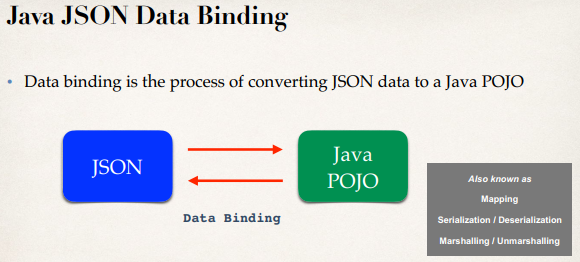
**57.2. Java JSON Data Binding**

**Java JSON Data Binding**:

Data binding is the process of converting JSON data to a Java POJO (Plain Old Java Object).



**JSON Data Binding with Jackson**:

1. Spring uses the Jackson Project behind the scenes
2. Jackson handles data binding between JSON and Java POJO
3. Details on Jackson Project: <https://github.com/FasterXML/jackson-databind>

**Jackson Data Binding**:

* Jackson Data Binding API
  + Package: com.fasterxml.jackson.databind
* Maven Dependency

<!-- Add jackson-databind for DataBind -->

<dependency>

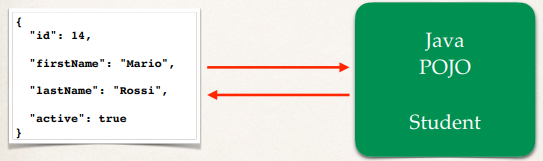
<groupId>com.fasterxml.jackson.core</groupId>

<artifactId>jackson-databind</artifactId>

<version>2.9.9.1</version>

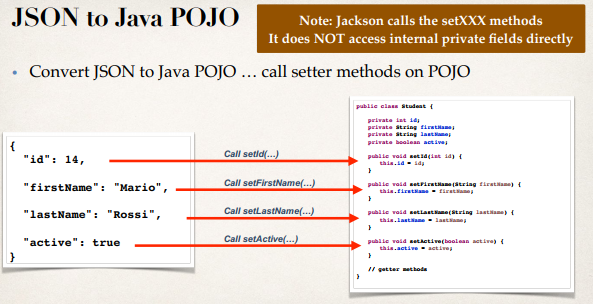
</dependency>

By default, Jackson will call appropriate getter / setter method.



**JSON to Java POJO**:

Convert JSON to Java POJO … call setter methods on POJO



**JSON to Java POJO Code**:

**package** com.ruhul.jackson.json.demo;

**import** java.io.File;

**import** com.fasterxml.jackson.databind.ObjectMapper;

**public** **class** Driver {

**public** **static** **void** main(String[] args) {

**try** {

// create object mapper

ObjectMapper mapper = **new** ObjectMapper();

// read JSON file and map/convert to java POJO

// data/simple-full.json

Student theStudent = mapper.readValue(**new**

File("data/sample-full.json"), Student.**class**);

// print the value

System.***out***.println("Student Id: " + theStudent.getId());

System.***out***.println("Student First Name: " +

theStudent.getFirstName());

System.***out***.println("Student Last Name: " +

theStudent.getLastName());

System.***out***.println("Student status: " + theStudent.isActive());

// print Address

System.***out***.println(theStudent.getAddress());

// print languages

System.***out***.print("Languages: ");

**for** (String languages : theStudent.getLanguages()) {

System.***out***.print(languages + " ");

}

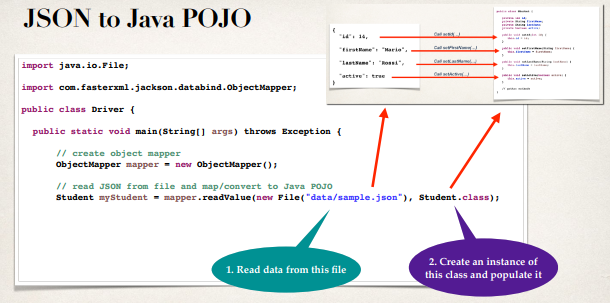
} **catch** (Exception e) {

e.printStackTrace();

}

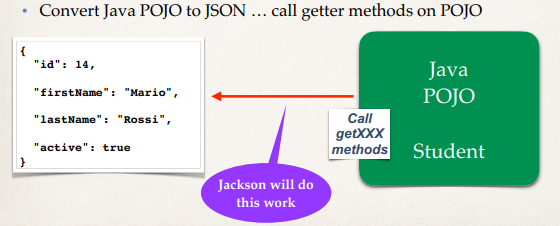
}

}



**Java POJO to JSON**:

Convert Java POJO to JSON … call getter methods on POJO



**Code**:

// create object mapper  
ObjectMapper mapper = new ObjectMapper();  
// read JSON from file and map/convert to Java POJO  
Student myStudent = mapper.readValue(new File("data/sample.json"), Student.class);

...

// now write JSON to output file

// indent the JSON output formapper.enable(SerializationFeature.INDENT\_OUTPUT);

// Jackson calls the getter methods on Student POJO to create JSON oututmapper.writeValue(new File("data/output.json"), myStudent);

**Spring and Jackson Support**:

* When building Spring REST applications
* Spring will automatically handle Jackson Integration
* JSON data being passed to REST controller is converted to POJO
* Java object being returned from REST controller is converted to JSON
* All this thing happens automatically behind the scenes.

**Development process**:

1. Download and import Maven starter files or create a maven project
2. Add Maven dependency for Jackson project
3. Create Student POJO Java class
4. Create main Driver App.

**Step-2: Add Maven dependency for Jackson project**:

<!-- Add jackson-databind for DataBind -->

<dependency>

<groupId>com.fasterxml.jackson.core</groupId>

<artifactId>jackson-databind</artifactId>

<version>2.9.9.1</version>

</dependency>

**Step-3: Create Student POJO Java class**:

**package** com.ruhul.jackson.json.demo;

**public** **class** Student {

**private** **int** id;

**private** String firstName;

**private** String lastName;

**private** **boolean** active;

**private** Address address;

**private** String[] languages;

// create getter and setter method

}

**Step-4: Create Main Driver App**:

**import** com.fasterxml.jackson.databind.ObjectMapper;

**public** **class** Driver {

**public** **static** **void** main(String[] args) {

**try** {

// create object mapper

ObjectMapper mapper = **new** ObjectMapper();

// read JSON file and map/convert to java POJO

// data/simple-full.json

Student theStudent = mapper.readValue(**new**

File("data/sample-full.json"), Student.**class**);

// print the value

System.***out***.println("Student Id: " + theStudent.getId());

System.***out***.println("Student First Name: " + theStudent.getFirstName());

System.***out***.println("Student Last Name: " + theStudent.getLastName());

System.***out***.println("Student status: " + theStudent.isActive());

// print Address

System.***out***.println(theStudent.getAddress());

// print languages

System.***out***.print("Languages: ");

**for** (String languages : theStudent.getLanguages()) {

System.***out***.print(languages + " ");

}

} **catch** (Exception e) {

e.printStackTrace();

}

}

}

**JSON data binding Ignore Properties**:

For ignore unknown property of JSON file we have to add " **@JsonIgnoreProperties** " annotation in POJO class.

@JsonIgnoreProperties(ignoreUnknown = **true**)

**public** **class** Student {

**private** **int** id;

**private** String firstName;

**private** String lastName;

**private** **boolean** active;

**private** Address address;

**private** String[] languages;

...

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*END\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*