



Framework Less

Hervé Letourneur & Alban Clevy

12/03/2024



Qui sommes-nous ?

Alban CLEVY

15 années d'expérience professionnelle dans l'informatique (Tech Lead / Architecte / Développeur / Responsable d'équipe / Chef de projet ...)

- 3 ans Société Général en tant que responsable d'équipe
- 12 ans au sein d'ESN dont 5 ans chez Meritis, consultant / RSI / responsable Practice Techlead - Architecture

Hervé LETOURNEUR

16 années d'expérience professionnelle dans le développement (TechLead, Developpeur, Responsable d'équipe)

16 ans au sein d'ESN dont 10 chez Méritis, Consultant/Reponsable Practice Java

Réseaux sociaux :



Spring Boot

Record

JDBI

Hibernate

Framework

Compilation

Lombok

Librairie

Martean

Spring Data

Javalin

C'est quoi un Framework ???

En programmation informatique, un framework est un ensemble cohérent de composants logiciels structurels qui sert à créer les fondations ainsi que les grandes lignes de tout ou partie d'un logiciel, c'est-à-dire une architecture.

Un Framework peut néanmoins être spécialisé dans un langage particulier.

Il impose un cadre de travail, dû à sa construction même, guidant l'architecture logicielle voire conduisant le développeur à respecter certains patrons de conception.

Besoins




BricoShop est une grande société de Bricolage

Son besoin :

Système de gestion de produit à installer sur des mini ordinateurs faibles en ressources

Choix de l'outil



Project

☒ Gradle - Groovy ☐ Gradle - Kotlin ☐ Maven

Language

☒ Java ☐ Kotlin ☐ Groovy

Spring Boot

☐ 3.3.0 (SNAPSHOT) ☐ 3.3.0 (M2) ☐ 3.2.4 (SNAPSHOT) ☒ 3.2.3 ☐ 3.1.10 (SNAPSHOT) ☐ 3.1.9

Project Metadata

Group

Artifact

Name

Description

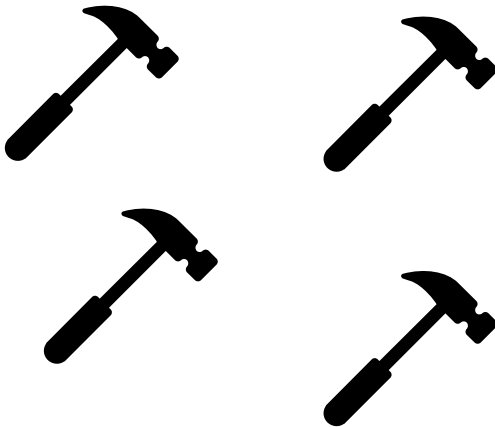
Package name

Packaging ☒ Jar ☐ War

Java ☒ 21 ☐ 17

Déjà du code ?

```
@SpringBootApplication
public class HammerApplication {
    public static void main(String[] args) {
        SpringApplication.run(HammerApplication.class);
    }
}
```



```
no usages
@RestController
public class HammerController {

    2 usages
    private final HammerService hammerService;

    no usages
    @Autowired
    public HammerController(HammerService hammerService) {
        this.hammerService = hammerService;
    }

    no usages
    @GetMapping("/helloworld")
    public ResponseEntity<String> helloworld() { return ResponseEntity.ok( body: "HelloWorld"); }

    no usages
    @GetMapping("/hammer/{id}")
    public ResponseEntity<Hammer> getHammer(@PathVariable(value = "id") UUID id) {
        Hammer hammer = hammerService.retrieveById(id);
        return ResponseEntity.ok(hammer);
    }
}
```

Et si on gagnait du temps et du Code ?

```
1 usage
public class Hammer {
    7 usages
    public int size;
    7 usages
    public int weight;

    @Override
    public String toString() {...}

    @Override
    public boolean equals(Object o) {...}

    @Override
    public int hashCode() { return Objects.hash(size, weight); }

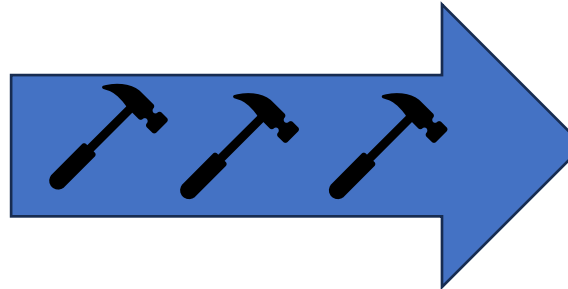
    no usages
    public Hammer(int size, int weight) {...}

    no usages
    public int getSize() { return size; }

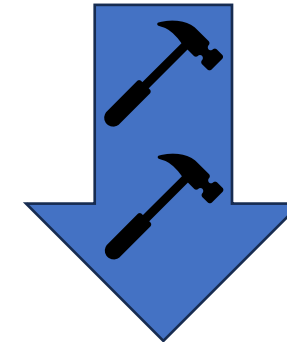
    no usages
    public void setSize(int size) { this.size = size; }

    no usages
    public int getWeight() { return weight; }

    no usages
    public void setWeight(int weight) { this.weight = weight; }
}
```

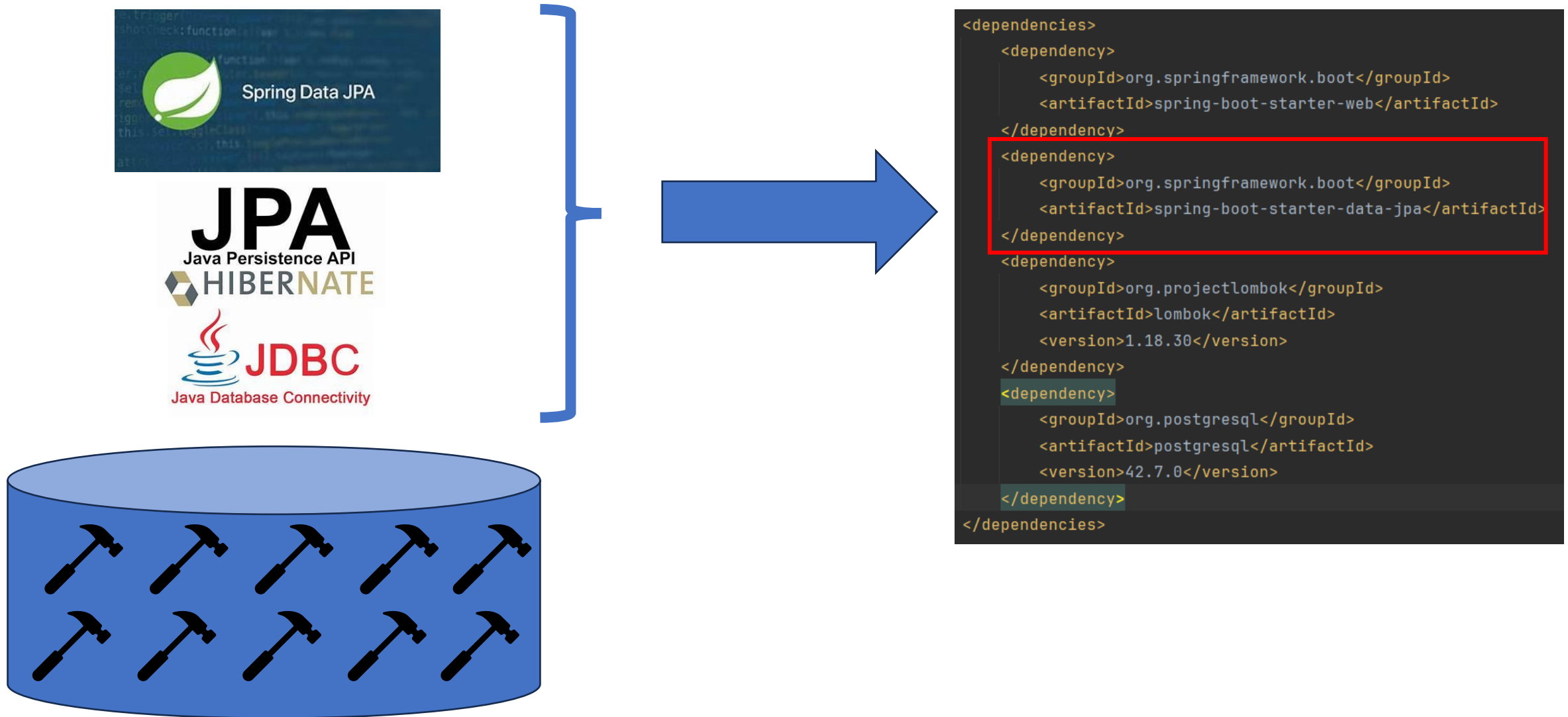


```
no usages
@Getter
@Setter
@ToString
@EqualsAndHashCode
@AllArgsConstructor
public class Hammer {
    public int size;
    public int weight;
}
```



```
no usages
@Data
public class Hammer {
    public int size;
    public int weight;
}
```


Mais comment communiquer avec la base ?



On modélise comment nos entités ?

```
usages
@Entity(name = "trade_mark_hammer")
@Data
@NoArgsConstructor
@AllArgsConstructor
public class HammerTradeMarkEntity {
    @Id
    private UUID id;

    @Column(name = "trade_mark_name")
    private String name;

    @OneToMany(mappedBy = "hammerTradeMark")
    @Fetch(FetchMode.JOIN)
    private List<HammerEntity> hammers;
}
```



```
@Data
@AllArgsConstructor
@NoArgsConstructor
@Entity(name = "Hammer")
public class HammerEntity {
    @Id
    private UUID id;

    @Column
    private String Name;

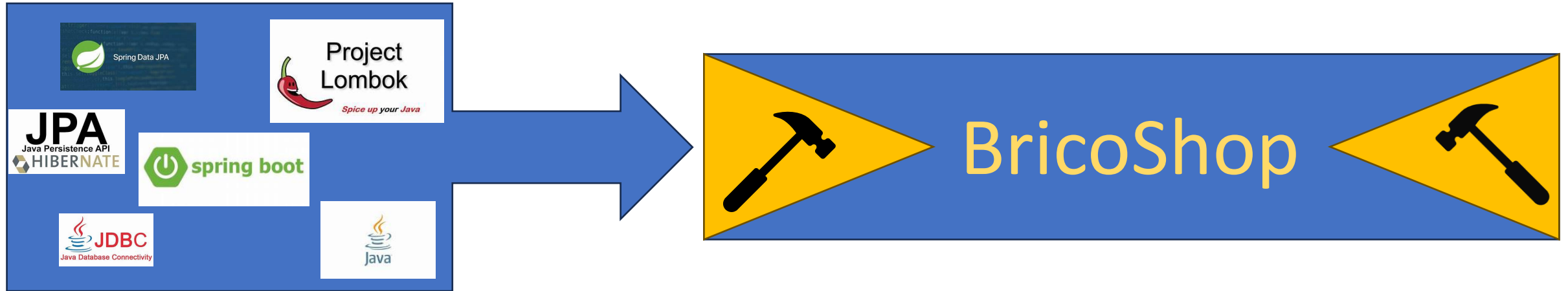
    @Column
    private int size;

    @Column
    private int weight;

    @ManyToOne
    @JoinColumn(name = "trade_mark_id")
    private HammerTradeMarkEntity hammerTradeMark;
}
```

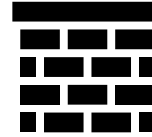
```
usages
public interface HammerRepository extends JpaRepository<HammerEntity, UUID> {
}
```

On livre en Production ?



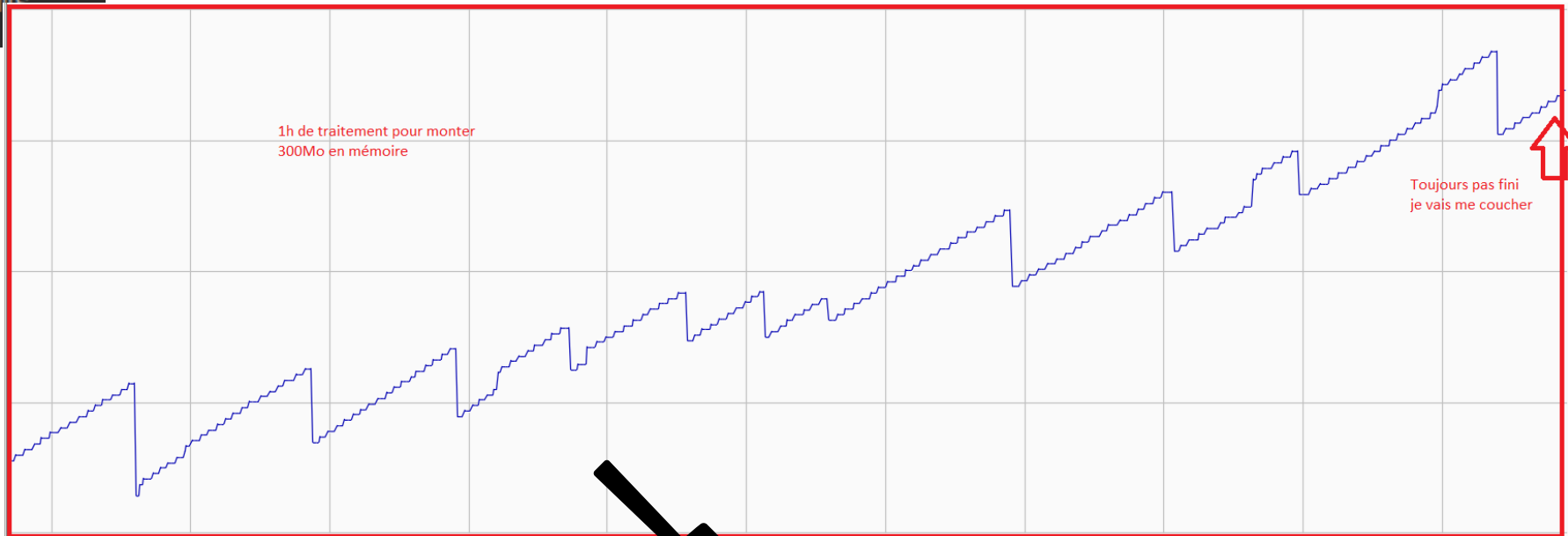
50 marques et 100 marteaux différents par marques au total 5050 entités chargés en mémoire en 60 ms

Est-ce que le client est content ?

[illegible]

50 000 marques et 100 marteaux différents par marques au total 5 050 000 entité à charger en mémoire

framework-1.0-SNAPSHOT-jar-with-dependencies.jar	10/03/2024 16:42	Fichier JAR	81 160 Ko
--	------------------	-------------	-----------



Besoins



BricoShop est une grande société de Bricolage

Son besoin :

Systeme de gestion de produit à installer sur des mini ordinateurs faibles en ressources

Choix de l'outil



A simple web framework for Java and Kotlin

Java

Kotlin

Copy

```
import io.javalin.Javalin;

public class HelloWorld {
    public static void main(String[] args) {
        var app = Javalin.create(/*config*/)
            .get("/", ctx -> ctx.result("Hello World"))
            .start(7070);
    }
}
```

Plus de code ?

```
public class HammerApplication {  
    public static void main(String[] args) {  
        Javalin.create(new ControllerFactory()::createAndBindController)  
            .start(port: 8081);  
    }  
}
```



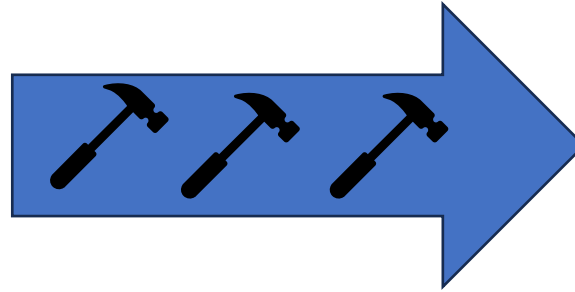
```
public class ControllerFactory {  
    1 usage  
    public void createAndBindController(JavalinConfig config) {  
        Jdbi jdbi = new JdbiConfiguration().createJdbi();  
        HammerDao hammerDao = new HammerDao(jdbi);  
        HammerTradeMarkDao hammerTradeMarkDao = new HammerTradeMarkDao(jdbi);  
        HammerService hammerService = new HammerService(hammerDao);  
        HammerTradeMarkService hammerTradeMarkService = new HammerTradeMarkService(hammerTradeMarkDao);  
        HammerController hammerController = new HammerController(hammerService, hammerTradeMarkService);  
        config.validation.register(HammerId.class, HammerId::fromString);  
        config.validation.register(HammerTradeMarkId.class, HammerTradeMarkId::fromString);  
        config.router.apiBuilder(hammerController::routes);  
    }  
}
```

```
public class HammerController implements Controller {  
    2 usages  
    private final HammerService hammerService;  
    3 usages  
    private final HammerTradeMarkService hammerTradeMarkService;  
  
    1 usage  
    public HammerController(HammerService hammerService, HammerTradeMarkService hammerTradeMarkService) {  
        this.hammerService = hammerService;  
        this.hammerTradeMarkService = hammerTradeMarkService;  
    }  
  
    1 usage  
    public void routes() {  
        get(path: "/helloworld", this::helloworld);  
        get(path: "/hammer/{id}", this::retrieveHammer);  
        get(path: "/tradeMark/{id}/hammers", this::retrieveHammerTradeMark);  
        get(path: "/tradeMarks/hammers", this::retrieveAll);  
    }  
  
    1 usage  
    public void helloworld(Context ctx) { ctx.result(resultString: "HelloWorld !").status(200); }  
  
    1 usage  
    public void retrieveHammer(Context ctx) {  
        Hammer hammer = hammerService.retrieveById(ctx.pathParamAsClass(key: "id", HammerId.class).get());  
        ctx.json(hammer).status(200);  
    }  
}
```



Et si on gagnait du temps et du Code ?

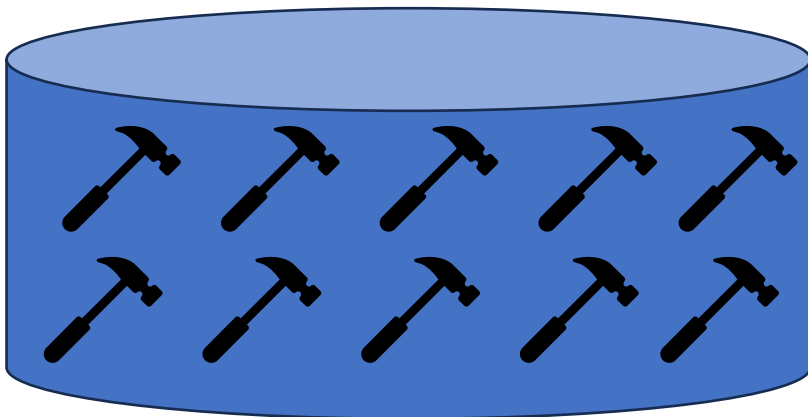
```
public class Hammer {  
    5 usages  
    public int size;  
    5 usages  
    public int weight;  
    public String name;  
  
    public Hammer(int size, int weight, String name) {  
        this.size = size;  
        this.weight = weight;  
        this.name = name;  
    }  
  
    @Override  
    public boolean equals(Object o) {  
        if (this == o)  
            return true;  
        if (o == null || getClass() != o.getClass())  
            return false;  
        Hammer hammer = (Hammer) o;  
        return size == hammer.size && weight == hammer.weight && Objects.equals(name, hammer.name);  
    }  
  
    @Override  
    public int hashCode() { return Objects.hash(size, weight, name); }  
  
    @Override  
    public String toString() { return "Hammer{" + "size=" + size + ", weight=" + weight + ", name=" + name + '\n' + '}'; }  
}
```



```
public record Hammer(double weight, double size, String name) {  
}
```



Mais comment communiquer avec la base ?



```
<dependency>
  <groupId>io.javalin</groupId>
  <artifactId>javalin</artifactId>
  <version>6.1.3</version>
</dependency>
<dependency>
  <groupId>org.jdbi</groupId>
  <artifactId>jdbi3-core</artifactId>
  <version>3.41.3</version>
  <scope>compile</scope>
</dependency>
<dependency>
  <groupId>org.postgresql</groupId>
  <artifactId>postgresql</artifactId>
  <version>42.7.0</version>
</dependency>
<dependency>
  <groupId>com.fasterxml.jackson.core</groupId>
  <artifactId>jackson-databind</artifactId>
  <version>2.16.1</version>
</dependency>
```

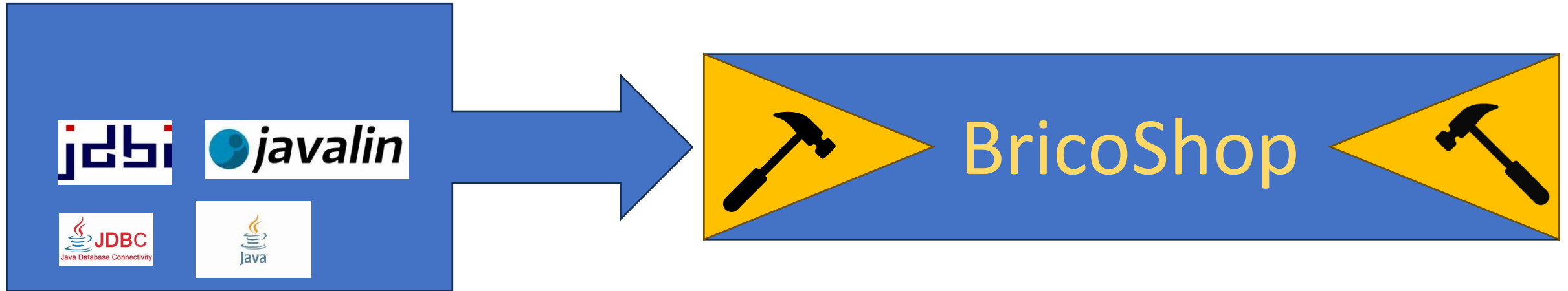
On modélise comment nos entités ?

```
public record HammerEntity(UUID id, String name, int size, int weight) {  
}
```



```
public class HammerDao {  
    2 usages  
    private final Jdbi jdbi;  
  
    1 usage  
    public HammerDao(Jdbi jdbi) {  
        this.jdbi = jdbi;  
        jdbi.registerRowMapper(ConstructorMapper.factory(HammerEntity.class));  
    }  
  
    1 usage  
    public HammerEntity retrieveHammerById(HammerId hammerId) {  
        return jdbi.withHandle(handle -> handle.createQuery( sql: ""  
            Select id, name, size, height  
            from hammer  
            where id = :id  
            """" ) Query  
            .bind( name: "id", hammerId.id()  
            .mapTo(HammerEntity.class) ResultIterable<HammerEntity>  
            .one());  
    }  
}
```

On livre en Production ?



50 000 marques et 100 marteaux différents par marques
au total 5 050 000 entités chargés en mémoire en 10 sec

Récapitulatif

Une application simple qui dit bonjour, retrouver des marteaux par id et chargé l'ensemble des marteaux par marques dans un environnements limité en ressources

Tout Framework :

Temps de démarrage	4,5 sec
Taille du livrable	80 Mo
Temps de build	10 sec
Nb dépendance	62
Quantité de ligne de code	Faible
Capacité à gérer du volume	50 000 entités

FrameworkLess :

Temps de démarrage	< 1 sec
Taille du livrable	7 Mo
Temps de build	6,5 sec
Nb dépendance	27
Quantité de ligne de code	Modéré
Capacité à gérer du volume	5 000 000 entités

Avantages / Inconvénient

Tout Framework

Standard sur le marché
Beaucoup de fonctionnalité déjà écrite
Moins de code

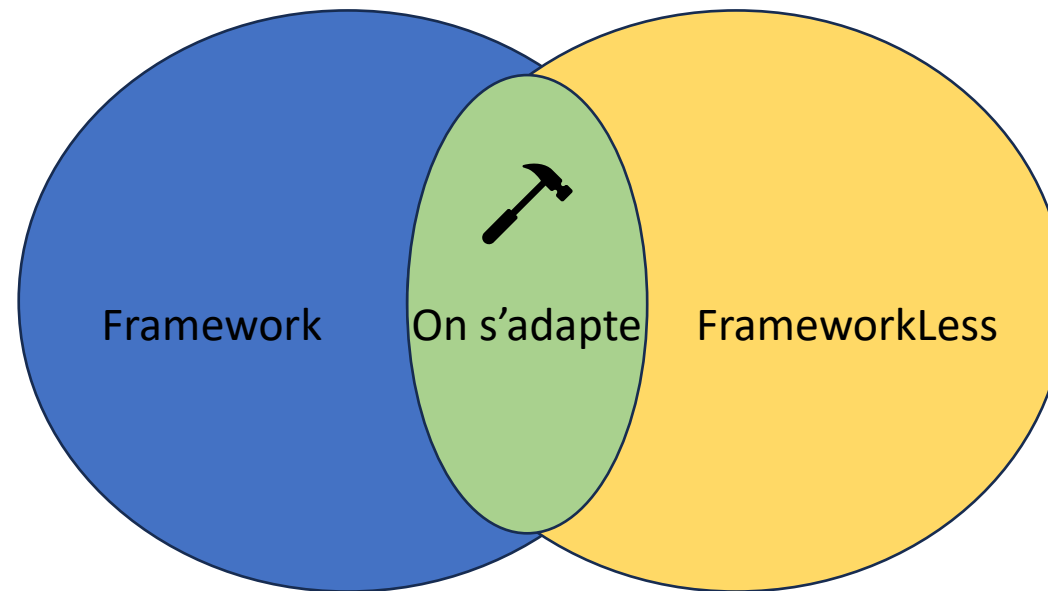
Contraint par le Framework
Boite noire
Lourd

(Moins) FrameworkLess

Léger
Peu contraignant
Stack technologique plus simple
Meilleure visibilité sur l'application

Plus de code à faire
Hors des Standards habituels

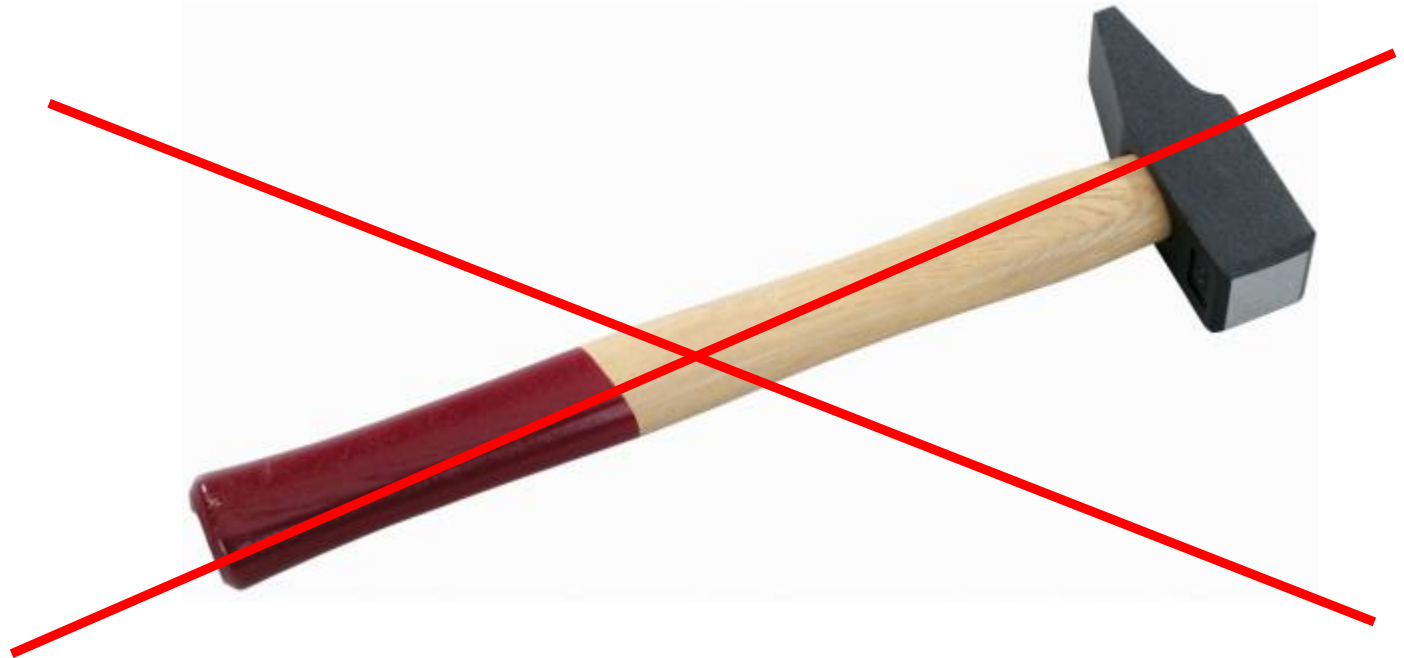
Notre Quotidien ?



Conclusion



Conclusion





Références

Git

- <https://github.com/b16d/FrameworkLess2>

Wiki

- <https://fr.wikipedia.org/wiki/Framework>

