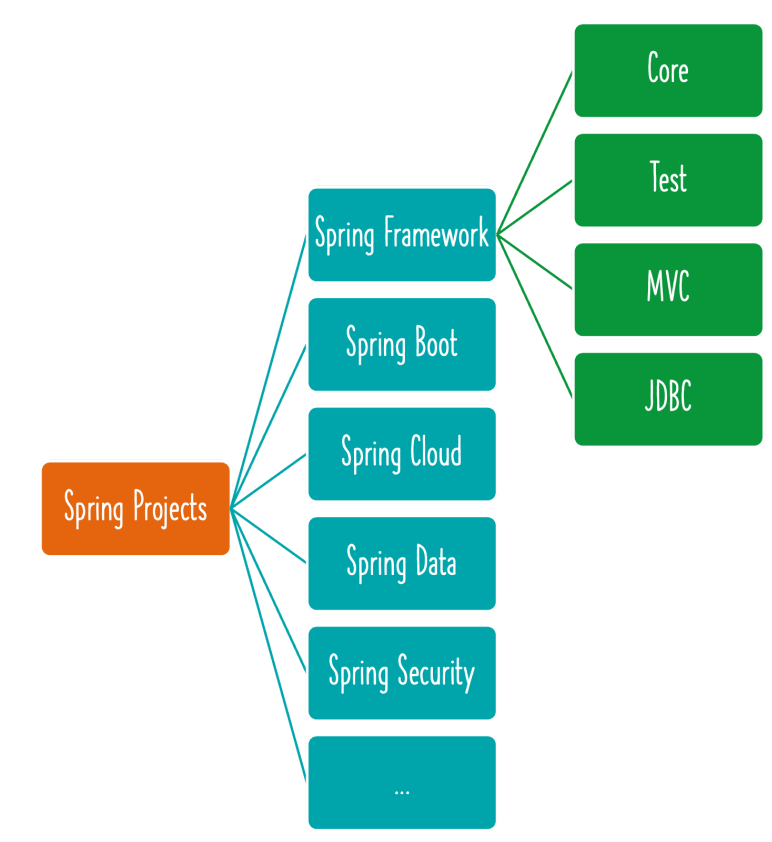
**EXPLORING SPRING BIG PICTURE – FRAMEWORK, MODULES AND PROJECTS**

1. Spring Core: IOC Container, Dependency Injection, Auto Wiring, ...
   1. These are the fundamental building blocks to:
      1. Building web applications
      2. Creating REST API
      3. Implementing authentication and authorization
      4. Talking to a database
      5. Integrating with other systems
      6. Writing great unit tests
2. Let's now get a Spring Big Picture:
   1. Spring Framework
   2. Spring Modules
   3. Spring Projects



**FRAMEWORK AND MODULES**

1. Spring Framework contains multiple Spring Modules:
   1. Fundamental Features: Core (IOC Container, Dependency Injection, Auto Wiring, ...)
   2. Web: Spring MVC etc (Web applications, REST API)
   3. Web Reactive: Spring WebFlux etc
   4. Data Access: JDBC, JPA etc
   5. Integration: JMS etc
   6. Testing: Mock Objects, Spring MVC Test etc
2. Why is Spring Framework divided into Modules?
   1. Each application can choose modules they want to make use of.
   2. They do not need to make use of everything in Spring framework!

Diagram

Description automatically generated

**SPRING PROJECTS**

1. Hierarchy: Spring Projects 🡪 Spring Framework 🡪 Spring Modules
2. Why is Spring Eco system popular?
   1. Loose Coupling: Spring manages creation and wiring of beans and dependencies.
      1. Makes it easy to build loosely coupled applications.
      2. Make writing unit tests easy! (Spring Unit Testing)
   2. Reduced Boilerplate Code: Focus on Business Logic
      1. Example: No need for exception handling in each method!
         1. All Checked Exceptions are converted to Runtime or Unchecked Exceptions
   3. Architectural Flexibility: Spring Modules and Projects
      1. You can pick and choose which ones to use (You DON'T need to use all of them!)
   4. Evolution with Time: Microservices and Cloud
   5. Diagram

      Description automatically generatedSpring Boot, Spring Cloud etc!

A picture containing text, businesscard, vector graphics

Description automatically generated