# **十二、集成Springboot**

Axon框架为Spring提供了广泛的支持，但并不要求您使用Spring来使用Axon。所有组件都可以通过编程方式配置，并且不需要在类路径上使用Spring。但是，如果您确实使用Spring，那么使用Spring的注释支持可以使大部分配置变得更简单。Axon在此基础上提供了SpringBoot启动器，因此您也可以从自动配置中获益。

## **12.1 自动配置**

到目前为止，Axon的springboot自动配置是开始配置Axon组件的最简单的选择。通过简单地声明对axonspringbootstarter的依赖性，axon将自动配置基础结构组件（命令总线、事件总线、查询总线），以及运行和存储聚合和saga所需的任何组件。

## **12.2揭开Axon Springboot的神秘面纱**

由于很多事情都是在后台发生的，有时候很难理解注释或者仅仅包含一个依赖项是如何支持这么多特性的。

axon spring boot starter在构造启动器时遵循一般的spring boot约定。它依赖于axon spring boot autoconfigure，它包含axon自动配置的具体实现。当AxonSpringBoot应用程序启动时，它会在类路径中查spring factory。此文件位于axon spring boot自动配置模块的META-INF目录中。

  这个文件映射了AxonSpringBoot，应用程序将尝试应用的不同配置类。因此，根据这个片段，springboot将尝试为AxonServerAutoConfiguration、AxonAutoConfiguration应用所有的配置类。

是否应用这些配置类，将取决于在这些类上定义的条件：

AxonServerAutoConfiguration将Axon服务器配置为命令总线、查询总线和事件存储的实现。它将在AxonAutoConfiguration之前应用，并且仅当org.axonframework.axonserver.connector.axonServer配置类在类路径中可用。可以通过设置已启用axon.axonserver属性在.properties/.yml文件中设置为false。

AxonAutoConfiguration配置命令总线、查询总线、事件存储/事件总线和其他axon组件的“非axon服务器”实现。这些组件只有在不在Spring应用程序上下文中时才会被初始化，例如@ConditionalOnMissingBean(事件总线.class). 由于AxonServerAutoConfiguration将在AxonServer自动配置之后应用，这些Axon组件将已经在Spring应用程序上下文中，因此Axon服务器的命令总线、查询总线和事件存储/事件总线将会成功实现。

Axon Spring Boot自动配置不具侵入性。它将只定义尚未在应用程序上下文中显式定义的Spring组件。这允许您通过在一个@Configuration类中定义自己的bean来完全覆盖自动配置的bean。

具体的Axon（Spring）组件配置将在本指南的以下部分进行详细说明。

除此之外，Axon包还提供了其他很多工具框架的支持：

# **十三、Axon服务器**

# **13.1安装**

本部分包括axon SE和axon EE，需要在java8或者java11的环境下安装。

# **13.1.1 Axon Server SE 本地安装**

二进制文件：

Axon server SE-ZIP下载包含服务器本身和CLI的可执行JAR文件。复制axonserver.jar/axonserver-cli.jar到你选择的目录。

运行Axon Server SE

从文件提取的位置，运行以下命令：$ ./axonserver.jar

这将使用默认端口启动Axon SE服务器~~8024用于HTTP/8124用于gRPC。

HTTP端口用于为管理UI和Axon server SE提供的rest api提供服务。gRPC端口由Axon框架客户端应用程序用来连接到Axon服务器SE。管理界面可以在“”http://localhost：8024“打开，而REST API可在”http://localhost：8024/v1打开。

Rest api在“/v1/public/me”提供了一个操作，以获取正在运行的Axon server SE实例的配置详细信息。下面给出了响应的表示。

{

  "authentication": false,

  "clustered": false,

  "ssl": false,

  "adminNode": true,

  "developmentMode": false,

  "storageContextNames": ["default"],

  "contextNames": [ "default" ],

  "httpPort": 8024,

  "grpcPort": 8124,

  "internalHostName": null,

  "grpcInternalPort": 0,

  "name": ${hostname},

  "hostName": ${hostname}

}

总结来说：

1. 默认情况下，未启用访问控制和SSL。以下各节详细介绍了如何配置此功能。
2. AxonServerSE为事件存储和消息路由提供了一个名为“default”的上下文。不可能创建任何其他上下文。
3. 默认端口为8024/8124。这些值可以通过配置进行更改。
4. 名称和主机名默认为运行Axon server SE的系统主机名。这些值可以通过配置更改（“axoniq.axonserver.name” / “axoniq.axonserver.hostname”).
5. 群集在Axon Server SE中不可用。
6. “主机名”和“内部端口”不适用于服务器

以上就完成了一个使用所有默认值的Axon服务器SE的快速设置。它现在可以作为事件存储和消息路由器使用。

访问控制

由于Axon服务器是一个事件存储，可能包含敏感数据，因此在生产环境和类似生产环境中启用访问控制始终是一个很好的实践。

访问控制部分详细介绍了在Axon服务器SE中设置访问控制所需的步骤。

SSL

Axon Server SE支持TLS/SSL（传输层安全/安全套接字层）来加密Axon Server SE的所有网络流量—从Axon框架客户端应用程序到Axon Server SE。

Axon Server SE有两个端口（HTTP/gRPC）需要为SSL启用，因此需要使用两组不同的设置，每个端口一组。

SSL部分详细介绍了在Axon服务器SE中设置SSL所需的步骤。

存储

默认情况下，Axon服务器SE将在当前目录中查找名为“data”的目录，并在其中查找“default”目录。这是存储“默认”上下文的事件和快照的位置。可以使用axoniq.axonserver.event.存储“和”axoniq.axonserver.snapshot.storage“设置。在“data”目录中还有一个小数据库，称为“ControlDB”，用于管理数据。您可以使用“…controldb path”设置自定义此位置。

配置部分详细说明了设置Axon Server SE存储所需的步骤。

开发模式

Axon Server SE可以在开发模式下启动，这使得一些特性能够为开发提供方便。

可以通过配置以下属性来启用此功能：

axoniq.axonserver.devmode.enabled=true

重置事件

在创建新特性的同时，可以方便地将Axon服务器恢复到干净的状态，而不存储任何事件。在针对系统编写和运行集成测试时，这也很有帮助。请注意，未存储在Axon服务器中的数据（例如跟踪令牌）不会从此功能中删除。必须手动删除或重置。

重置Axon服务器可以通过CLI以及UI和REST接口完成。

注意：当Axon服务器以群集模式运行时，此功能被禁用。