PayCore Java Spring Bootcamp [2.3]

Homework 6#

18.02.2022

QUESTION 1: What is the difference between manual testing and automated testing?

Answer:

In manual testing, software tests are executed manually by QA Analyst. It is performed to discover bugs in software under development. In this testing, the tester checks all the essential features of software or application.

In automated testing, tests are executed by another codes which is written by testers. For this code, testers use appropriate automation tools to develop the test scripts and validate the software. The goal is to complete test execution in a less amount of time.

QUESTION 2: What does Assert class?

Answer:

Asserts class is used for determining whether our test case fails or pass. They are orıvided by org.junit.Assert class which extends java.lang.Object class.

There are many types of Assert methods like Boolean, Null, Identical etc. in this class. Generally, this class provides a set of assertion methods, useful for writing tests.

QUESTION 3: How can be tested 'private' methods?

Answer:

We don’t use unit test on private methods directly. This is a wrong approach for developers. Instead we test their effects on the public methods that call them.

Long story short, we shouldn’t test our private methods directly. Instead we should test your public interface. If the methods that call your private methods are working as you expect, you then assume by extension that your private methods are working correctly.

QUESTION 4: What is Monolithic Architecture?

Answer:

Monolithic architecture means that the software is designed as self-contained. The components in this architecture are designed as interdependent rather than loosely coupled. In general, monolithic architecture is the development of all functionalities in a self-sufficient application under a single roof.

The advantages of Monolithic Architecture:

1. Easy to manage and monitor.
2. Easy to develop and maintain for small projects. Application can be developed quickly.
3. The interoperability of functionalities is consistent.
4. Transaction management is easy.

The disadvantages of Monolithic Architecture:

1. As the application grows, it becomes more difficult to develop new features and maintain existing code.
2. With the increase in the number of teams and employees working on the project, development and maintenance becomes more difficult.
3. Due to their interdependencies, a change in one functionality may affect other places.
4. There is no possibility to scale a specific functionality. (For example, an invoice is constantly created in the application you are developing, and this is the bottleneck of the application. Even if you want to run this functionality in more than one instance, your application is in monolithic architecture, so you have to scale the entire application instead of just the relevant service.)
5. Version management becomes difficult.
6. The same programming language and the same frameworks should be used in the application.

QUESTION 5: What are the best practices to write a Unit Test Case?

Answer:

1. The smallest particle should be tested.
2. Only one scenario is tested.
3. The steps used are determined.
4. The name of the test method should be a reflection of the scenario being tested.
5. The tested part should be independent from other parts.
6. Our tests should run fully automatically.
7. It should run fast and give quick results.
8. It should be legible, understandable and maintainable.
9. When the test fails, it should stop and return a good error report.

QUESTION 6: Why does JUnit only report the first failure in a single test?

Answer:

Because our tests has to be simple for our senarious. Each test must have generally one reporting fail. Otherwise it is too much for unit test. But Junit is designed to do small test tasks. And it is working way better like that.

QUESTION 7: What are the benefits and drawbacks of Microservices?

Answer:

The benefits of Microservices:

1. Whether the application is very large or very small, it is easy to add new features and maintain existing code. Because it is sufficient to make changes only within the relevant service.
2. The code base of the service will be quite simple as each service is independent from each other and has only its own business logic.
3. Teams can work more efficiently and quickly. Friends who have just started the project can adapt easily without losing the code base.
4. Services can be scaled independently of each other.
5. Versioning is done easily.
6. A change in a service does not require deploying other services. . It is sufficient to just deploy the relevant service.
7. Services can be written in different languages ​​and with different frameworks.

The drawbacks of Microservices:

1. Since there are more than one service and more than one database, transaction management will be difficult.
2. The manageability and monitoring of these services will become difficult.
3. If the services are more independent than necessary, it will be difficult to manage. For example, let's say you have a service called user-service that performs CRUD operations on user. If you step up and want to separate each functionality of the user-service (create, update, delete, get… etc.) into separate services, it will be much more difficult to manage.

QUESTION 8: What is the role of actuator in spring boot?

Answer:

Spring Boot Actuator automatically activates the production-ready features of applications (health check, disk usage, heap dump etc.) and offers a structure that allows interacting with different HTTP endpoints.

QUESTION 9: What are the challenges that one has to face while using Microservices?

Answer:

There are few challenges for Microservices. These are:

1. Design: Designing is more difficult in microservices rather than monolithic. Designing microservices requires creating them within a bounded context. Therefore, each microservice should clarify, encapsulate, and define a specific responsibility.
2. Security: Due to its distributed framework, setting up access controls and administering secured authentication to individual services poses not only a technical challenge but also increases the attack surface substantially.
3. Testing: The testing phase of any[software development lifecycle (SDLC)](https://www.bmc.com/blogs/sdlc-software-development-lifecycle/) is increasingly complex for microservices-based applications.  Given the standalone nature of each microservice, you have to test individual services independently.
4. Increased Operational Complexity: Each microservice’s team is usually tasked with deciding the technology to use and manage it. As each service should be deployed and operated independently, maintaining operations may open a can of worms for those who are not prepared.
5. Communication: Independently deployed microservices act as miniature standalone applications that communicate with each other. To achieve this, you have to configure infrastructure layers that enable resource sharing across services.

QUESTION 10: How independent microservices communicate with each other?

Answer: Generally microservices are distributedand microservices communicatewith each other by inter-service communicationon network level. Each microservice has its own instance and process. Therefore, services must interact using an inter-servicecommunication protocols like HTTP, gRPC or message brokers AMQP protocol. We should be careful when considering communication types and manage them into design phases. Because microservices are complex structure into independently developed.

We have 2 communication types in microservice. Synchronous and Asynchronous communication. In Asynchronous communication, the client sends a request but it doesn’t wait for a response from the service. In synchronous communication, the client sends a request with using http protocols and waits for a response from the service. The synchronous communication protocols can be HTTPor HTTPS.

QUESTION 11: What do you mean by Domain driven design?

Answer:

Domain Driven Design is an approach that tries to provide solutions to the basic problems that are frequently encountered in the development of complex software systems and in ensuring the continuity of our applications after these complex projects are implemented.

Generally, Domain Driven Design advocates a philosophy of how software should be modeled to adapt it to the digital world by creating real-world business models with a common language (Ubiquitous Language) that everyone can understand.

QUESTION 12: What is container in Microservices?

Answer:

Basicly, we can say that containers are packages of our software that include everything that it needs to run. This includes code, dependencies, libraries, binaries, and more.

Containers are immutable: After they’re deployed, they can’t (or shouldn’t) be altered. If a new version of code becomes available, the container is destroyed, and a new container with the latest code is deployed in its place. Containers can launch in seconds or milliseconds, so additional service components can be deployed immediately when and where they are needed.

QUESTION 13: What are the main components of Microservices architecture?

Answer:

There are 8 main components for Microservice architecture:

1. Clients
2. Identity Providers
3. API Gateway
4. Messaging Formats
5. Databases
6. Static Content
7. Management
8. Service Discovery

QUESTION 14: How does a Microservice architecture work?

Answer:

A microservice architecture has main idea. This idea is that applications are simpler to build and maintain when broken down into smaller pieces that work seamlessly together. When using microservices, we isolate software functionality into multiple independent modules that are individually responsible for performing precisely defined and standalone tasks. These modules communicate with each other through simple, universally accessible application programming interfaces (APIs).