luv2code

Spring Boot Back End

Java Development Environment

- We assume that you are already an experienced Spring Boot Developer

- You should have the following items already installed
  - Java Development Kit (JDK)
  - Java IDE (we'll use IntelliJ in the videos, but any Java IDE will work)
  - Maven
  - MySQL Database and MySQL Workbench

About IntelliJ

- In this course, we will use the free version of IntelliJ
  - Known as **IntelliJ Community Edition**
  - Download from: Super Amazing IDE!!! **n/idea/download**
  - Select **Community Edition**
- You can also use the Ultimate Edition ($) ... free trial version is available

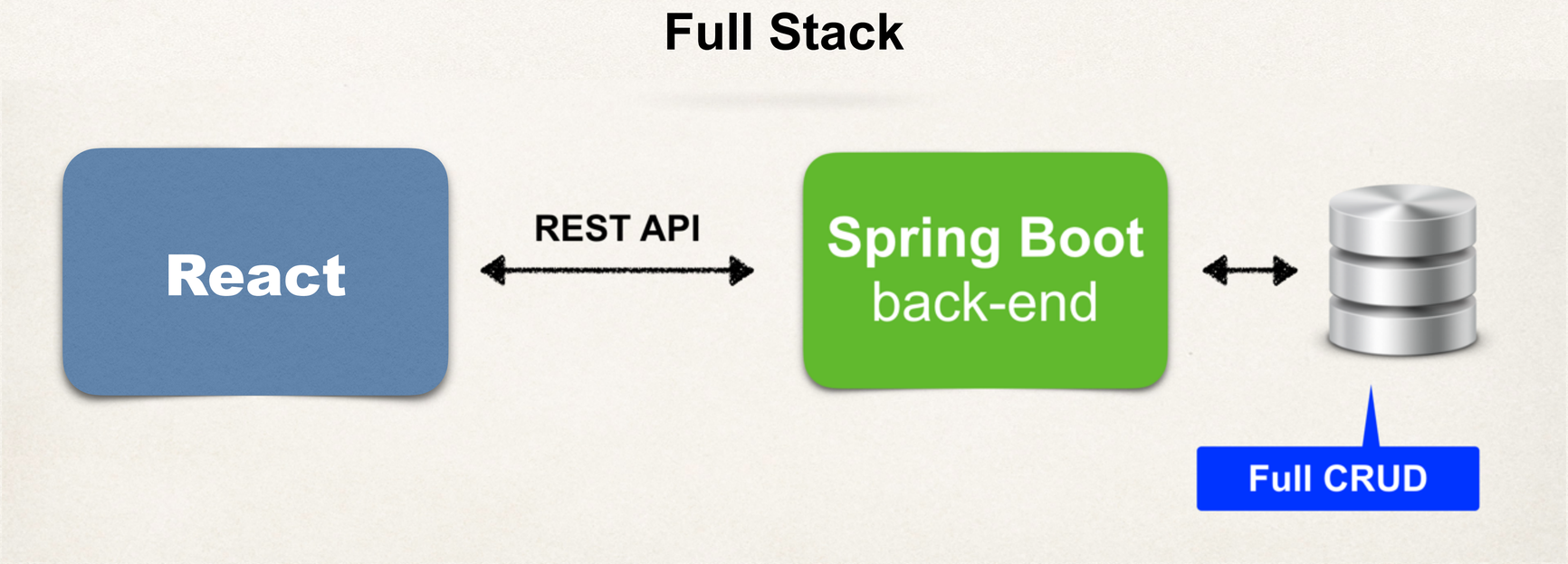
Additional Java IDEs

- You are free to use other Java IDEs such as Eclipse, VS Code, NetBeans
- All you need is a Java IDE that supports Maven ... that's it!
- You can easily follow along with any Java IDE

- We will provide tech support for IntelliJ, Eclipse, VS Code, NetBeans

Spring Boot Back End

- Leverage Spring Data REST for REST API
- Minimizes the coding for Spring Boot back end



Create Repository

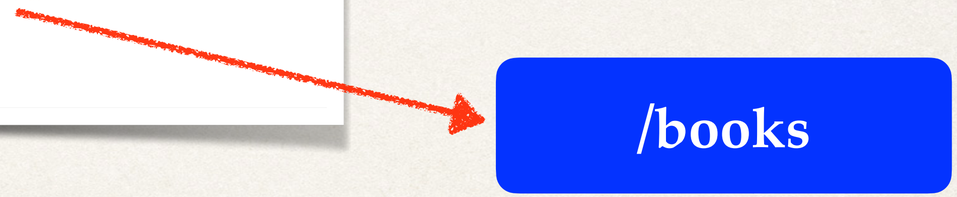
- Spring Data REST will scan your project for **JpaRepository**
- Expose REST APIs for each entity type for your **JpaRepository**

```
public interface BookRepository extends JpaRepository<Book, Long> {

}
```

REST Endpoints

- By default, Spring Data REST will create endpoints based on entity type
- Simple pluralized form
  - First character of Entity type is lowercase
  - Then just adds an “s” to the entity

```
public interface BookRepository extends JpaRepository<Book, Long> {

}
```



REST API

- Spring Data REST will expose these endpoints for free!

HTTP Method		CRUD Action
POST	/books	**C**reate a new book
GET	/books	**R**ead a list of books
GET	/books/{id}	**R**ead a single book
PUT	/books/{id}	**U**pdate an existing book
DELETE	/books/{id}	**D**elete an existing book

Five Database Scripts

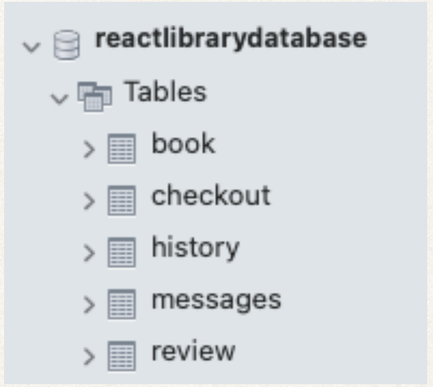
- **React-Springboot-Add-Tables-Script-1.sql**
- **React-SpringBoot-Add-Books-Script-2.sql**
- **React-SpringBoot-Add-Books-Script-3.sql**
- **React-SpringBoot-Add-Books-Script-4.sql**
- **React-SpringBoot-Add-Books-Script-5.sql**

MySQL user for our application

1. Create a new MySQL user for our application
    - user id: **root**
    - password: **test1234!**

About: React-Springboot-Add-Tables-Script-1.sql

1. Creates new tables for our MySQL database
    - Book
    - Checkout
    - History
    - Messages
    - Reviews



React-Springboot-Add-Books-Script-(2-5).sql

1. Add books to our database
2. Books hold blobs for images which are memory heavy
3. Books are split among 4 files

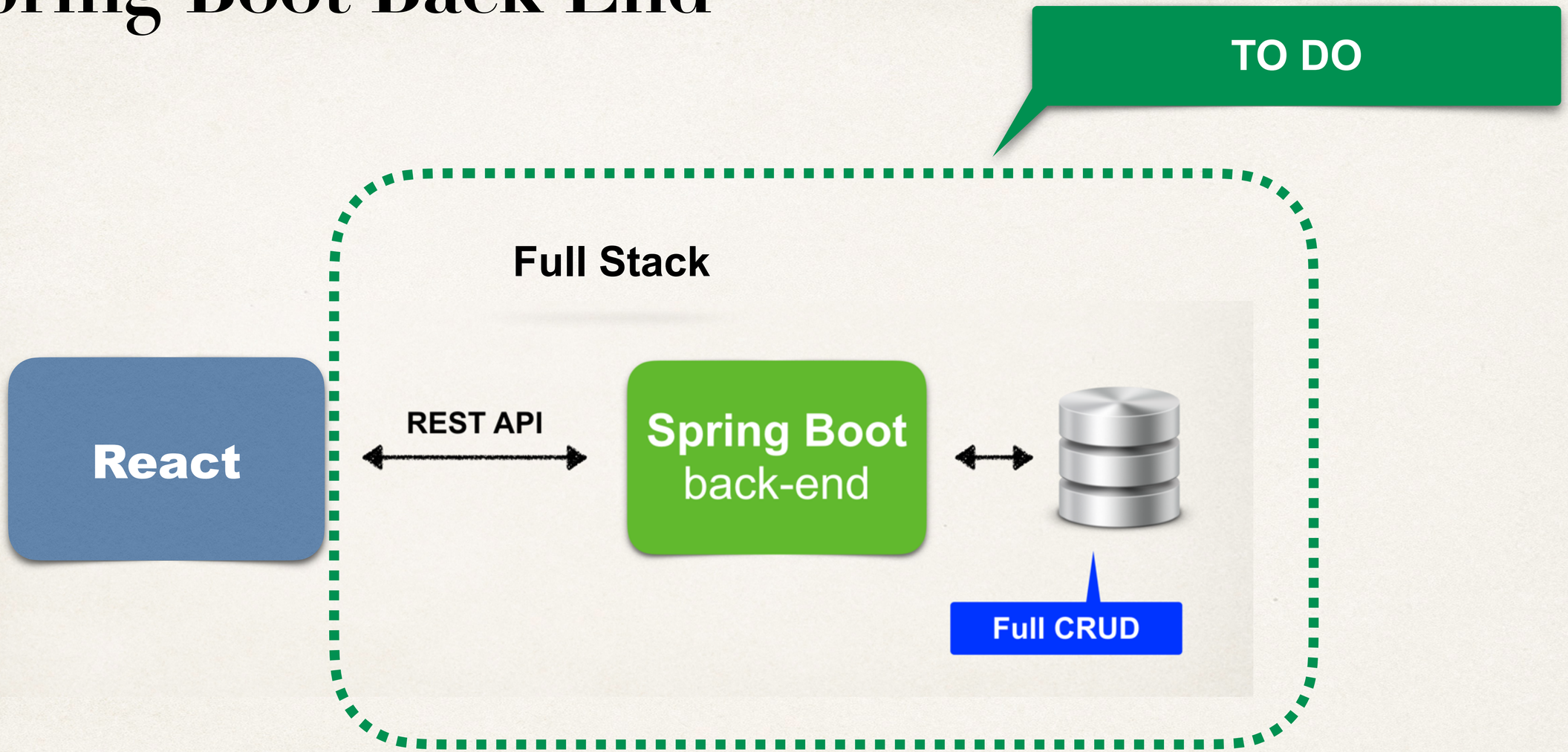
```
1 •  select * from book;
2
```

00% 19:1

Result Grid | Filter Rows: Search | Edit: | Export/Import:

id	title	author	description	copies	copies_available	category	img
1	JavaScript Cookbook	Luv, Zofia	Lorem ipsum dolor sit amet, consectetur adipisc…	10	10	FE	BLOB
2	Become a Guru in JavaScript	Luv, Lena	Pellentesque varius aliquam lacus quis rhoncus.…	15	15	FE	BLOB
3	Exploring Vue.js	Luv, Jakub	Proin at urna faucibus, pretium mi in, dapibus m…	10	10	FE	BLOB
4	Advanced Techniques in Big Data	Luv, Alexander	Nunc eget lorem ac neque tincidunt mollis. Fusc…	12	12	Data	BLOB
5	Crash Course in Big Data	Luv, Judy	Morbi eu tempus eros, in imperdiet sem. Nulla s…	20	20	Data	BLOB
6	Beginners Guide to SQL	Luv, Anup	Lorem ipsum dolor sit amet, consectetur adipisc…	20	20	Data	BLOB
7	Advanced Techniques in JavaScript	Luv, Yasemin	Pellentesque varius aliquam lacus quis rhoncus.…	20	20	FE	BLOB
8	Introduction to Spring Boot	Luv, Fatma	Lorem ipsum dolor sit amet, consectetur adipisc…	10	10	BE	BLOB
9	Become a Guru in React.js	Luv, Andrey	Pellentesque varius aliquam lacus quis rhoncus.…	15	15	FE	BLOB
10	Beginners Guide to Data Science	Luv, Ajay	Proin at urna faucibus, pretium mi in, dapibus m…	10	10	Data	BLOB
11	Advanced Techniques in Java	Luv, Anna	Nunc eget lorem ac neque tincidunt mollis. Fusc…	12	12	BE	BLOB
12	The Expert Guide to SQL	Luv, Poornima	Morbi eu tempus eros, in imperdiet sem. Nulla s…	20	20	Data	BLOB
13	Exploring DevOps	Luv, Mariana	Lorem ipsum dolor sit amet, consectetur adipisc…	20	20	DevOps	BLOB
14	Introduction to SQL	Luv, Hugo	Pellentesque varius aliquam lacus quis rhoncus.…	20	20	Data	BLOB
15	The Expert Guide to JavaScript	Luv, Elena	Lorem ipsum dolor sit amet, consectetur adipisc…	10	10	FE	BLOB
16	Exploring React.js	Luv, Filip	Pellentesque varius aliquam lacus quis rhoncus.…	15	15	FE	BLOB
17	Advanced Techniques in React.js	Luv, Mary	Proin at urna faucibus, pretium mi in, dapibus m…	10	10	FE	BLOB
18	Introduction to C#	Luv, Bill	Nunc eget lorem ac neque tincidunt mollis. Fusc…	12	12	BE	BLOB
19	Crash Course in JavaScript	Luv, Frank	Morbi eu tempus eros, in imperdiet sem. Nulla s…	20	20	FE	BLOB
20	Introduction to Machine Learning	Luv, Camila	Lorem ipsum dolor sit amet, consectetur adipisc…	20	20	Data	BLOB
21	Become a Guru in Java	Luv, Priya	Lorem ipsum dolor sit amet, consectetur adipisc…	20	20	BE	BLOB
22	Introduction to Python	Luv, Juan	Pellentesque varius aliquam lacus quis rhoncus.…	20	20	BE	BLOB
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Spring Boot Back End



Development Process

1. Set up the database tables
2. Create a Spring Boot starter project (start.spring.io)

```
spring-boot-starter-data-jpa
spring-boot-starter-data-rest
mysql-connector-java
lombok
```

3. Develop the Entity: **Book**
4. Create REST APIs with Spring Data JPA Repositories and Spring Data REST

Project Lombok

- Modern Java project
- Lombok automagically generates the getters/setters (behind the scenes)
- No need for the developer to manually define getters/setters, etc ...
- Easy-to-use Annotations to eliminate boilerplate code

http://www.projectlombok.org

Project Lombok

Before Lombok

```
@Entity
@Table(name = "book")
public class Book {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name="id")
    private Long id;

    @Column(name = "title")
    @NotBlank
    private String title;

    public Book() {
    }

    public Long getId() {
        return id;
    }

    public void setId(Long id) {
        this.id = id;
    }

    public String getTitle() {
        return title;
    }

    public void setTitle(String title) {
        this.title = title;
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass())
          return false;
        Book book = (Book) o;
        return Objects.equals(id, book.id) &&
                Objects.equals(title, book.title);
    }

    @Override
    public int hashCode() {
        return Objects.hash(id, title);
    }
}
```

Lombok annotation

After Lombok

```
@Entity
@Table(name = "book")
@Data
public class Book {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name="id")
    private Long id;

    @Column(name = "title")
    @NotBlank
    private String title;
}
```

That's It!!!
Absolutely no need to generate getters and setters

Lombok will do this work for you automagically behind the scenes