SPRINGBOOT QUICKSTART

(2) Requests → Get - read data from server
→ Post - send new data to server
→ Put - updat existing data
→ Delete - delet a resource by its ID

(3) Maven — you mention what you need in pom.xml. It contains list of dependencies that Maven needs to know
Maven goes to the repository where everything is kept and brings them.

meta dependency - pulls all the dependencies you need. You just need to bring this one.

(10) SpringApplication.run(App.class, args) → starts TOMCAT server
→ Performs class path scan

sets up default config. starts spring application context

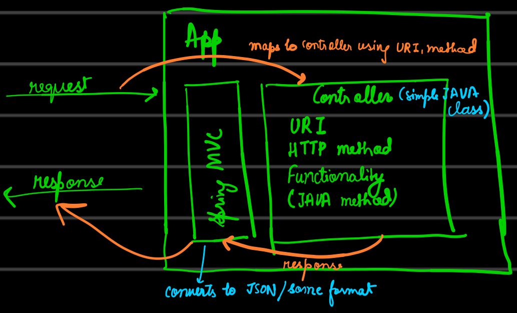
(11) Controller - JAVA class with certain annotations
These annotations let Spring know what URL to map to and what happens when requests come to URL

REST API - representational stat transfer - is a way for different software systems to communicate over the internet using HTTP methods like GET, POST, PUT, DELETE. Data is usually exchanged in JSON/XML format.

@RequestMapping("/hello") → whenever request at URL /hello, method below will be executed. Works for all methods but you've to mention. By default GET.

(12) Return object is automatically converted into JSON and sent as HTTP response.

(13) Spring MVC


App
maps to controller using URI, method
request
Controller (simple JAVA class)
URI
HTTP method
Functionality
(JAVA method)
Spring MVC
response
response
converts to JSON/some format

17 In spring, by default, all beans are singleton scoped, meaning only 1 object of a class is created and shared across the application - even if it is used in multiple places.

@Service - tells spring to creat a bean of that class automatically
- makes the class available for dependency injection

```
@Service
public class calc {
  public int add(int a, int b){
     return a+b;
  }
}
```

```
@RestController
public class A{
@Autowired
 privat calc service;
@GetMapping("/add")
 public int add (@RequestParam int a, " int b)
     return service.add(a,b);
  }
```

@Autowired is an annotation used to automatically inject (connect) one class into another.

```
@Component
public class Engine {
   public void start()
      Sopln("Engine started");
}
```

```
@Component
 public class Car {
@Autowired
private Engine engine;
 public void drive()
 {    engine.start();
     Sopln("car is moving");
 }
}
```

18 Path with variable id

```
@RequestMapping("/topics/{id}")
public Topic getTopic(@PathVariable String id)
 { return topicService.getTopic(id); }
```

19

```
@RequestMapping(method = RequestMethod.POST, value = "/topics")
public void addTopic(@RequestBody Topic topic)
    topicService.addTopic(topic);
```

@RequestBody → instructs Spring to bind the incoming JSON payload to the Topic object

---

26 ORM → Object Relational Mapping → map JAVA objects to database Tables

JPA → Java Persistence API → helps in ORM

28 @Id → marks the primary key

@Entity → annotation that tells springboot that this Java class should be mapped to a table in database

Repository contains all methods to be used.

24 We make an interface of the CrudRepository and autowire it. Then we can use its inbuilt methods.

public interface TopicRepository extends CrudRepository $\langle \underset{\text{entity}}{\text{Topic}}, \underset{\text{type of primary key}}{\text{String}} \rangle$