**Handling Exceptions with Future in Java Executors 🚀**

When using submit(), exceptions are **not thrown immediately** like in execute(). Instead, they are **captured inside the Future** and rethrown when calling future.get().

**📌 Example: Handling Exceptions in Future**

import java.util.concurrent.\*;

public class FutureExceptionHandling {

public static void main(String[] args) {

ExecutorService executor = Executors.newFixedThreadPool(2);

Callable<Integer> task = () -> {

System.out.println(Thread.currentThread().getName() + " is running...");

if (true) throw new RuntimeException("Something went wrong!");

return 100; // This will never execute

};

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

try {

Integer result = future.get(); // Exception is thrown here

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

} catch (InterruptedException e) {

System.out.println("Thread was interrupted!");

} catch (ExecutionException e) {

System.out.println("Exception occurred: " + e.getCause()); // Get original exception

} finally {

executor.shutdown();

}

}

}

**🔴 Output**

pool-1-thread-1 is running...

Exception occurred: java.lang.RuntimeException: Something went wrong!

**⚡ Key Takeaways**

1. **Exceptions are NOT thrown immediately** when using submit().
2. **They are stored inside Future** and rethrown when calling future.get().
3. Use **ExecutionException.getCause()** to retrieve the actual exception.

**✅ Alternative: Handling Exceptions in execute()**

If we used execute(), the exception would be thrown immediately and crash the program.

executor.execute(() -> {

throw new RuntimeException("Error in execute!");

});

☝ **This will throw an exception immediately** and crash the thread.

**❓ Would you like an example using CompletableFuture for better exception handling? 🚀**

**Why does submit() not throw exceptions immediately, but execute() does?**

This difference is due to how **submit()** and **execute()** handle exceptions internally in Java's Executor framework.

**1️⃣ execute() (Immediate Exception Handling)**

* execute() directly runs the task on a thread from the thread pool.
* If an exception occurs, **it is thrown immediately** and not caught by the executor.
* If there is no explicit try-catch, the **thread will crash**.

**Example:**

import java.util.concurrent.ExecutorService;

import java.util.concurrent.Executors;

public class ExecuteExceptionExample {

public static void main(String[] args) {

ExecutorService executor = Executors.newFixedThreadPool(2);

executor.execute(() -> {

System.out.println(Thread.currentThread().getName() + " running...");

throw new RuntimeException("Error in execute!");

});

executor.shutdown();

}

}

**🔴 Output**

pool-1-thread-1 running...

Exception in thread "pool-1-thread-1" java.lang.RuntimeException: Error in execute!

* The exception **is thrown directly in the thread** and **crashes it**.

**2️⃣ submit() (Exception Stored in Future)**

* submit() does **not throw exceptions immediately**.
* Instead, it **wraps the exception inside a Future object**.
* The exception is **only thrown when calling future.get()**.

**Example:**

import java.util.concurrent.\*;

public class SubmitExceptionExample {

public static void main(String[] args) {

ExecutorService executor = Executors.newFixedThreadPool(2);

Future<?> future = executor.submit(() -> {

System.out.println(Thread.currentThread().getName() + " running...");

throw new RuntimeException("Error in submit!");

});

try {

future.get(); // Exception is thrown here

} catch (InterruptedException | ExecutionException e) {

System.out.println("Caught exception: " + e.getCause());

}

executor.shutdown();

}

}

**🟢 Output**

pool-1-thread-1 running...

Caught exception: java.lang.RuntimeException: Error in submit!

* The **exception is stored inside Future** and thrown when future.get() is called.

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

| **Feature** | **execute(Runnable)** | **submit(Runnable/Callable)** |
| --- | --- | --- |
| Returns a result? | ❌ No | ✅ Yes (Future) |
| Exception Handling | ❌ **Exception is thrown immediately** | ✅ **Exception is captured in Future and thrown on future.get()** |
| Thread crash? | ✅ **Yes** (if uncaught) | ❌ No (Exception is handled in Future) |
| Recommended for? | **Fire-and-forget tasks** | **Tasks that return results** |

**When to Use Which?**

✅ Use **execute()** when you **don’t need a result**, and exceptions should be handled immediately.  
✅ Use **submit()** when you **need a result** (e.g., Future) or want to handle exceptions later.

Would you like an example of **exception handling using CompletableFuture**, which provides better async handling? 🚀