Study - Selenium Java TestNG Interview FAQs

Retryanalyyzer vs innotation transformer

Great one 👍 let's break it down carefully. Both `IRetryAnalyzer` and `IAnnotationTransformer` are TestNG extension interfaces — but they serve **different purposes**.

🔁 `IRetryAnalyzer`

- **Goal**: Rerun failed tests automatically.
- **How**: You implement the interface `IRetryAnalyzer` and override `retry(ITestResult result)` → return `true` if you want TestNG to rerun the test, `false` if not.

Example:

java

```java
public class RetryAnalyzer implements IRetryAnalyzer {
    private int count = 0;
    private static final int MAX_RETRY = 2;

    @Override
    public boolean retry(ITestResult result) {
        if (count < MAX_RETRY) {
            count++;
            return true; // rerun test
        }
        return false; // stop retrying
    }
}
```

Attach it at test level:

java

```java
@Test(retryAnalyzer = RetryAnalyzer.class)
public void testLogin() {
```

```
    Assert.assertTrue(false); // will retry 2 times
}
```

👉 Use case: When tests are flaky due to network/UI slowness, you can rerun automatically.

🎭 `IAnnotationTransformer`

- **Goal**: Modify TestNG annotations **at runtime** (before tests run).
- **How**: You implement `IAnnotationTransformer` and override `transform()` method.
- You can programmatically change things like:
  - RetryAnalyzer assignment
  - Invocation count
  - Enabled/disabled
  - Priority
  - Groups

Example:

```java
public class RetryTransformer implements IAnnotationTransformer {
    @Override
    public void transform(ITestAnnotation annotation, Class testClass,
                          Constructor testConstructor, Method testMethod) {
        // Assign retry analyzer dynamically if not already present
        if (annotation.getRetryAnalyzer() == null) {
            annotation.setRetryAnalyzer(RetryAnalyzer.class);
        }
    }
}
```

Register transformer in `testng.xml`:

```xml
<suite name="Suite" parallel="false">
    <listeners>
        <listener class-name="com.example.RetryTransformer"/>
    </listeners>
```

```
    <test name="Test">
      <classes>
        <class name="com.example.MyTests"/>
      </classes>
    </test>
</suite>
```

👉 Use case: Instead of annotating **every test** with `@Test(retryAnalyzer=...)`, you apply retry globally via transformer.

✅ Key Differences

Feature	`IRetryAnalyzer`	`IAnnotationTransformer`
Purpose	Retry failed tests	Modify test annotations dynamically
Scope	Works on one test (where attached)	Can affect all tests globally
How to use	Attach via `@Test(retryAnalyzer=...)`	Register in `testng.xml` listeners
Control	Only controls retry attempts	Can change retry, priority, groups, enabled, etc.
When executed	After a test fails	Before tests start running

👉 Think of it like this:

- **RetryAnalyzer = local fix (per test)**
- **AnnotationTransformer = global policy setter (all tests, runtime control)**

Do you want me to also show you a **combined example** where `IAnnotationTransformer` automatically applies a `RetryAnalyzer` to all tests (so you don't have to write `retryAnalyzer=...` everywhere)?