

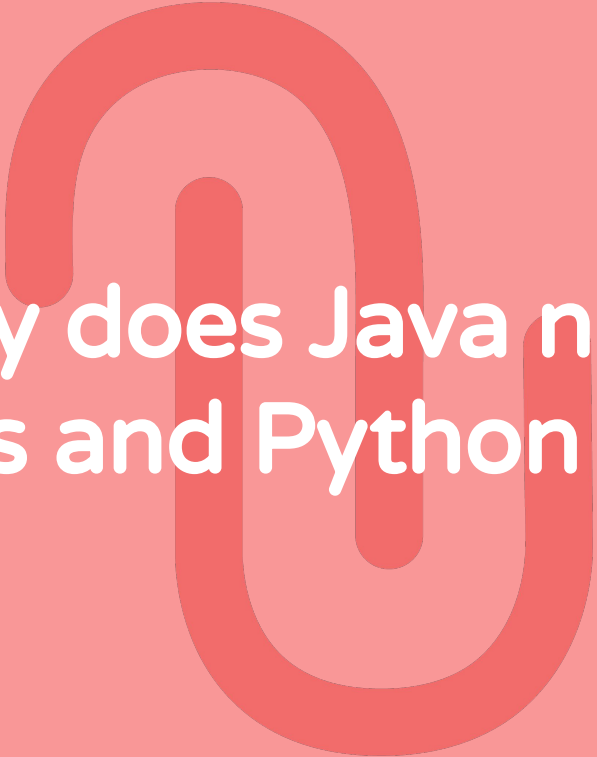
Little Helpers

Useful Tools for
(Java) Developers



- ❖ Why do some tech stacks need tools?
- ❖ Fighting Verbosity: Lombok
- ❖ Better Tooling: Google Guava
- ❖ Java Swiss Army Knife: Apache Commons
- ❖ Back to Excel: Apache POI

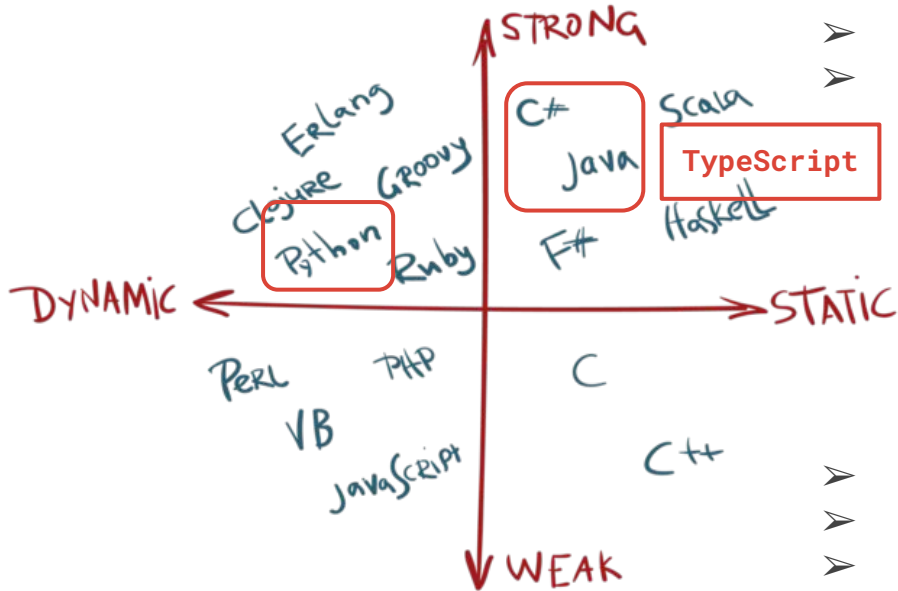
- ❖ (Aspect-Oriented Programming: AspectJ / Spring)
- ❖ (Rapid Application Development: JHipster)
- ❖ (Spring with more Fun: Spring Boot Dev Tools)
- ❖ (Better Testing: Mockito, AssertJ & ArchUnit)



Why does Java need
Tools and Python not?



Static vs Dynamic and Weak vs Strong



- **strong** and **weak** typing refers to **type consistency**
- **static** and **dynamic** typing refers to the **time** when names are **bound to types**

- **C#, Java** and **Typescript** are **static & strong**
- **Python, Ruby** and **Erlang** are **dynamic & strong**
- **JavaScript** and **Perl** are **weak & dynamic**



Static vs Dynamic and Weak vs Strong



Python: Dynamic Strong Typing

```
value = 21;           # type assigned as int at runtime.
value = value + "dot"; # type-error, string and int cannot be concatenated.
print(value);
```

JavaScript: Dynamic Weak Typing

```
value = 21;
value = value + "dot"; # "21dot"
console.log(value);    # no error, implicit conversion between unrelated types
```

Java: Static Strong Typing

```
var value = 21;           // type int set at compile time (type inference)
value = value + "dot";    // compile-time-error
System.out.println(value);
```

TypeScript: Static (Pre-Compile) Strong Typing

```
var value = 21;           # type assigned as int at compile-time.
value = value + "dot";    # (pre-)compile-time-error
console.log(value);
```



Static vs Dynamic and Weak vs Strong



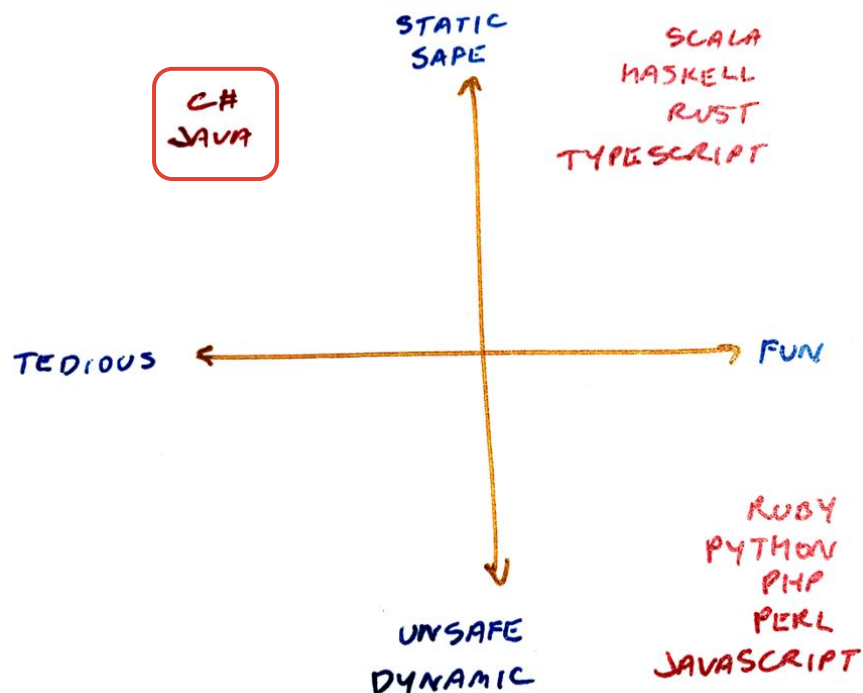
Java is a **strongly** and **statically typed** language and favors **simplicity of syntax**.

With these characteristics, Java tends to be **very verbose**, requiring a lot of **boilerplate code** and **tedious typing**.

Mitigation of these shortcomings is typically done by using **highly dynamic frameworks**, using **Reflection and Dynamic Proxies** (like Spring Framework), favour **Convention-over-Configuration** (like Spring Boot and Quarkus) or heavily rely on **annotation processing** and **code generation frameworks** (like Lombok).



Fun vs Tedious vs Static vs Dynamic



How can we bring FUN back to Java?

- ❖ Avoid boilerplate code, focus on business logic
- ❖ Provide useful libraries for missing functionality
- ❖ Copy patterns from other languages / tech stacks
- ❖ Extend existing codebases
- ❖ Provide more choices for developers



Fighting Verbosity: Lombok



Ease Java Development & Focus on Logic

```
public class NoLombokPojo {  
  
    private String attribute;  
    private LocalDate created;  
    private BigDecimal value;  
  
    public NoLombokPojo(String attribute, LocalDate created, BigDecimal  
value) {  
        this.attribute = attribute;  
        this.created = created;  
        this.value = value;  
    }  
    public String getAttribute() {  
        return attribute;  
    }  
    public void setAttribute(String attribute) {  
        this.attribute = attribute;  
    }  
    public LocalDate getCreated() {  
        return created;  
    }  
    public void setCreated(LocalDate created) {  
        this.created = created;  
    }  
    public BigDecimal getValue() {  
        return value;  
    }  
    public void setValue(BigDecimal value) {  
        this.value = value;  
    }  
    @Override  
    public boolean equals(Object o) {  
        if (this == o) return true;  
        if (o == null || getClass() != o.getClass()) return false;  
        NoLombokPojo that = (NoLombokPojo) o;  
        return attribute.equals(that.attribute) &&  
            created.equals(that.created) && value.equals(that.value);  
    }  
}
```



lombok

- Avoid repetitive code
- Less error-prone code
- Consistent behaviour over projects
- Best practices baked in

```
@Data  
public class LombokPojo {  
  
    private String attribute;  
    private LocalDate created;  
    private BigDecimal value;  
  
}
```

@Data is the short form for

@RequiredArgsConstructor, @ToString, @EqualsAndHashCode, @Getter, @Setter



Constructors: Automatic Constructor Creation

```
public class LombokConstructorPojo {  
  
    private @NonNull String id;  
    private LocalDate created;  
    private BigDecimal value;  
  
    public LombokConstructorPojo(@NonNull String id) {  
        this.id = id;  
    }  
  
    public LombokConstructorPojo(  
        @NonNull String id,  
        LocalDate created,  
        BigDecimal value) {  
        this.id = id;  
        this.created = created;  
        this.value = value;  
    }  
  
    public LombokConstructorPojo() {  
    }  
  
}
```

- Automatically create different kind of constructors (required, no-args, all-args)
- Beware of conflicts with **@Builder**



lombok

```
@NoArgsConstructor  
@RequiredArgsConstructor  
@AllArgsConstructor  
public class LombokPojo {  
  
    private String attribute;  
    private LocalDate created;  
    private BigDecimal value;  
  
}
```



@Value: Immutable Classes made easy

```
public final class LombokConstructorPojo {  
  
    private final @NonNull String id;  
    private final LocalDate created;  
    private final BigDecimal value;  
  
    public LombokConstructorPojo(  
        @NonNull String id,  
        LocalDate created,  
        BigDecimal value) {  
        this.id = id;  
        this.created = created;  
        this.value = value;  
    }  
  
    public @NonNull String getId() {  
        return this.id;  
    }  
  
    public LocalDate getCreated() {  
        return this.created;  
    } [...]  
}
```



lombok

- Create Immutable classes easily
- Useful for Value classes, thread-safety and statelessness

```
@Value  
public class LombokConstructorPojo {  
  
    private @NonNull String id;  
    private LocalDate created;  
    private BigDecimal value;  
  
}
```



@Builder: Immutable Classes made easy

```
public class LombokConstructorPojo {  
  
    private @NonNull String id;  
    private LocalDate created;  
    private BigDecimal value;  
  
    public static LombokConstructorPojoBuilder builder() {  
        return new LombokConstructorPojoBuilder();  
    }  
  
    public static class LombokConstructorPojoBuilder {  
        private @NonNull String id;  
        private LocalDate created;  
        private BigDecimal value;  
  
        public LombokConstructorPojoBuilder id(@NonNull String id) {  
            this.id = id;  
            return this;  
        }  
  
        public LombokConstructorPojoBuilder created(LocalDate creatd){  
            this.created = creatd;  
            return this;  
        }  
  
        public LombokConstructorPojo build() {  
            return new LombokConstructorPojo(id, created, value);  
        }  
    }  
}
```



lombok

- Produces complex builder APIs for your classes.

```
@Builder  
public class LombokConstructorPojo {  
  
    private @NonNull String id;  
    private LocalDate created;  
    private BigDecimal value;  
  
}
```

```
City.builder()  
    .capital(true)  
    .founded(LocalDate.now())  
    .name("New Java City")  
    .population(42).build();
```



Automatic Printer: @ToString

- Generates **toString** for you with configurable attributes.

```
public class City {  
  
    private @NonNull String name;  
    private @NonNull Integer population;  
    private LocalDate founded;  
    private @ToString.Exclude boolean capital;  
  
    public String toString() {  
        return "City(name=" + this.getName() +  
            ", population=" + this.getPopulation() + ",  
            founded=" + this.getFounded() + ")";  
    }  
}
```



lombok

```
@ToString  
public class City {  
  
    private @NonNull String name;  
    private @NonNull Integer population;  
    private LocalDate founded;  
    private @ToString.Exclude boolean capital;  
  
}
```



Are we the same? @EqualsAndHashCode

```
public class NoLombokPojo {

    public boolean equals(final Object o) {
        if (o == this) return true;
        if (!(o instanceof City)) return false;
        final City other = (City) o;
        if (!other.canEqual((Object) this)) return false;
        final Object this$name = this.getName();
        final Object other$name = other.getName();
        if (this$name == null ? other$name != null :
            !this$name.equals(other$name)) return false;
        final Object this$population = this.getPopulation();
        final Object other$population = other.getPopulation();
        if (this$population == null ? other$population != null :
            !this$population.equals(other$population))
            return false;
        final Object this$founded = this.getFounded();
        final Object other$founded = other.getFounded();
        if (this$founded == null ? other$founded != null :
            !this$founded.equals(other$founded)) return false;
        if (this.isCapital() != other.isCapital()) return false;
        return true;
    }

    public int hashCode() {
        final int PRIME = 59;
        int result = 1;
        final Object $name = this.getName();
        result = result * PRIME + ($name == null ? 43 : $name.hashCode());
        final Object $population = this.getPopulation();
        result = result * PRIME + ($population == null ? 43 :
            $population.hashCode());
        final Object $founded = this.getFounded();
        result = result * PRIME + ($founded == null ? 43 : $founded.hashCode());
        result = result * PRIME + (this.isCapital() ? 79 : 97);
        return result;
    }
}
```

lombok

- Generates **equals** and **hashCode** so that semantics are correct.

```
@EqualsAndHashCode
public class City {

    private @NonNull String name;
    private @NonNull Integer population;
    private LocalDate founded;
    private @ToString.Exclude boolean capital;

}
```



Pluggable Logging with SLF4J



- Logging against **Interfaces**
- Helps with **quick switching** of Implementation

```
public class LombokLog {  
  
    private static final Logger log =  
        org.slf4j.LoggerFactory.getLogger(LombokConstructorPojo.class);  
  
    [...]
```

lombok

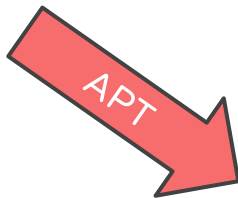
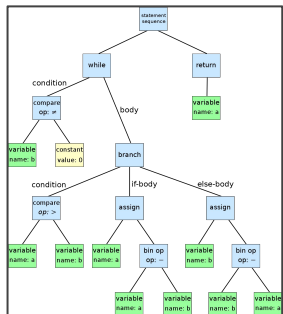
```
@Slf4j  
public class LombokLog {  
  
    [...]  
    log.info("Hello from Logger");  
}
```




How does Lombok work?

```
@Slf4j
public class LombokLog {

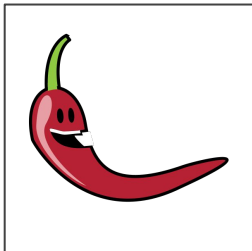
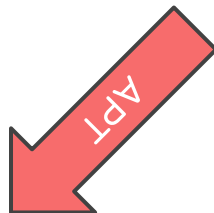
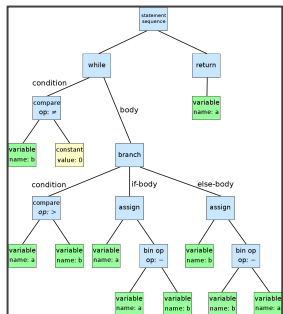
    [...]
    log.info("Hello from Logger");
}
```



JSR 269
Pluggable
Annotation
Processing
API



```
public static void
main(java.lang.String[]);
descriptor: ([Ljava/lang/String;)V
flags: (0x0009) ACC_PUBLIC, ACC_STATIC
Code:
stack=2, locals=4, args_size=1
0: iconst_1
1: istore_1
2: iconst_2
3: istore_2
4: iload_1
5: iload_2
6: iadd
7: istore_3
8: return
```

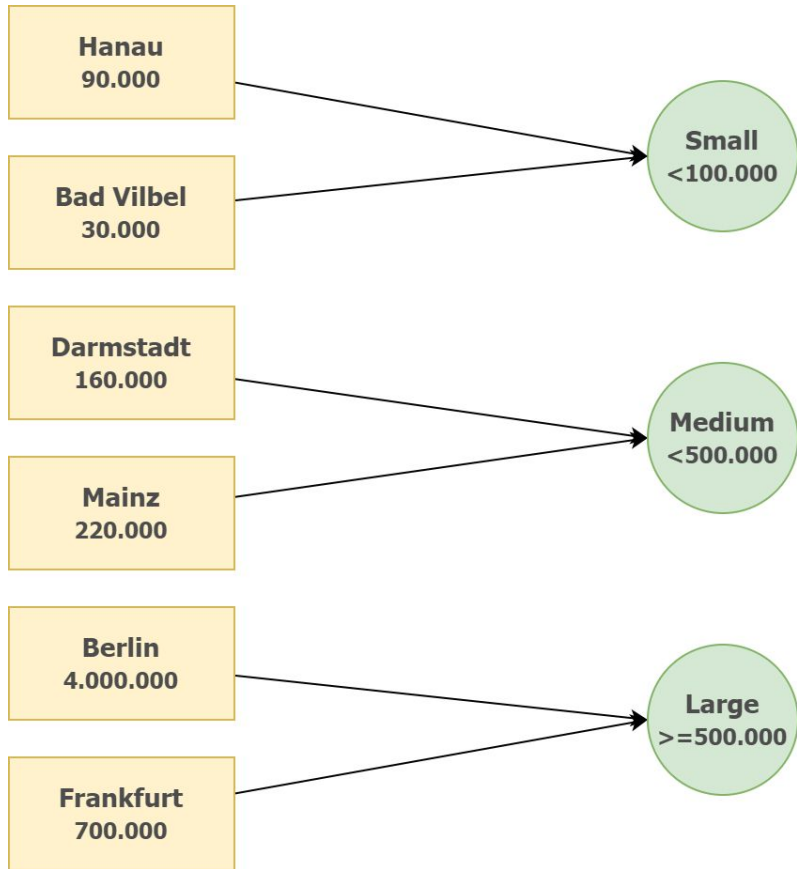




Google Core Libraries for Java: Guava



City Lights - Sample Collection



```
@Data
public class City {

    private @NonNull String name;
    private @NonNull Integer population;
    private LocalDate founded;
    private boolean capital;

}
```

Java Swiss Army Knife: Apache Commons



Reason for Payment - Sample Texts

Abbreviate the text to max 20 characters.

Please make sure that no words are cut.

"Thanks for the nice Lunch."

"Max Muster Invoice No 551662781"

"Remittance Item #188655"

"September 2022 Loan Repayment Max Mustermann"

"Invoice Reference No 8988826/2022/145"

"Thanks for the nice..."

"Max Muster Invoice..."

"Remittance Item #188..."

"September 2022 Loan..."

"Invoice Reference..."

"Thanks for the ni..."

"Max Muster Invoic..."

"Remittance Item #..."

"September 2022 Lo..."

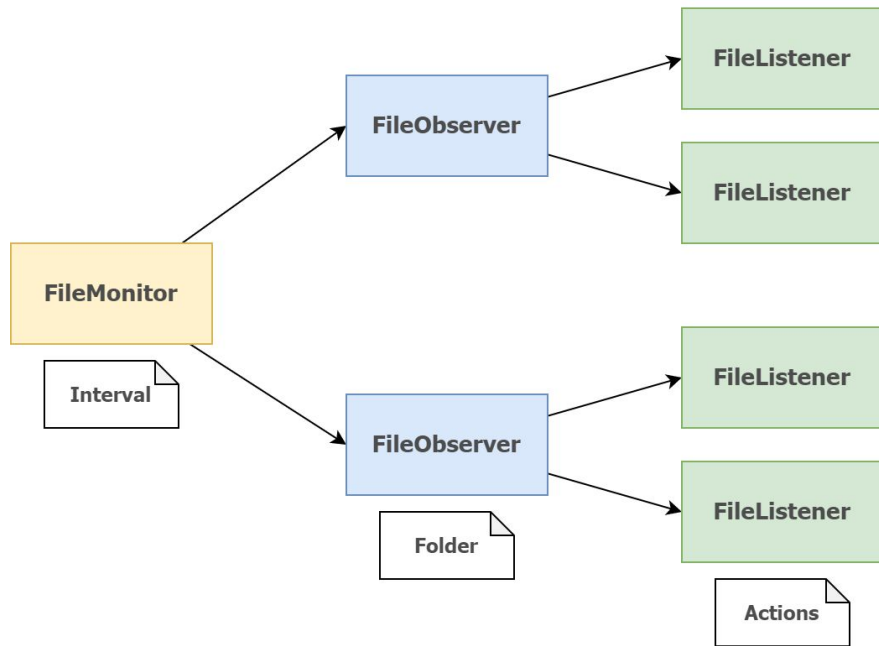
"Invoice Reference..."



Watch Out! ...for File Changes...

If there is a new Excel file, import it.

This might also be other files. And other folders. And different intervals.



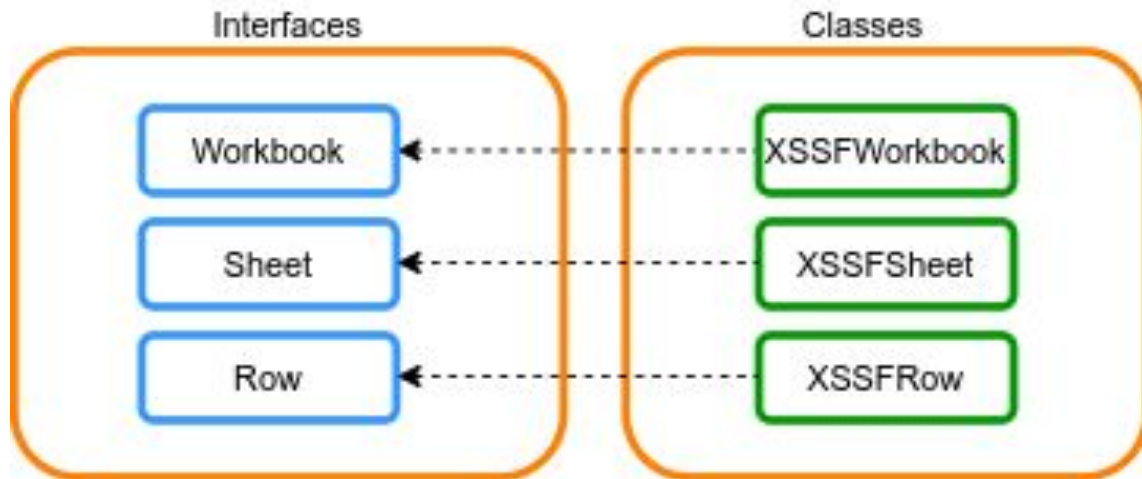
The Java API for Microsoft Documents: Apache POI



Back to Excel with Apache POI

Change this one field in Excel with the correct value.

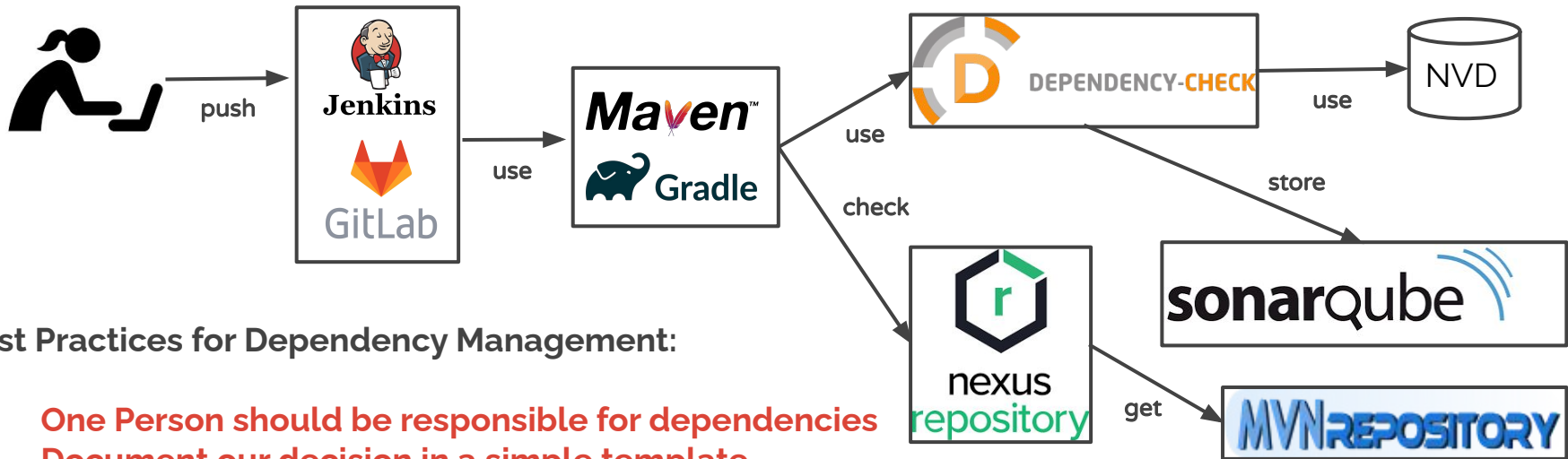
But please leave everything else as is, the people love it like it is!



How to add a
dependency -
without the team
hating you?



How to add this dependency - without all others hating you?



Best Practices for Dependency Management:

- One Person should be responsible for dependencies
- Document our decision in a simple template
- Use OWASP Dependency Check
- Regularly check for outdated dependencies
- Decide on KPIs for dependencies beforehand

Dependency Management: Maven, Gradle

Static Code Analysis Tools: SonarQube

Continuous Integration: Jenkins, GitLab

Component Dependency Analysis: Dependency Check/Track



Links and other information

Project Lombok

- ❖ Project Lombok: <https://projectlombok.org/features/all>
- ❖ Baeldung on Lombok: <https://www.baeldung.com/intro-to-project-lombok>

Google Guava

- ❖ Baeldung on Guava Maps: <https://www.baeldung.com/guava-maps>
- ❖ Baeldung on Guava StringUtils: <https://www.baeldung.com/guava-string-charmatcher>
- ❖ Java Dev Central on Guava: <https://javadevcentral.com/category/google-guava>
- ❖ Guava Documentation: <https://github.com/google/guava/wiki/>

Apache Commons

- ❖ Apache Commons (Proper): <https://commons.apache.org/>
- ❖ Java Dev Central on Commons: <https://javadevcentral.com/category/apache-commons>
- ❖ Java Dev Central on Commons Text: <https://javadevcentral.com/apache-commons-text-wordutils>
- ❖ Java Dev Central on Commons Lang: <https://javadevcentral.com/apache-commons-lang-randomstringutils>



Links and other information

Apache POI

- ❖ Baeldung on Apache POI: <https://www.baeldung.com/java-microsoft-excel>
- ❖ RFC 4180 (CSV): <https://datatracker.ietf.org/doc/html/rfc4180>

Decision Documentation

- ❖ Decision Templates: <https://github.com/joelparkerhenderson/architecture-decision-record>