# What is the difference between manual testing and automated testing ?

|  |  |
| --- | --- |
| Manual Testing | Automated Testing |
| Manual testing is not accurate at all times due to human error, hence it is less reliable. | Automated testing is more reliable, as it is performed by tools and/or scripts. |
| Manual testing is time-consuming, taking up human resources. | Automated testing is executed by software tools, so it is significantly faster than a manual approach. |
| Investment is required for human resources. | Investment is required for testing tools. |
| Manual testing is only practical when the test cases are run once or twice, and frequent repetition is not required. | Automated testing is a practical option when the test cases are run repeatedly over a long time period. |
| Manual testing allows for human observation, which may be more useful if the goal is user-friendliness or improved customer experience. | Automated testing does not entail human observation and cannot guarantee user-friendliness or positive customer experience. |

# What does Assert class ?

As you may have figured out from the simple test, most of the secret of implementing JUnit unit tests, is in the use of the assert methods in the class org.junit.Assert.

Here is a list of the assert methods:

*assertArrayEquals()*

*assertEquals()*

*assertTrue() + assertFalse()*

*assertNull() + assertNotNull()*

*assertSame() + assertNotSame()*

*assertThat()*

# How can be tested 'private' methods ?

Private methods are great for breaking logic into smaller parts. If a method gets too big, you should refactor it and break it down into private methods so it’s easier to read. But you don’t need to test those private methods individually.

To test private methods, you just need to test the public methods that call them. Call your public method and make assertions about the result or the state of the object. If the tests pass, you know your private methods are working correctly.

# What is Monolithic Architecture ?

A monolithic architecture is the traditional unified model for the design of a software program. Monolithic, in this context, means composed all in one piece. According to the Cambridge dictionary, the adjective monolithic also means both too large and unable to be changed.

Monolithic software is designed to be self-contained; components of the program are interconnected and interdependent rather than loosely coupled as is the case with modular software programs. In a tightly-coupled architecture, each component and its associated components must be present in order for code to be executed or compiled.

# What are the best practices to write a Unit Test Case ?

* Write readable tests
* Avoid magic numbers and magic strings
* Write deterministic tests
* Avoid test interdependencies
* Avoid logic in tests
* Refrain multiple asserts in a single unit test
* Keep your tests away from too much implementation details
* Write tests during development, not after it
* Automate tests using CI/CD tools
* Update the tests periodically

# Why does JUnit only report the first failure in a single test ?

Reporting multiple failures in a single test is generally a sign that the test does too much and it is too big a unit test. JUnit is designed to work best with a number of small tests. It executes each test within a separate instance of the test class.

# What are the benefits and drawbacks of Microservices ?

**Advantages**

* **Improved fault isolation**: Larger applications can remain mostly unaffected by the failure of a single module.
* **Eliminate vendor or technology lock-in**: Microservices provide the flexibility to try out a new technology stack on an individual service as needed. There won’t be as many dependency concerns and rolling back changes becomes much easier. With less code in play, there is more flexibility.
* **Ease of understanding:**With added simplicity, developers can better understand the functionality of a service.
* **Smaller and faster deployments**: Smaller codebases and scope = quicker deployments, which also allow you to start to explore the benefits of Continuous Deployment.
* **Scalability**: Since your services are separate, you can more easily scale the most needed ones at the appropriate times, as opposed to the whole application. When done correctly, this can impact cost savings.

**Disadvantages**

* **Communication between services is complex**: Since everything is now an independent service, you have to carefully handle requests traveling between your modules. In one such scenario, developers may be forced to write extra code to avoid disruption. Over time, complications will arise when remote calls experience latency.
* **More services equals more resources**: Multiple databases and transaction management can be painful.
* **Global testing is difficult**: Testing a microservices-based application can be cumbersome. In a monolithic approach, we would just need to launch our WAR on an application server and ensure its connectivity with the underlying database. With microservices, each dependent service needs to be confirmed before testing can occur.
* **Debugging problems can be harder**: Each service has its own set of logs to go through. Log, logs, and more logs.
* **Deployment challengers**: The product may need coordination among multiple services, which may not be as straightforward as deploying a WAR in a container.
* **Large vs small product companies**: Microservices are great for large companies, but can be slower to implement and too complicated for small companies who need to create and iterate quickly, and don’t want to get bogged down in complex orchestration.

# What is the role of actuator in spring boot ?

Spring Boot Actuator is a sub-project of the Spring Boot Framework. It includes a number of additional features that help us to monitor and manage the Spring Boot application. It contains the actuator endpoints (the place where the resources live). We can use HTTP and JMX endpoints to manage and monitor the Spring Boot application. If we want to get production-ready features in an application, we should use the Spring Boot actuator.

# What are the challenges that one has to face while using Microservices ?

* Confusions regarding each microservice’s size and a higher number of Moving parts (Services, Databases, Processes, Containers, Frameworks).
* Optimal boundaries and connection points between each microservice
* Designing the entire framework to integrate services is harder and time-consuming.

# How independent microservices communicate with each other?

Every microservice in order to communicate either synchronously or asynchronously with other microservices.

“Synchronous - HTTP is a synchronous protocol. The client sends a request and waits for a response from the service. That is independent of the client code execution that could be synchronous (thread is blocked) or asynchronous (thread is not blocked, and the response will reach a callback eventually). The important point here is that the protocol (HTTP/HTTPS) is synchronous and the client code can only continue its task when it receives the HTTP server response.”

“Asynchronous - Other protocols like AMQP (a protocol supported by many operating systems and cloud environments) use asynchronous messages. The client code or message sender usually does not wait for a response. It just sends the message as when sending a message to a RabbitMQ queue or any other message broker.”

# 11. What do you mean by Domain driven design ?

Microservices is an architecture design model with a specific bounded context, configuration, and dependencies. These result from the architectural principles of the domain-driven design and DevOps. Domain-driven design is the idea of solving problems of the organization through code.

The business goal is important to the business users, with a clear interface and functions. This way, the microservice can run independently from other microservices

# 12. What is container in Microservices ?

Containers are a form of operation system virtualization. Single container can be used for running a

small microservice to a software process of a larger application. Container contains all the necessary

executables , binary code, libraries, and configuration files. However , when compared to servers

they do not contain operation system images. This makes them more lightweight and portable.

# 13. What are the main components of Microservices architecture ?

* clients
* identity providers
* api gateway
* messaging formats
* databases
* static content
* management
* service discovery

# 14. How does a Microservice architecture work?

The microservice architecture contains components depending on the business requirements.

API Gateway- Clients need API Gateway as it is an entry point, which forwards the call to the specific services on the back end. Here API gateway helps in collecting the responses from different services and returns the response to the client.

Microservices- As the name itself suggests that microservices are the services that help in dividing the service into small services that perform a certain business capability like user registration, current orders, or wish list.

Database- Microservices can either share the same database or an independent database.

Inter-microservices communication- REST or Messaging are the protocol to interact with each other.

