Sebastian Lammering

Endlich mit Struktur

Services mit Spring Modulith designen

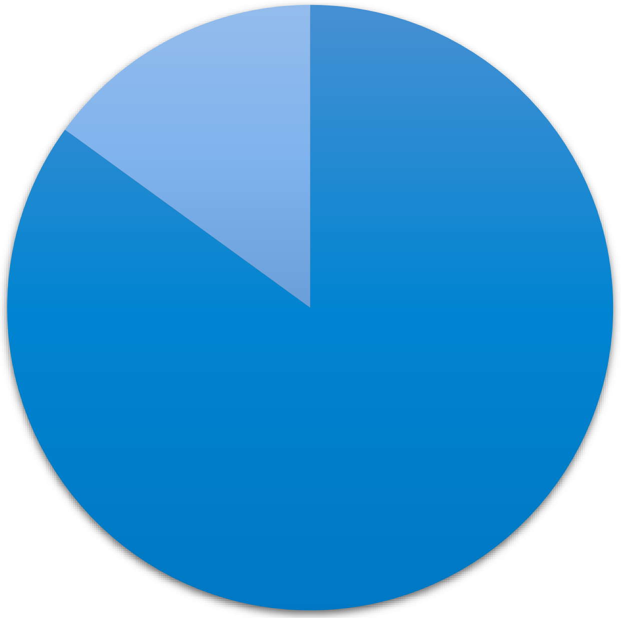
code of change

Hyand by GOD | MT

RUHRPULLS

Microservices are established

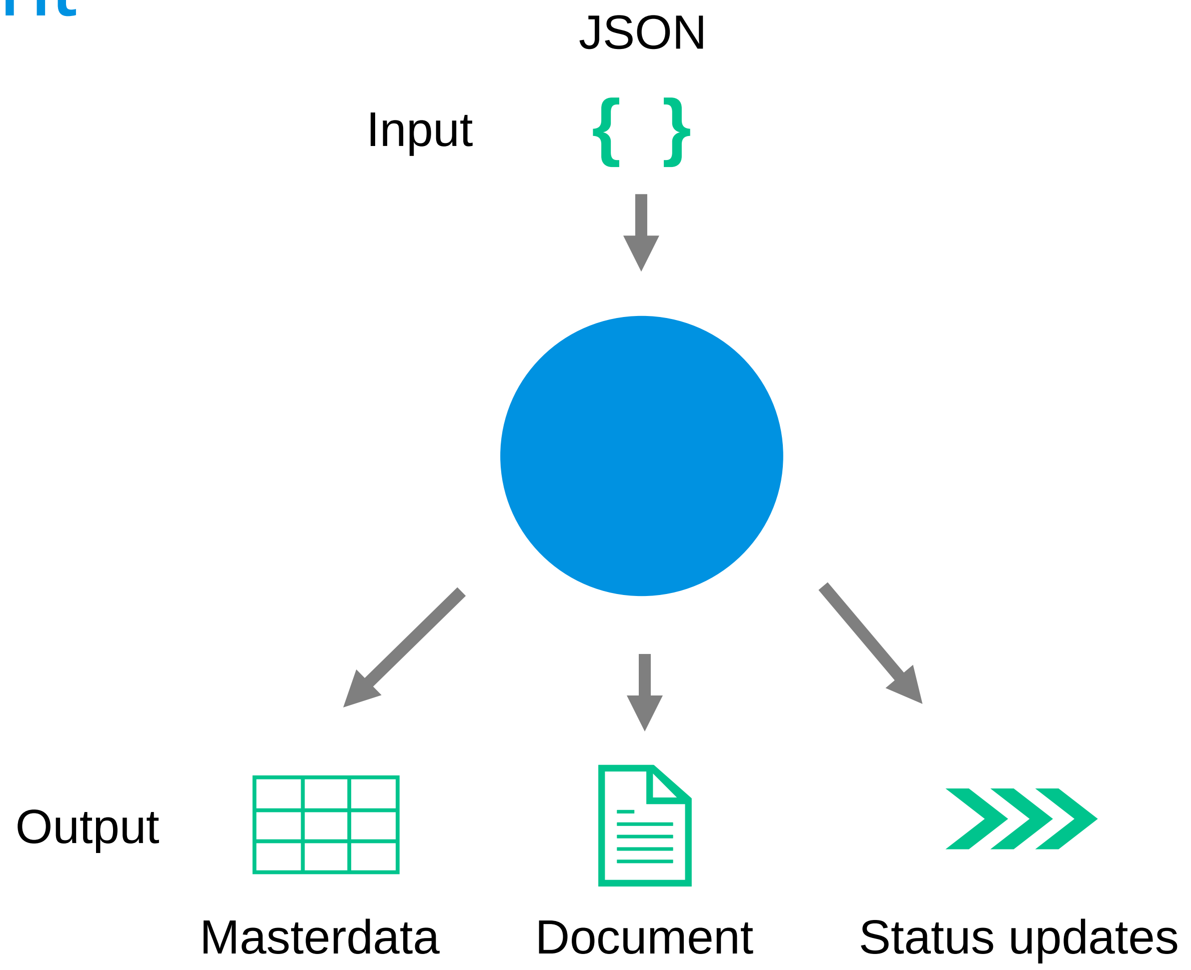
Usage of microservices



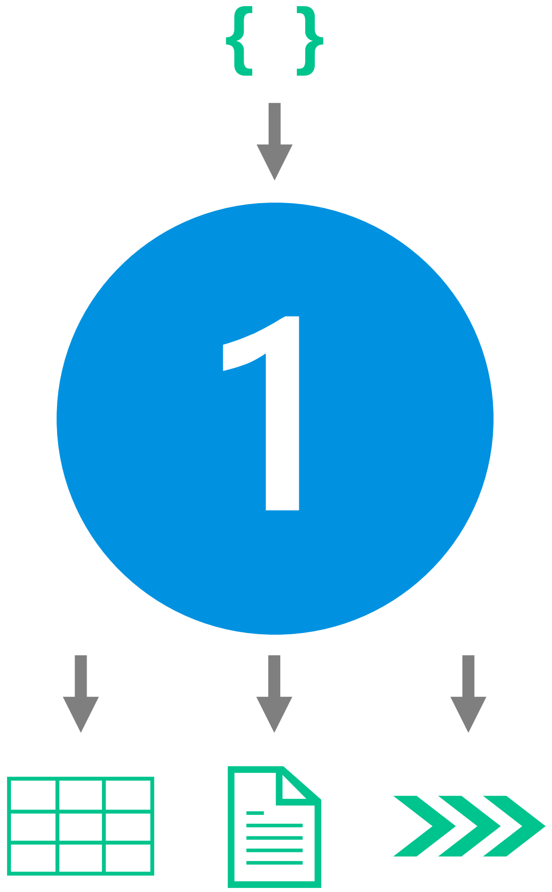
In 2021, **85 percent** of respondents from large organizations with 5,000 or more employees indicated that they are currently using microservices.

https://www.statista.com/statistics/1236823/microservices-usage-per-organization-size/

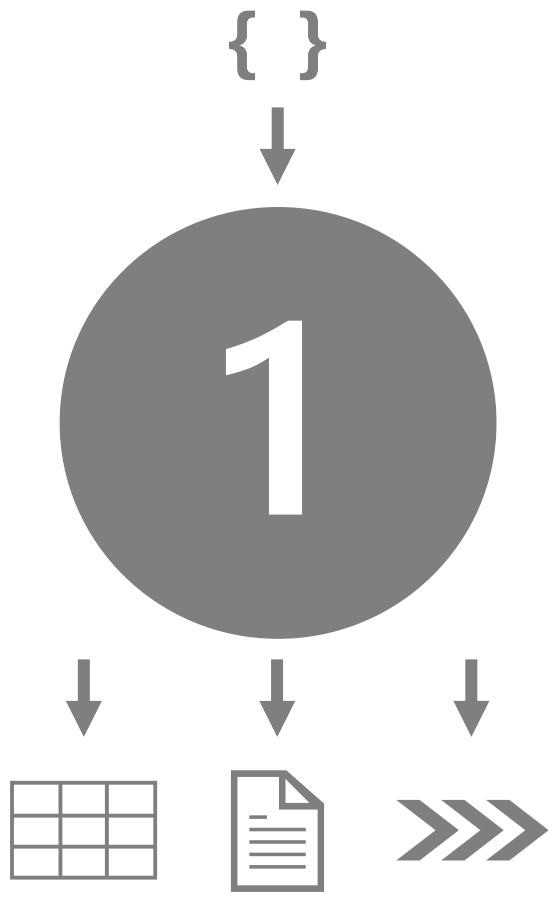
Requirement

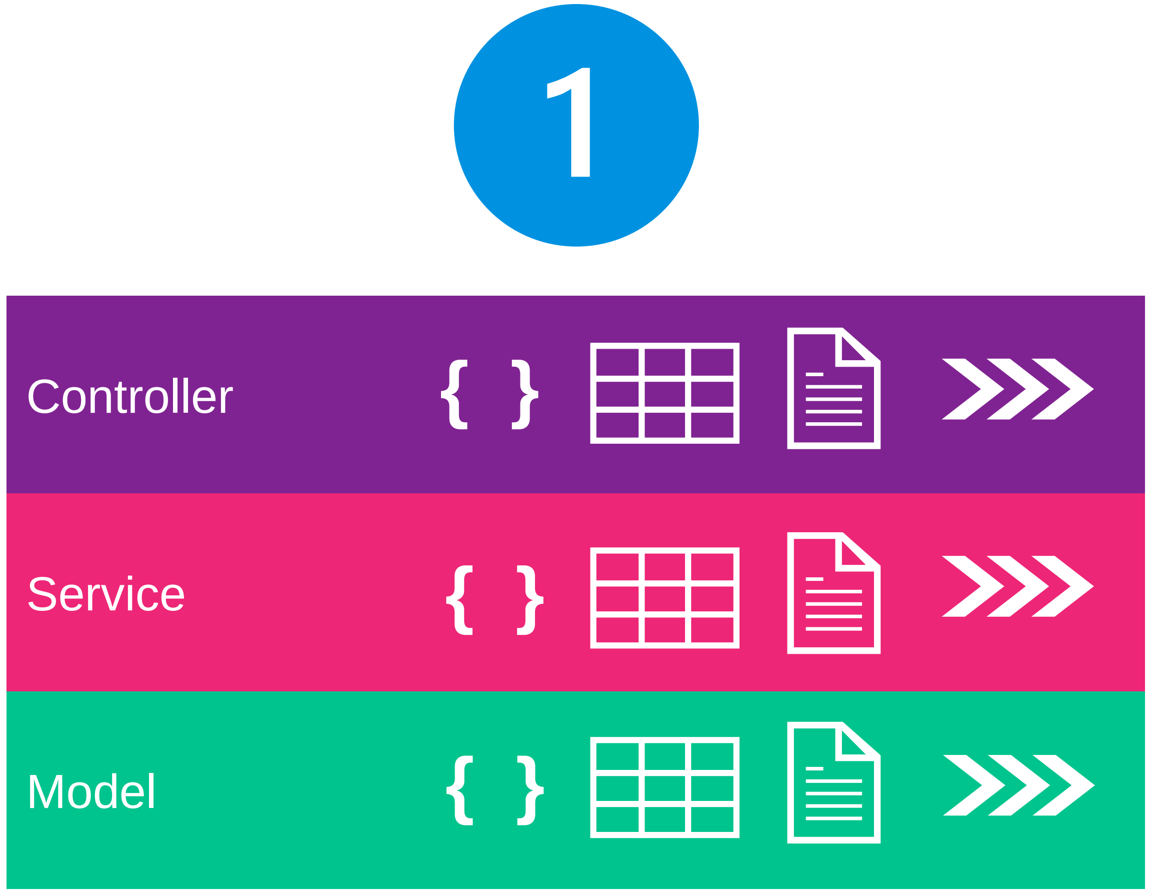


One service

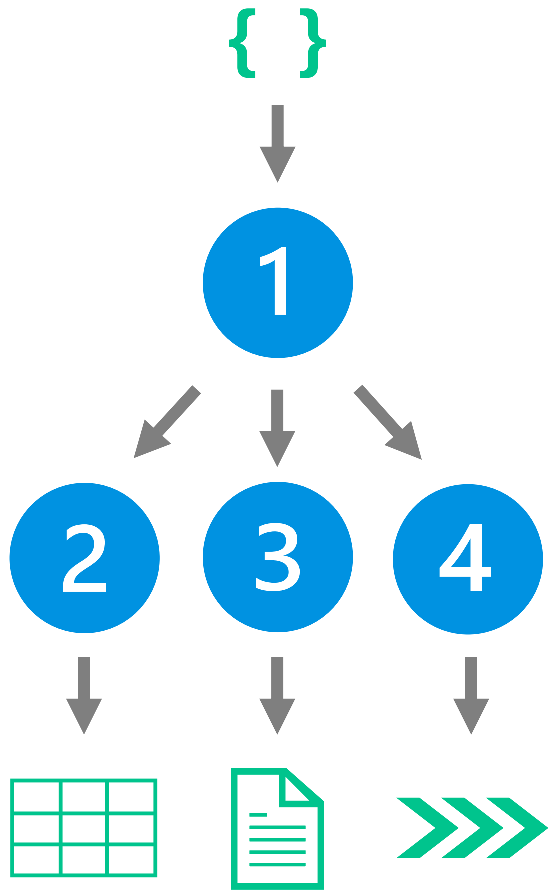


One service

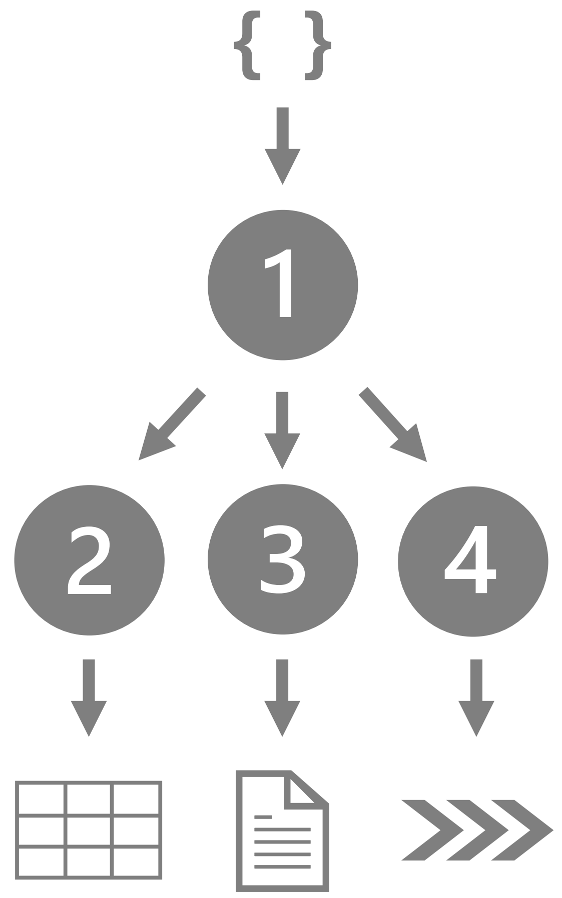


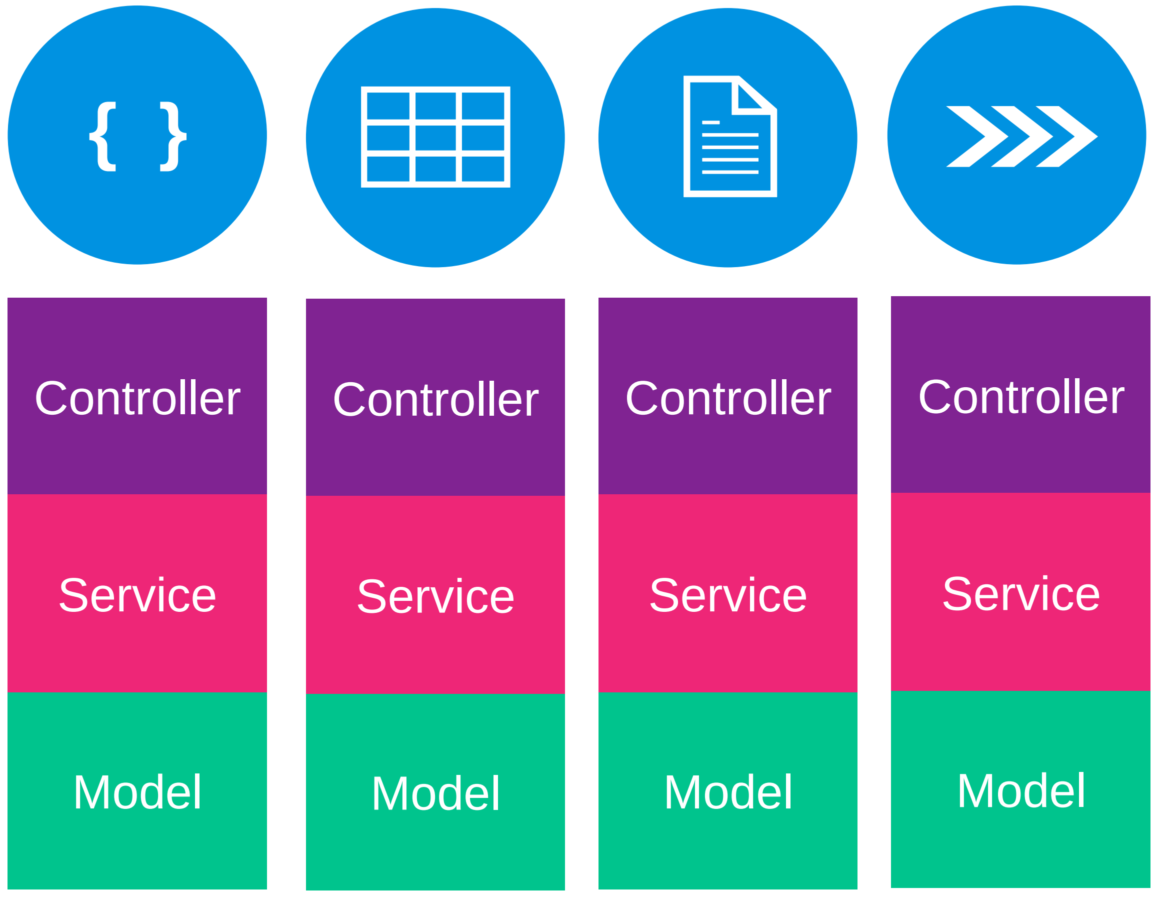


Multiple services

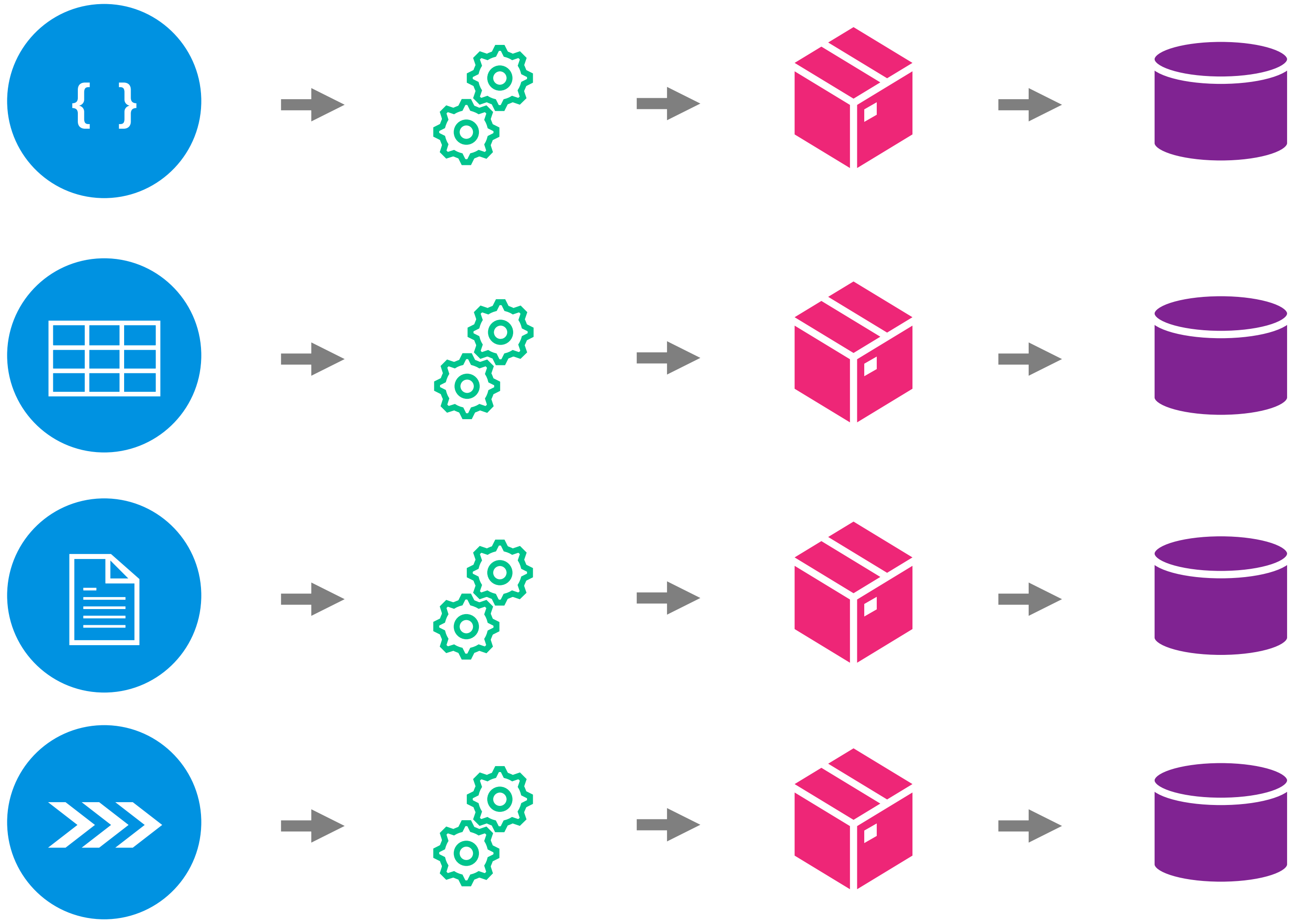


Multiple services

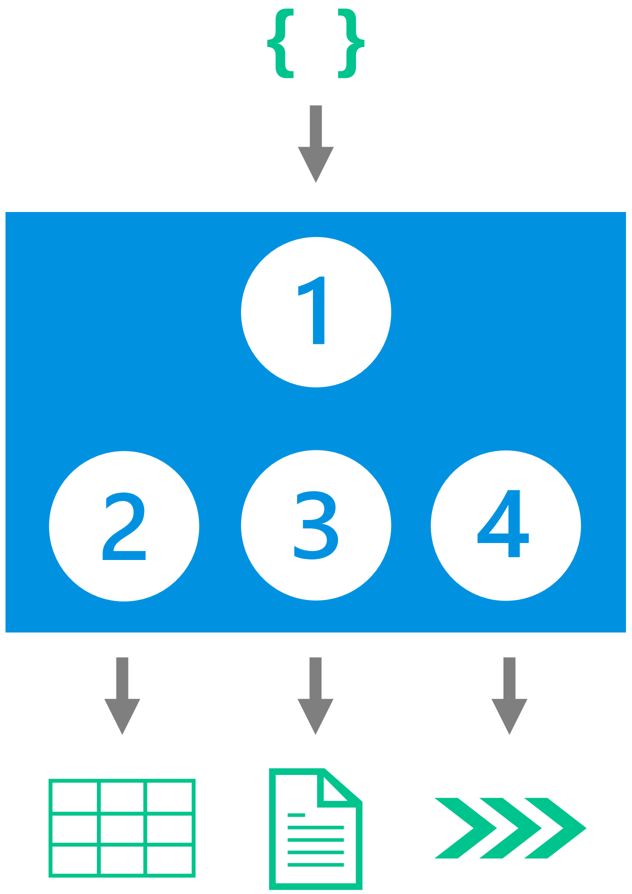




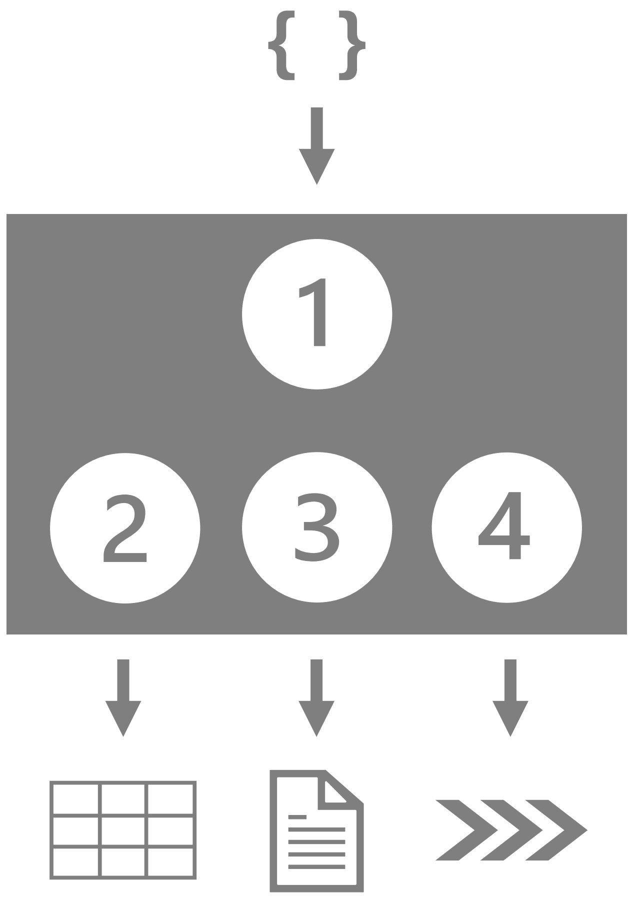
Multiple services

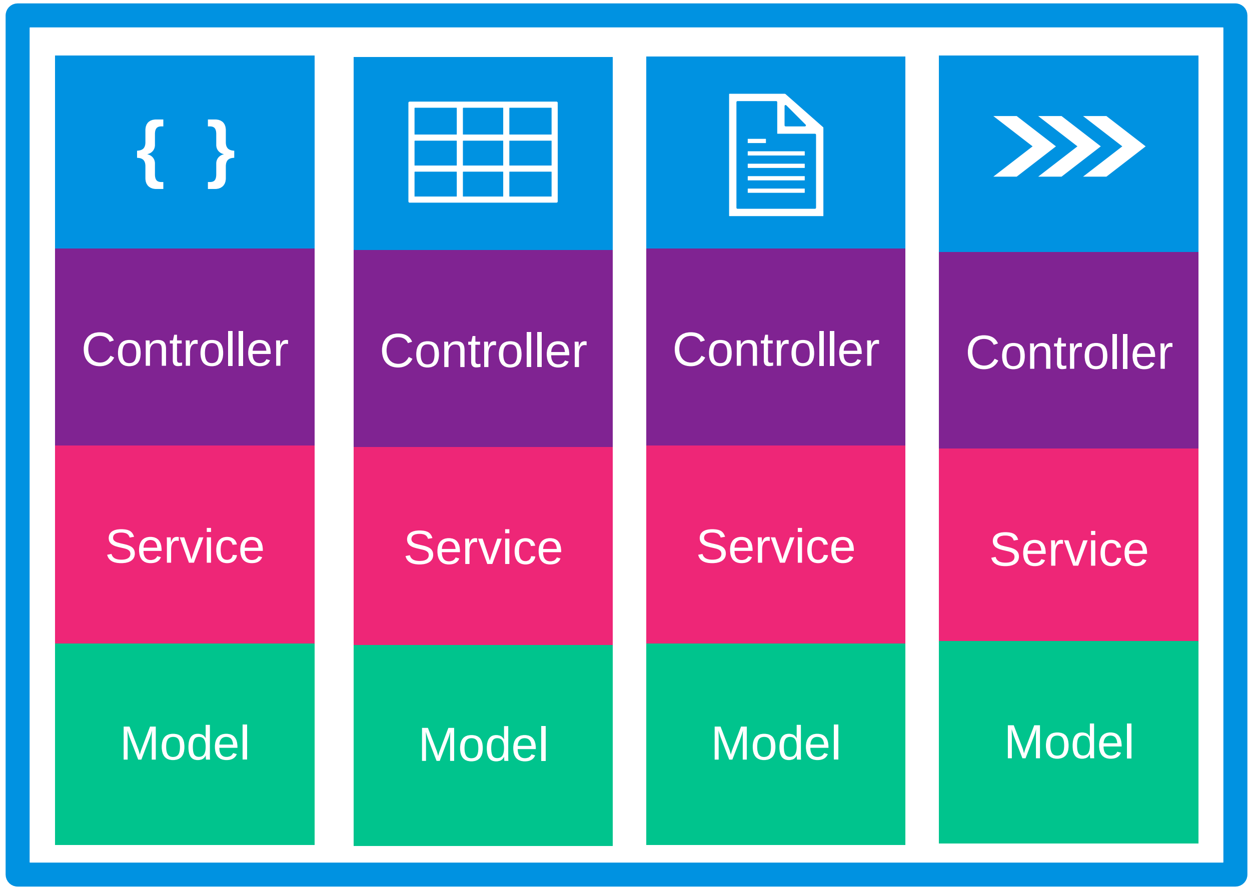


Service with modules



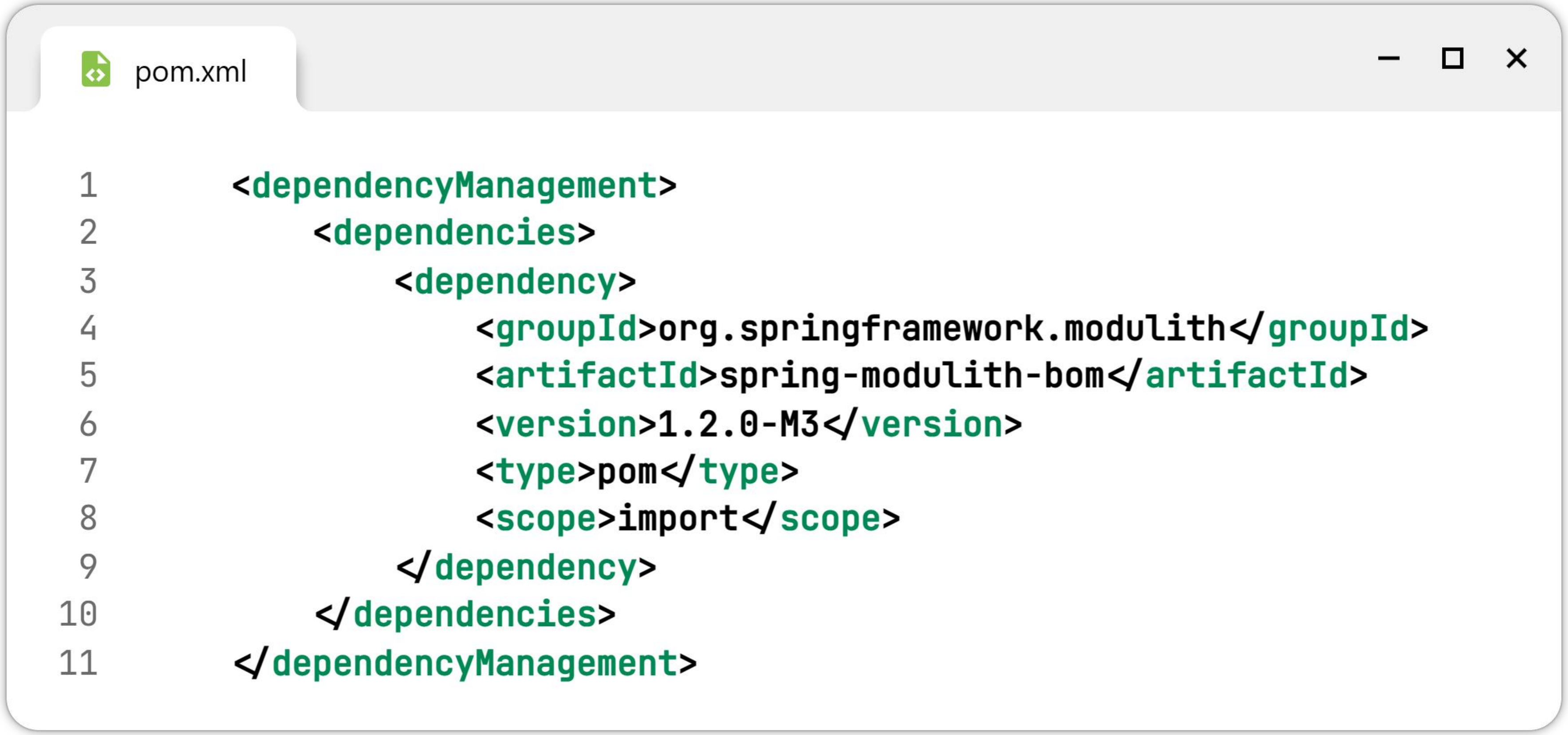
Service with modules





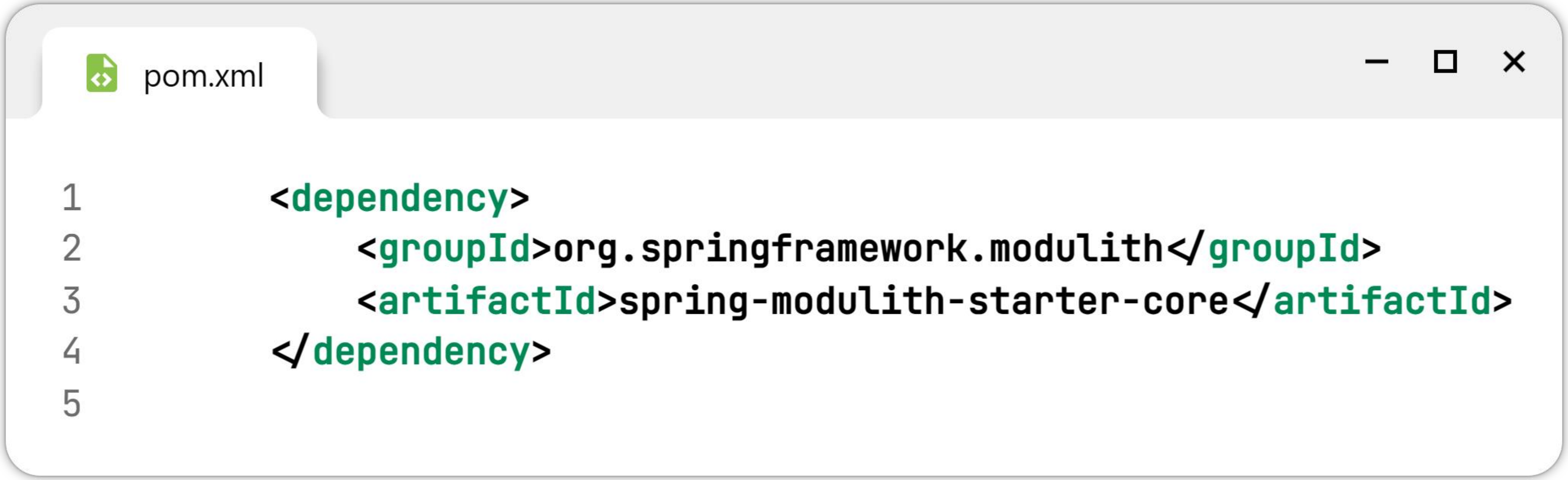
Spring Modulith

Spring Modulith pom



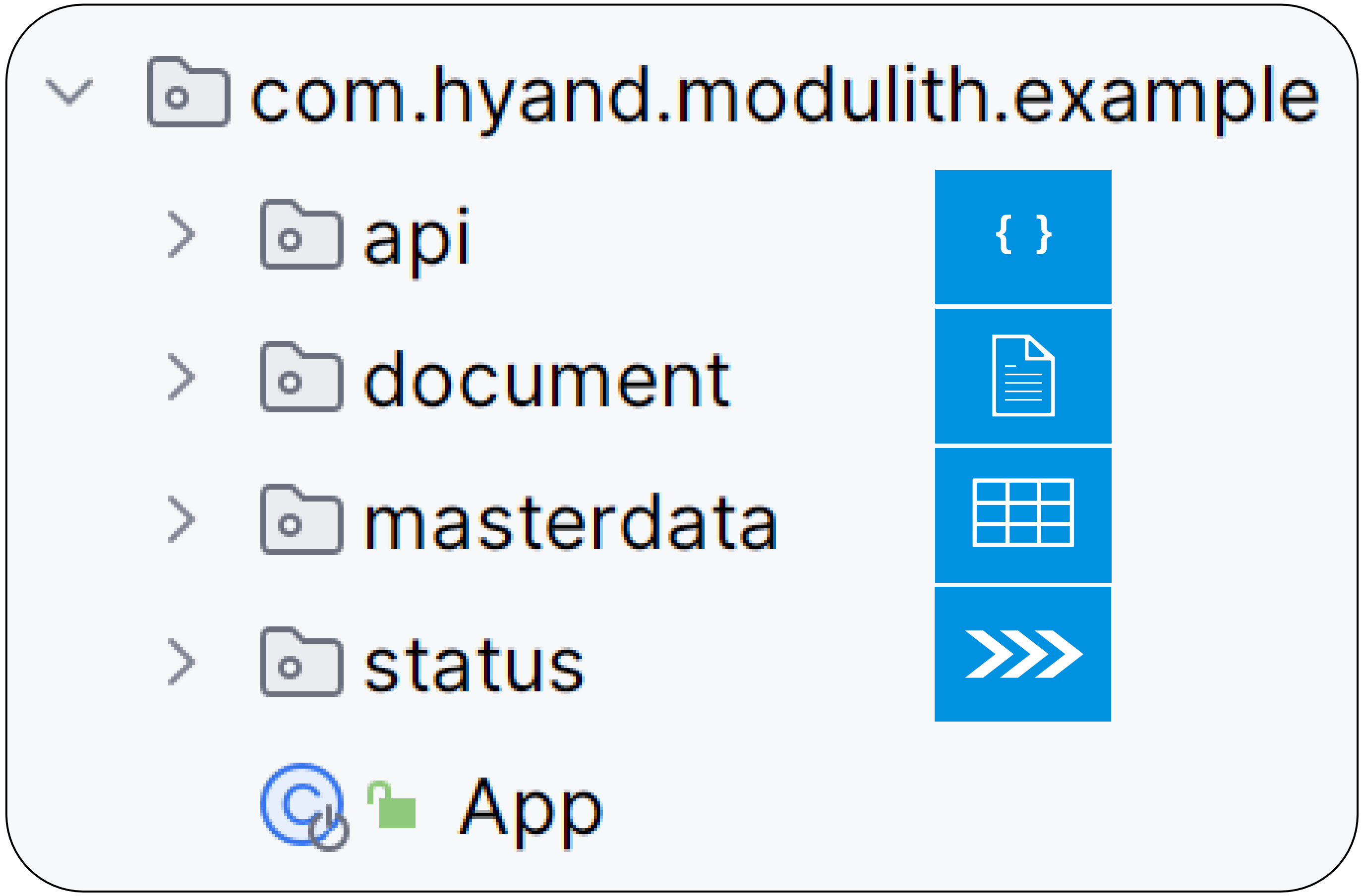
```xml
<dependencyManagement>
    <dependencies>
        <dependency>
            <groupId>org.springframework.modulith</groupId>
            <artifactId>spring-modulith-bom</artifactId>
            <version>1.2.0-M3</version>
            <type>pom</type>
            <scope>import</scope>
        </dependency>
    </dependencies>
</dependencyManagement>
```

Spring Modulith starter core



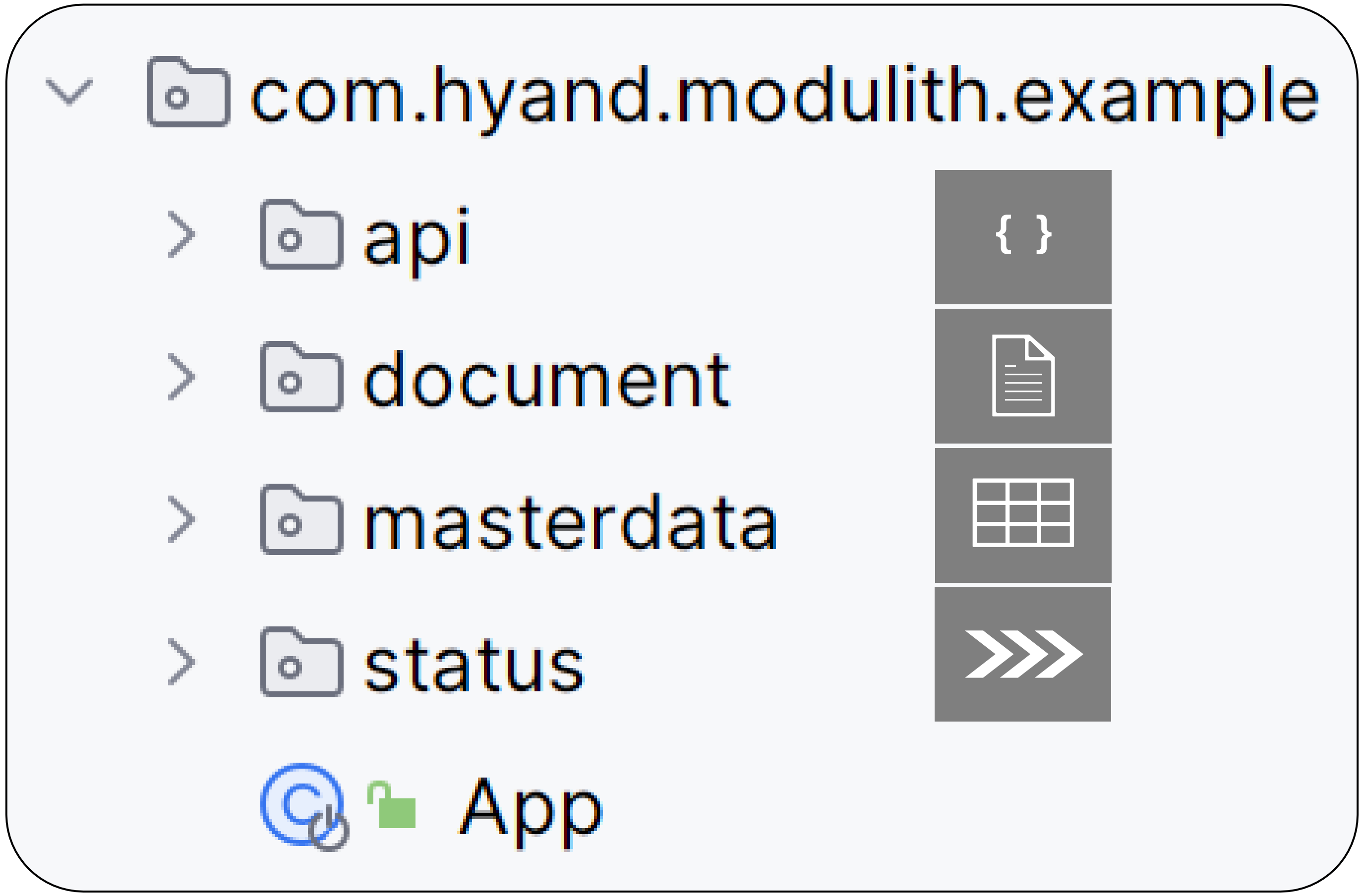
```xml
        <dependency>
            <groupId>org.springframework.modulith</groupId>
            <artifactId>spring-modulith-starter-core</artifactId>
        </dependency>

```

Project structure

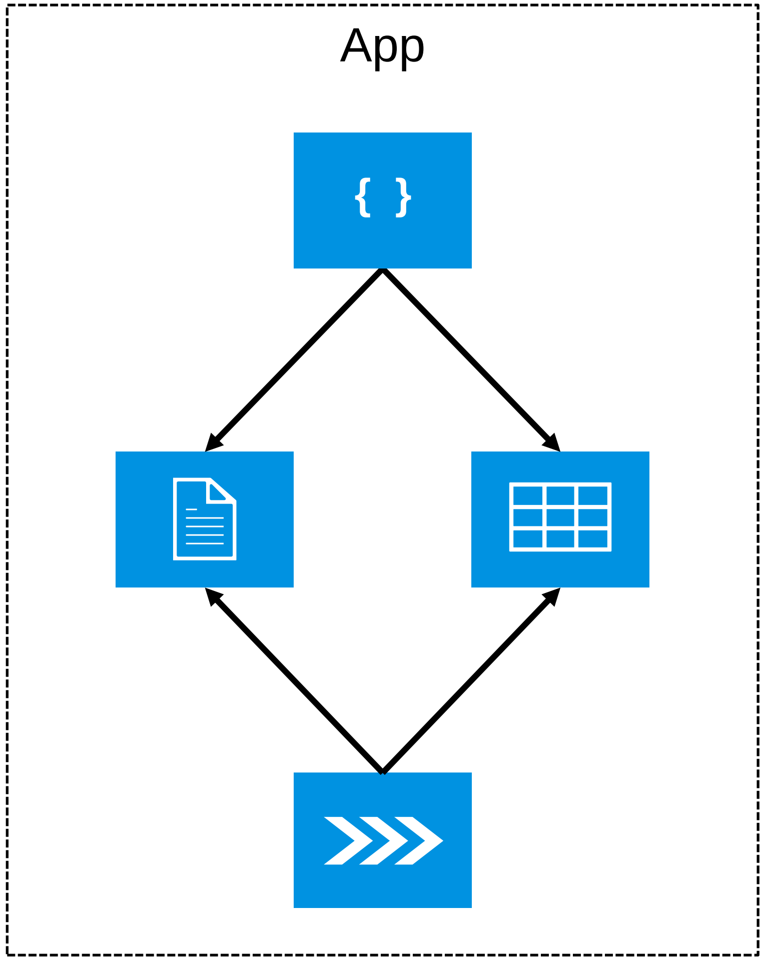


```
com.hyand.modulith.example
    api
    document
    masterdata
    status
    App
```

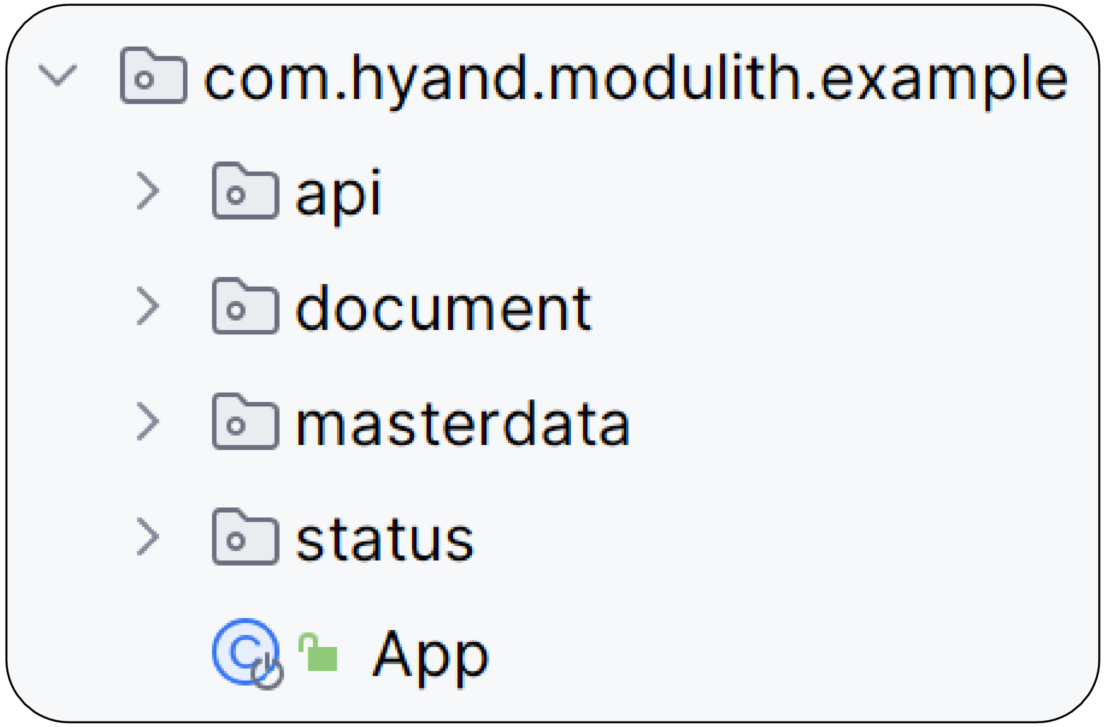
Project structure



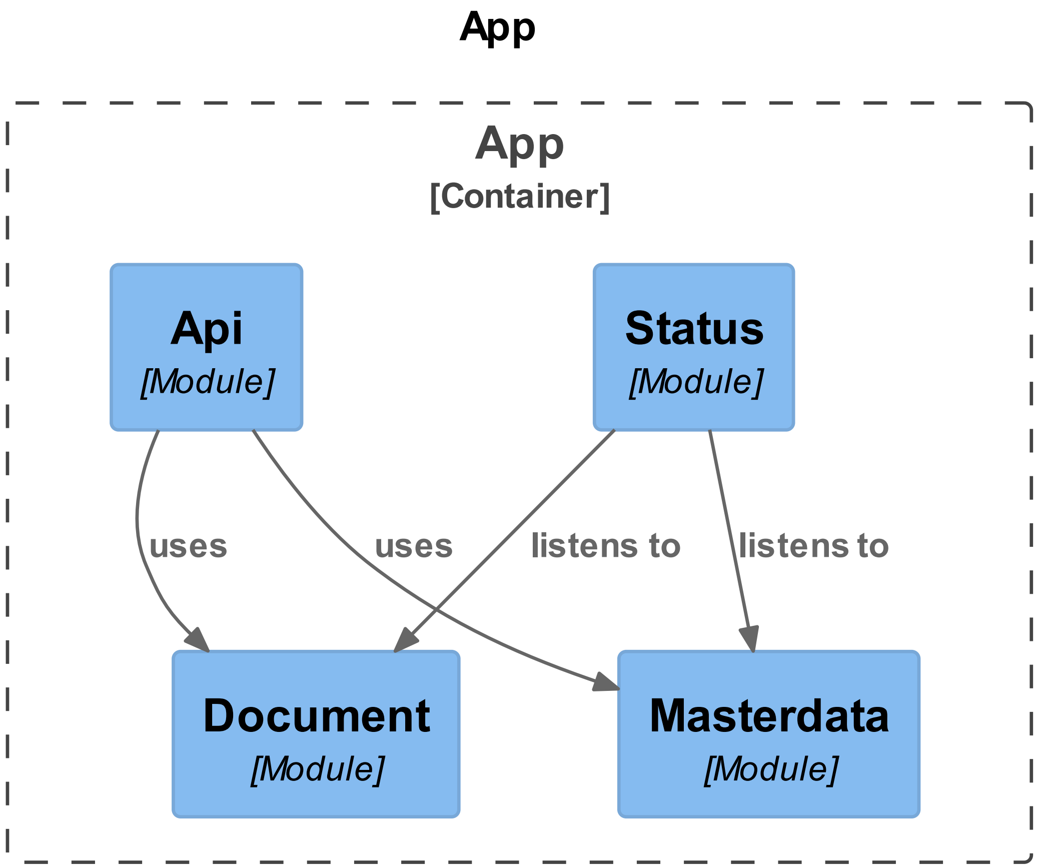
com.hyand.modulith.example
- api
- document
- masterdata
- status
- App



Project structure



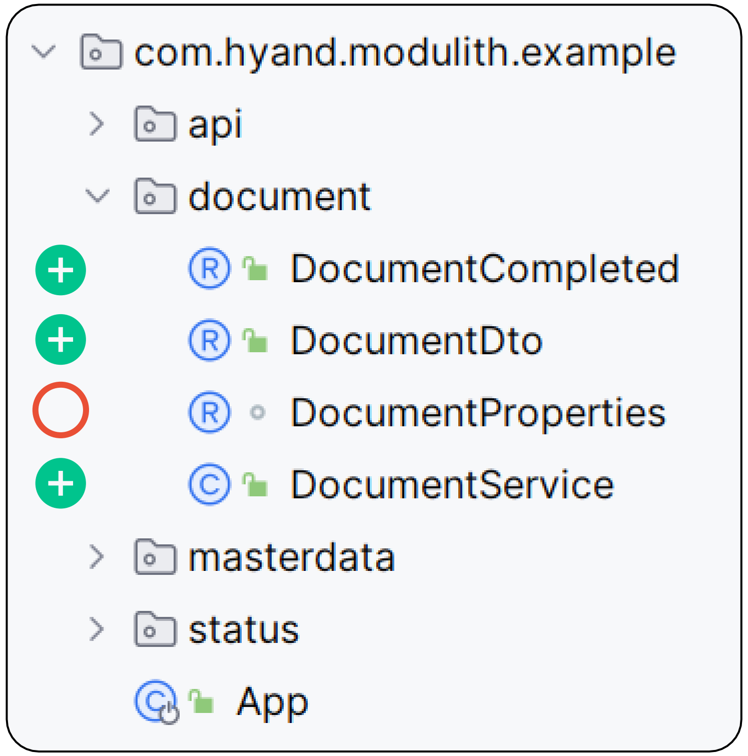
```
com.hyand.modulith.example
    api
    document
    masterdata
    status
    App
```



Legend

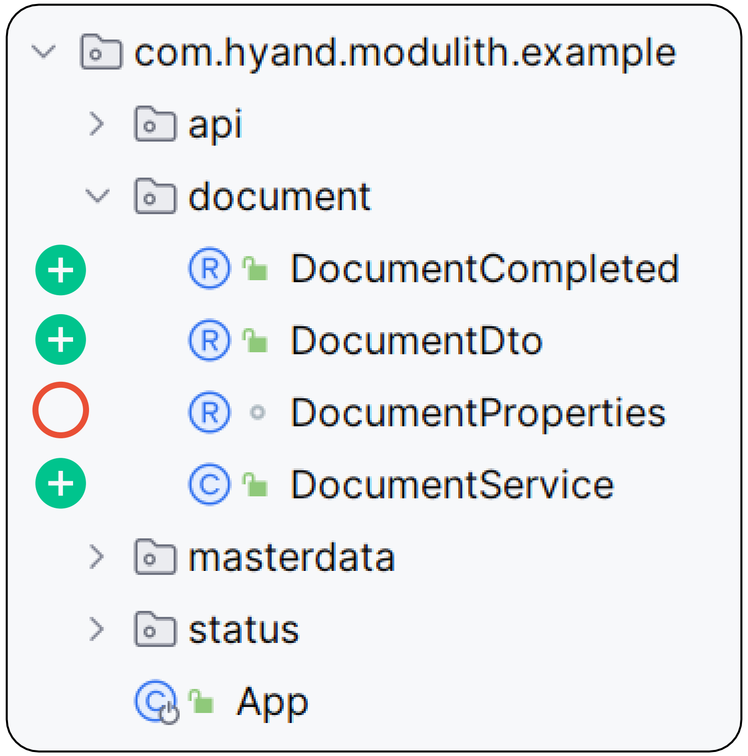
🟦	component
▢	container boundary (dashed)

Simple



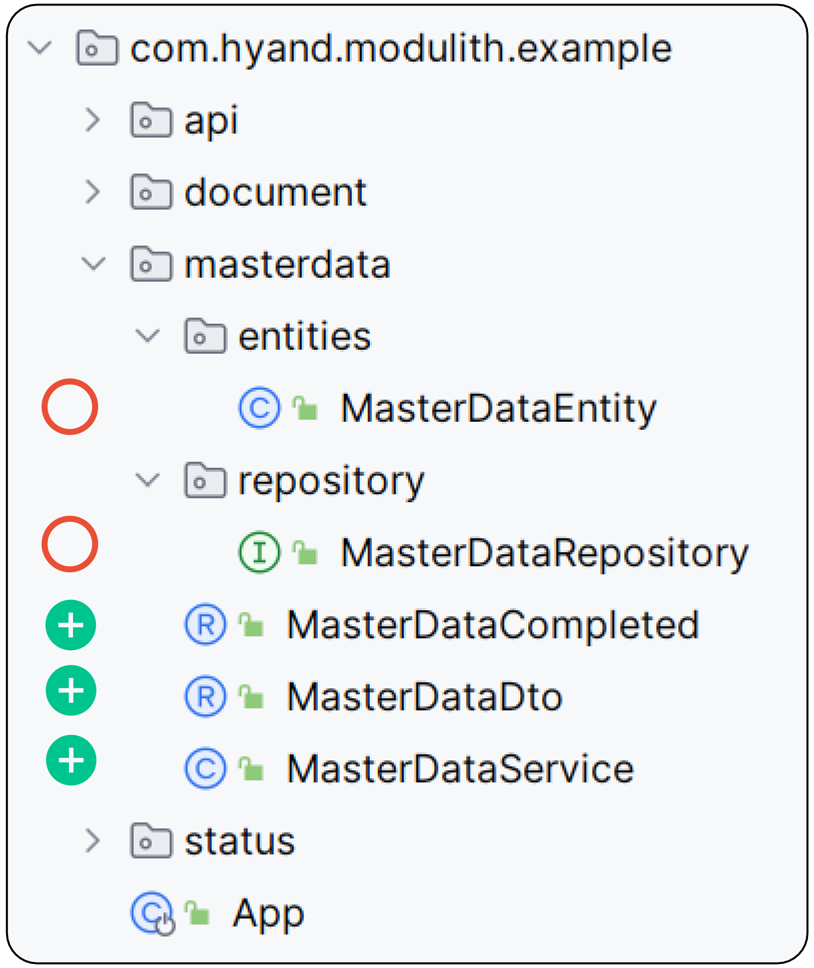
```
com.hyand.modulith.example
  api
  document
    DocumentCompleted
    DocumentDto
    DocumentProperties
    DocumentService
  masterdata
  status
  App
```

Simple



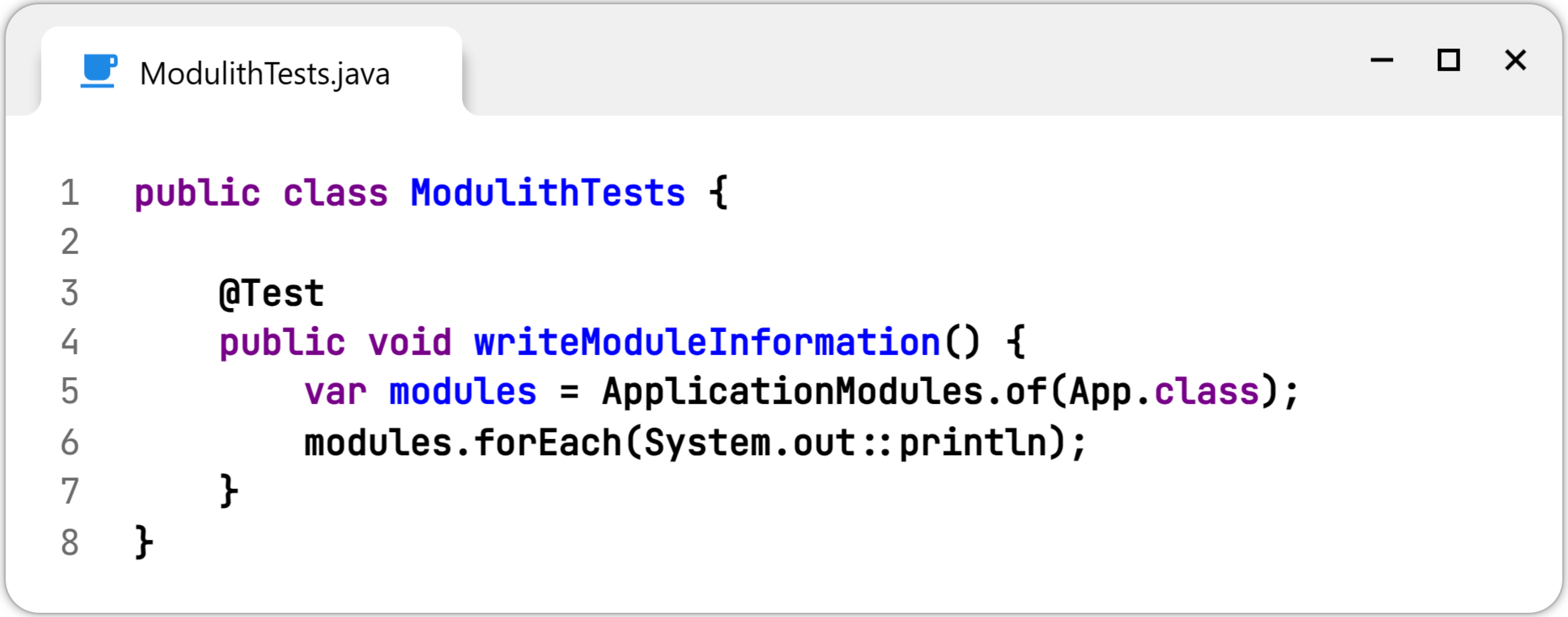
```
⌄ com.hyand.modulith.example
  › api
  ⌄ document
      DocumentCompleted
      DocumentDto
      DocumentProperties
      DocumentService
  › masterdata
  › status
    App
```

Extended



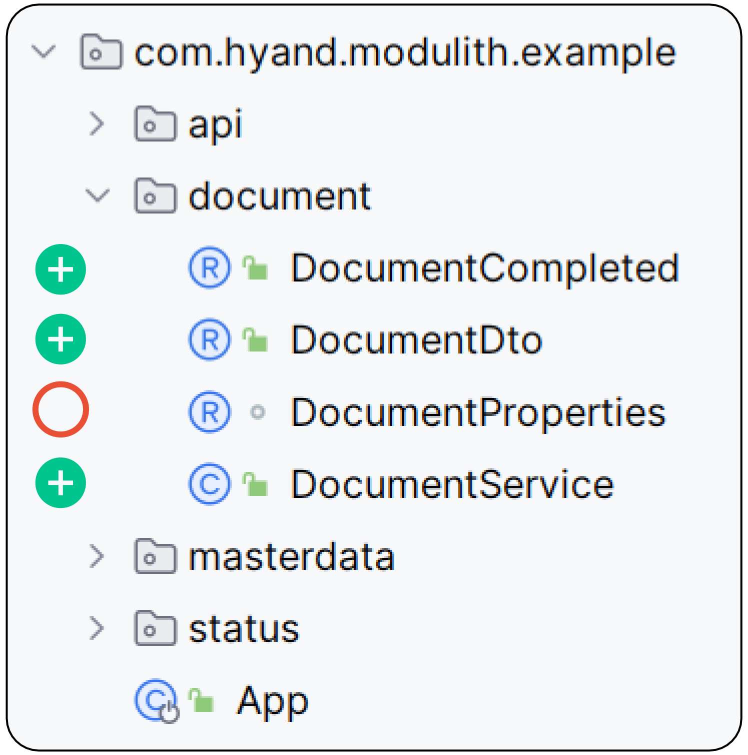
```
⌄ com.hyand.modulith.example
  › api
  › document
  ⌄ masterdata
    ⌄ entities
        MasterDataEntity
    ⌄ repository
        MasterDataRepository
      MasterDataCompleted
      MasterDataDto
      MasterDataService
  › status
    App
```

Type ApplicationModules


ModulithTests.java

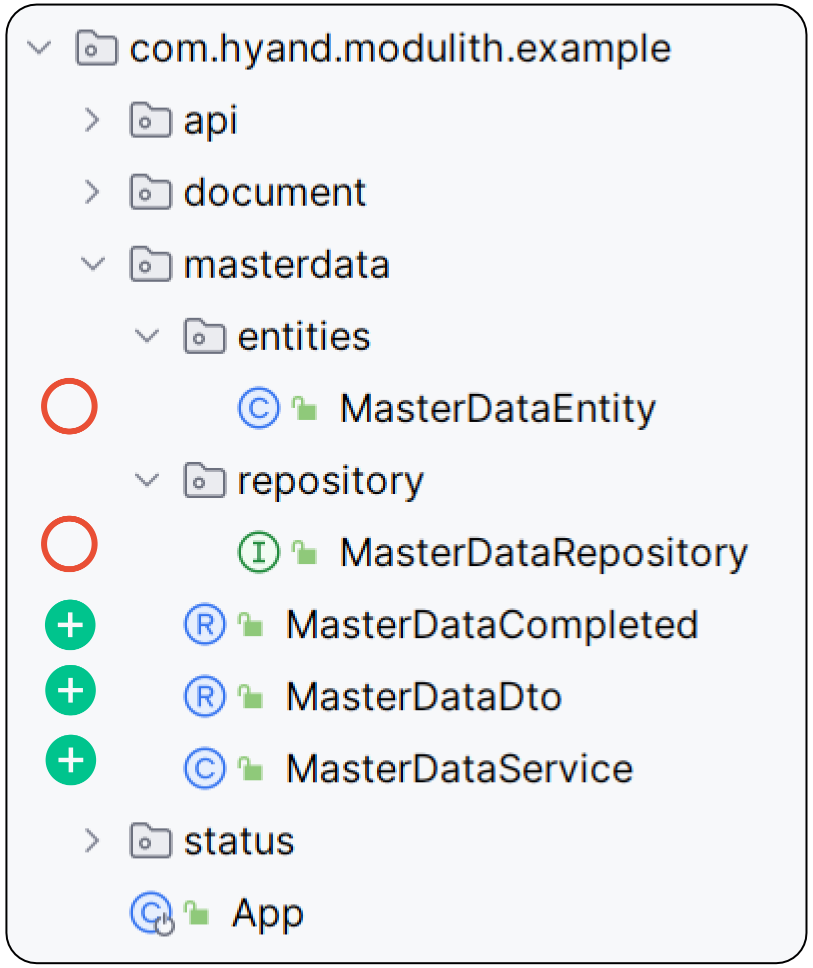
```java
public class ModulithTests {

    @Test
    public void writeModuleInformation() {
        var modules = ApplicationModules.of(App.class);
        modules.forEach(System.out::println);
    }
}
```

Output for module Document



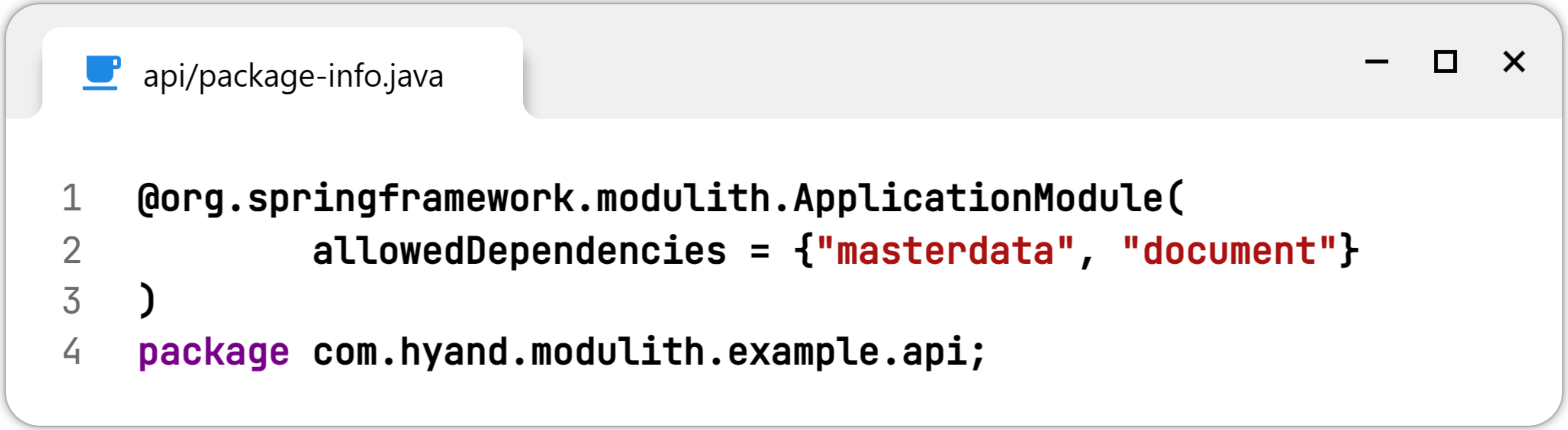
```
# Document
> Logical name: document
> Base package: com.hyand.modulith.example.document
> Spring beans:
  o ….DocumentProperties
  + ….DocumentService
```

Output for module Masterdata


```
com.hyand.modulith.example
  api
  document
  masterdata
    entities
      MasterDataEntity
    repository
      MasterDataRepository
    MasterDataCompleted
    MasterDataDto
    MasterDataService
  status
  App
```

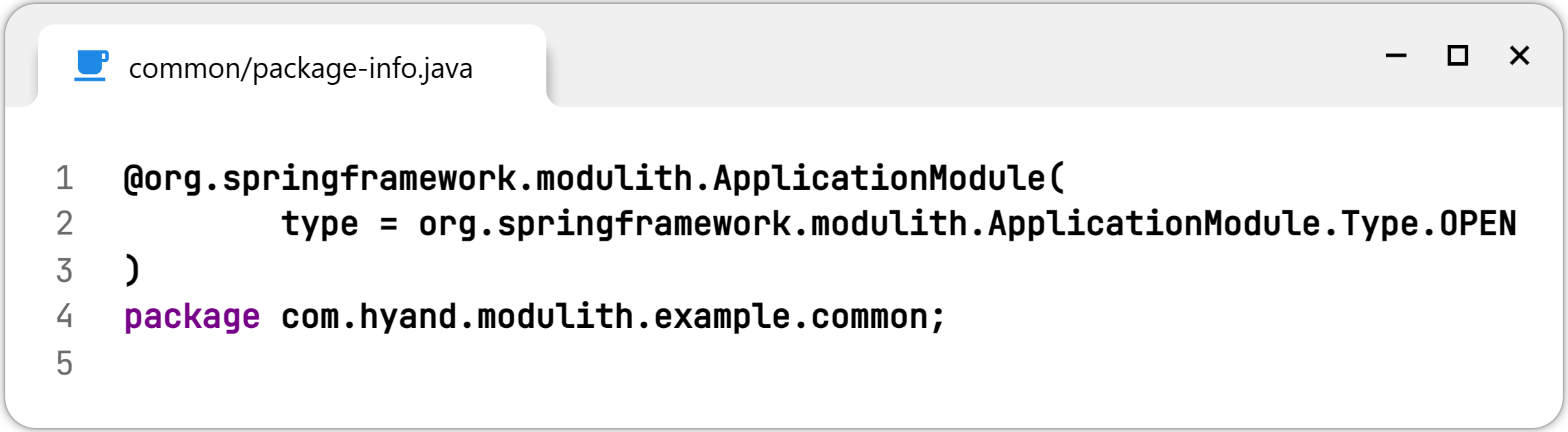
```
# Masterdata
> Logical name: masterdata
> Base package: com.hyand.modulith.example.masterdata
> Spring beans:
  + ….MasterDataService
  o ….repository.MasterDataRepository
```

Explicit Dependencies (optional)



```
@org.springframework.modulith.ApplicationModule(
        allowedDependencies = {"masterdata", "document"}
)
package com.hyand.modulith.example.api;
```

Application modules
Type OPEN



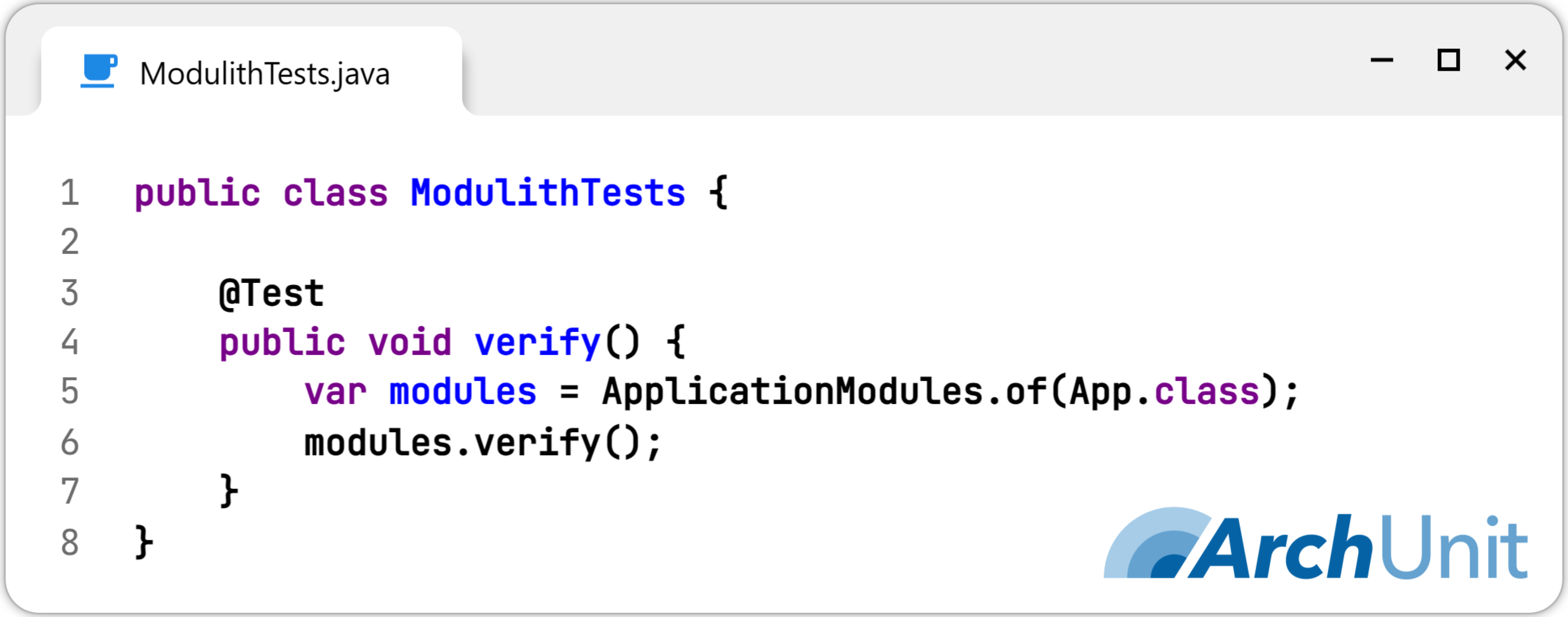
```java
@org.springframework.modulith.ApplicationModule(
        type = org.springframework.modulith.ApplicationModule.Type.OPEN
)
package com.hyand.modulith.example.common;

```

Spring Modulith 1.2 required

Verification

Verification

Unit test


ModulithTests.java

```java
public class ModulithTests {

    @Test
    public void verify() {
        var modules = ApplicationModules.of(App.class);
        modules.verify();
    }
}
```

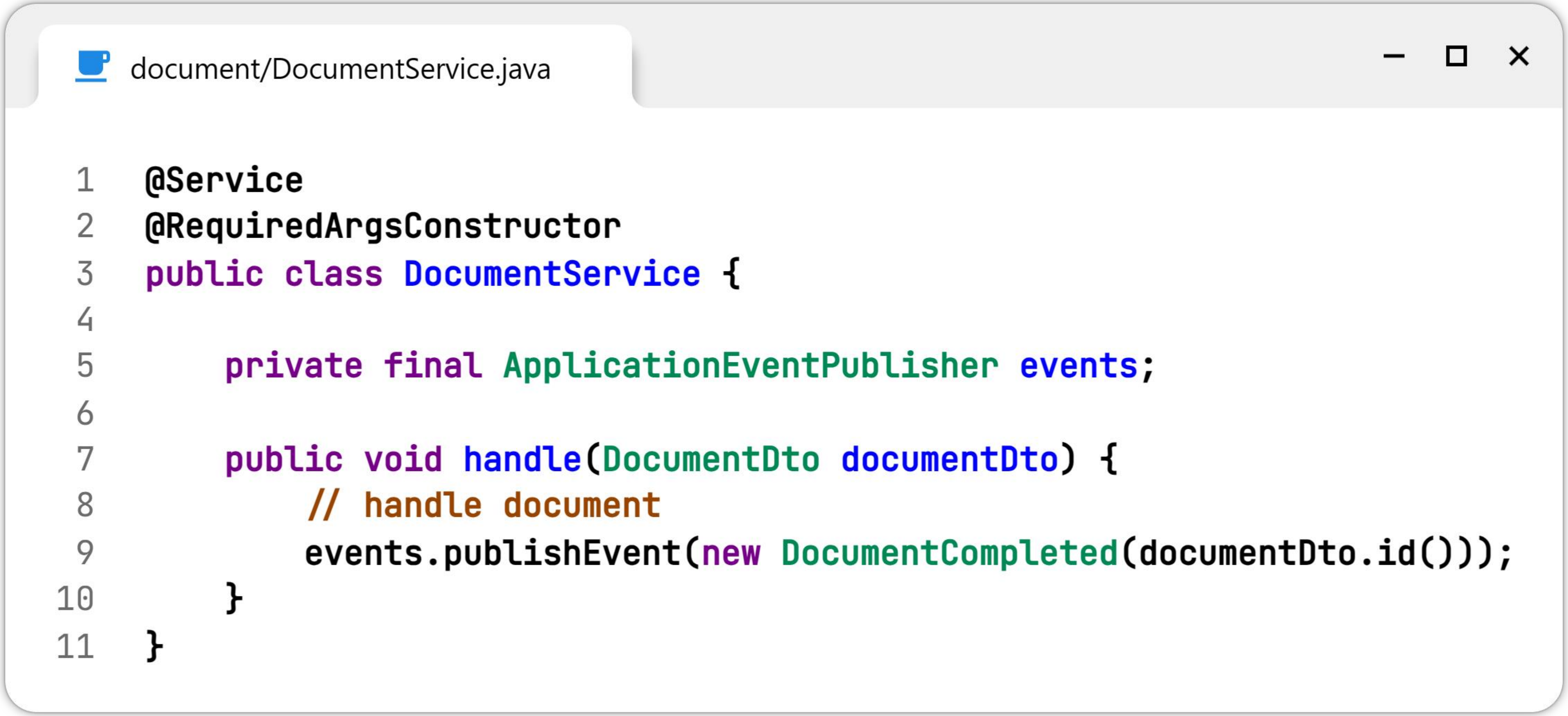
Result of a failed unit test

```
org.springframework.modulith.core.Violations: - Cycle detected: Slice api ->
                Slice document ->
                Slice api
  1. Dependencies of Slice api
    - Constructor <com.hyand.modulith.example.api.Controller.<init>(com.hyand.modulith.
    - Field <com.hyand.modulith.example.api.Controller.documentService> has type <com.h
    - Method <com.hyand.modulith.example.api.Controller.mapDocument(com.hyand.modulith.
    - Method <com.hyand.modulith.example.api.Controller.receive(com.hyand.modulith.exam
    - Method <com.hyand.modulith.example.api.Controller.mapDocument(com.hyand.modulith.
  2. Dependencies of Slice document
    - Constructor <com.hyand.modulith.example.document.DocumentService.<init>(org.sprin
    - Field <com.hyand.modulith.example.document.DocumentService.controller> has type <
```

Hyand

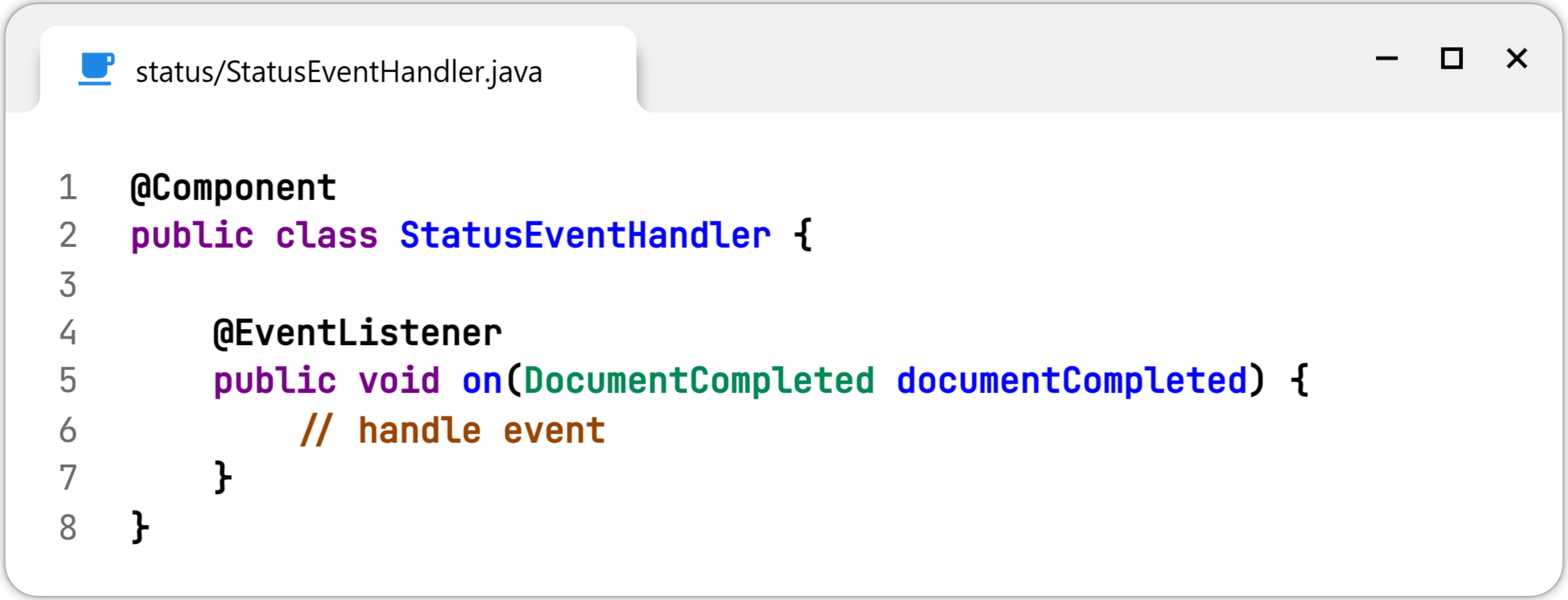
Events

Publish events



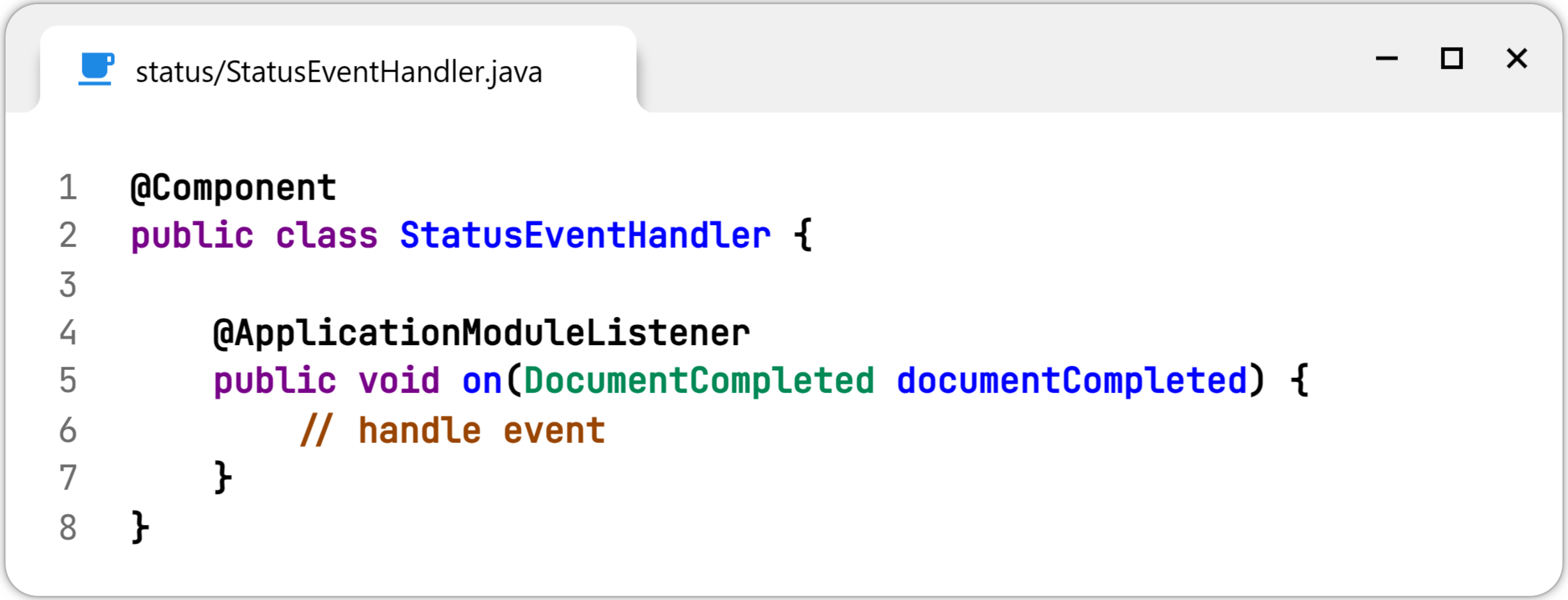
```java
@Service
@RequiredArgsConstructor
public class DocumentService {

    private final ApplicationEventPublisher events;

    public void handle(DocumentDto documentDto) {
        // handle document
        events.publishEvent(new DocumentCompleted(documentDto.id()));
    }
}
```

Spring: EventListener


status/StatusEventHandler.java

```java
@Component
public class StatusEventHandler {

    @EventListener
    public void on(DocumentCompleted documentCompleted) {
        // handle event
    }
}
```

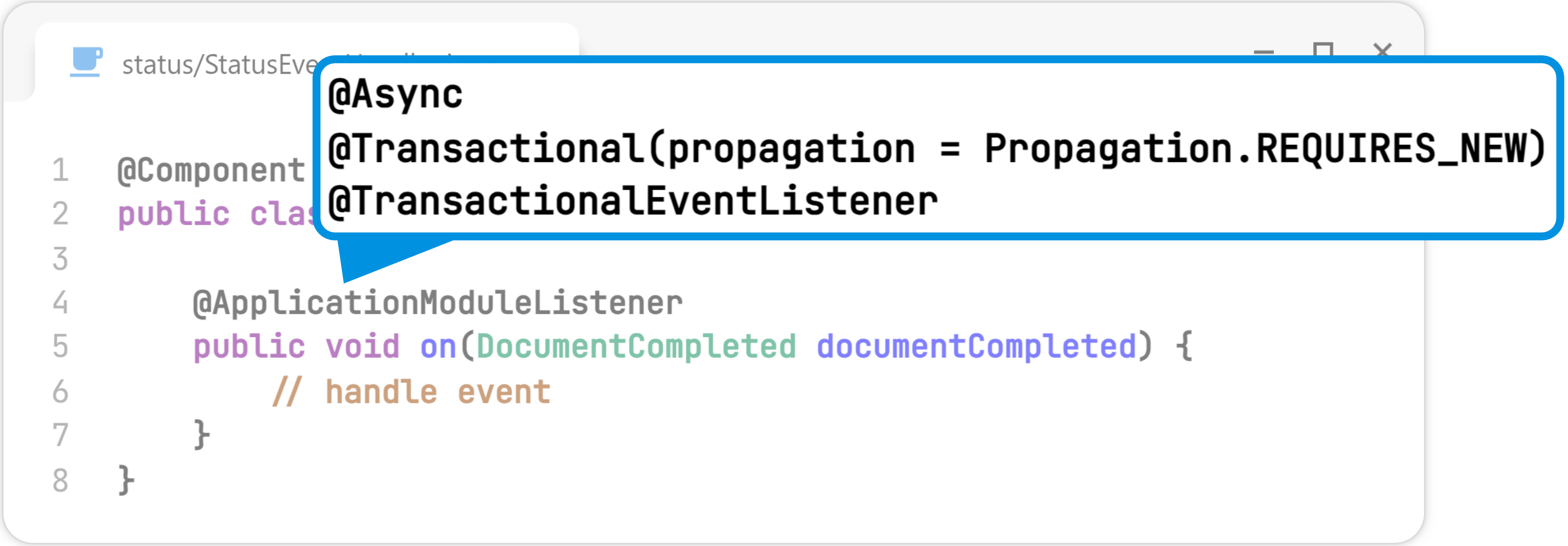
Modulith: ApplicationModuleListener



status/StatusEventHandler.java

```java
@Component
public class StatusEventHandler {

    @ApplicationModuleListener
    public void on(DocumentCompleted documentCompleted) {
        // handle event
    }
}
```

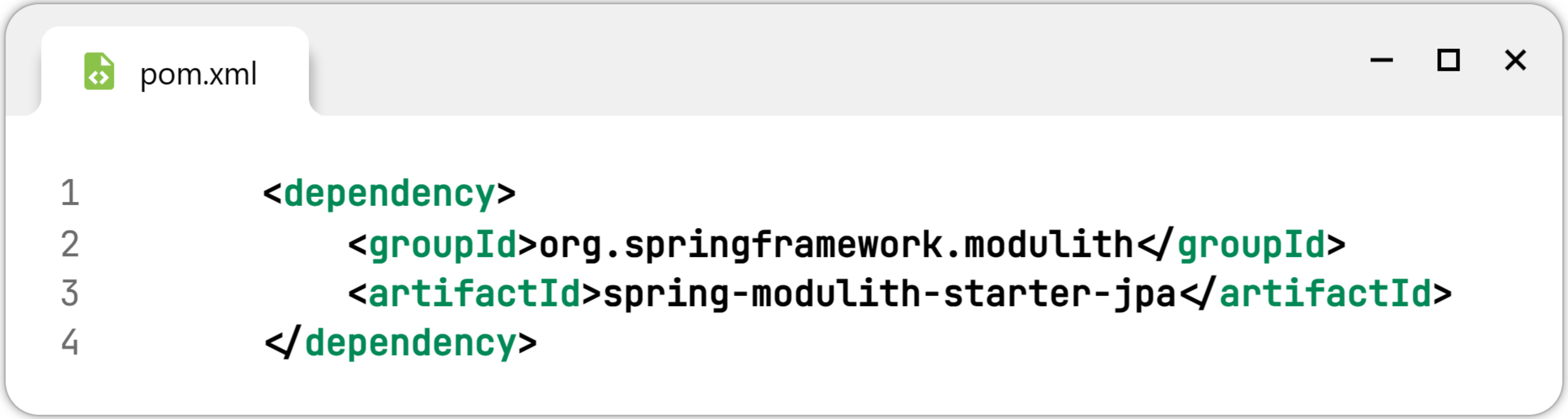
Modulith: ApplicationModuleListener


status/StatusEve

```
@Component
public clas

    @ApplicationModuleListener
    public void on(DocumentCompleted documentCompleted) {
        // handle event
    }
}
```

```
@Async
@Transactional(propagation = Propagation.REQUIRES_NEW)
@TransactionalEventListener
```

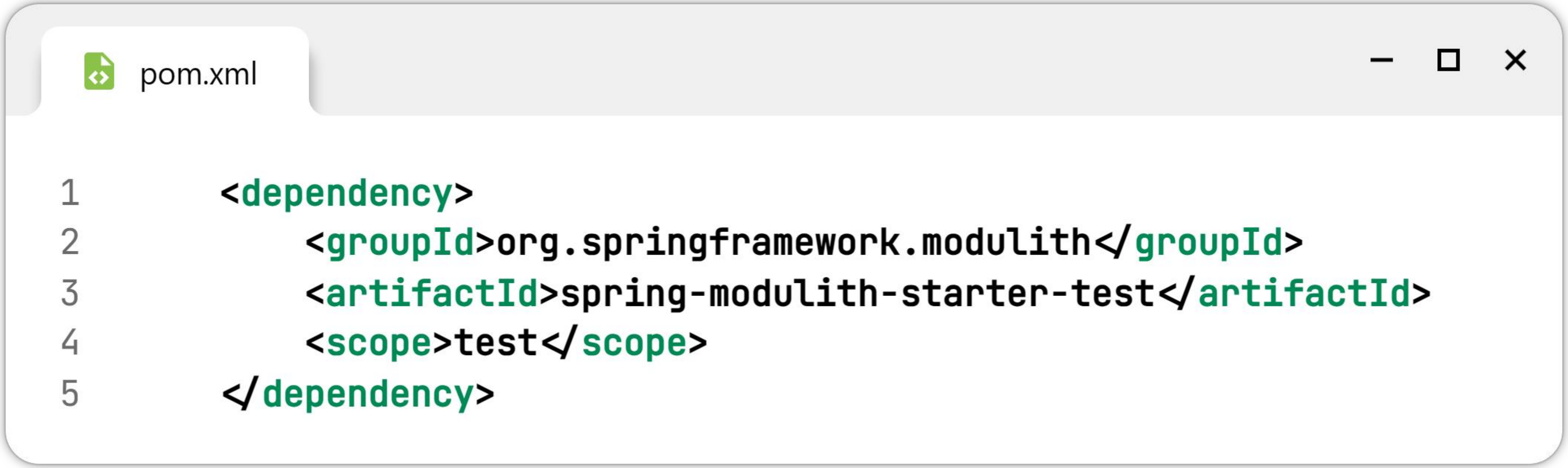
Spring Modulith starter JPA


pom.xml

```xml
        <dependency>
            <groupId>org.springframework.modulith</groupId>
            <artifactId>spring-modulith-starter-jpa</artifactId>
        </dependency>
```

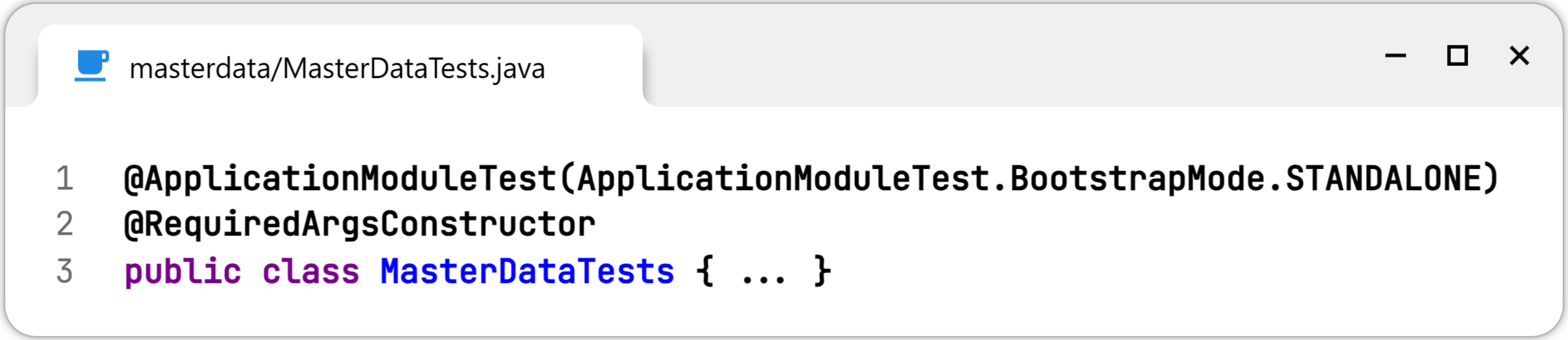
Integration testing

Spring Modulith starter test



```xml
<dependency>
    <groupId>org.springframework.modulith</groupId>
    <artifactId>spring-modulith-starter-test</artifactId>
    <scope>test</scope>
</dependency>
```

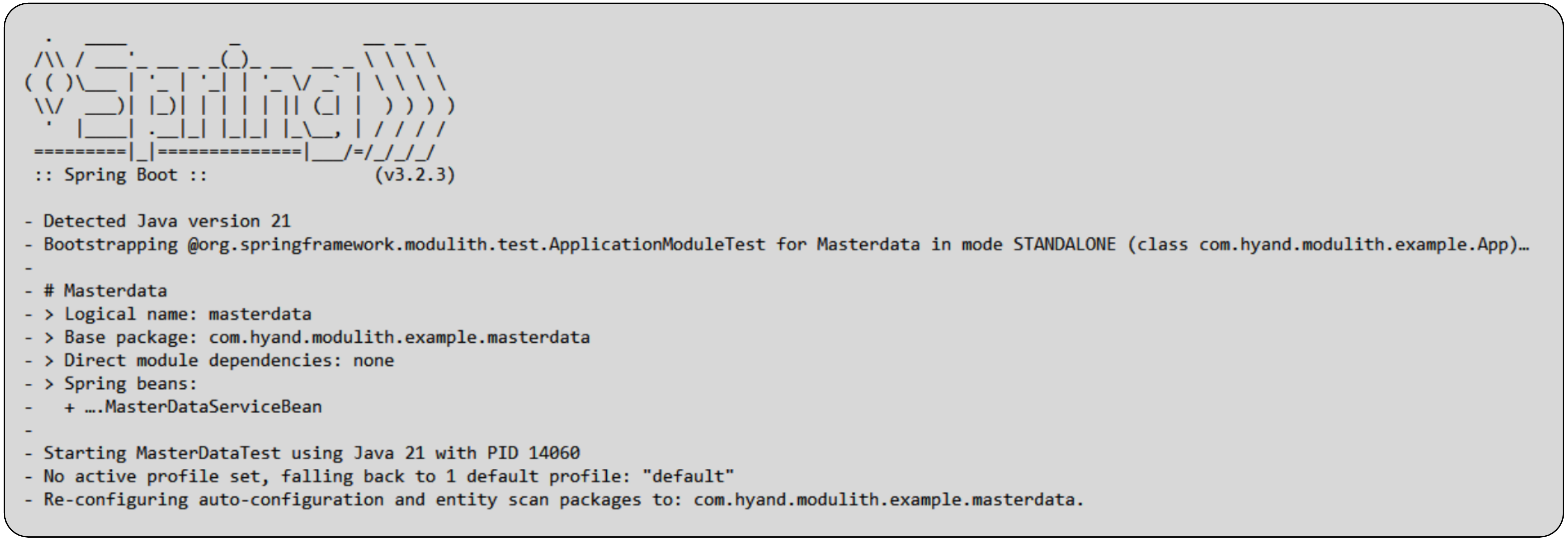
Bootstrap


masterdata/MasterDataTests.java

```java
@ApplicationModuleTest(ApplicationModuleTest.BootstrapMode.STANDALONE)
@RequiredArgsConstructor
public class MasterDataTests { ... }
```

- STANDALONE (default)
- DIRECT_DEPENDENCIES
- ALL_DEPENDENCIES

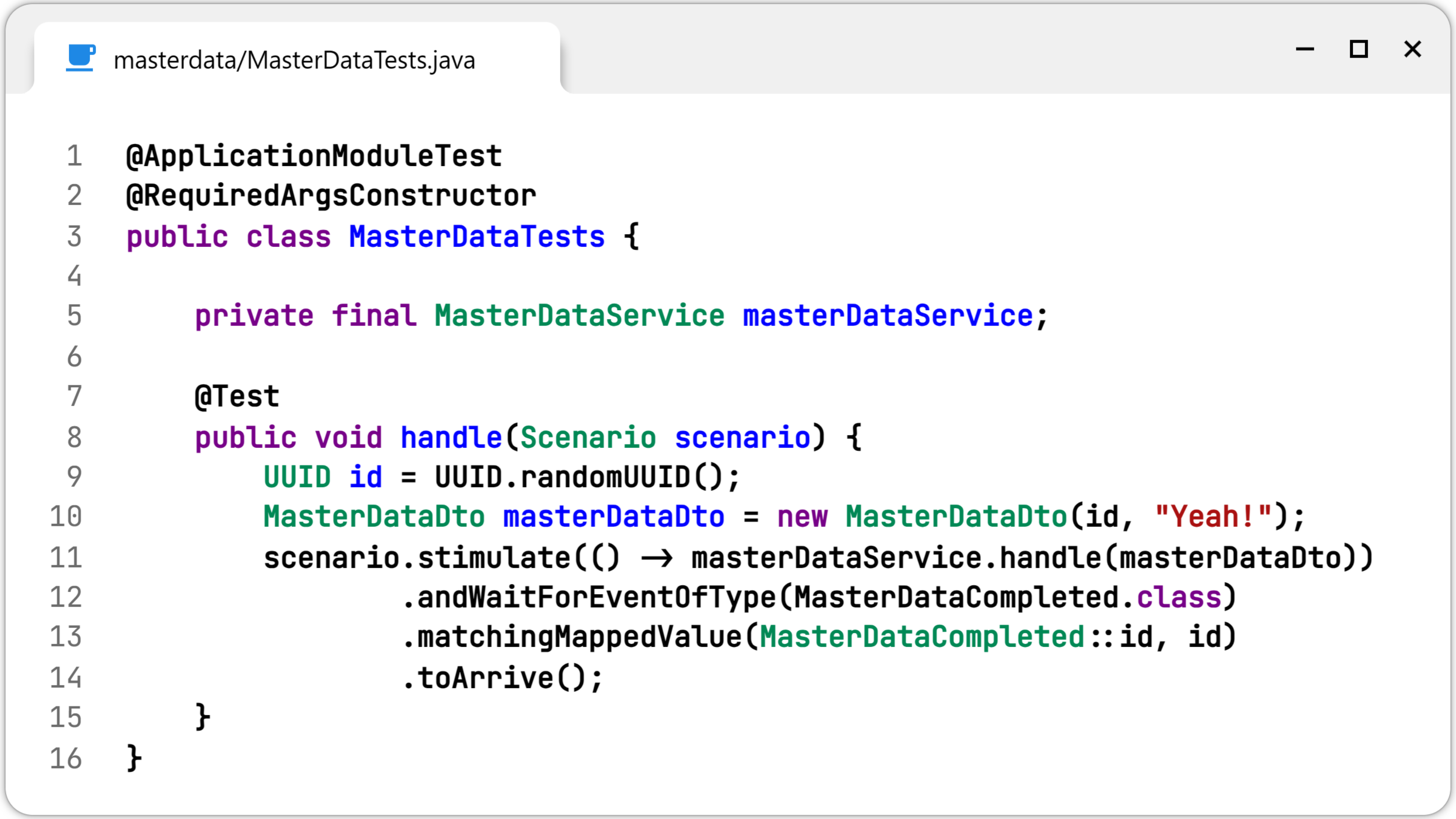
Output of bootstrapping: Module masterdata



```
  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::                (v3.2.3)

- Detected Java version 21
- Bootstrapping @org.springframework.modulith.test.ApplicationModuleTest for Masterdata in mode STANDALONE (class com.hyand.modulith.example.App)…
-
- # Masterdata
- > Logical name: masterdata
- > Base package: com.hyand.modulith.example.masterdata
- > Direct module dependencies: none
- > Spring beans:
-   + ….MasterDataServiceBean
-
- Starting MasterDataTest using Java 21 with PID 14060
- No active profile set, falling back to 1 default profile: "default"
- Re-configuring auto-configuration and entity scan packages to: com.hyand.modulith.example.masterdata.
```

Hyand

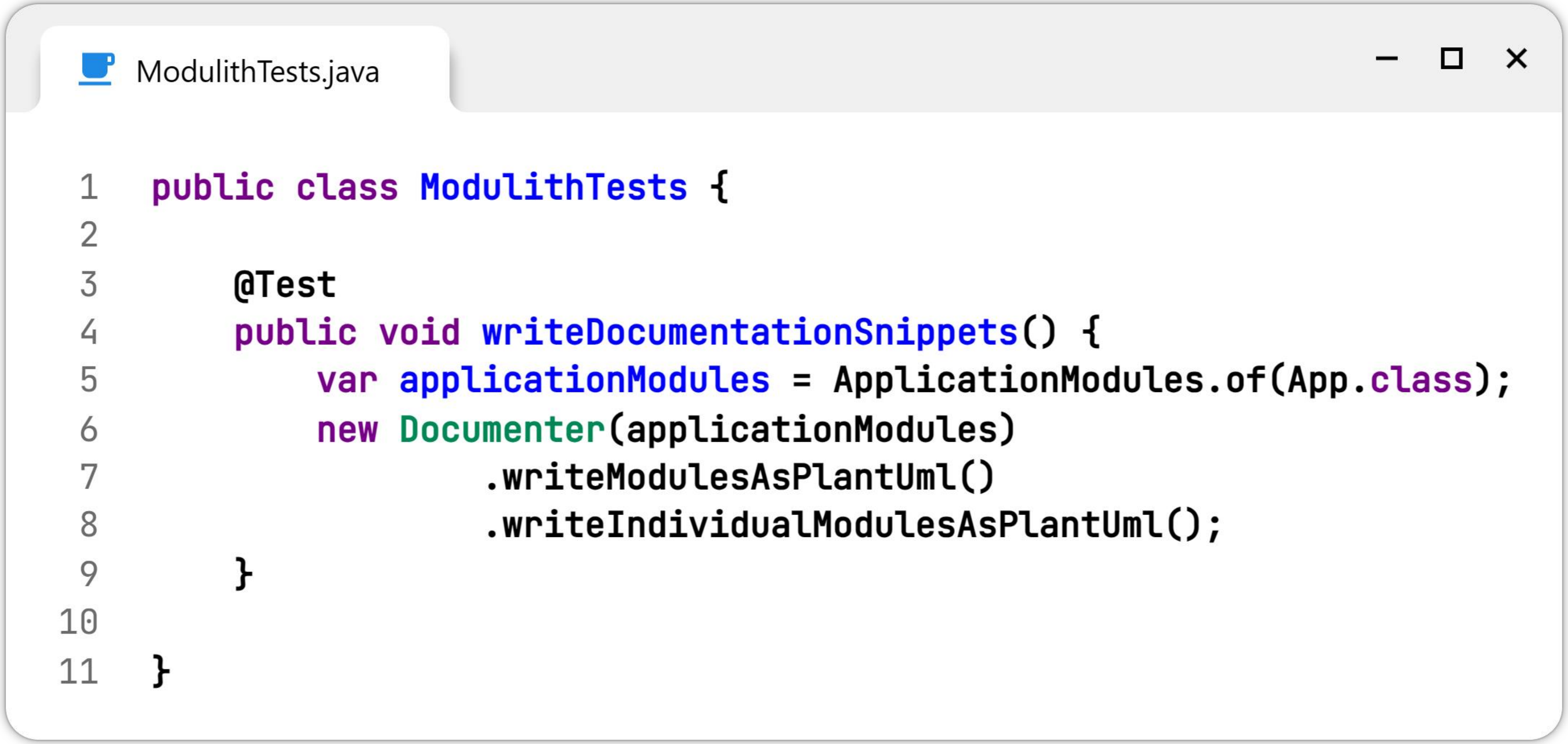
Scenario


masterdata/MasterDataTests.java

```java
@ApplicationModuleTest
@RequiredArgsConstructor
public class MasterDataTests {

    private final MasterDataService masterDataService;

    @Test
    public void handle(Scenario scenario) {
        UUID id = UUID.randomUUID();
        MasterDataDto masterDataDto = new MasterDataDto(id, "Yeah!");
        scenario.stimulate(() -> masterDataService.handle(masterDataDto))
                .andWaitForEventOfType(MasterDataCompleted.class)
                .matchingMappedValue(MasterDataCompleted::id, id)
                .toArrive();
    }
}
```

Documentation

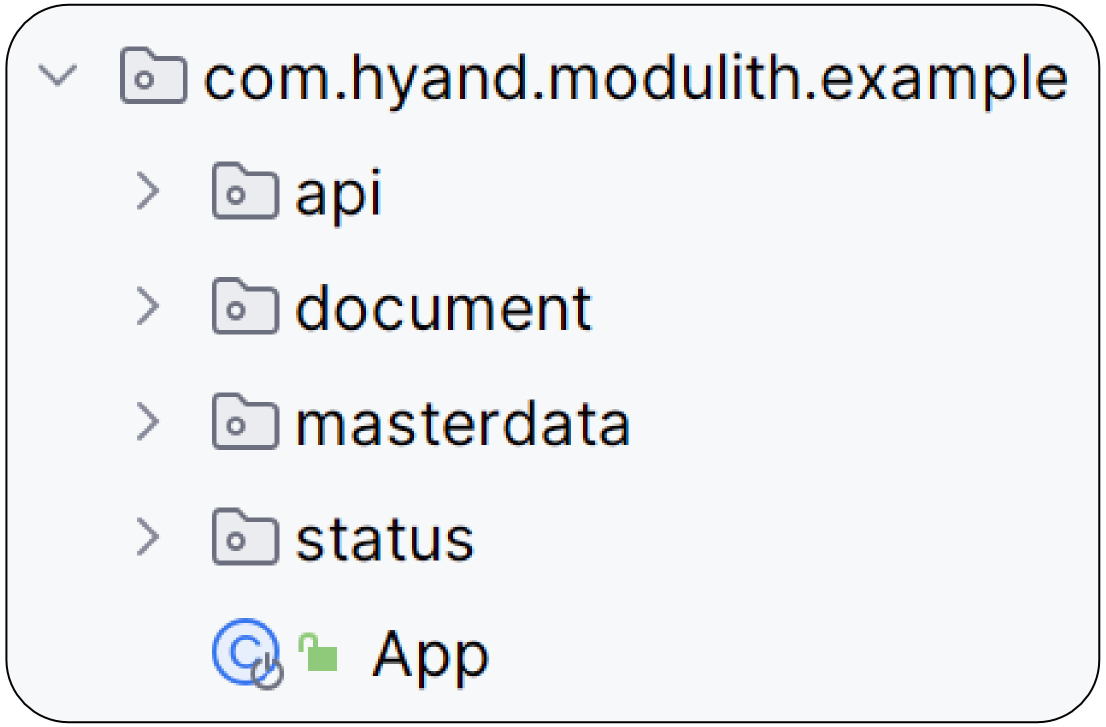
Component diagram



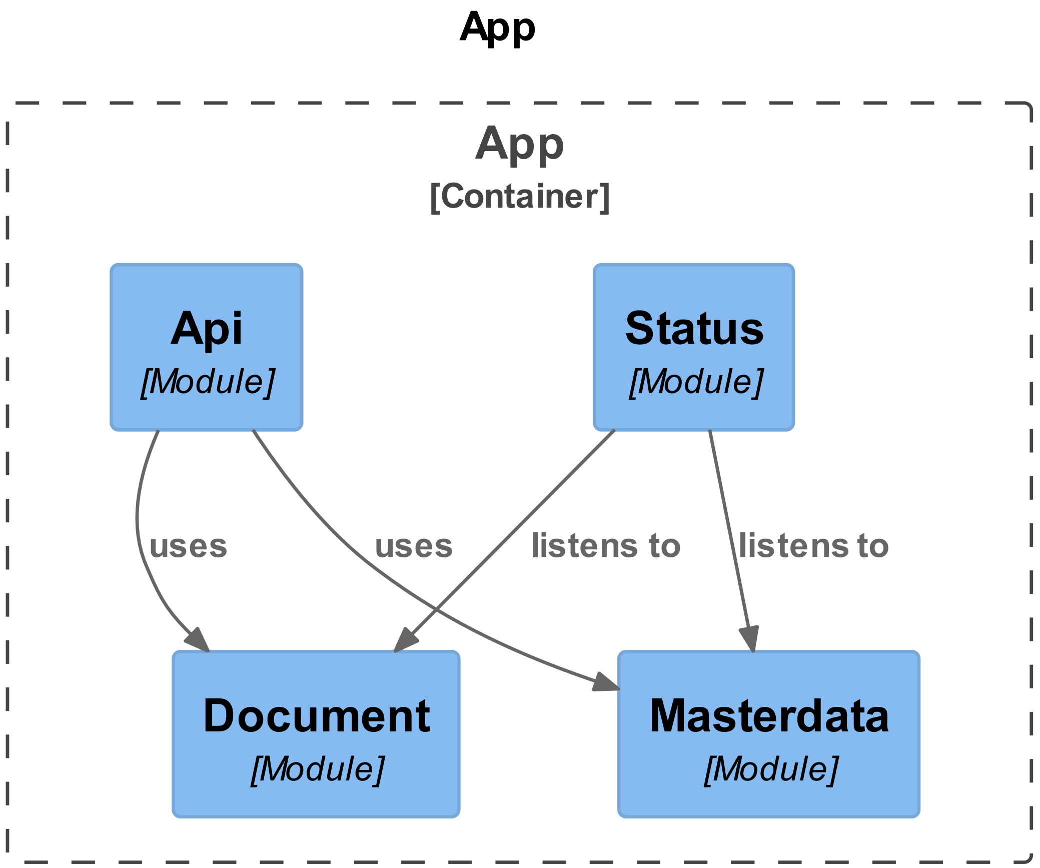
ModulithTests.java

```java
public class ModulithTests {

    @Test
    public void writeDocumentationSnippets() {
        var applicationModules = ApplicationModules.of(App.class);
        new Documenter(applicationModules)
                .writeModulesAsPlantUml()
                .writeIndividualModulesAsPlantUml();
    }

}
```

Component diagram



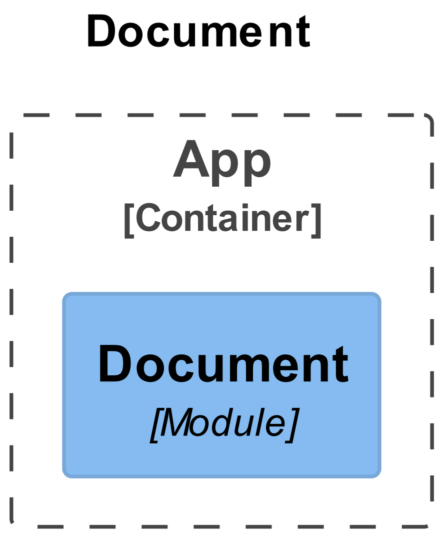
```
com.hyand.modulith.example
  api
  document
  masterdata
  status
  App
```

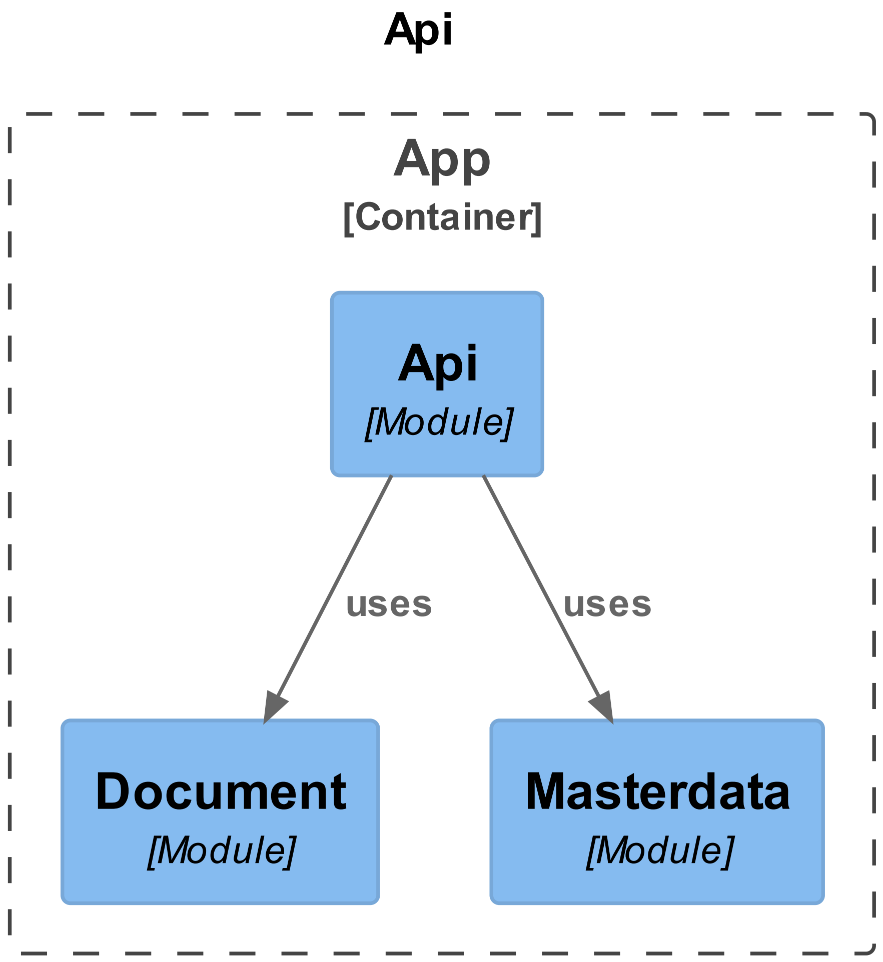


Legend

🟦	component
☐	container boundary (dashed)

Component diagrams
Individual modules





Legend

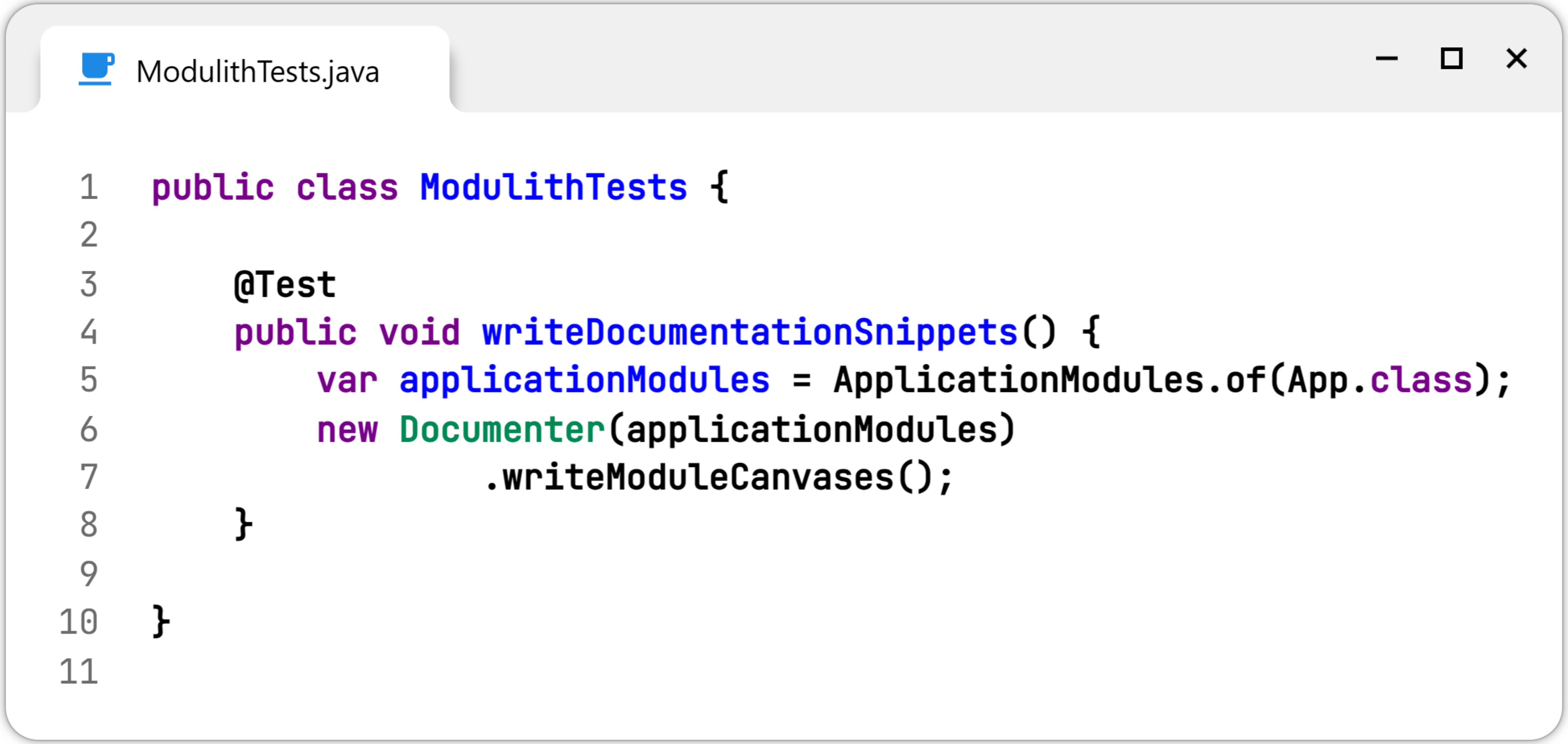
🟦	component
☐	container boundary (dashed)

Legend

🟦	component
☐	container boundary (dashed)

Hyand

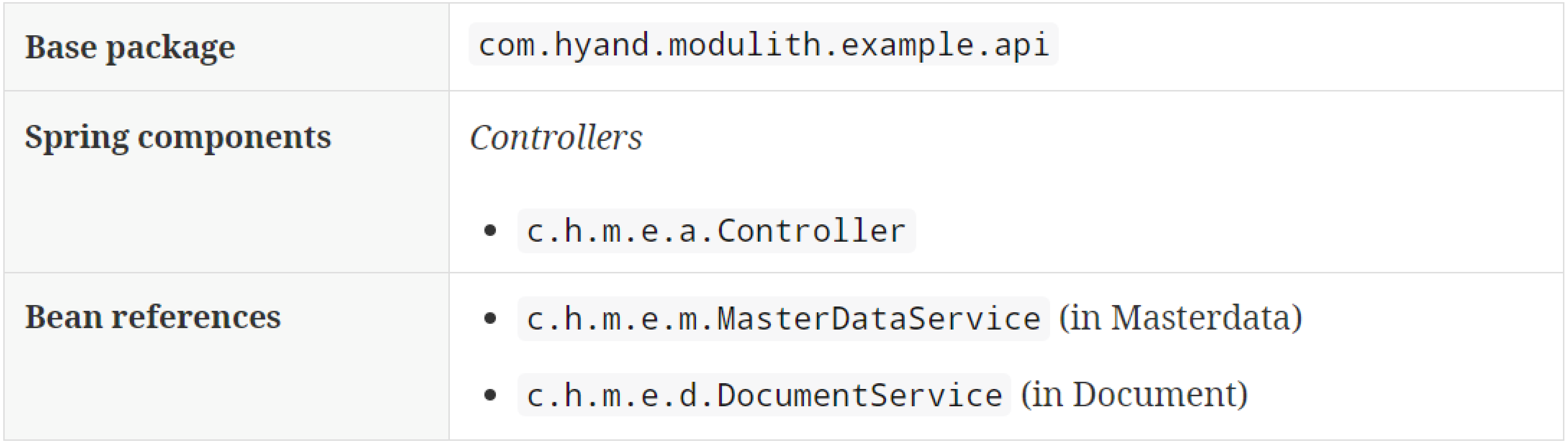
Canvases



ModulithTests.java

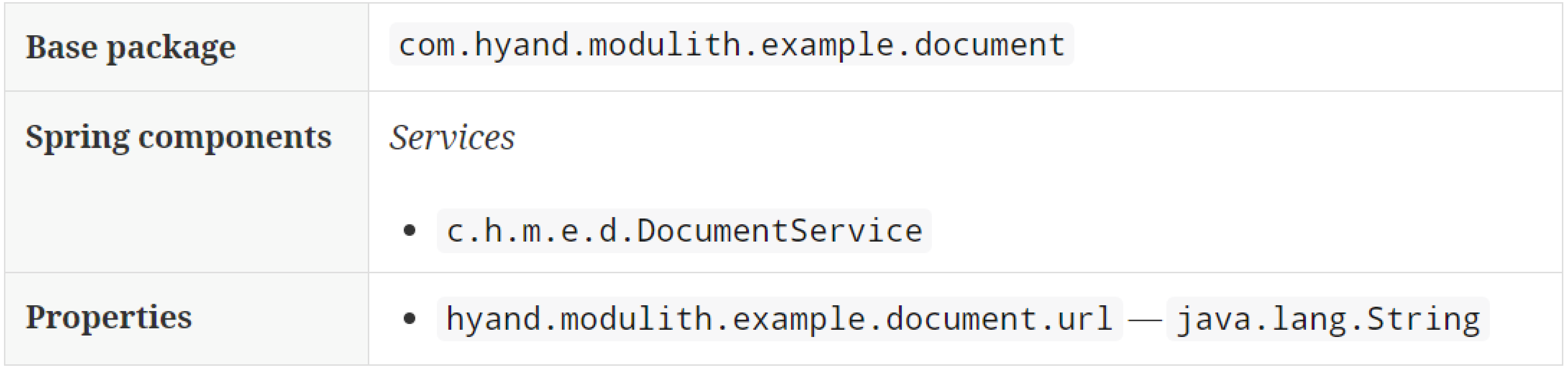
```java
public class ModulithTests {

    @Test
    public void writeDocumentationSnippets() {
        var applicationModules = ApplicationModules.of(App.class);
        new Documenter(applicationModules)
                .writeModuleCanvases();
    }

}
```

Canvas: Module Api



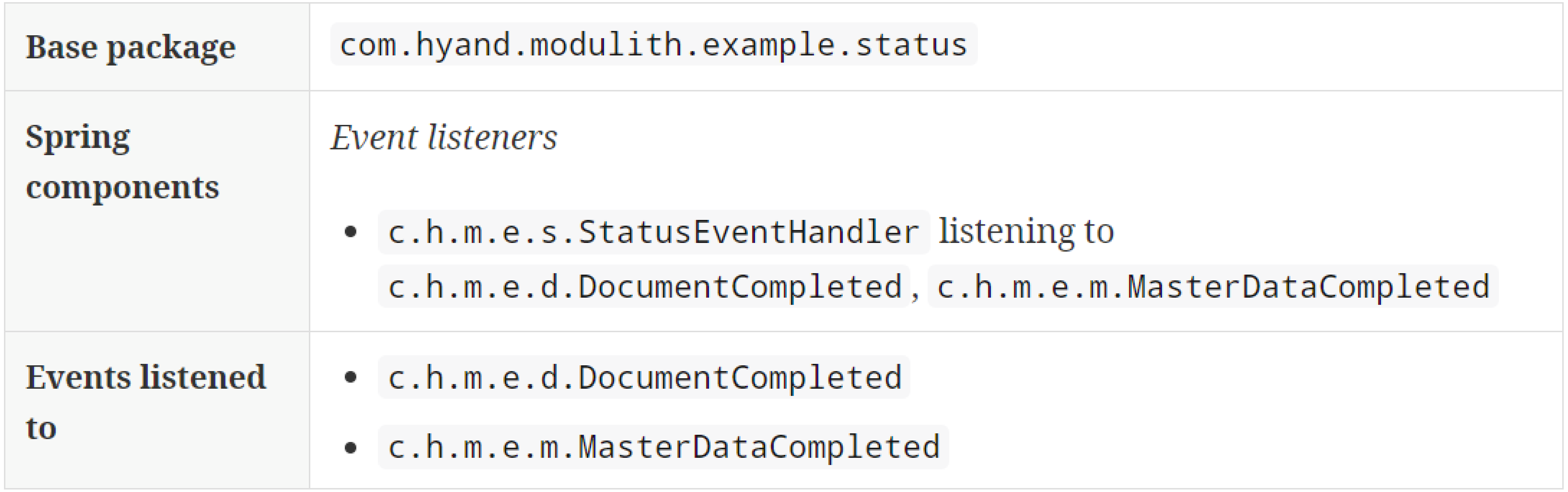
Base package	`com.hyand.modulith.example.api`
Spring components	*Controllers* • `c.h.m.e.a.Controller`
Bean references	• `c.h.m.e.m.MasterDataService` (in Masterdata) • `c.h.m.e.d.DocumentService` (in Document)

Canvas: Module Document



Base package	`com.hyand.modulith.example.document`
Spring components	*Services* • `c.h.m.e.d.DocumentService`
Properties	• `hyand.modulith.example.document.url` — `java.lang.String`

Canvas: Module Status



Base package	`com.hyand.modulith.example.status`
Spring components	*Event listeners* • `c.h.m.e.s.StatusEventHandler` listening to `c.h.m.e.d.DocumentCompleted`, `c.h.m.e.m.MasterDataCompleted`
Events listened to	• `c.h.m.e.d.DocumentCompleted` • `c.h.m.e.m.MasterDataCompleted`

Livedemo

Summary

Summary of Spring Modulith

- Follows the Convention-over-Configuration principle
- Identifies the application modules based on the package structure
- Verifies the application modules with automated tests
- Supports integration tests for application modules
- Extends Spring Events to support loose coupling
- Generates documentation about application modules

Summary of Spring Modulith

- Follows the Convention-over-Configuration principle ✓
- Identifies the application modules based on the package structure ✓
- Verifies the application modules with automated tests ✓
- Supports integration tests for application modules
- Extends Spring Events to support loose coupling
- Generates documentation about application modules ✓

Thank You for Your Attention!

Sebastian Lammering
https://www.linkedin.com/in/sebastian-lammering

Example

https://github.com/slammering/modulith-example

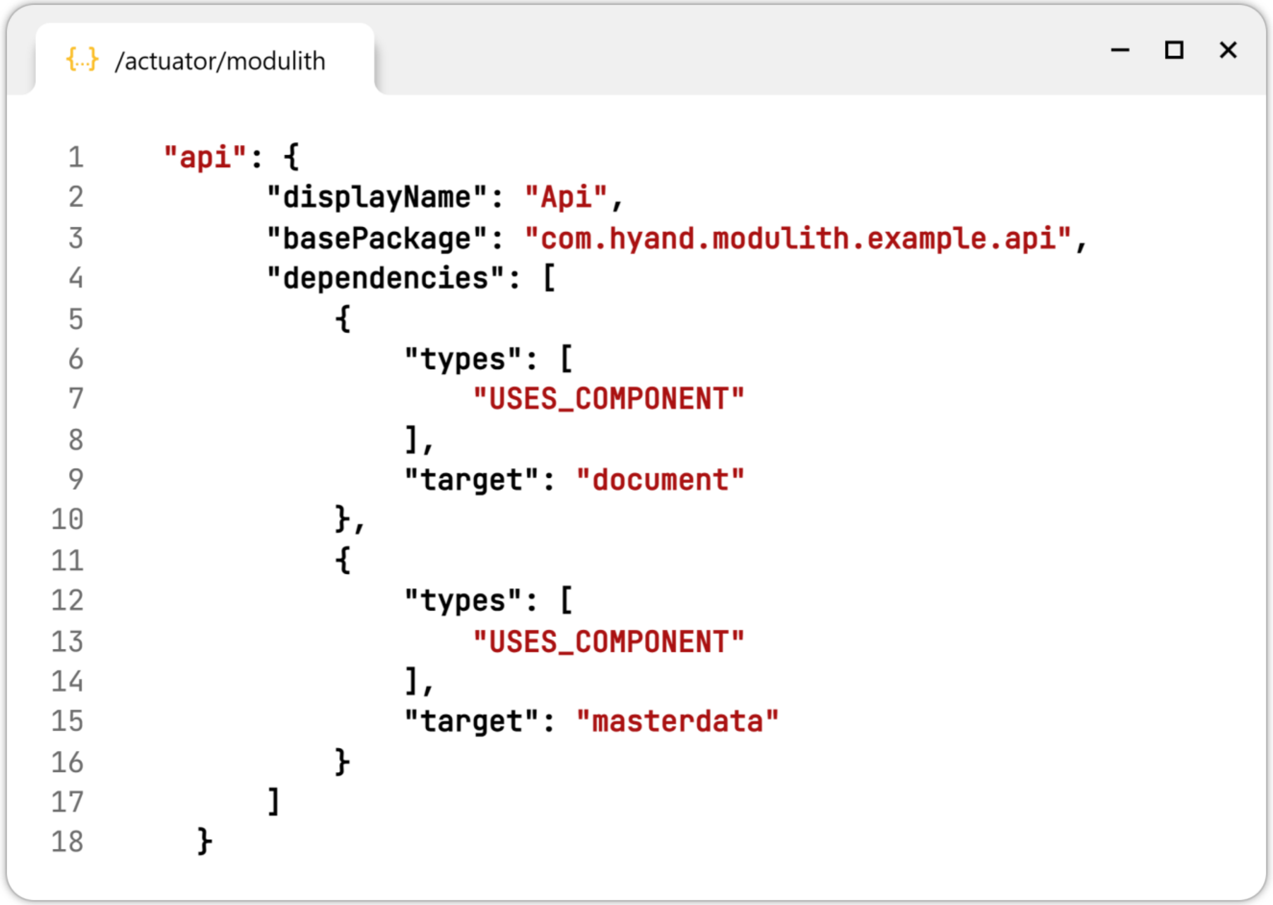
Hyand

Discussion

Hyand

Runtime

Actuator



```
"api": {
      "displayName": "Api",
      "basePackage": "com.hyand.modulith.example.api",
      "dependencies": [
          {
              "types": [
                  "USES_COMPONENT"
              ],
              "target": "document"
          },
          {
              "types": [
                  "USES_COMPONENT"
              ],
              "target": "masterdata"
          }
      ]
  }
```