Em **pytest**, o **escopo** de uma fixture define o período de tempo (ou ciclo de vida) durante o qual a fixture é criada e permanece ativa. Basicamente, o escopo determina:

1. **Quando a fixture será criada.**
2. **Quando a fixture será destruída.**

O pytest oferece quatro níveis de escopo para fixtures:

**1. function (Padrão)**

* **Descrição**: A fixture é criada e destruída para cada função de teste que a utiliza.
* **Quando usar**: Quando os recursos configurados pela fixture só são relevantes para uma única função de teste.
* **Exemplo**:

Python:

import pytest

@pytest.fixture

def recurso\_temporario():

return "dados temporários"

def test\_exemplo(recurso\_temporario):

assert recurso\_temporario == "dados temporários"

* + Neste caso, recurso\_temporario será criado e descartado para cada teste que o utilize.

**2. class**

* **Descrição**: A fixture é criada uma vez por classe de teste e compartilhada entre todos os métodos dessa classe.
* **Quando usar**: Quando os métodos de teste dentro de uma mesma classe podem compartilhar o mesmo recurso.
* **Exemplo**:

Python:

@pytest.fixture(scope="class")

def recurso\_da\_classe():

return "dados para a classe"

class TestClasse:

def test\_um(self, recurso\_da\_classe):

assert recurso\_da\_classe == "dados para a classe"

def test\_dois(self, recurso\_da\_classe):

assert recurso\_da\_classe == "dados para a classe"

**3. module**

* **Descrição**: A fixture é criada uma vez por módulo de teste e compartilhada entre todos os testes do mesmo arquivo.
* **Quando usar**: Quando múltiplos testes no mesmo módulo (arquivo) podem usar o mesmo recurso configurado.
* **Exemplo**:

Python:

@pytest.fixture(scope="module")

def recurso\_do\_modulo():

return "dados para o módulo"

def test\_exemplo\_um(recurso\_do\_modulo):

assert recurso\_do\_modulo == "dados para o módulo"

def test\_exemplo\_dois(recurso\_do\_modulo):

assert recurso\_do\_modulo == "dados para o módulo"

**4. session**

* **Descrição**: A fixture é criada apenas uma vez para toda a sessão de testes (independente de quantos módulos ou testes sejam executados).
* **Quando usar**: Para recursos globais, como configurações de banco de dados ou instâncias que podem ser reutilizadas em todos os testes.
* **Exemplo**:

Python:

@pytest.fixture(scope="session")

def recurso\_da\_sessao():

return "dados para a sessão inteira"

def test\_um(recurso\_da\_sessao):

assert recurso\_da\_sessao == "dados para a sessão inteira"

def test\_dois(recurso\_da\_sessao):

assert recurso\_da\_sessao == "dados para a sessão inteira"

**Como definir o escopo de uma fixture?**

* Use o parâmetro scope no decorator @pytest.fixture.
* Exemplo:

Python:

@pytest.fixture(scope="module")

def minha\_fixture():

return "dados compartilhados"

**Resumo do Ciclo de Vida:**

* **Criação**: A fixture é criada no início do escopo.
* **Destruição**: A fixture é automaticamente descartada após o fim do escopo, geralmente por meio de um *teardown* (usando yield ou um gancho de finalização, como request.addfinalizer).