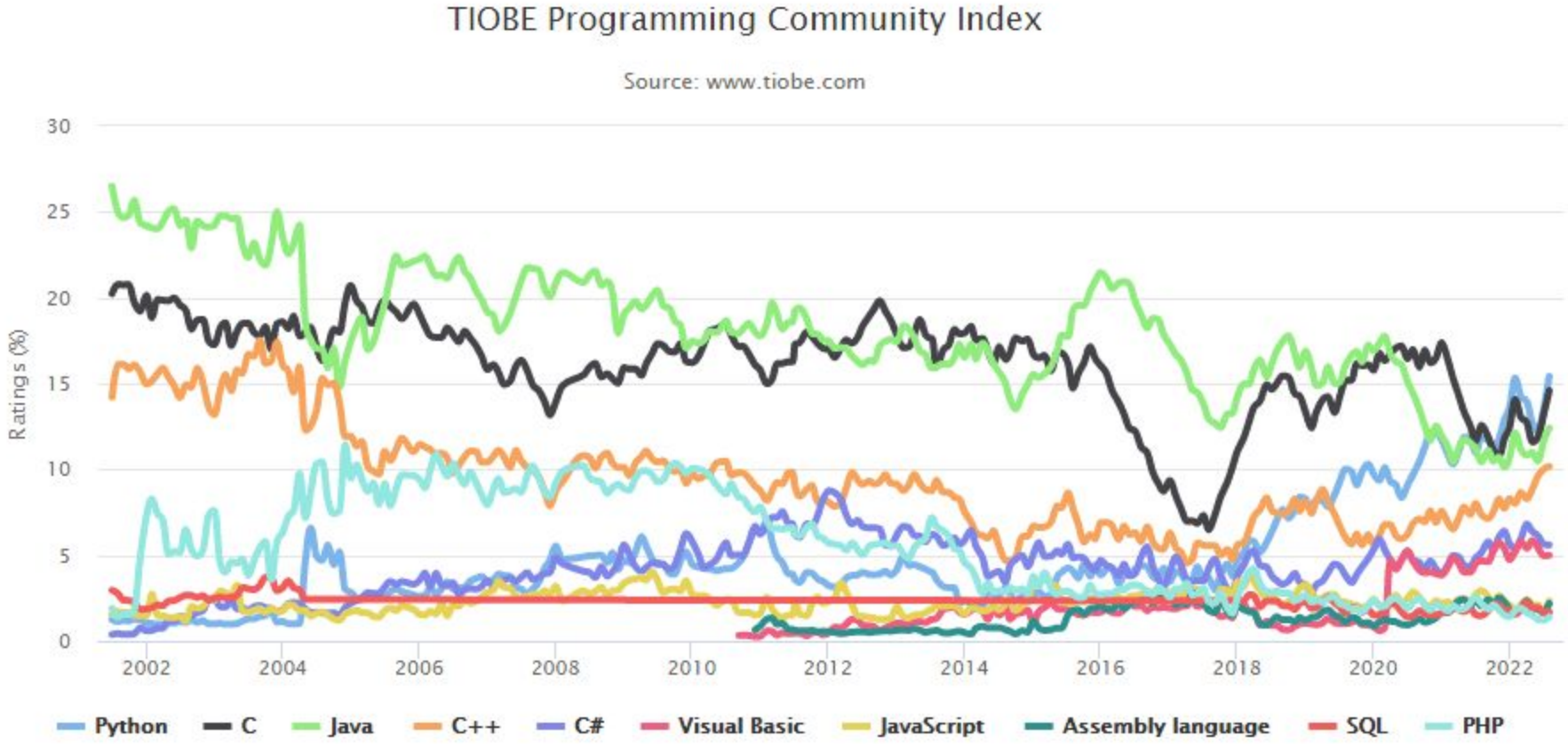
Java/Jakarta EE

Matthias Colin

Programmation Orienté Objets

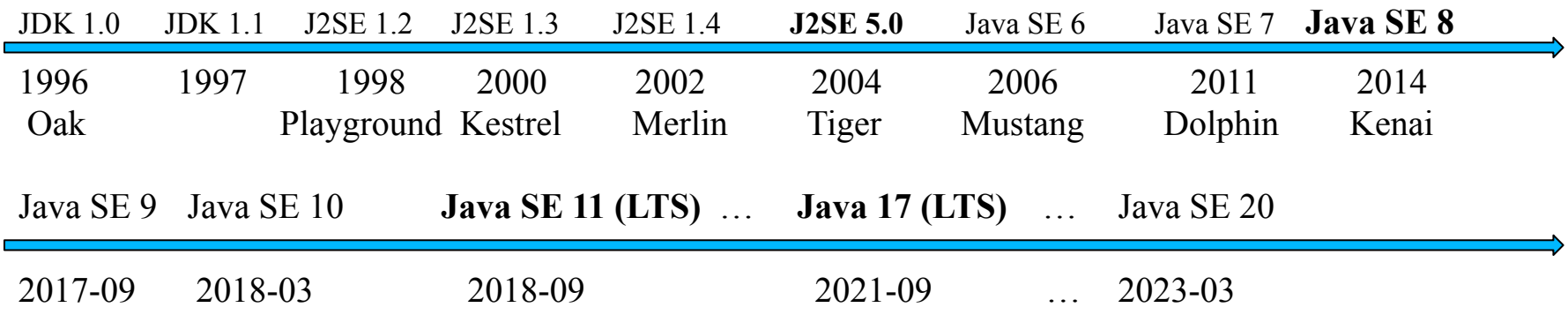
- POO / OOP
- C++
- Java
- .NET (C#, VB.NET, ..)
- Python
- JavaScript, TypeScript, Google App Script
- Php


TIOBE Programming Community Index
Source: www.tiobe.com
Ratings (%)
30
25
20
15
10
5
0
2002
2004
2006
2008
2010
2012
2014
2016
2018
2020
2022
Python
C
Java
C++
C#
Visual Basic
JavaScript
Assembly language
SQL
PHP

Java éditions

- Java SE (Standard Edition)
  - JVM : Java Virtual Machine
  - JRE : Java Runtime Environment (JVM + lib Java SE)
  - JDK : Java Development Kit (JRE + tools javac, jar, …)
  - Providers: Oracle et Openjdk
- Java ME (Mobile Edition)
- Java EE / Jakarta EE (Entreprise Edition)
  - collection de spécifications
  - servlet (http), jsp (Java Server Page), el, jstl
  - JPA (Java Persistence API) : RDBMS + SQL
  - JAX-WS : web services (SOAP+WSDL), axé XML
  - JAX-RS : API Rest avec data XML ou JSON
  - JNDI : annuaire
  - Java Bean Validation
  - ...

Java SE


JDK 1.0
JDK 1.1
J2SE 1.2
J2SE 1.3
J2SE 1.4
J2SE 5.0
Java SE 6
Java SE 7
Java SE 8
1996
Oak
1997
1998
Playground
2000
Kestrel
2002
Merlin
2004
Tiger
2006
Mustang
2011
Dolphin
2014
Kenai
Java SE 9
Java SE 10
Java SE 11 (LTS)
…
Java 17 (LTS)
…
Java SE 20
2017-09
2018-03
2018-09
2021-09
…
2023-03

Vocabulary

- **JVM**: Java Virtual Machine (java or java.exe)
  - execute Java Bytecode
- **JRE**
  - JVM + libraries included in the language
- **JDK**
  - JRE + tools
    - javac: compiler
    - jar: packager
    - javadoc: documentation generator
    - jshell: Java interpreter
    - jconsole: monitoring
  -

Tools

- java (JVM)
- javac (compiler)
  - *.java (source) => *.class (bytecode)
- jar (package)
  - contains: bytecode, resource, other jar
  - examples:
    - appli.jar, library.jar
      - java -jar appli.jar
    - webapp.war (to be deployed in application server)
- javadoc (documentation)
- many others …

Project Manager

- Managers
  - Ant (deprecated)
  - **Maven**: default configuration file **pom.xml**
    - download dependencies
    - plugins
  - **Gradle**: default configuration file **build.gradle**
    - download dependencies
    - plugins
    - many languages, speed
- Dependencies
  - Maven Repository
  - Maven Central

IDE: Integrated Development Environment

- **IntelliJ Idea** (JetBrains)
- Eclipse
- Netbeans
- VS Code

Java/Jakarta EE

JDK 1.0	JDK 1.1	J2SE 1.2	J2SE 1.3	J2SE 1.4	J2SE 5.0	Java SE 6	Java SE 7	**Java SE 8**	**Java SE 11**		
1996	1997	1998	2000	2002	2004	2006	2011	2014	2018		
		J2EE 1.2	J2EE 1.3	J2EE 1.4	JEE 5	**JEE 6**	**JEE 7**	**JEE 8**	**Jakarta EE 8,9,10**		
		1999	2001	2003	2006	2009	2013	**2017**	**2018**		
Servlets :		2.2	2.3	2.4	2.5	3.0	3.1	4.0			
JSP :		1.1	1.2	2.0	2.1	2.2	2.3				
JSTL :			1.0	1.1	1.2	-	-				
EL :					2.1	2.2	3.0				
JPA :					1.0	2.0	2.1	2.2		3.0	3.1
Bean validation :					1.0		2.0				
Outils :		JDBC		WS		JAXWS	JAXRS		JSON	JSON-B	
		JNDI			JSF		SAAJ	WebProfile	WebSocket		

Gouvernance Java

- Java SE: Sun Microsystems puis Oracle
  - package java.yyy et javax.zzz
- Java EE: Sun Microsystems puis Oracle (fin 2018, Java EE 8)
  - package javax.yyy
- Jakarta EE: Eclipse Foundation (depuis 2018)
  - Java EE 8 -> Jakarta EE 8: package javax.yyy
  - Jakarta EE 9, 10: package jakarta.yyy

Providers

Java SE:

- Oracle
- IBM
- Openjdk optimized by debian, redhat, mac, docker

Java EE / Jakarta EE

- Oracle via Weblogic
- IBM via Websphere
- Red Hat via Wildfly/JBoss, Hibernate
- Apache via Tomcat, …

Migration Java EE => Jakarta EE

- Change dependencies spec JEE + provider: maven, gradle, …
  - Ex: javax.validation:validation-api:2.0.1.Final
    => jakarta.validation:jakarta.validation-api:3.0.2
- Code source:
  - change import javax.yyy => jakarta.yyy
  - Ex: import javax.validation.constraints.NotBlank
    => import jakarta.validation.constraints.NotBlank
  - Attention aux faux amis: import javax.sql.DataSource (in Java SE)

- Configuration

Bean Validation

https://beanvalidation.org/
https://www.baeldung.com/javax-validation

- introduced in Java EE 7
- constraints: package javax.validation or jakarta.validation
  - @NotBlank, @NotNull
  - @Min, @Max, @Size
  - @Pattern,@Email
  - @Past, @Present
  - custom
- validation: validator => list of constraint violations

Bean Validation provider: Hibernate

Hibernate Validator	8.0	7.0	6.2
Java	11 or 17	8, 11 or 17	8, 11 or 17
Bean Validation	N/A	N/A	N/A
Java EE	N/A	N/A	8
Jakarta Bean Validation	3.0	3.0	2.0
Jakarta EE	10	9	8

Bean Validation API

Bean Validation 2.0 (Java EE 8)

https://docs.jboss.org/hibernate/validator/6.2/api/

Bean Validation 3.0 (Java EE 9, 10)

https://jakarta.ee/specifications/bean-validation/3.0/apidocs/

RDBMS

- Relational Database Management System
- Created in the 70’s
- Main vendors
  - Oracle Database
  - Microsoft SQL Server
  - PostgreSQL
  - MySQL / MariaDB
  - Sqlite
- Standard SQL (1974 - 2023)

Java application with Relational Database

- communication appli Java <-> RDBMS
- langage commun de communication SQL
- JDBC : Java Database Connectivity (inclus Java SE)
  - Comment gérer des requêtes (insert, update, delete, select)
  - package java.sql et javax.sql
    - Driver : spécification d'un driver éditeur
    - Connection : établir une connexion avec la base de données
      - host, port, dbname, user (, password)
    - DataSource : pool de connexion(s)
    - Statement : exécuter une requête
      - select * from movies where year = 2020
    - PreparedStatement : exécuter une requête préparée
      - select * from movies where year = ?
      - paramètre #1 pourra être 2020, 2021, …
    - ResultSet : résultat d'une requête
  - Driver JDBC apporté par l'éditeur ou la communauté
    - postgresql-42.2.20.jar
- JNDI : externaliser les settings JDBC de l'appli => serveur appli

Java with RDBMS (2)

- JDBC
- specification java JEE: JPA (Java Persistence API) :
  - main provider Hibernate
  - ORM: Object Relational Mapper
- Spring Data: JPARepository

ORM

ORM = Object Relational Mapper

- entity: class Java: Movie <-> table DB: movies
  - attribute: String title <-> column title
    - **id** <-> column id (Primary Key)
- association:
  - Movie-Person director <-> column director_id (FK)
  - Movie-Person; actors <-> table play
- crud
  - save(movie: Movie) <-> insert into movies …
  - List<Movie> res = read(...) <-> select … from movies where …

ORM in Object Oriented Languages

- Java: Java/Jakarta EE JPA + provider Hibernate ORM (or Eclipse Link)
- Python: Django ORM, SQLAlchemy
- .NET: Entity Framework
- php: symfony doctrine

ORM JPA

JPA = Java Persistence API

- J2EE, Java EE, JEE: JPA 1 and JPA 2
  - Hibernate 1 to 5
  - package javax.persistence.*
- Jakarta EE: JPA 2 and 3 (3.1)
  - Hibernate 6
  - package jakarta.persistence.*

JPA

- class tagged with @Entity
  - default constructor
  - getter/setter for each persistent field
  - by default all fields are persistent
  - primary key: @Id, @GeneratedValue
    - strategy: IDENTITY, SEQUENCE, AUTO, TABLE, UUID
  - tuning names, constraints
    - class: @Table
    - fields: @Column

SQL vocabulary

CRUD: DML = **Data Manipulation** Language

- INSERT
- UPDATE
- DELETE
- SELECT

DDL = **Data Definition** Language: table, view, index, user, …

- CREATE
- ALTER
- DROP

Hibernate settings

- dialect: H2, MariaDB, MySQL, PostgreSQL, …

  https://docs.jboss.org/hibernate/orm/6.2/userguide/html_single/Hibernate_User_Guide.html#database-dialect

- hbm2ddl.auto: DDL (JPA: jakarta.persistence.schema-generation.database.action)
  - none: no ddl generation (production)
  - update: create new table, alter existing table
  - create: drop previous version of all tables, then create all tables
  - create-drop: idem create + drop all tables when shutting down hibernate
- show_ql: show DDL and DML SQL queries
- format_sql: pretty print SQL

JPA Queries

- Traduction to native SQL according to chosen dialect (MariaDB, PostgreSQL, …)
  - Advantage: Java code independent from DB vendor
  - Techniques
    - JPQL: pseudo SQL with entities (not tables)
    - API Criteria: Java Code with methods .from(), .where(), .join()
- Native SQL
  - Drawback: Java code dependent from DB vendor

JPA Associations

- One to one
- Many to one
- One to many
- Many to many

Java/Jakarta EE - Web Profile

Specifications:

- Servlets: HTTP protocol
- JSP: HTML dynamic page (Java and HTML mixed code)
- EL: Expression Language (simplified Java expression)
- Websockets
- JSTL: if, for with HTML tags (no Java code)

Java Application Server:

- Apache Tomcat: Java/Jakarta EE Web Profile only
- Wildfly/JBoss EAP sponsored by RedHat: Full Java/Jakarta EE profile
- others: IBM Websphere, Oracle Weblogic

Other Java/Jakarta EE Specificications

- JAX-WS: Web Services SOAP/WSDL (W3C + XML)
  - include in Java SE from 1.6 to 1.8, remove from Java SE 9+
- JAX-RS: API Rest
- JNDI: Java Object Directory
  - Datasource Configuration in Java Application Server

Old Slides

Following Slides are Old Slides

Enjoy ;)

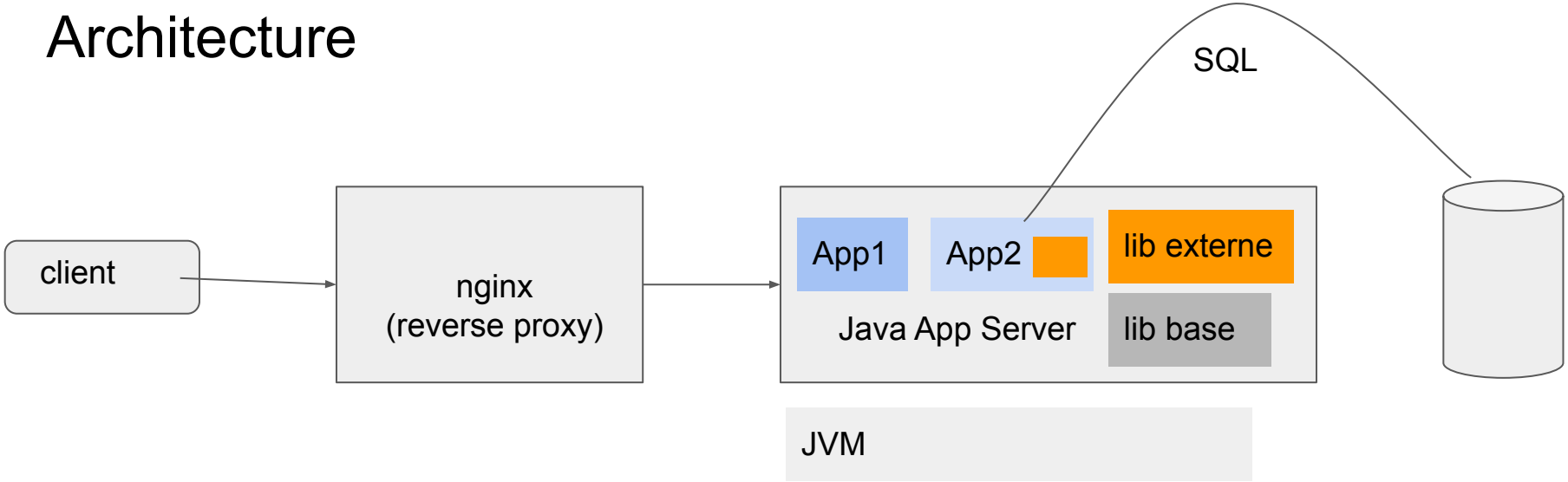
Application Java Backend

- Webapp MVC (Model View Controller)
- API Rest (Web Services)
  - micro-services
  - framework Spring
- Plateforme d'exécution (Java EE, JEE, Jakarta EE)
  - JVM : java / java.exe
  - Serveur Application Java
    - Tomcat (JEE Profile Web)
    - Full JEE:
      - JBoss / WildFly (RedHat)
      - Oracle WebLogic
      - IBM WebSphere

Exemple

- Développement d'un application avec
  - Java SE 11
  - servlet (spec JEE)
  - JPA (spec JEE)
  - dependance spring
- Déploiement & exécution
  - JRE 11
  - serveur application Java
    - Profile Web JEE : tomcat + jar JPA + conf + jar spring
    - Full JEE :
      - RedHat Wildfly + jar spring

Architecture



Wildfly

- https://www.wildfly.org/
- modes
  - standalone : 1 process java
  - domain : plusieurs processus java
    - 1 domain controller (process controller)
    - 1 host controller par host/machine
    - server(s)
- interfaces
  - public : 8080, 8443
  - management : 9990
- HAL: appli web d’administration
- jboss-cli : client en ligne de commande d’administration

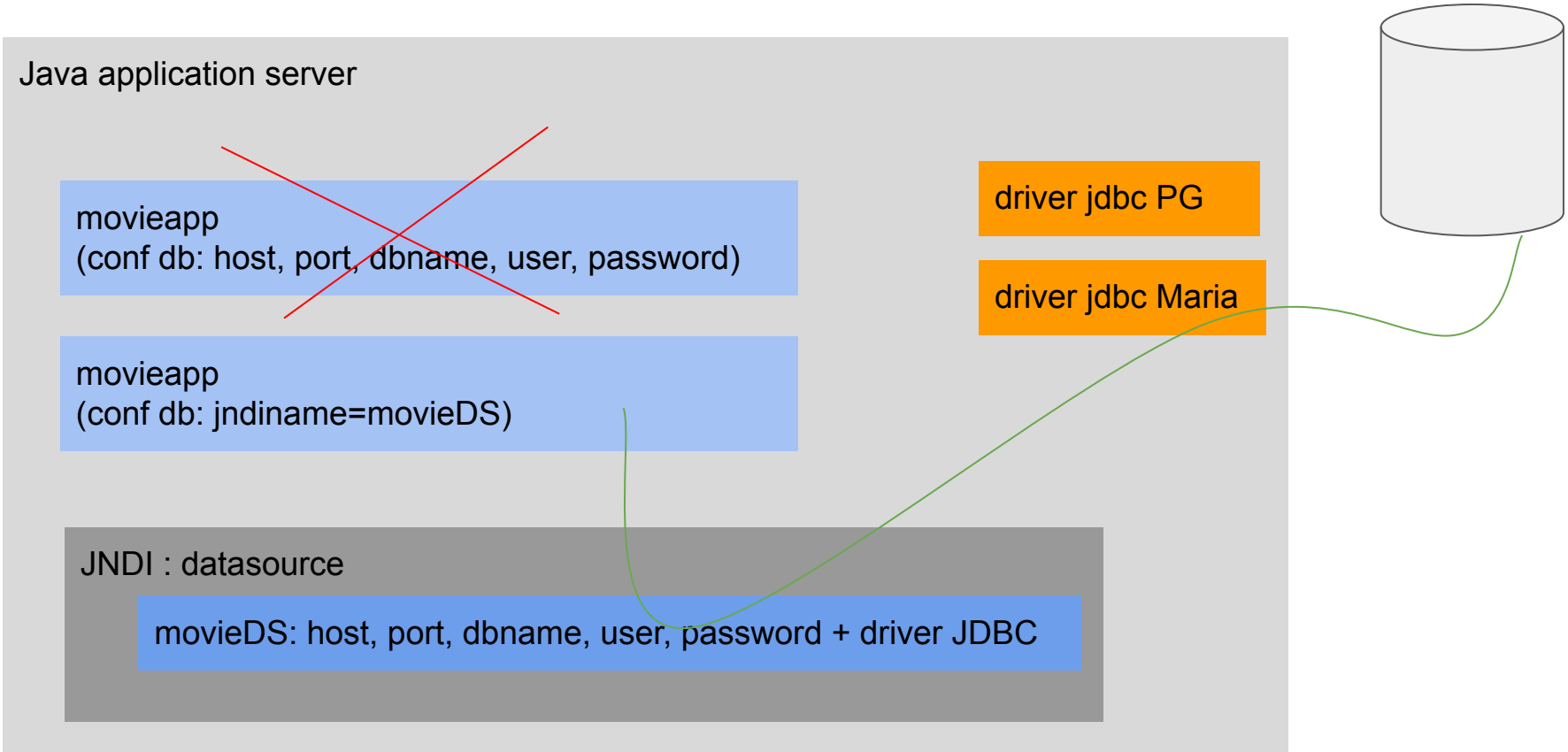
Déploiement d’une webapp

http://192.168.56.106:8081/bonjour/index.html

http://192.168.56.106:8081/bonjour/Goodbye

- 192.168.56.106 : hostname ou ip serveur
- 8081 : port pour atteindre le serveur
- /bonjour : contexte de l‘application
- /index.html : ressource publique (html, css, image, jsp)
- /Goodbye : ressource privé routée (servlet, api rest, …)

JNDI


Java application server
movieapp
(conf db: host, port, dbname, user, password)
movieapp
(conf db: jndiname=movieDS)
JNDI : datasource
movieDS: host, port, dbname, user, password + driver JDBC
driver jdbc PG
driver jdbc Maria

Configure JNDI Datasource

- tomcat: ajouter en XML l’entrée JNDI
  - http://tomcat.apache.org/tomcat-9.0-doc/jndi-datasource-examples-howto.html
- wildfly