

# Injection de dépendances

Une application :

- doit répondre aux besoins actuels
- mais aussi être évolutive

Objectif :


faciliter la maintenance du code en impactant le moins possible l'existant.

Solution → réduire les dépendances au sein du code.

# Couplage fort

Product ne peut pas être utilisé  
ni testé indépendamment de Command

```
public class Command {  
    private Product product;  
  
    public Command() {  
        this.product = new Product("something");  
        System.out.println(product);  
    }  
}
```

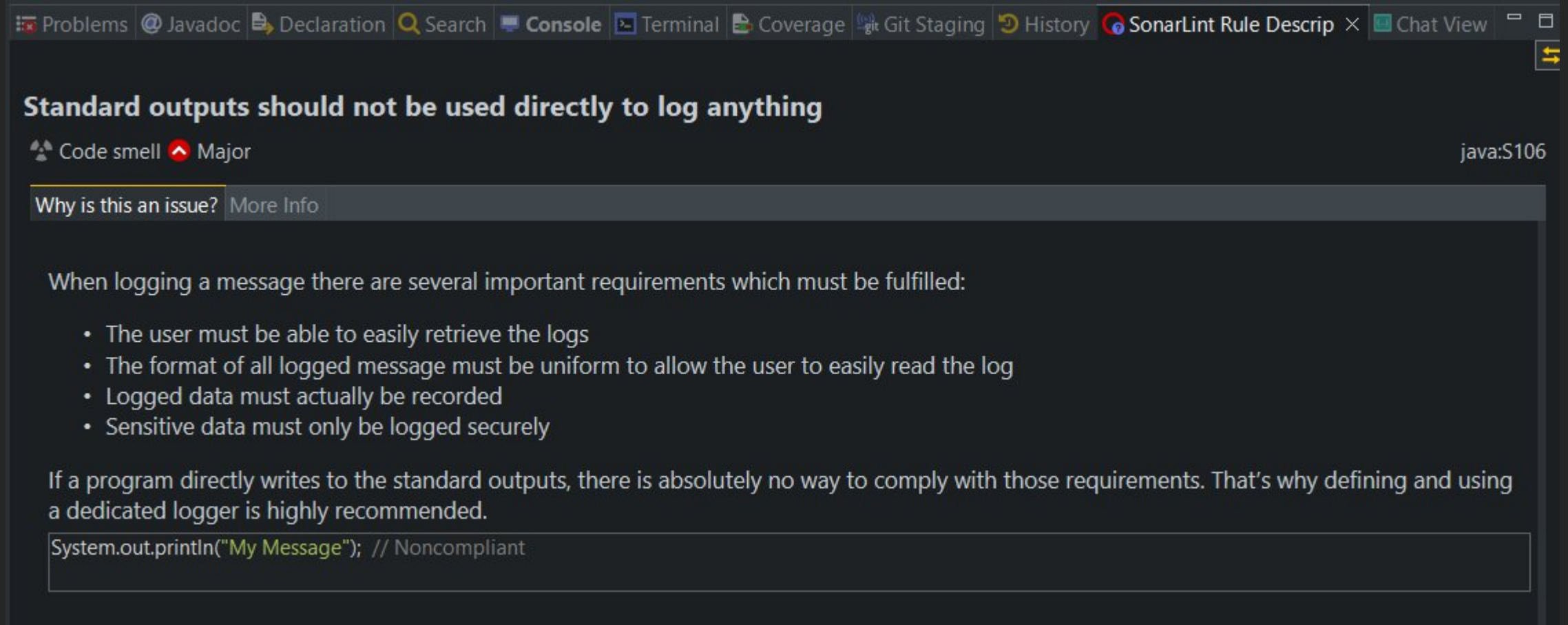


```
public class MainDependanceExemple {  
  
    public static void main(String[] args) {  
        Command command = new Command();  
    }  
}
```

```
public class Product {  
    private String productName = "";  
  
    public Product(String productName) {  
        this.productName = productName;  
    }  
  
    @Override  
    public String toString() {  
        return "Product [productName=" + productName + "]";  
    }  
}
```

# Note sur le logging

`System.out.println(product);` → Règle générée/rapellée par Sonarlint



The screenshot shows the SonarLint interface in an IDE. The top toolbar includes tabs for Problems, Javadoc, Declaration, Search, Console, Terminal, Coverage, Git Staging, History, SonarLint Rule Descrip (active), and Chat View. The main panel displays a rule violation titled "Standard outputs should not be used directly to log anything" with a severity of "Major" (indicated by a red upward arrow) and a category of "Code smell". The rule ID is "java:S106". Below the title, there are two tabs: "Why is this an issue?" (selected) and "More Info". The content under "Why is this an issue?" explains that logging to standard outputs is non-compliant because it fails several requirements: easy retrieval, uniform format, actual recording, and secure logging of sensitive data. It recommends using a dedicated logger. A code snippet is shown in a box: `System.out.println("My Message"); // Noncompliant`. A red arrow points to this code snippet from the left.

**Standard outputs should not be used directly to log anything**  
Code smell Major java:S106

Why is this an issue? More Info

When logging a message there are several important requirements which must be fulfilled:

- The user must be able to easily retrieve the logs
- The format of all logged message must be uniform to allow the user to easily read the log
- Logged data must actually be recorded
- Sensitive data must only be logged securely

If a program directly writes to the standard outputs, there is absolutely no way to comply with those requirements. That's why defining and using a dedicated logger is highly recommended.

```
System.out.println("My Message"); // Noncompliant
```

Le logging pour : déboguer, traces d'exécution, recherches d'anomalie, etc.

Voir : <https://www.jmdoudoux.fr/java/dej/chap-logging.htm> , logiciel Log4j

# Pattern Dependency Injection

New Product() a disparu (Command ne s'occupe plus de l'instanciation)

setProduct() insère l'objet instancié dans l'objet Command : *injection par le setter*

```
public class Command {
    private Product product;

    public Product getProduct() {
        return product;
    }

    public void setProduct(Product product) {
        this.product = product;
    }

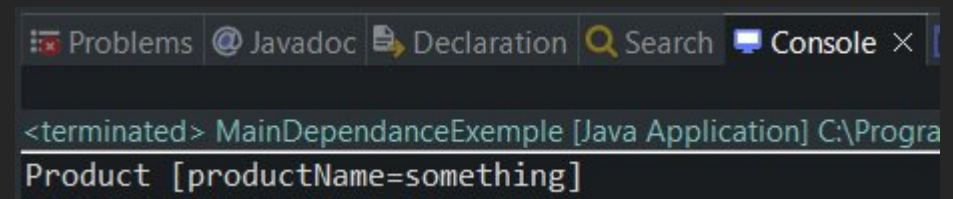
    public Command() {
        // TODO
    }
}
```

```
public class MainDependanceExemple {

    public static void main(String[] args) {

        Product product = new Product("something");
        Command command = new Command();
        command.setProduct(product);
        System.out.println(command.getProduct());

    }
}
```



The screenshot shows the bottom part of an IDE window with a console tab. The console output displays the result of the program execution: "Product [productName=something]".

```
<terminated> MainDependanceExemple [Java Application] C:\Progra
Product [productName=something]
```

# Pattern Dependency Injection

setProduct() est "caché" dans Command (diapo d'avant), alors :

Injection de l'objet Product dans le constructeur de Command : *injection par le constructeur*

Idem avec l'ajout de l'objet Delivery supplémentaire

```
class Command {
    private Product product;
    private Delivery delivery;

    public Command(Product product, Delivery delivery)
    {
        this.product = product;
        this.delivery = delivery;
    }
}
```

```
public class MainDependanceExemple {

    public static void main(String[] args) {

        // dependencies
        Product product = new Product("something");
        Delivery delivery = new Delivery(product);

        // injections
        Command command = new Command(product, delivery);
    }
}
```

```
class Delivery {
    private Product product;

    public Delivery(Product product) {
        this.product = product;
    }
}
```

# Inversion of Control (IoC)

La construction des objets est déléguée à un conteneur IoC (IoC container)

Une interface s'écrit comme une classe

Spring Framework fournit avant tout un conteneur IoC (création d'objets, dépendances entre eux)

```
public class IoC {  
  
    public static void main(String[] args) {  
  
        IoC container = new IoC();  
        // On peut aussi bien mettre MySQLDatabase que  
        // OracleDatabase  
        User user = container.new User(container.new OracleDatabase());  
        user.add("This is some data!");  
    }  
  
    // Business Layer Logic  
    public class User {  
  
        private IDatabase database;  
  
        public User(IDatabase database) {  
            this.database = database;  
        }  
  
        public void add(String data) {  
            database.persist(data);  
        }  
    }  
}
```

```
public interface IDatabase {  
    void persist(String data);  
}  
  
// Database Access Layer  
public class MySQLDatabase implements IDatabase {  
    @Override  
    public void persist(String data) {  
        System.out.println("Persisting: " + data);  
    }  
}  
  
public class OracleDatabase implements IDatabase {  
    @Override  
    public void persist(String data) {  
        System.out.println("Persisting: " + data);  
    }  
}  
}
```

# Framework Spring

Si beaucoup de classes → beaucoup de New @Bean avec le Framework Spring :

- Instantiation (singleton par défaut)
- gestion du cycle de vie des objets
- code plus clair

Spring Boot peut être considéré comme extension plus pratique de Spring.

Spring Tools 4 for Eclipse (outils de dev.)

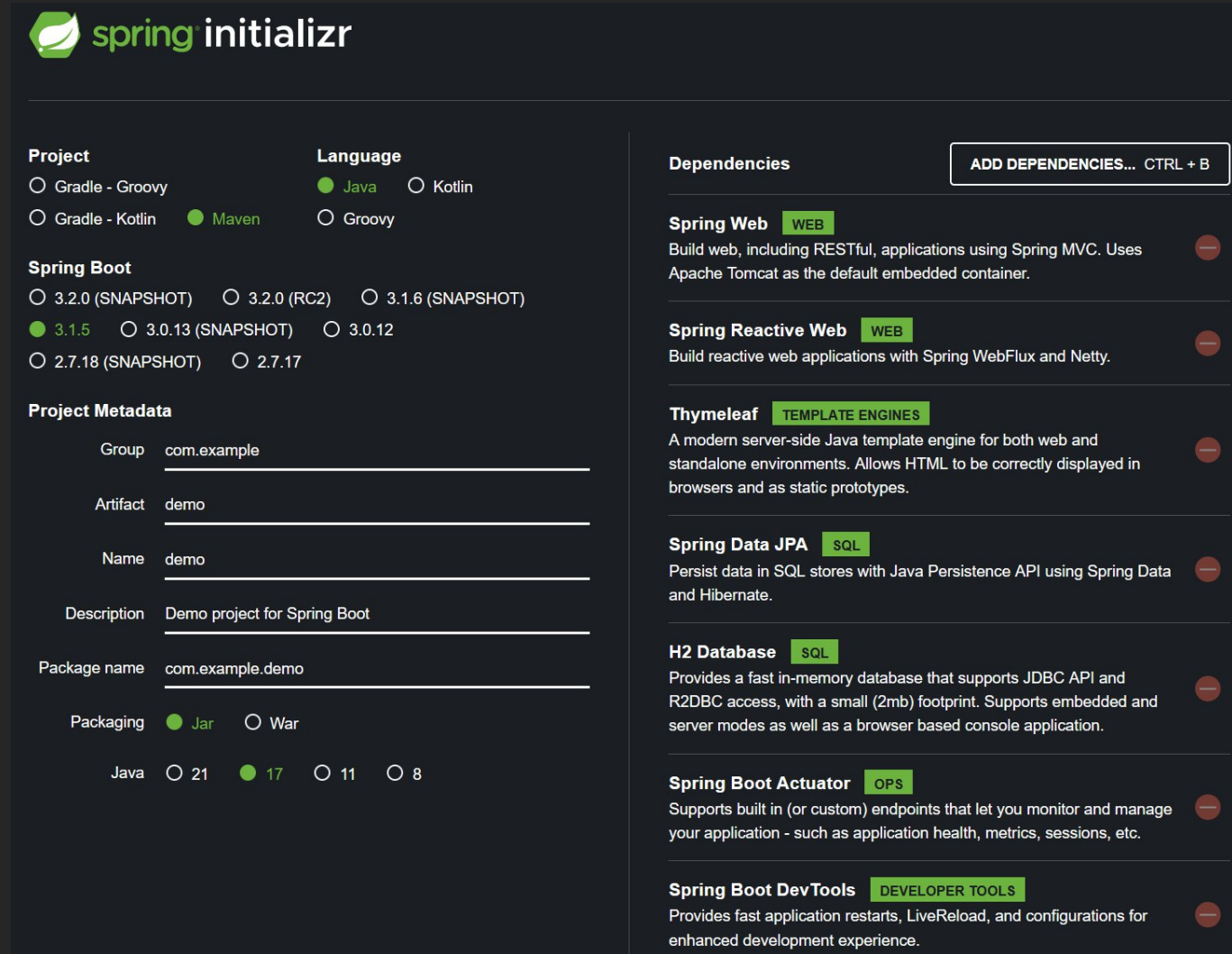
<https://spring.io/tools>

Spring initializr :

<https://start.spring.io/>

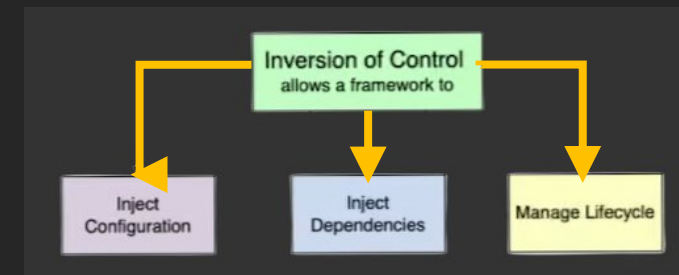
Dependency Injection and IoC :

<https://www.youtube.com/watch?v=EPv9-cHEmQw>



The screenshot shows the Spring Initializr web application interface. It is divided into several sections:

- Project:** Options for Gradle (Groovy, Kotlin) and Maven (selected).
- Language:** Options for Java (selected), Kotlin, and Groovy.
- Spring Boot:** Version selection (3.1.5 selected, others are snapshots or RC2).
- Project Metadata:** Fields for Group (com.example), Artifact (demo), Name (demo), Description (Demo project for Spring Boot), Package name (com.example.demo), and Packaging (Jar selected, War unselected).
- Dependencies:** A list of dependencies with category tags and descriptions:
  - Spring Web (WEB):** Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.
  - Spring Reactive Web (WEB):** Build reactive web applications with Spring WebFlux and Netty.
  - Thymeleaf (TEMPLATE ENGINES):** A modern server-side Java template engine for both web and standalone environments. Allows HTML to be correctly displayed in browsers and as static prototypes.
  - Spring Data JPA (SQL):** Persist data in SQL stores with Java Persistence API using Spring Data and Hibernate.
  - H2 Database (SQL):** Provides a fast in-memory database that supports JDBC API and R2DBC access, with a small (2mb) footprint. Supports embedded and server modes as well as a browser based console application.
  - Spring Boot Actuator (OPS):** Supports built in (or custom) endpoints that let you monitor and manage your application - such as application health, metrics, sessions, etc.
  - Spring Boot DevTools (DEVELOPER TOOLS):** Provides fast application restarts, LiveReload, and configurations for enhanced development experience.





# Framework Spring



## Microservices

Quickly deliver production-grade features with independently evolvable microservices.



## Reactive

Spring's asynchronous, nonblocking architecture means you can get more from your computing resources.



## Cloud

Your code, any cloud—we've got you covered. Connect and scale your services, whatever your platform.



## Web apps

Frameworks for fast, secure, and responsive web applications connected to any data store.



## Serverless

The ultimate flexibility. Scale up on demand and scale to zero when there's no demand.



## Event Driven

Integrate with your enterprise. React to business events. Act on your streaming data in realtime.



## Batch

Automated tasks. Offline processing of data at a time to suit you.

## Difference Between Spring Framework and Spring Boot



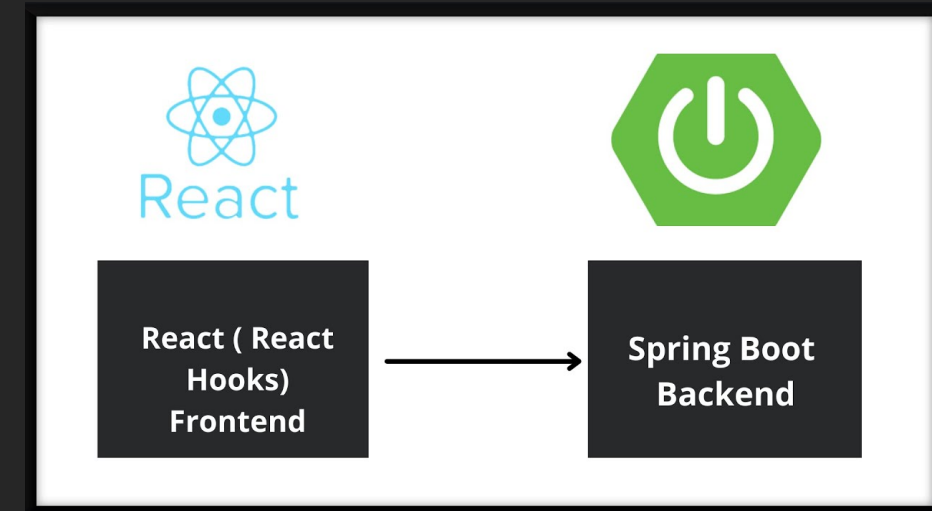
Spring Framework	Spring Boot
Spring is an open-source lightweight framework widely used to develop enterprise applications.	Spring Boot is built on top of the conventional spring framework, widely used to develop REST APIs.
To test the Spring project, we need to set up the sever explicitly.	Spring Boot offers <b>embedded server</b> such as <b>Jetty</b> and <b>Tomcat</b> , etc.
It helps to create a loosely coupled application.	It helps to create a stand-alone application.
The most important feature of the Spring Framework is dependency injection.	The most important feature of the Spring Boot is Autoconfiguration.
It does not provide support for an in-memory database.	It offers several plugins for working with an embedded and <b>in-memory</b> database such as <b>H2</b> .
Developers manually define dependencies for the Spring project in <b>pom.xml</b> .	Spring Boot comes with the concept of <b>starter</b> in pom.xml file that internally takes care of downloading the dependencies <b>JARs</b> based on Spring Boot Requirement.
Configurations are done in xml file	In spring boot, XML configuration is eliminated as it has autoconfiguration feature
To create a Spring application, the developers write lots of code.	It reduces the lines of code.



# Spring Boot et REST

## Avantages de Spring Boot :

- Optimisation de la gestion des dépendances (starters)
- Autoconfiguration (permet de se concentrer sur le code)
- Gestion des propriétés (fichier applications.properties)
- Monitoring et gestion du programme (endpoints Actuator)
- Déploiement (un simple fichier jar)

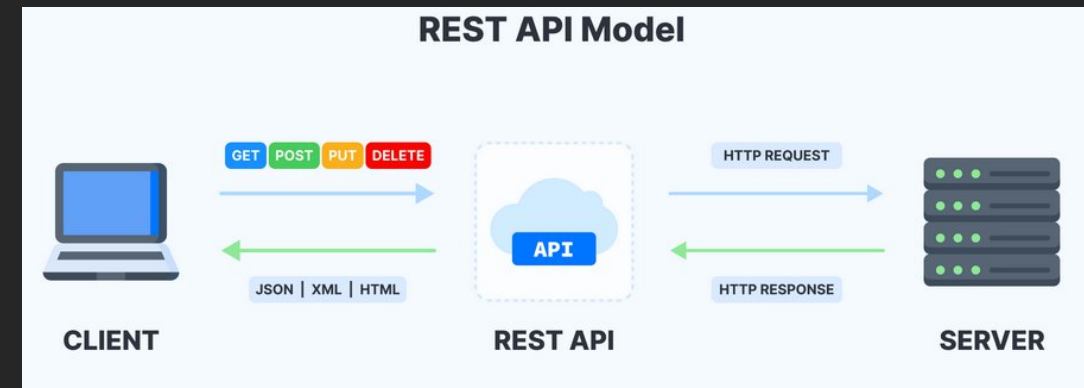


## REST (Representational State Transfer) :

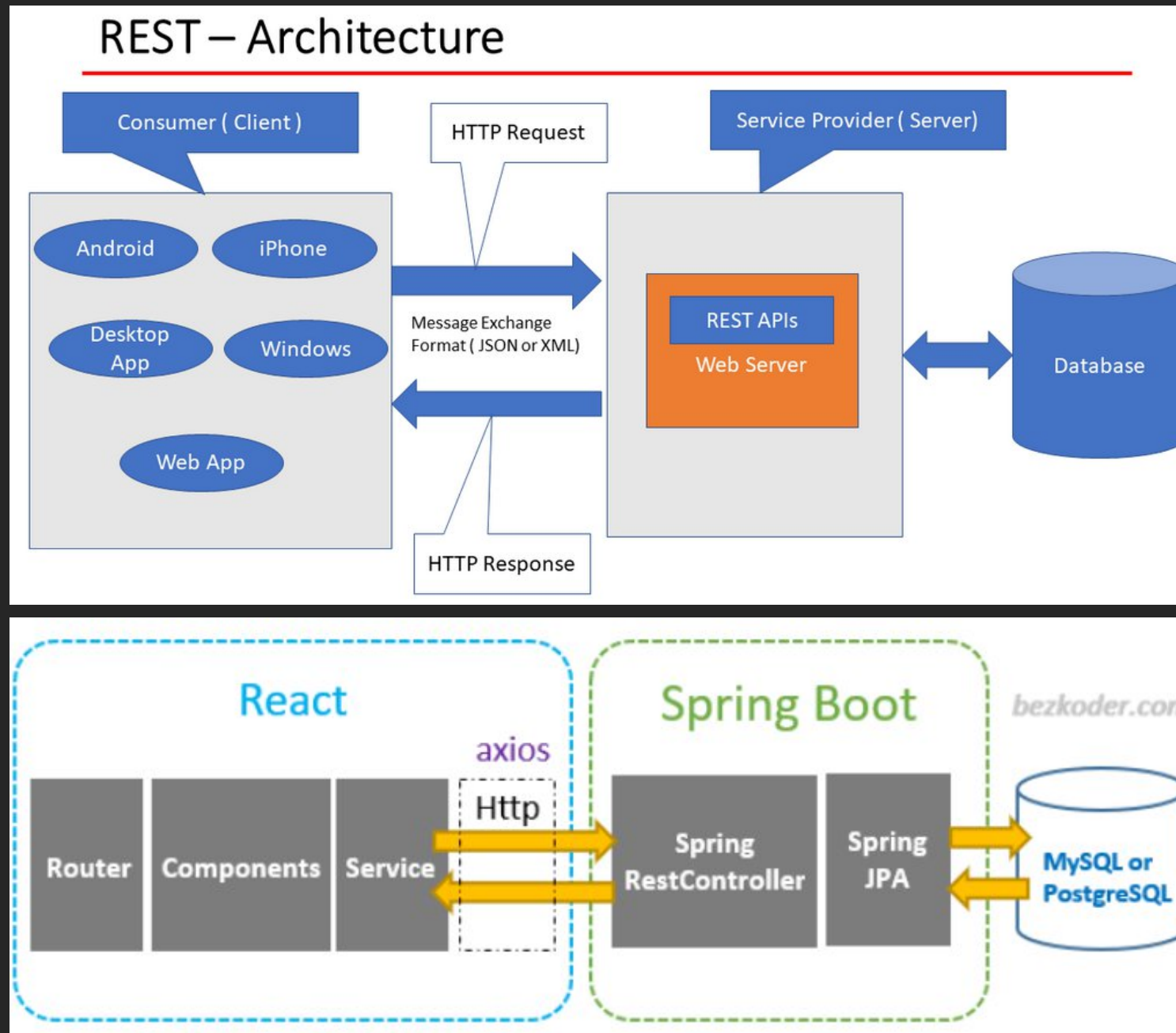
( JSON , data , HTTP Protocol )

la norme pour la création de services Web sur le Web  
("modèle objet de HTTP") :

- Facilité de développement
- Facilité d'utilisation
- Réponses à des requêtes en HTML, XML, JSON
- Utilise les méthodes HTTP (GET, POST, PUT, DELETE)  
pour gérer le CRUD



# REST Architecture



# REST API VS RESTful API

**tous les services Web sont des API  
mais toutes les API ne sont pas des services Web.**

<https://www.geeksforgeeks.org/know-the-difference-between-rest-api-and-restful-api/>

<https://medium.com/@shikha.ritu17/rest-api-architecture-6f1c3c99f0d3>

# REST API VS Web Socket API

En IoT, un Web service peut être implémenté soit :

- en utilisant REST
- en utilisant le protocole Web Socket

[https://www.geeksforgeeks.org/difference-between-rest-api-and-web-socket-api/?ref=next\\_article](https://www.geeksforgeeks.org/difference-between-rest-api-and-web-socket-api/?ref=next_article)

# Codes HTTP

**200: OK**

**201: Created**

**400: Bad Request**

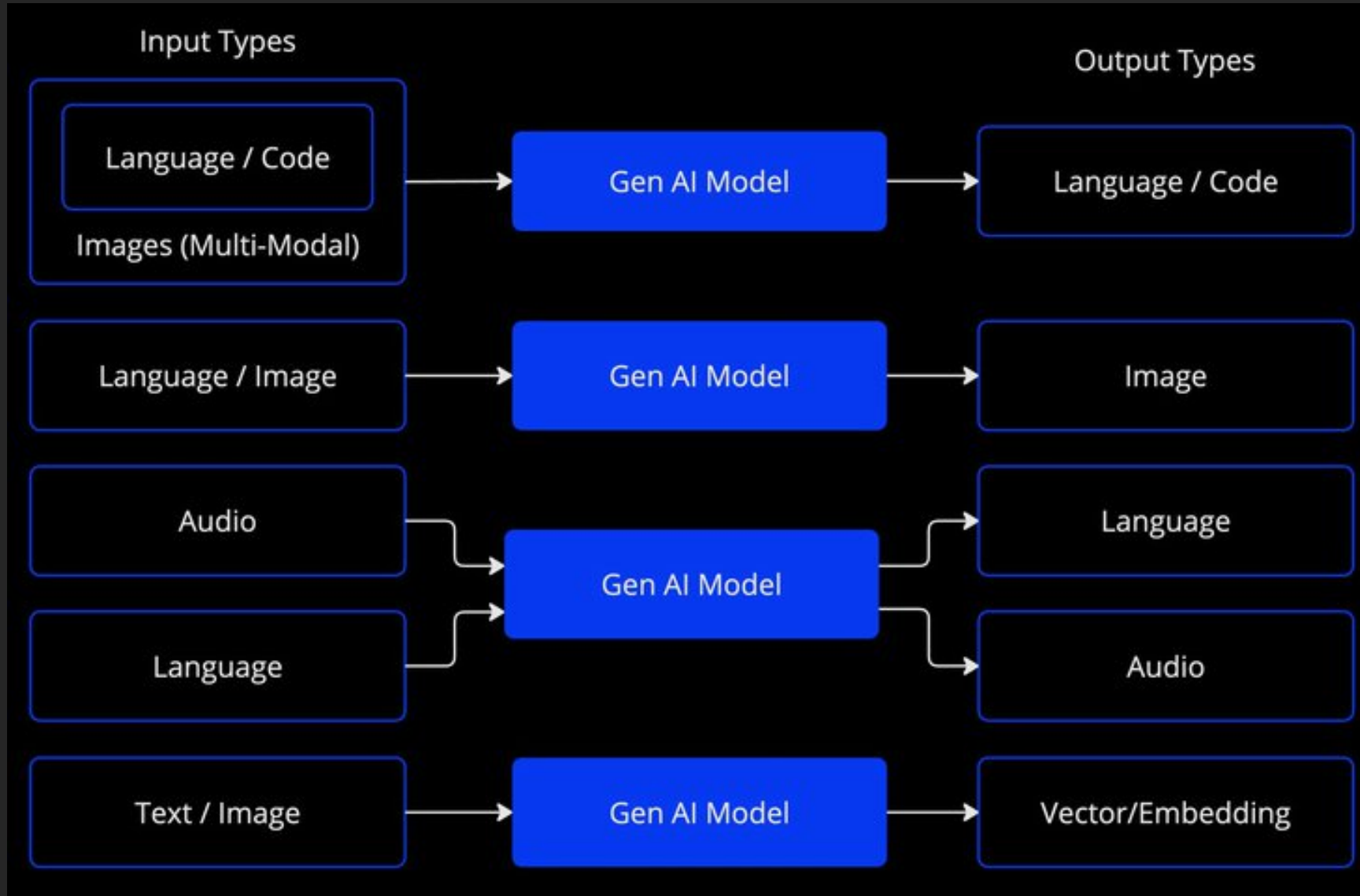
**404: Not Found**

**500: Internal Server Error**

**[https://fr.wikipedia.org/wiki/Liste\\_des\\_codes\\_HTTP](https://fr.wikipedia.org/wiki/Liste_des_codes_HTTP)**

# Spring AI

<https://docs.spring.io/spring-ai/reference/index.html>





# Audit Site Web

**Prompt :**

Établis une checklist pour un audit de sécurité informatique complet de l'entreprise.

**Outils :**

– vitesse de téléchargement des pages Web

**PageSpeed Insights :**

[https://pagespeed.web.dev/?utm\\_source=psi&utm\\_medium=redirect&hl=fr](https://pagespeed.web.dev/?utm_source=psi&utm_medium=redirect&hl=fr)

– analyse d'un site (gratuit) : SEO mais aussi technique

<https://hellotools.org/fr/>

## Vérifier la pertinence du nommage dans les classes métiers

## Astuce du nuage de mots

