SPRINGBOOT QUICKSTART

② Requests ⟶ Get - read data from server
          ⟶ Post - send new data to server
          ⟶ Put - update 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
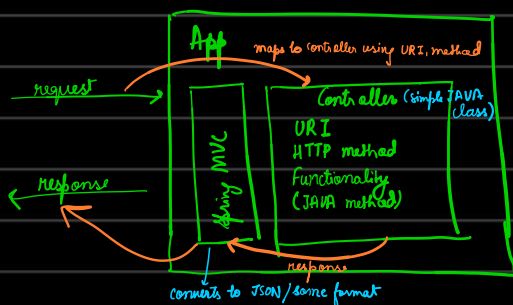
---

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)
          Spring MVC   URI
                        HTTP method
                        Functionality
                        (JAVA method)
response ⟵
          converts to JSON/some format
                        response

(12) 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 create a bean of that class automatically
 — makes the class available for dependency injection

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

```java
@RestController
public class A {
  @Autowired
  private 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.

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

```java
@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.

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

public interface TopicRepository extends CrudRepository <Topic, String>
                                                              ↓   ↓
                                                           entity   type of primary
                                                                          key