现在我们有两个测试方法，testEntityManager()和testEntityManagerQuery()，在两个测试方法中，都使用bootKernel()方法来启动内核准备容器，我们可以把启动内核准备容器的这个方法，统一放到一个setUp()方法中进行处理。

我们打开phpunit文档，在第四章，phpunit提供共享，建立基境的代码。在运行某个测试方法之前，会调用一个名叫setUp()的模板方法。setUp()是创建测试所用对象的地方，当测试方法进行结束后，不管是成功还是失败，都会调用另外一个方法tearDown()方法。tearDown()是清理测试所用对象的方法。

setUp()方法和tearDown()方法是对应的，我们添加setUp()方法，直接覆盖方法。在setUp()方法中，我们启动内核，注释19行代码，注释43行代码。bootKernel()方法最终的返回值是静态变量$kernel，所以我们直接可以在21行使用self::$kernel，另外$entitymanager我们可能在以后的测试中还会经常用到，所以我们把$entitymanager的获取也放到setUp()方法中。

我们把$entitymanager变量设置为私有属性，注释24行代码，修改代码，我们可以在setUp()方法中执行数据库的清除。，这样每次在执行测试时都会首先清空数据库，我这里就直接复制代码了，我们创建一个私有方法。

```
class EntityManagerTest extends KernelTestCase
{
    private $entitymanager;

    protected function setUp(): void
    {
        self::bootKernel();
        $this->entitymanager = static::getContainer()-
>get('doctrine.orm.entity_manager');

        $this->truncateEntities([
            Post::class
        ]);
    }

    public function testEntityManager(): void
    {
//        $kernel = self::bootKernel();

        $this->assertSame('test', self::$kernel->getEnvironment());
//        $entitymanager = static::getContainer()-
>get('doctrine.orm.entity_manager');
        $this->assertInstanceOf(EntityManagerInterface::class, $this-
>entitymanager);
        //$myCustomService = self::$container->get(CustomService::class);

        $factory = static::getContainer()->get(PostFactory::class);
        $this->assertInstanceOf(PostFactory::class, $factory);

        $post1 = $factory->create('Post title 01', 'Post Body 01');
        $post2 = $factory->create('Post title 02', 'Post Body 02', null,
'published');
        $post3 = $factory->create('Post title 03', 'Post Body 03');
        $post4 = $factory->create('Post title 04', 'Post Body 04');
```

```
        $this->entitymanager->persist($post1);
        $this->entitymanager->persist($post2);
        $this->entitymanager->persist($post3);
        $this->entitymanager->persist($post4);

        $this->entitymanager->flush();

        $postRepo = static::getContainer()->get(PostRepository::class);

        $this->assertInstanceOf(PostRepository::class, $postRepo);

        $posts = $postRepo->findAll();
        $this->assertCount(4, $posts);
    }

//    public function testEntityManagerQuery():void
//    {
////        $kernel = self::bootKernel();
//
//        $postRepo = static::getContainer()->get(PostRepository::class);
//
//        $this->assertInstanceOf(PostRepository::class, $postRepo);
//
//        $posts = $postRepo->findAll();
//        $this->assertCount(4, $posts);
//    }

    private function truncateEntities(array $entities)
    {
        $connection = $this->entitymanager->getConnection();
        $databasePlatform = $connection->getDatabasePlatform();
        if ($databasePlatform->supportsForeignKeyConstraints()) {
            $connection->executeQuery('SET FOREIGN_KEY_CHECKS=0');
        }
        foreach ($entities as $entity) {
            $query = $databasePlatform->getTruncateTableSQL(
                $this->entitymanager->getClassMetadata($entity)->getTableName()
            );
            $connection->executeQuery($query);
        }
        if ($databasePlatform->supportsForeignKeyConstraints()) {
            $connection->executeQuery('SET FOREIGN_KEY_CHECKS=1');
        }
    }
}
```

truncateEntities()这个方法中，orm框架会根据不同的数据库创建不同的清空数据库表的SQL语句，然后执行SQL语句来清空数据库，我们在setUp()方法中直接调用清空数据库的方法，这里我们需要清空的是post数据库表，phpunit中的每个测试方法都是独立的，在testEntityManager()方法运行之前，它会启动内核，清空数据库 然后再插入数据。

`testEntityManagerQuery()`方法执行之前，它也会执行setUp()方法，这时数据库它是没有内容的，所以测试代码还会失败。我们把 `testEntityManagerQuery()`方法中的代码粘贴到上个方法中,然后注释掉`testEntityManagerQuery()`方法,所有的测试代码都在 `testEntityManager()`方法中。在testEntityManager()方法执行之前，它会启动内核，清空数据库，然后获取EntityManager对象，再执行插入方法，插入方法执行完之后，读取数据库，然后验证数据库。

我们再次执行测试代码，测试代码出错了，在清空数据库之前，首先要获取到EntityManager对象，我们调整一下代码，再次执行测试代码。这次测试就通过了，我们再次执行，测试仍然通过。

我们已经使用了findAll()方法，在下一节课，我们将测试PostRepository提供另外三个方法。