Java Project

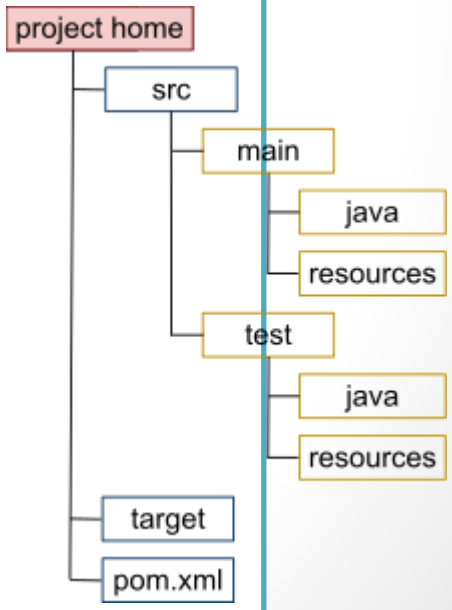
Xplain Iberica

Java Technologies

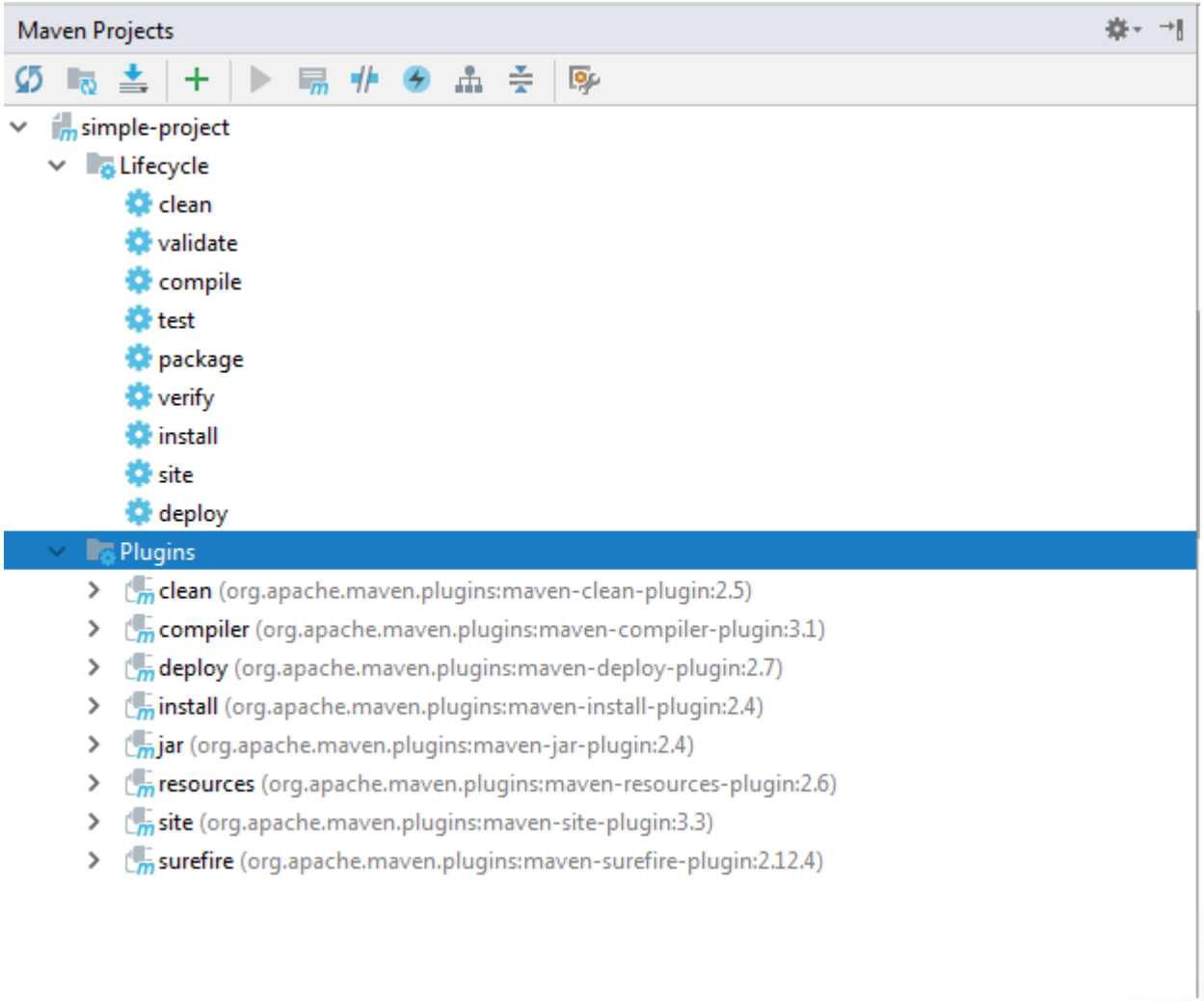
Maven

- It is a build automation tool used primarily for Java projects.
- Maven projects are configured using a Project Object Model, which is stored in a pom.xml-file.

```
<?xml version="1.0" encoding="UTF-8"?>

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
 <modelVersion>4.0.0</modelVersion>
 <groupId>com.packt.cookbook</groupId>
 <artifactId>simple-project</artifactId>
 <version>1.0-SNAPSHOT</version>
<name>simple-project</name>
 <!-- FIXME change it to the project's website -->
 <url>http://www.example.com</url>

<properties>
 <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
 <maven.compiler.source>1.7</maven.compiler.source>
 <maven.compiler.target>1.7</maven.compiler.target>
</properties>

<dependencies>
 <dependency>
  <groupId>junit</groupId>
  <artifactId>junit</artifactId>
  <version>4.11</version>
  <scope>test</scope>
 </dependency>
</dependencies>
```



Maven Project



Spring Framework

- The **Spring Framework** is an application framework and inversion of control container for the Java platform.

- Modules:
  - **Spring Core Container**: this is the base module of Spring and provides spring containers (BeanFactory and ApplicationContext).
  - **Aspect-oriented programming:** enables implementing cross-cutting concerns.
  - **Authentication and authorization**: configurable security processes that support a range of standards, protocols, tools and practices via the Spring Security sub-project (formerly Acegi Security System for Spring).
  - **Convention over configuration:** a rapid application development solution for Spring-based enterprise applications is offered in the Spring Roo module
  - **Data access:** working with relational database management systems on the Java platform using Java Database Connectivity (JDBC) and object-relational mapping tools and with NoSQL databases
  - **Inversion of control container:** configuration of application components and lifecycle management of Java objects, done mainly via dependency injection
  - **Messaging:** configurative registration of message listener objects for transparent message-consumption from message queues via Java Message Service (JMS), improvement of message sending over standard JMS APIs
  - **Model–view–controller:** an HTTP- and servlet-based framework providing hooks for extension and customization for web applications and RESTful (representational state transfer) Web services.
  - **Remote access framework**: configurative remote procedure call (RPC)-style marshalling of Java objects over networks supporting Java remote method invocation (RMI), CORBA (Common Object Request Broker Architecture) and HTTP-based protocols including Web services (SOAP (Simple Object Access Protocol))
  - **Transaction management**: unifies several transaction management APIs and coordinates transactions for Java objects
  - **Remote management**: configurative exposure and management of Java objects for local or remote configuration via Java Management Extensions (JMX)
  - **Testing**: support classes for writing unit tests and integration tests

Configuration.xml

```
package com.tutorialspoint;
public class HelloWorld {
private String message;
public void setMessage(String message){
this.message = message; }

public void getMessage(){
System.out.println("Your Message : " + message);
 }
}
```

```
<?xml version = "1.0" encoding = "UTF-8"?>
<beans xmlns = "http://www.springframework.org/schema/beans"
        xmlns:xsi = http://www.w3.org/2001/XMLSchema-instance
       xsi:schemaLocation = "http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-3.0.xsd">
    <bean id = “singleProject" class = "com.packt.cookbook">
          <property name = "message" value = "Hello World!"/>
   </bean>
</beans>
```

The *ApplicationContext* container includes all functionality of the *BeanFactory*container, so it is generally recommended over *BeanFactory*. BeanFactory can still be used for lightweight applications like mobile devices or applet-based applications where data volume and speed is significant.

Hibernate

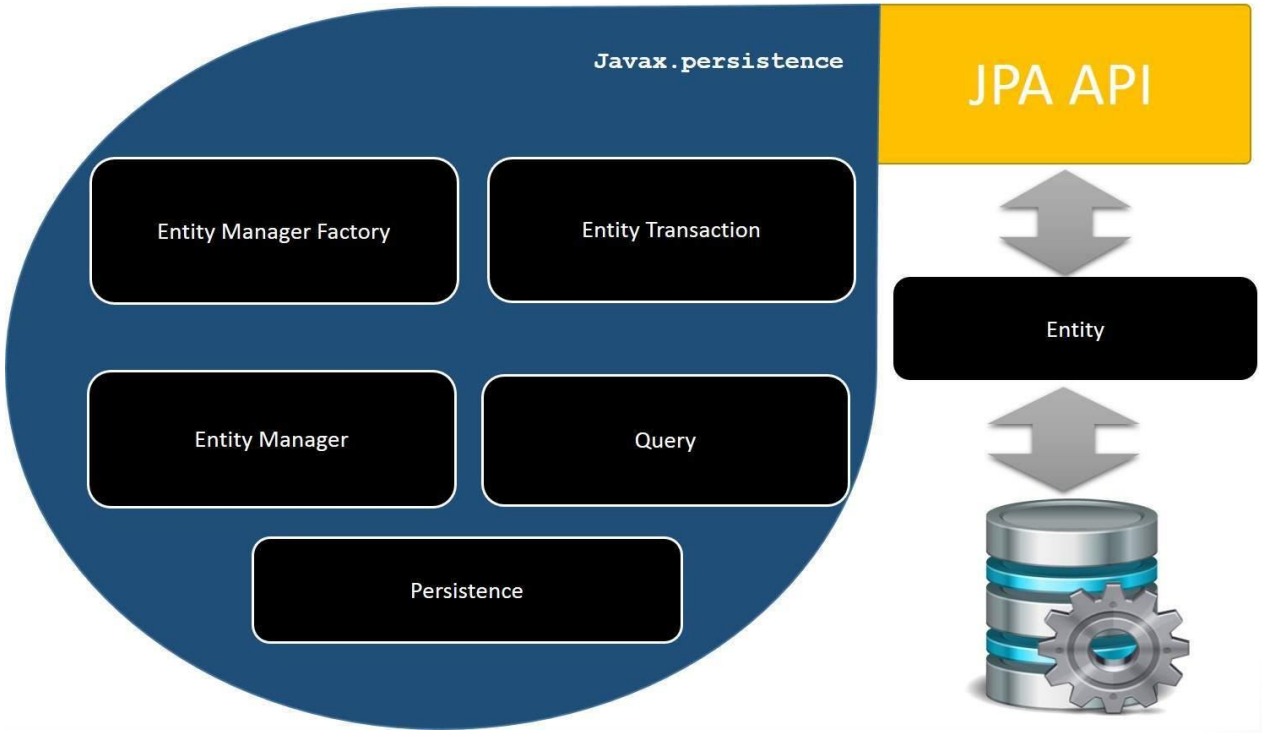
- It is an object-relational mapping tool for the Java programming language. It provides a framework for mapping an object-oriented domain model to a relational database.
- The mapping of Java classes to database tables is implemented by the configuration of an XML file or by using Java Annotations. When using an XML file, Hibernate can generate skeleton source code for the persistence classes. This is auxiliary when annotations are used. Hibernate can use the XML file or the Java annotations to maintain the database schema.

Hibernate Configuration

```
<?xml version='1.0' encoding='utf-8'?>
<!DOCTYPE hibernate-configuration PUBLIC
   "-//Hibernate/Hibernate Configuration DTD//EN"
   "http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">
<hibernate-configuration>
   <session-factory>
      <property name="connection.url">jdbc:oracle:thin:@//mig-bj-sp-name:1521/NAME</property>
      <property name="connection.driver_class">oracle.jdbc.OracleDriver</property>
      <property name="connection.username">name</property>
      <property name="connection.password">123456</property>
     <mapping class="ch.xplain.migtrova.model.Class"/>
  </session-factory>
</hibernate-configuration>
```

JPA

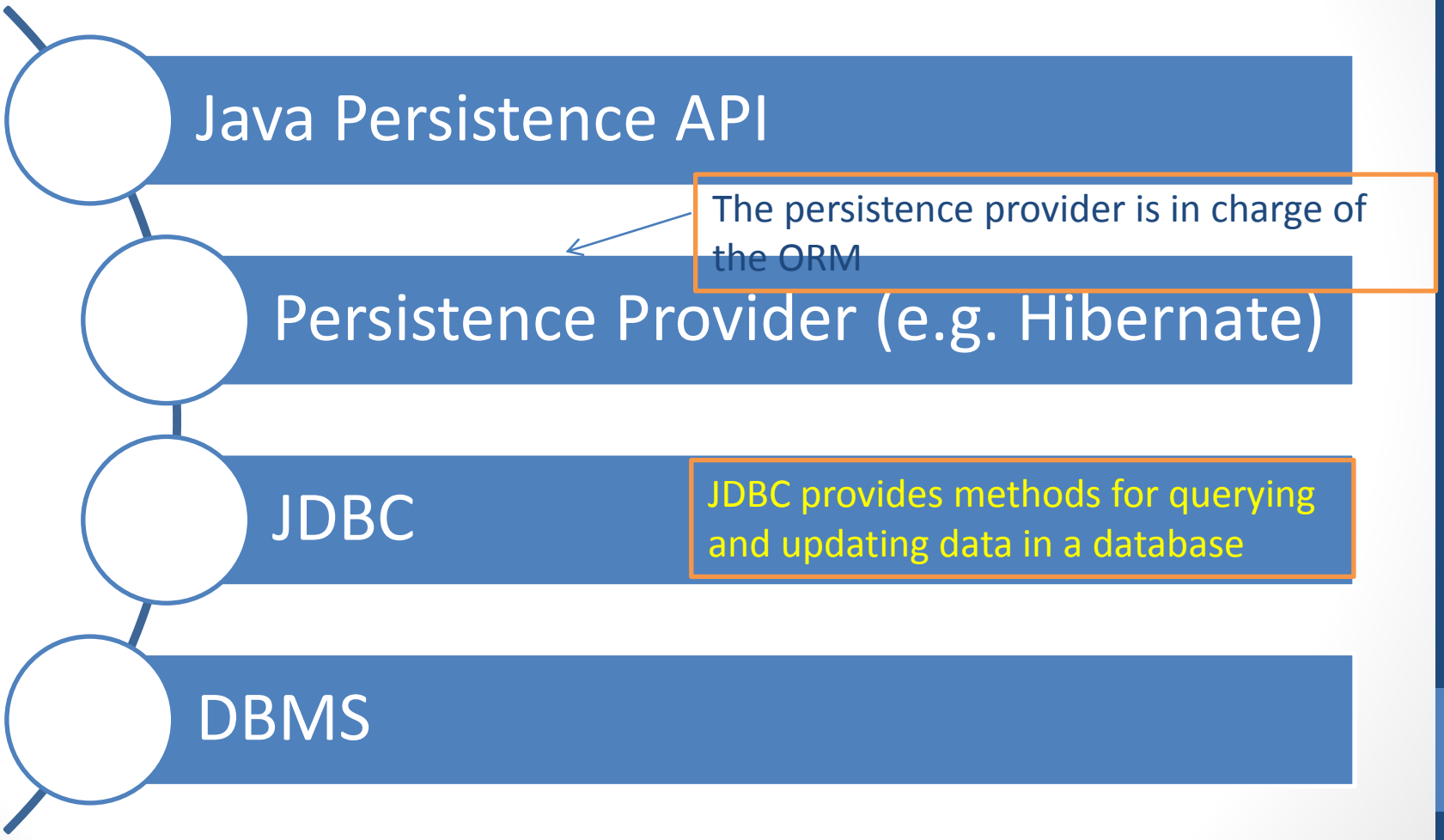
- JPA or Java Persistence API is the Java standard in charge of automating the persistence of our objects in the database as much as possible.



Persistence.xml

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<persistence xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://java.sun.com/xml/ns/persistence"
        version="2.0"
        xsi:schemaLocation="http://java.sun.com/xml/ns/persistence
http://java.sun.com/xml/ns/persistence/persistence_2_0.xsd">

  <persistence-unit name="oraclePersistenceUnit" transaction-
type="RESOURCE_LOCAL">
    <!--<provider>org.hibernate.ejb.HibernatePersistence</provider>-->
    <non-jta-data-source>dataSourceOracle</non-jta-data-source>
    <class>ch.xplain.migtrova.model.CLASS</class>
```

JPA Architecture



Sonarqube, Bamboo & Nexus

- **SonarQube** provides the capability to not only show health of an application but also to highlight issues newly introduced. With a Quality Gate in place, you can fix the leak and therefore improve code quality systematically.
- **Bamboo** is a continuous integration and continuous deployment server developed by Atlassian.
- **Nexus Repository OSS** is a centralized repository for managing all component.

XPATH

- A syntax for defining the parts of an XML document.
- A set of expressions for selecting parts of an XML document.
- A set of standard functions for processing chains, dates, numerical values, etc.

```
<?xml version="1.0" encoding="UTF-8"?>
<mapping>
 <contextPath>
  ch.infocar.intern.wscustomer.macs_v1:ch.infocar.schemas.wscustomer.macs_v1
 </contextPath>
 <person xPath="/infoc:sPersonSucheResponse/infoc:sPersonSucheResult/infoc:personListType|/infoc:sPersonDetailResponse/infoc2:person"
      sourceId="@pin" sourceTable="@kanton" inferMultiCol="true">
  <!-- Starting short details -->
  <property key="importableECH" short="true" full="false" xPath="." type="FixBoolean" typeParam="true"/>
  <property key="familyNames" short="true" full="false" xPath="@name" type="String"/>
  <property key="birthDate" short="true" full="false" xPath="@gebdat" type="DateAsString" typeParam="dd.MM.yyyy"/>
  <property key="sex" short="true" full="false" xPath="@sex" type="Sex"/>
  <!-- Starting extended short details-->
  <!--<property key="address" full="false" short_extended="true" xPath="concat(@strasse,'\, ',@ort, ', ',@plz,', ',@heimatort)" type="String"/>-->
  <property key="additionalInfo" full="false" short_extended="true" xPath="@stat" type="SingleString"/>
```

QueryDSL

- It is an extensive Java framework, which helps with creating and running **type-safe queries in a domain specific language that is similar to SQL**.

Basic query

```
List<Person> persons = queryFactory.selectFrom(person)
  .where(
    person.firstName.eq("John"),
    person.lastName.eq("Doe"))
  .fetch();
```

Order

```
List<Person> persons = queryFactory.selectFrom(pe
  .orderBy(person.lastName.asc(),
           person.firstName.desc())
  .fetch();
```

Subqueries

```
List<Person> persons = queryFactory.selectFrom(person)
  .where(person.children.size().eq(
    JPAExpressions.select(parent.children.size().max())
                  .from(parent)))
  .fetch();
```

Tuple projection

```
List<Tuple> tuples = queryFactory.select(
    person.lastName, person.firstName, person.yea
  .from(person)
  .fetch();
```