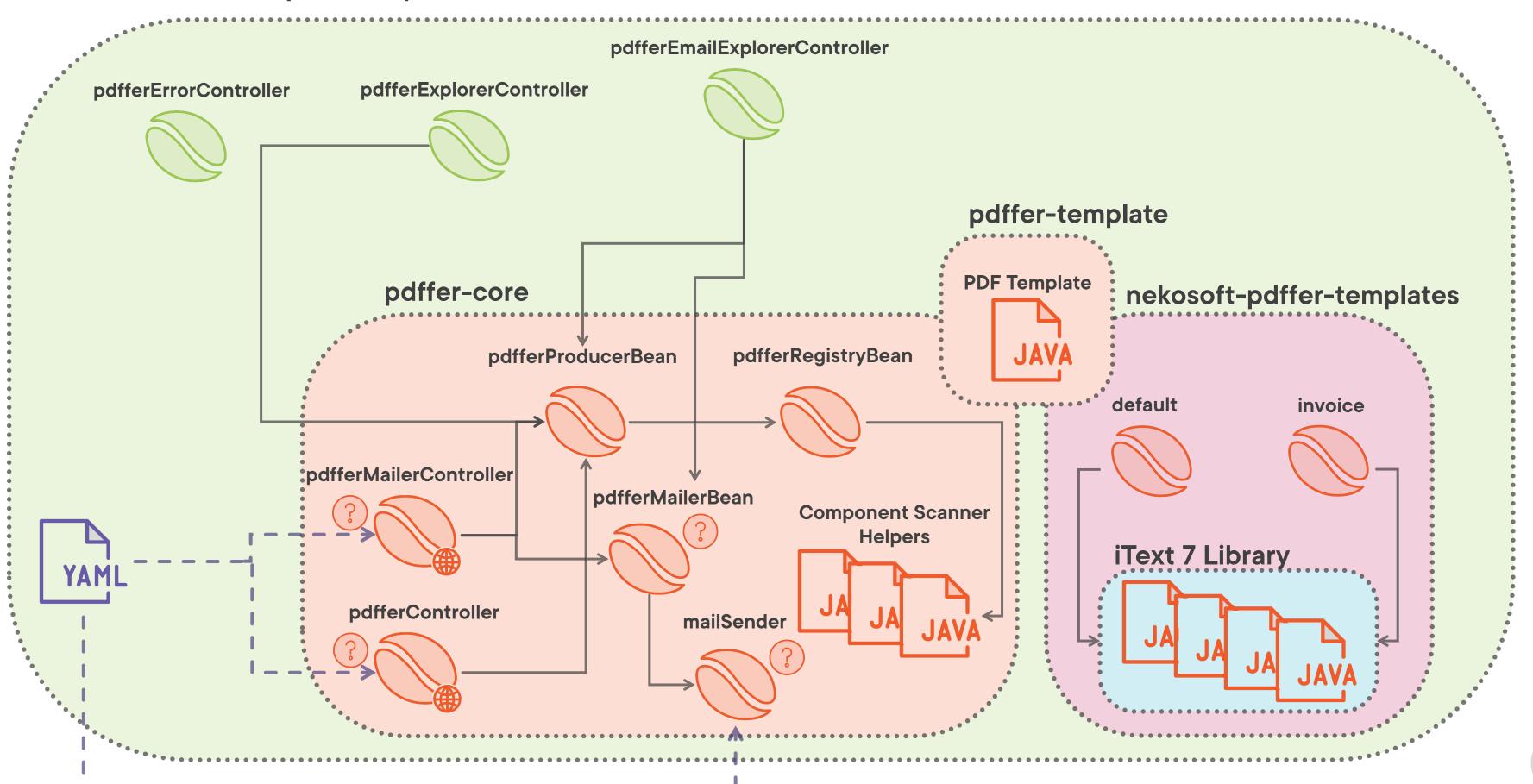
- We will externalize everything users of the library might want control over
 - Read values from files external to the source code
 - Using Spring application properties
- Also use property conditionals
 - To select individual PDFfer beans



nekosoft-pdffer-explorer



nekosoft-pdffer-explorer



Spring Configuration Data

Common need to externalize options

Spring Boot offers its own ways

- Sophisticated layered approach

Code has access to config properties

- For whatever reason it needs them!

Inject a property value with \${ }

- Using @Value, @RequestMapping, etc.
- Can provide defaults too

Using \${ } in @Value

```
@Component
public class PdfferMailerBean {
   @Value("${pdffer.mailer.send-from.name}")
    private String sendFromName;
    @Value("${pdffer.mailer.send-from.address}")
    private String sendFromAddress;
    @Value("${pdffer.mailer.reply-to.name}")
    private String replyToName;
    @Value("${pdffer.mailer.reply-to.address}")
    private String replyToAddress;
    // ...
```

Using \${ } in Controllers, with defaults

```
@RestController
@RequestMapping("${pdffer.web.controller.base_uri:pdffer}")
public class PdfferController {
   @PostMapping("${pdffer.web.controller.download_uri:download}/{templateId}")
    public ResponseEntity<byte[]> download(@PathVariable String templateId) {
        // ...
    @PostMapping("${pdffer.web.controller.save_uri:save}/{templateId}")
    public void save(@PathVariable String templateId) {
        // ...
```

You can have a nested default too!

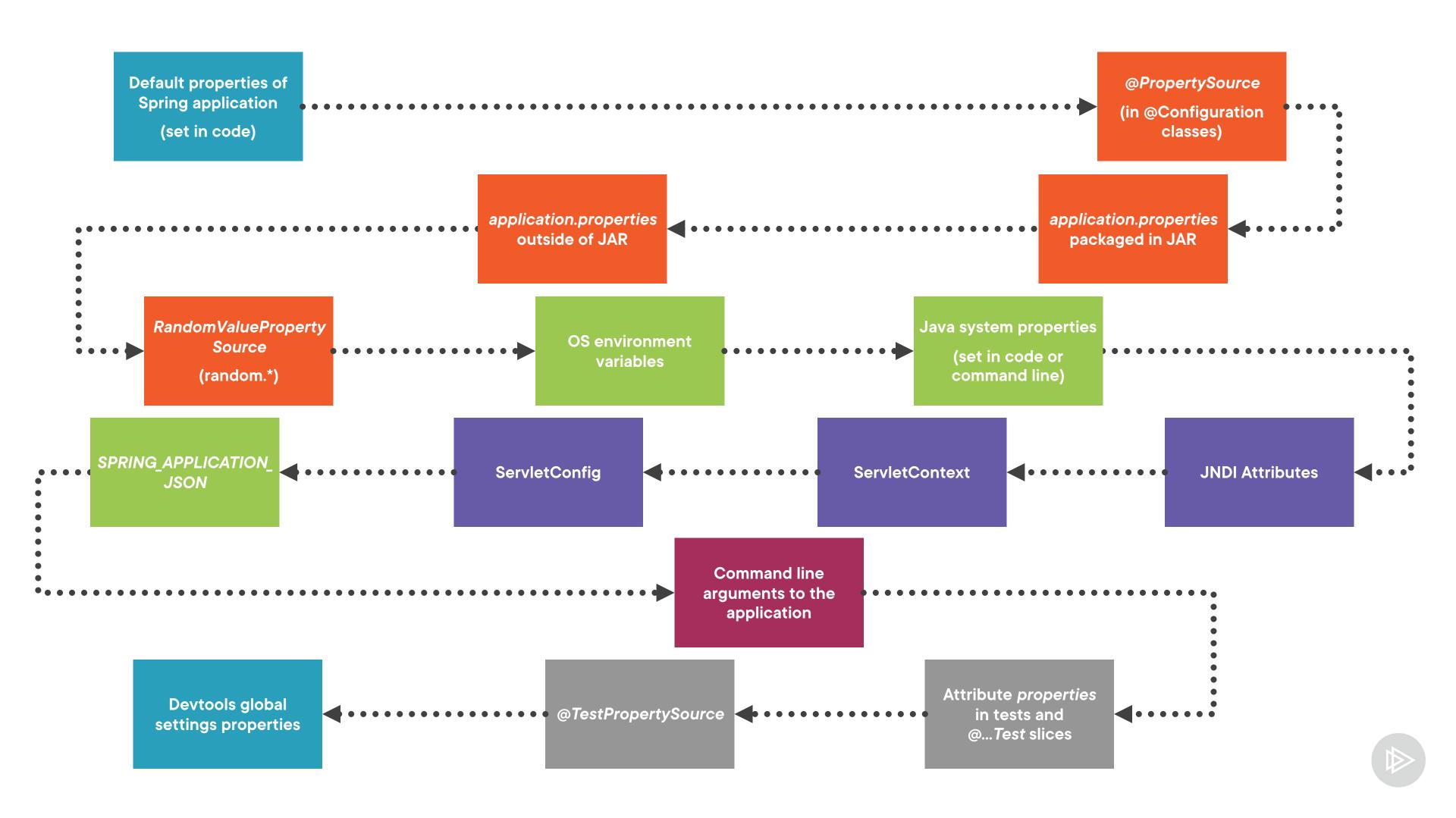
```
@RestController
@RequestMapping(
    "${pdffer.mailer.controller.base_uri:${pdffer.web.controller.base_uri:pdffer}}"
public class PdfferMailerController {
    @PostMapping("${pdffer.web.controller.email_uri:mail}/{templateId}")
    public void email(@PathVariable String templateId, @RequestBody EmailRequestData) {
       // ...
```

Where does Spring Boot get these configuration values from?



Spring Boot Configuration Data





application.properties

```
@SpringBootApplication
public class MySpringBootApp {
    public static void main(String[] args) {
        SpringApplication application = new SpringApplication(MySpringBootApp.class);
        Properties properties = new Properties();
        properties.setProperty("pdffer.web.controller.base-uri", "module6");
        properties.setProperty("pdffer.web.controller.save-uri", "store");
        application.setDefaultProperties(properties);
        application.run(args);
```

application.properties

.properties or .yaml both ok

```
@SpringBootApplication
@PropertySource("classpath:org/nekosoft/pdffer/pdffer-defaults.yaml")
public class MySpringBootApp {
    public static void main(String[] args) {
        SpringApplication.run(MySpringBootApp.class, args);
    }
}
```

Spring Boot Configuration Data

Config data in classpath root

- application.properties
- application.yaml

You should stick to one format only

- Properties takes precedence if both present

application.properties

```
pdffer.mailer.send-from.name=Terminal PDF Generator
pdffer.mailer.send-from.address=termpdfgen@pdffer.domain
pdffer.mailer.reply-to.name=Terminal PDF Generator Admin
pdffer.mailer.reply-to.address=termpdfgenadmin@pdffer.domain
pdffer.mailer.smtp.host=in-v3.mailjet.com
pdffer.mailer.smtp.port=587
pdffer.mailer.smtp.username=yourmailerusernameyourmaileruser
pdffer.mailer.smtp.password=yourmailerpasswordyourmailerpass
pdffer.mailer.smtp.java-mail-properties.mail.transport.protocol="smtp"
pdffer.mailer.smtp.java-mail-properties.mail.smtp.auth="false"
pdffer.mailer.smtp.java-mail-properties.mail.smtp.starttls.enable="false"
pdffer.mailer.smtp.java-mail-properties.mail.debug="false"
pdffer.web.controller.base-uri=nekopdf
pdffer.web.controller.download-uri=download
```

application.yaml

```
pdffer:
 mailer:
    send-from:
      name: PDFfer
      address: pdf@pdffer.domain
    reply-to:
      name: PDFfer Admin
      address: admin@pdffer.domain
    smtp:
      host: in-v3.mailjet.com
      port: 587
      username: yourmailerusernameyourmaileruser
      password: yourmailerpasswordyourmailerpass
web:
    controller:
      base-uri: nekopdf
      download-uri: download
```

application.properties

```
@Component
public class AppSecrets {
    @Value("${random.int}")
    private int secretRandomInt;
    @Value("${random.long(0,1000)}")
    private long secretRandomLong;
    @Value("${random.uuid}")
    private UUID secretRandomUUID;
    @Value("${random.anything}")
    private byte[] secretRandomBytes;
    // getters
```

OS Env Vars

- > export PDFFER_WEB_CONTROLLER_BASE_URI=module6a
- > export PDFFER_WEB_CONTROLLER_SAVE_URI=store1
- > java -jar terminal-pdf-generator-1.0.0.jar

Base URI for this instance of PDFfer: module6a

Full URL for Save endpoint: http://localhost:8080/module6a/store1

Java System Properties

Base URI for this instance of PDFfer: module6b

Full URL for Save endpoint: http://localhost:8080/module6b/store2

SPRING_APPLICATION_JSON

Full URL for Save endpoint:

http://localhost:8080/m6c/st3

```
> export SPRING_APPLICATION_JSON='{"pdffer.web.controller.base-uri":"m6c","pdffer.web.controller.save-uri":"st3"}'
> java -jar terminal-pdf-generator-1.0.0.jar
======|_|========|__/=/_/_/
 :: Spring Boot :: (v2.5.3)
Base URI for this instance of PDFfer:
m6c
```

Command Line Arguments

```
> java -jar terminal-pdf-generator-1.0.0.jar --pdffer.web.controller.base-uri=module6d \
     --pdffer.web.controller.save-uri=store4
```

Base URI for this instance of PDFfer: module6d

Full URL for Save endpoint: http://localhost:8080/module6d/store4

Spring Boot DevTools

Property defaults and global properties

- Defines some defaults useful in dev
 - E.g. disabling caching
- Global properties across projects
 - In .spring-boot-devtools.properties under user's home directory

Automatic restart

- Whenever changes are made

Live reload

Of browser pages when plugin installed

Remote debugging and updates

With appropriate server, package and IDE configurations



Inject a property value with \${ }

- Using @Value, @RequestMapping, etc.
- Can provide defaults too

Inject the full Environment object

And access properties with the Environment API

Using the Environment API

```
@Component
public class ConsoleAppBean implements CommandLineRunner {
   @Autowired
    Environment env;
   @Override
    public void run(String... args) throws Exception {
        System.out.println("Base URI for this instance of PDFfer: ");
        System.out.println(env.getProperty("pdffer.web.controller.base_uri"));
        System.out.println("Full URL for Save endpoint: ");
        System.out.println(env.resolvePlaceholders(
 "http://localhost/${pdffer.web.controller.base_uri}/${pdffer.web.controller.save_uri}"
        ));
```

Inject a property value with \${ }

- Using @Value, @RequestMapping, etc.
- Can provide defaults too

Inject the full Environment object

And access properties with the Environment API

Create Property POJOs

 Which can automatically be mapped to properties in a convenient manner



Enabling @ConfigurationProperties

```
@Configuration
@ComponentScan
@ConfigurationPropertiesScan(basePackages = "org.nekosoft.pdffer.props")
public class PdfferCoreConfiguration {
}
```

Creating a Configuration Properties POJO

Mutable with getters and setters

PdfferWebProps.java

```
@ConfigurationProperties(
   prefix = "pdffer.web.controller"
)
public class PdfferWebProps {
    String baseUri;
   String downloadUri;
   String saveUri;

   // getters and setters
}
```

application.yaml

```
pdffer:
    web:
    controller:
        base-uri: nekopdf
        download-uri: download
        save-uri: save
```

Creating a Configuration Properties POJO

Immutable with constructor binding and only getters

PdfferWebProps.java

```
@ConfigurationProperties
 prefix = "pdffer.web.controller"
@ConstructorBinding
public class PdfferWebProps {
    // field declarations
    public PdfferWebProps()
        String baseUri,
        String downloadUri,
        String saveUri) {
            // set field values
    // getters only
```

application.yaml

```
pdffer:
    web:
    controller:
        base-uri: nekopdf
        download-uri: download
        save-uri: save
```

Using Configuration Properties POJOs

```
@Component
public class ConsoleAppBean implements CommandLineRunner {
   @Autowired
    PdfferWebProps webProps;
   @Override
    public void run(String... args) throws Exception {
        System.out.println("Base URI for this instance of PDFfer: ");
        System.out.println(webProps.getBaseUri());
        System.out.println("Full URL for Save endpoint: ");
        System.out.println(String.format("http://localhost:8080/%s/%s",
            webProps.getBaseUri(), webProps.getSaveUri())
```

Creating a Configuration Properties POJO

Immutable with constructor binding and only getters

application.yaml

```
pdffer:
 mailer:
    send-from:
     name: PDFfer
     address: pdf@pdffer.domain
    reply-to:
     name: PDFfer Admin
      address: admin@pdffer.domain
    smtp:
     host: in-v3.mailjet.com
     port: 587
     username: yourmailerusernameyourmaileruser
      password: yourmailerpasswordyourmailerpass
      java-mail-properties:
        mail.transport.protocol: "smtp'
        mail.smtp.auth: "true"
        mail.smtp.starttls.enable: "true"
        mail.debug: "true"
```

EmailAddressInfo.java

```
public class EmailAddressInfo {
    // field declarations
    public EmailAddressInfo(
        String name,
        String address
        // set field values
    // getters only
```

Conditionals on Properties



Conditions on beans

- @ConditionalOnBean

Conditions on files and classes

- @ConditionalOnClass
- @ConditionalOnResource

Conditions on environment and set-up

- @ConditionalOnWebApplication
- @ConditionalOnJava

Conditions on property values

- @ConditionalOnProperty
- @ConditionalOnExpression

Conditional Controllers and Controller Methods

```
@RestController
@ConditionalOnWebApplication
@ConditionalOnBean(type = "org.nekosoft.pdffer.PdfferProducerBean")
@ConditionalOnProperty(
    name = "pdffer.skip.web.controller",
    havingValue = "false",
    matchIfMissing = true
@RequestMapping("${pdffer.web.controller.base_uri:pdffer}")
public class PdfferController {
    // implementation omitted
```

@ConditionalOnProperty

```
@RestController
@ConditionalOnWebApplication
@ConditionalOnBean(type = "org.nekosoft.pdffer.PdfferProducerBean")
@ConditionalOnExpression("${pdffer.skip.web.controller} eq 'false'")
@RequestMapping("${pdffer.web.controller.base_uri:pdffer}")
public class PdfferController {
    // implementation omitted
}
```

Spring Boot Configuration Metadata

Configuration Annotation Processor

Looks for configuration properties defined in your code

Lists them all together in a JSON file

- With basic information
- Can enrich with further documentation

```
<dependency>
     <groupId>org.springframework.boot</groupId>
          <artifactId>spring-boot-configuration-processor</artifactId>
          <optional>true</optional>
</dependency>
```

Adding the Spring Boot Configuration Processor

With Maven

```
dependencies {
   annotationProcessor "org.springframework.boot:spring-boot-configuration-processor"
}
```

Adding the Spring Boot Configuration Processor

With Gradle









> .mvn

Project •

✓ build

classes

✓ java

✓ main

✓ META-INF

spring-configuration-metadata.json

> org

> generated

> libs

> resources

> tmp

bootJarMainClassName

✓ src

∨ **main**

> java

✓ resources

✓ META-INF

additional-spring-configuration-metadata.json

spring.factories

> org.nekosoft.pdffer._misc

application.yaml

abanner.txt

gitignore.

File generated automatically in build folder









> .mvn

Project •

✓ build

classes

✓ java

✓ main

✓ META-INF

spring-configuration-metadata.json

> org

> generated

> libs

> resources

> tmp

bootJarMainClassName

✓ src

> java

✓ resources

✓ META-INF

additional-spring-configuration-metadata.json

spring.factories

> org.nekosoft.pdffer._misc

application.yaml

abanner.txt

agitignore.

Additional overriding metadata is possible

Metadata Groups

```
"groups": [
    "name": "pdffer.mailer",
    "type": "org.nekosoft.pdffer.props.PdfferMailerProps",
    "sourceType": "org.nekosoft.pdffer.props.PdfferMailerProps"
    "name": "pdffer.mailer.controller",
    "type": "org.nekosoft.pdffer.props.PdfferMailerControllerProps",
    "sourceType": "org.nekosoft.pdffer.props.PdfferMailerControllerProps"
    "name": "pdffer.web.controller",
    "type": "org.nekosoft.pdffer.props.PdfferWebControllerProps",
    "sourceType": "org.nekosoft.pdffer.props.PdfferWebControllerProps"
```

Metadata Properties

```
"properties": [
    "name": "pdffer.mailer.controller.base-uri",
    "type": "java.lang.String",
    "description": "Base URI at which the mailer controller for the PDFfer endpoints will be registered",
    "sourceType": "org.nekosoft.pdffer.props.PdfferMailerControllerProps"
    "name": "pdffer.mailer.reply-to.address",
    "type": "java.lang.String",
    "description": "The email address to which replies should be sent"
    "name": "pdffer.mailer.reply-to.name",
    "type": "java.lang.String",
    "description": "The name or label associated to the email address to which replies should be sent"
```

Metadata Properties

```
"hints": [
    "name": "spring.jpa.hibernate.ddl-auto",
    "values": [
        "value": "none",
        "description": "Disable DDL handling."
        "value": "validate",
        "description": "Validate the schema, make no changes to the database."
        "value": "update",
        "description": "Update the schema if necessary."
      },
```

Summary



Effective development

- Spring Initialzr
 - From the web, console, or IDE
- Navigating Spring configurations in IDE
- Testing HTTP endpoints
- Externalizing configuration
 - With property POJOs
 - And configuration metadata

Summary



Effective configuration

- Spring Boot autoconfiguration
 - The spring.factories file
- Hierarchical contexts
- Advanced component scanning
 - And custom type filters
- Built-in and custom conditional beans
- Spring Boot configuration properties
- Conditionals on property values

Up Next:

Offering a Stand-alone Mode for Our Spring Boot Library