**submit() vs execute() in Java Executors 🚀**

Both submit() and execute() are methods in the **ExecutorService** interface for running tasks asynchronously. However, they have key differences.

**1️⃣ execute() Method**

* Part of **Executor** interface.
* **Returns void** → No result is returned.
* Used when the task is **fire-and-forget**.
* Throws **exceptions directly** if the task fails.

**📌 Example of execute()**

import java.util.concurrent.ExecutorService;

import java.util.concurrent.Executors;

public class ExecuteExample {

public static void main(String[] args) {

ExecutorService executor = Executors.newFixedThreadPool(2);

Runnable task = () -> System.out.println(Thread.currentThread().getName() + " is running...");

executor.execute(task); // Fire-and-forget

executor.shutdown();

}

}

**🔴 Key Points of execute()**

✔ Runs the task asynchronously.  
✔ No way to get the result.  
✔ Exceptions are thrown immediately.

**2️⃣ submit() Method**

* Part of **ExecutorService** interface.
* **Returns a Future<T>** → Allows getting results.
* Suitable for **tasks that return values**.
* **Catches exceptions** and stores them in Future.

**📌 Example of submit()**

import java.util.concurrent.\*;

public class SubmitExample {

public static void main(String[] args) throws ExecutionException, InterruptedException {

ExecutorService executor = Executors.newFixedThreadPool(2);

Callable<Integer> task = () -> {

Thread.sleep(1000);

return 10 \* 2;

};

Future<Integer> future = executor.submit(task);

System.out.println("Doing other work...");

// Get result (blocks until task completes)

Integer result = future.get();

System.out.println("Result: " + result);

executor.shutdown();

}

}

**🔴 Key Points of submit()**

✔ Returns Future<T> → Can get the result later.  
✔ Exceptions are captured and retrieved via future.get().  
✔ Can handle **Runnable & Callable** tasks.

**3️⃣ Key Differences Between execute() and submit()**

| **Feature** | **execute()** | **submit()** |
| --- | --- | --- |
| **Returns a Result?** | ❌ No | ✅ Yes (via Future<T>) |
| **Handles Exceptions?** | ❌ No (throws immediately) | ✅ Yes (captures inside Future) |
| **Accepts Callable<T>?** | ❌ No | ✅ Yes |
| **Blocking?** | ❌ No | ✅ Only when calling get() |
| **When to Use?** | Fire-and-forget tasks | Tasks requiring a result |

**4️⃣ When to Use What?**

* **Use execute()** when **no result is needed**, and the task should run asynchronously.
* **Use submit()** when **you need a result** or want to **handle exceptions properly**.

**✅ Final Thoughts**

* If you don’t care about the result → Use execute().
* If you need a result or exception handling → Use submit().

Would you like an example of handling exceptions with Future? 🚀