Built-in Test containers

During the integration tests, we’ll spin up a Docker container containing the database server. Since the database port exposed by the container will be dynamically allocated, we cannot define the database URL in the properties file. **As a result, for a Spring Boot application with a version prior to 3.1, we’d need to use @DynamicPropertySource annotation in order to add these properties to a DynamicPropertyRegistry:**

@DynamicPropertySource  
static void setProperties(DynamicPropertyRegistry dynamicPropertyRegistry){  
 dynamicPropertyRegistry.add("spring.data.mongodb.uri",***mongoDBContainer***::getReplicaSetUrl);  
}

Or

@Testcontainers  
@SpringBootTest(webEnvironment = DEFINED\_PORT)  
class DynamicPropertiesIntegrationTest {  
 @Container  
 Static MongoDBContainer mongoDBContainer = new MongoDBContainer(DockerImageName.parse("mongo:4.0.10"));  
 @DynamicPropertySource  
 static void setProperties(DynamicPropertyRegistry registry) {  
 registry.add("spring.data.mongodb.uri", mongoDBContainer::getReplicaSetUrl);  
 }  
 *// ...*}

For the integration test, we’ll use the **@SpringBootTest** annotation to start the application on the port defined in the configuration files. Additionally, we’ll use Testcontainers for setting up the environment.

[Intro to the Jackson ObjectMapper](https://www.baeldung.com/jackson-object-mapper-tutorial)

This tutorial focuses on understanding the Jackson ObjectMapper class and how to serialize Java objects into JSON and deserialize JSON string into Java objects.