Red Hat Fuse 7.13

API 설계

OpenShift에서 Fuse 애플리케이션을 위한 REST API 설계

Last Updated: 2024-07-23

Red Hat Fuse 7.13 API 설계

OpenShift에서 Fuse 애플리케이션을 위한 REST API 설계

법적 공지

초록

웹 기반 REST API Creator 사용 가이드

차례

보다 포괄적 수용을 위한 오픈 소스 용어 교체

Red Hat은 코드, 문서, 웹 속성에서 문제가 있는 용어를 교체하기 위해 최선을 다하고 있습니다. 먼저 마스터(master), 슬레이브(slave), 블랙리스트(blacklist), 화이트리스트(whitelist) 등 네 가지 용어를 교체하고 있습니다. 이러한 변경 작업은 향후 여러 릴리스에 대해 단계적으로 구현될 예정입니다. 자세한 내용은 CTO Chris Wright의 메시지 에서 참조하십시오.

1장. API WINDOW의 개요

Red Hat Fuse on OpenShift는 API 서비스에 대한 벤더 중립 및 이식 가능한 오픈 설명 형식인 OpenAPI 사양(버전 3 또는 2)을 준수하는 REST API 를 설계하는 데 사용할 수 있는 웹 기반 API 편집기인 API opportunity를 제공합니다. API Cryostat는 Apicurio Studio 오픈 소스 프로젝트(https://www.apicur.io/)의 "조명" 버전입니다. 즉, API Builder 세션은 상태 비저장이며 각 세션이 끝나면 API 정의를 JSON 파일로 저장해야 합니다.

API window를 사용하여 REST API 정의를 기반으로 사전 Fuse 프로젝트를 생성할 수도 있습니다. 그런 다음 Fuse 개발 환경에서 프로젝트의 Camel 경로를 완료하고 프로젝트를 빌드할 수 있습니다. 마지막으로 OpenShift에서 Fuse에 결과 REST 서비스를 배포할 수 있습니다.

다음은 API Cryostat를 사용하여 Fuse on OpenShift 애플리케이션 솔루션에 REST API를 통합하는 방법에 대한 개요입니다.

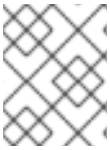
1. OpenShift 프로젝트에 API Cryostat를 서비스로 추가합니다.
2. API desktop에서 다음을 수행합니다.
   - API Cryostat를 사용하여 API 정의를 만듭니다. REST API 정의를 JSON 파일로 로컬 파일 시스템에 저장합니다. API 정의가 완료되지 않은 경우에도 편집 세션 중 언제든지 API 정의를 저장할 수 있습니다.
   - API Cryostat에 API 정의를 업로드합니다.
   - 현재 REST API 정의를 기반으로 Fuse Camel 프로젝트를 생성합니다. API Cryostat는 전체 Maven 프로젝트가 포함된 다운로드 가능한 zip 파일을 제공합니다.
3. Fuse 개발 환경에서 생성된 Fuse 프로젝트에서 제공하는 스케일톤 구현을 완료합니다.
4. Fuse 애플리케이션을 빌드하고 OpenShift에 배포합니다.
5. (선택 사항) 3scale 서비스 검색 기능을 사용하여 Fuse 애플리케이션을 Red Hat 3scale API Management와 통합하여 Fuse 애플리케이션을 찾고 구성합니다.

2장. OPENSHIFT 클러스터에 API CRYOSTAT를 서비스로 추가

2.1. OPENSHIFT 4 프로젝트에 API CRYOSTAT를 서비스로 추가

OpenShift 4.x의 경우 OpenShift 관리자가 OpenShift 가이드의 Fuse에 설명된 대로 프로젝트에 API Policy Operator를 설치했는지 확인해야 합니다.

선택적으로 OpenShift 관리자는 API window를 프로젝트에 서비스로 추가할 수도 있습니다. 그렇지 않은 경우 해당 작업을 수행해야 합니다.



참고

API Builder, **Apicurito** 의 이전 이름은 API Cryostat Operator의 인터페이스에 계속 표시됩니다.

사전 요구 사항

OpenShift 관리자가 OpenShift 프로젝트에 API Policy operator를 설치했습니다.

프로세스

1. 웹 브라우저에서 OpenShift 콘솔을 열고 자격 증명(예: 사용자 이름 **developer** 및 암호 **developer**)을 사용하여 로그인합니다.
2. API Creator Operator가 포함된 프로젝트를 선택합니다.
3. **토폴로지** 를 선택하고 **fuse-apicurito** 레이블이 지정된 아이콘이 표시되는지 확인합니다. **apicurito-service-ui** 및 **apicurito-service-generator** 에 대한 아이콘이 있는 경우 OpenShift 관리자가 이미 API Cryostat를 프로젝트에 서비스로 추가하고 나머지 단계를 건너뛸 수 있습니다.

   apicurito-service-ui 및 **apicurito-service-generator** 에 대한 아이콘이 없는 경우 다음 단계를 계속합니다.
4. 왼쪽 탐색 창에서 **추가를 클릭합니다**.
5. **개발자 카탈로그** 섹션에서 **Operator Backed** 를 클릭합니다.
6. 검색 필드에 **Apicurito** 를 입력하여 카탈로그 항목을 필터링합니다.
7. **Red Hat 카드가 제공하는 Apicurito**를 클릭합니다.
8. **생성**을 클릭합니다.
   API Builder 인스턴스에 대한 최소 시작 템플릿이 있는 기본 양식이 열립니다. 기본값을 허용하거나 선택적으로 편집합니다.
9. **만들기** 를 클릭하여 새 API의 pod, 서비스 및 기타 구성 요소의 시작을 트리거합니다.
10. API Creator 서비스가 프로젝트에 추가되었는지 확인하려면 **Topology** 를 선택한 다음 **apicurito-service-ui** 및 **apicurito-service-generator** 에 대한 아이콘이 표시되는지 확인합니다.
11. APIDesign을 열려면 **apicurito-service-ui** 아이콘에서 URL 링크를 클릭합니다.

2.2. OPENSHIFT 3.11 프로젝트에 API CRYOSTAT를 서비스로 추가

명령줄에서 API Cryostat 템플릿을 배포하여 OpenShift 3.11 프로젝트에 API window를 서비스로 추가할 수 있습니다.

사전 요구 사항

- OpenShift 시스템 관리자가 권장하는 지침에 따라 API Builder에 액세스할 수 있는 호스트 이름을 가져옵니다.
- 명령 창에서 다음 명령을 실행하여 **apidesigner-ui** 및 **fuse-apidesigner-generator** 를 포함한 OpenShift 이미지 및 템플릿의 Fuse가 OpenShift 클러스터에 설치되어 있는지 확인합니다.

```
oc get is -n openshift
```

이미지 및 템플릿이 사전 설치되지 않았거나 제공된 버전이 최신 버전이 아닌 경우 OpenShift 이미지 및 템플릿에 Fuse를 설치(또는 업데이트)하여 OpenShift 가이드 의 Fuse에 설명된 대로 설치(또는 업데이트)합니다.

프로세스

명령줄에서 API Builder 서비스를 추가하려면 다음을 수행합니다.

1. 명령 창에서 OpenShift 서버에 로그인합니다.

```
oc login -u developer -p developer
```

2. 새 프로젝트 네임스페이스를 생성합니다. 예를 들어 다음 명령은 **myproject** 라는 새 프로젝트를 생성합니다.

```
oc new-project myproject
```

3. 다음 명령을 실행하여 APIoauth 템플릿을 기반으로 새 애플리케이션을 생성합니다(여기서 **myproject** 는 프로젝트 이름임).

```
oc new-app -n myproject -f https://raw.githubusercontent.com/jboss-fuse/application-templates/application-templates-2.1.0.fuse-7_13_0-00014-redhat-00001/fuse-apicurito.yml -p ROUTE_HOSTNAME=myhost
```

참고: 필요한 경우 **oc new-app** 명령에 **-p** 옵션을 추가하여 다른 템플릿 매개변수를 지정할 수 있습니다. 예를 들어 기본 **openshift** 네임스페이스 이외의 네임스페이스에 OpenShift 이미지 및 템플릿에 Fuse를 설치한 경우 **IMAGE_STREAM_NAMESPACE** 를 설정하여 Fuse 이미지 스트림이 설치된 네임스페이스를 지정할 수 있습니다.

```
oc new-app -n myproject -f https://raw.githubusercontent.com/jboss-fuse/application-templates/application-templates-2.1.0.fuse-7_13_0-00014-redhat-00001/fuse-apicurito.yml -p ROUTE_HOSTNAME=myhost -p IMAGE_STREAM_NAMESPACE=othernamespace
```

4. 다음 명령을 실행하여 API Creator 배포의 상태 및 URL을 가져옵니다.

```
oc status
```

API window가 배포되지 않은 경우 다음 명령을 실행하여 올바른 버전의 **apicurito-ui** 및 **fuse-apicurito-generator** 이미지를 설치했는지 확인합니다.

```
oc get is -n openshift | grep "apicurito"
```

5. 브라우저에서 API window에 액세스하려면 제공된 URL(예: https://apicurito.192.168.64.12.nip.io)을 사용합니다.

3장. API CRYOSTAT를 사용하여 API 정의 설계 및 개발

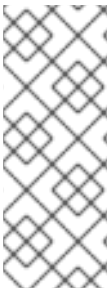
API Builder를 사용하여 OpenAPI 3(또는 2) 사양을 준수하는 REST API 정의를 설계 및 개발할 수 있습니다.

사전 요구 사항

- OpenShift 프로젝트를 생성하셨습니다.
- OpenShift 프로젝트에 API Creator 서비스를 추가했습니다.

3.1. API CRYOSTAT에서 REST API 정의 생성

다음 단계에서는 REST API 정의를 생성하는 방법을 설명합니다.



참고

- OpenShift에서 Fuse Online 또는 Fuse에서 API Builder 사용자 인터페이스에 액세스할 수 있습니다.
- OpenShift의 Fuse만의 경우 API Builder는 상태 비저장이므로 OpenShift 세션 간에 작업을 저장하지 않습니다. 세션 간에 API를 로컬 파일 시스템에 저장해야 합니다.

예제 정보

작업 관리 API 예제에서는 영업 컨설턴트가 고객 연락처와 상호 작용할 때 필요한 작업을 추적하는 데 사용할 수 있는 간단한 API를 시뮬레이션합니다. 예를 들어 "to-do" 작업은 "새 연락처에 대한 계정을 생성"하거나 "기존 연락처에 대한 주문 배치"일 수 있습니다. 작업 관리 API 예제를 구현하려면 두 개의 경로(하나는 작업용 및 특정 작업용 경로)를 생성합니다. 그런 다음 작업을 생성하고, 모든 작업 또는 특정 작업을 검색하고, 작업을 업데이트하고, 작업을 삭제하는 작업을 정의합니다.

사전 요구 사항

- 생성하려는 API의 끝점을 알고 있습니다. 작업 관리 API 예제의 경우 **/todo** 및 **/todo/{id}** 의 끝점이 두 개 있습니다.
- OpenShift의 Fuse만 사용하면 OpenShift 프로젝트를 생성하고 OpenShift 프로젝트에 API Creator 서비스를 추가했습니다.

프로세스

1. Fuse Online을 사용하는 경우 2 단계로 건너뜁니다.
   OpenShift에서 Fuse를 사용하는 경우:

   a. OpenShift 웹 콘솔에 로그인한 다음 API Creator가 포함된 프로젝트를 엽니다.

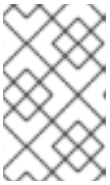
   b. OpenShift 4.x의 경우 **Topology** 를 선택한 다음 **apicurito-service-ui** 아이콘에서 URL 링크를 클릭합니다.
   OpenShift 3.11의 경우 애플리케이션 목록에서 APIFilter의 URL을 클릭합니다(예: https://apidesigner-myproject.192.168.64.43.nip.io) **.**

   API의 새 브라우저 창 또는 탭이 열립니다.

참고

API Builder는 Apicurio Studio 오픈 소스 프로젝트 의 "조명" 버전이므로 "Apicurio"가 API의 Cryostat 인터페이스에 표시됩니다.

2. **New API** 를 클릭합니다. 새 API 페이지가 열립니다.
기본적으로 API Builder는 OpenAPI 3 사양을 사용합니다. OpenAPI 2 사양을 사용하려면 **새 API 버튼 옆에 있는 화살표를 클릭한 다음 OpenAPI 2** 를 선택합니다.



참고

OpenAPI 2 사양을 기반으로 API를 여는 경우, API desktop의 **Convert를 OpenAPI 3 옵션으로 사용하여 API** 를 변환하여 OpenAPI 3 사양을 준수할 수 있습니다.

3. API 이름을 변경하려면 다음을 수행합니다.

   a. 커서를 이름 위에 마우스로 이동한 다음 표시되는 편집 아이콘()을 클릭합니다.

   b. 이름을 편집합니다. 예를 들어 **Task API** 를 입력합니다.

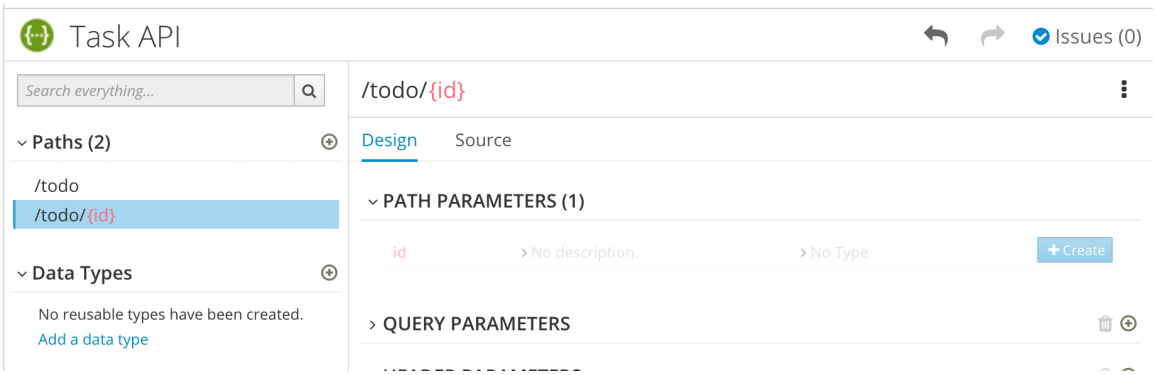
   c. 확인 표시 아이콘을 클릭하여 이름 변경을 확인합니다.

4. 선택적으로 다음을 수행합니다.

   - 버전 번호 및 설명을 입력합니다.
   - 연락처 정보를 추가합니다(이름, 이메일 주소, URL).
   - 라이센스를 선택합니다.
   - 태그를 정의합니다.
   - 하나 이상의 서버를 정의합니다.
   - 보안 스키마를 구성합니다.
   - 보안 요구 사항을 지정합니다.

5. API의 각 개별 끝점에 대한 상대 경로를 정의합니다. 필드 이름은 슬래시(/)로 시작해야 합니다.
작업 관리 API 예제에 대해 다음 두 경로를 생성합니다.

   - 작업 경로: **/todo**
   - ID별 특정 작업의 경로: **/todo/{id}**



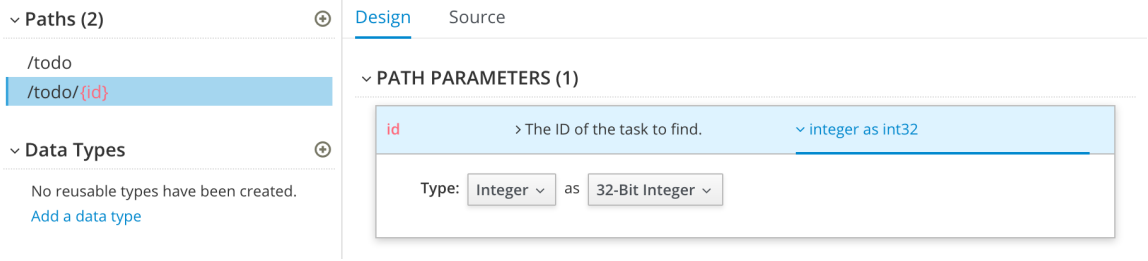
6. 경로 매개변수 유형을 지정합니다.
예제 **id** 매개변수의 경우:

   a. **경로** 목록에서 **/todo/{id}** 를 클릭합니다.
   id 매개변수는 **PATH PARAMETERS** 섹션에 표시됩니다.

   b. **생성**을 클릭합니다.

   c. description: **찾을 작업의 ID를 입력합니다.**

   d. 유형에 대해 정수를 **32bit 정수** 로 선택합니다.



7. **데이터 유형** 섹션에서 API에 재사용 가능한 유형을 정의합니다.

   a. **데이터 유형 추가를** 클릭합니다.

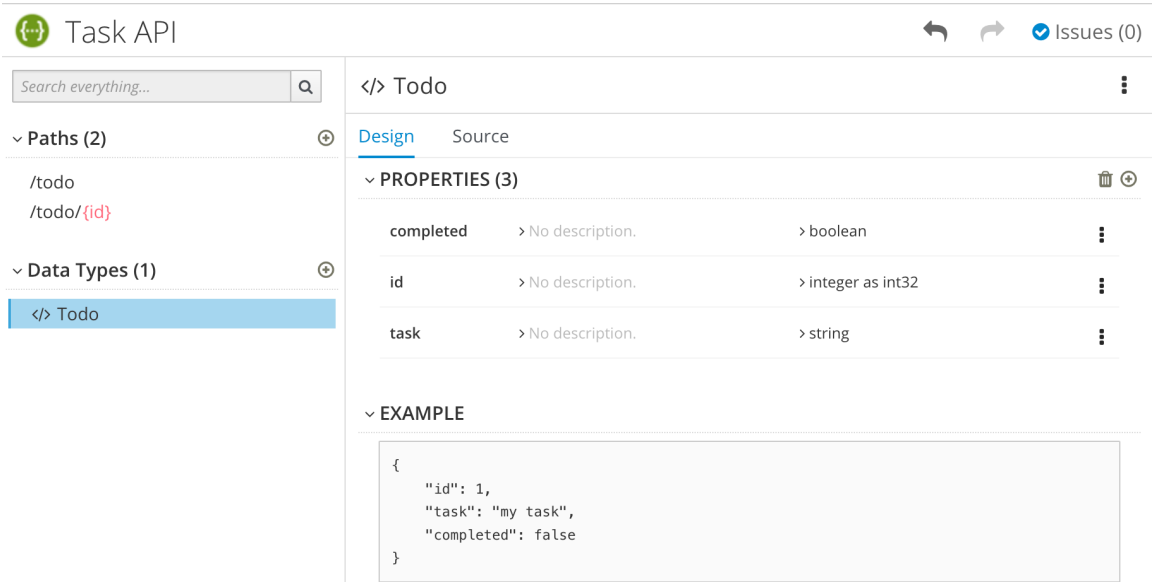
   b. **데이터 유형 추가** 대화 상자에서 이름을 입력합니다. 작업 관리 API 예제에 대해 **Todo** 를 입력합니다.

   c. 선택적으로 API window가 데이터 형식의 스키마를 생성하는 예제를 제공할 수 있습니다. 그런 다음 생성된 스키마를 편집할 수 있습니다.
   작업 관리 API 예제의 경우 다음 JSON 예제로 시작합니다.

```
{
   "id": 1,
   "task": "my task",
   "completed": false
}
```

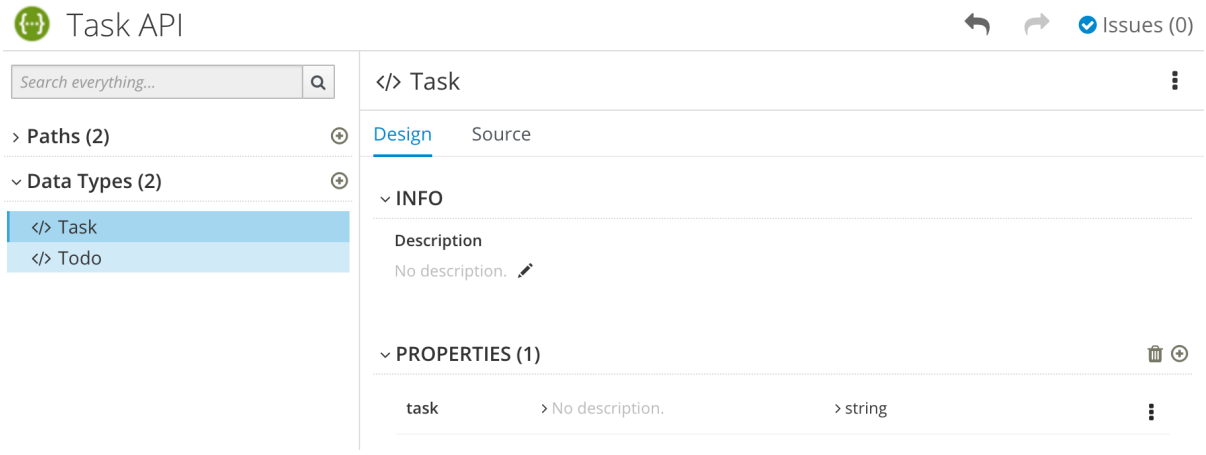
   d. 선택적으로 데이터 유형으로 REST 리소스를 생성하도록 선택할 수 있습니다.

   e. **저장**을 클릭합니다. 예제를 제공한 경우 API window는 다음 예제에서 스키마를 생성합니다.



8. 선택적으로 스키마 속성 편집을 추가하고 새 속성을 추가할 수 있습니다.Optionally, you can add edit the schema properties and add new ones.

9. Task Management API 예제의 경우 **문자열** 형식의 **task** 라는 하나의 속성을 사용하여 **Task** 라는 다른 데이터 유형을 생성합니다.



10. 각 경로에 대해 작업(GET, PUT, POST, DELETE, OPTIONS, HEAD 또는 PATCH)을 정의합니다. 작업 관리 API 예제의 경우 다음 표에 설명된 대로 작업을 정의합니다.

표 3.1. 작업 관리 API 작업

경로	작업	설명	요청 본문	응답
/todo	**POST**	새 작업을 생성합니다.	미디어 유형: **애플리케이션/json** 데이터 유형: **Task**	• 상태 코드: **201** 설명: **생성된 작업** 응답 본문: 미디어 유형: **application/json** 데이터 유형: **Todo**
/todo	**GET**	모든 작업을 가져옵니다.	*해당 없음*	• 상태 코드: **200** 설명: **작업 목록**

경로	작업	설명	요청 본문	응답
/todo/{id}	**GET**	ID로 작업을 가져옵니다.	*해당 없음*	• 상태 코드: 200 설명: **ID 응답 본문에 대해 발견된 작업**: 미디어 유형: **application/json** 데이터 유형: **Task** • 상태 코드: **404** description: **no task with provided identifier found.**
/todo/{id}	**PUT**	작업을 ID로 업데이트합니다.	요청 본문 유형: **Task**	• 상태 코드: 200 설명: **완료됨** • 상태 코드: 400 설명: **작업이 업데이트되지 않음**
/todo/{id}	**DELETE**	ID로 작업을 삭제합니다.	*해당 없음*	• 상태 코드: 200 설명: **작업이 삭제됨** • 상태 코드: 400 설명: **작업이 삭제되지 않음**

11. API Splunk의 유효성 검사 문제에 설명된 대로 모든 문제를 해결합니다 .

12. OpenShift의 Fuse 전용의 경우 **Save As** 를 클릭한 다음 **JSON** 또는 **YAML** 형식을 선택하여 API 사양을 저장합니다.
JSON 또는 YAML 파일이 로컬 다운로드 폴더에 다운로드됩니다. 기본 파일 이름은 적절한 파일 확장자가 있는 **openapi-spec** 입니다.

추가 리소스

- OpenAPI 사양에 대한 자세한 내용은 https://github.com/OAI/OpenAPI-Specification로 이동하십시오.

3.2. API CRYOSTAT의 유효성 검사 문제 해결

API를 생성 및 편집할 때 API Creator는 느낌표(!) 아이콘과 API Cryostat 제목 표시줄의 문제 목록과 함께 해결해야 하는 문제를 식별합니다.

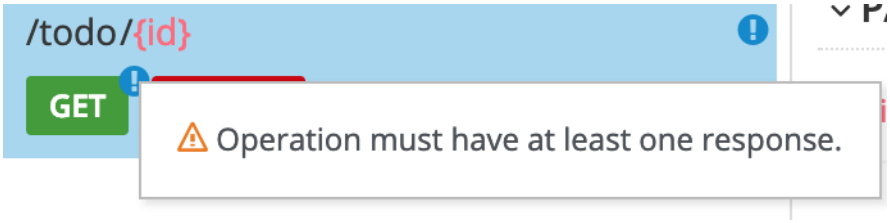
사전 요구 사항

- API window에서 API를 엽니다.

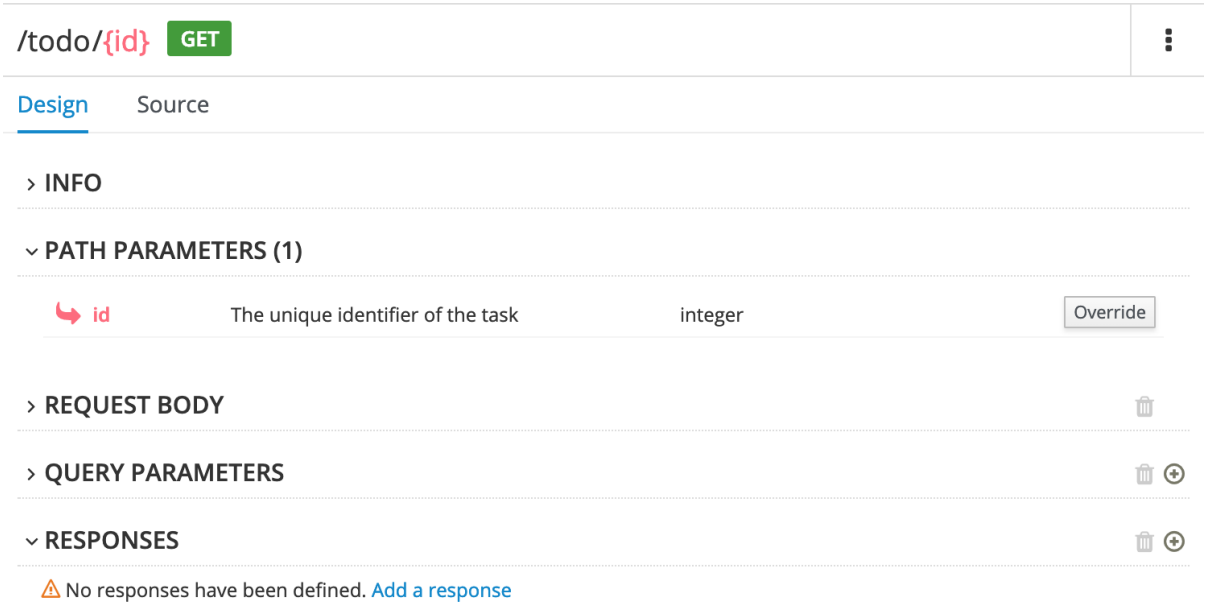
프로세스

1. 느낌표(!) 아이콘으로 표시된 문제를 찾습니다. 예를 들면 다음과 같습니다.

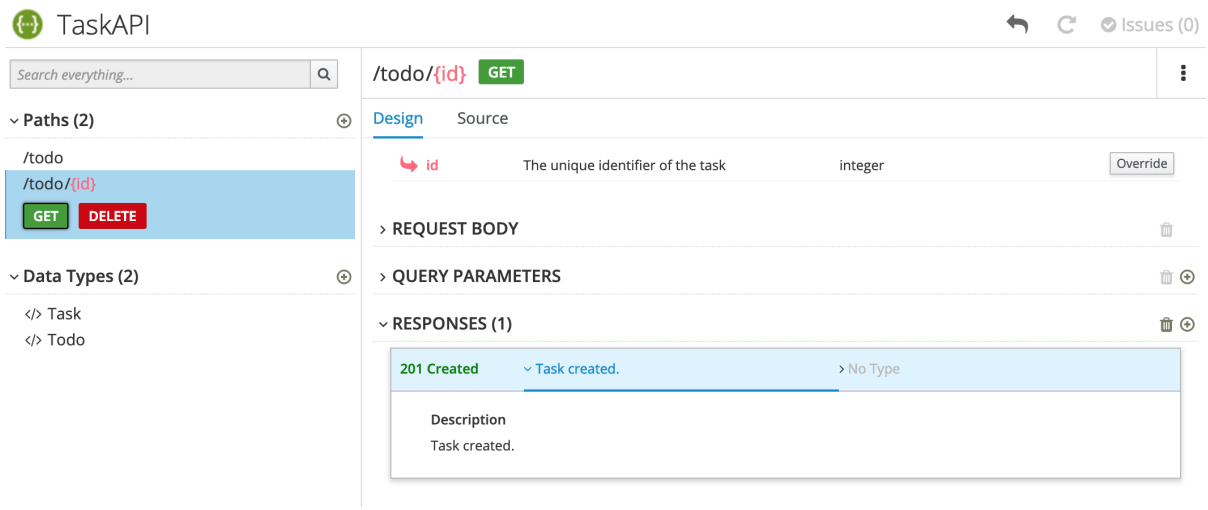
2. 문제에 대한 설명을 보려면 느낌 아이콘을 클릭합니다. 예를 들면 다음과 같습니다.



3. 문제 설명에서 제공하는 정보를 기반으로 문제의 위치로 이동하여 수정합니다.
예를 들어, "작업에 하나 이상의 응답이 있어야 함" 문제를 해결하려면 **GET** 작업을 클릭하여 **응답 추가**를 클릭합니다.



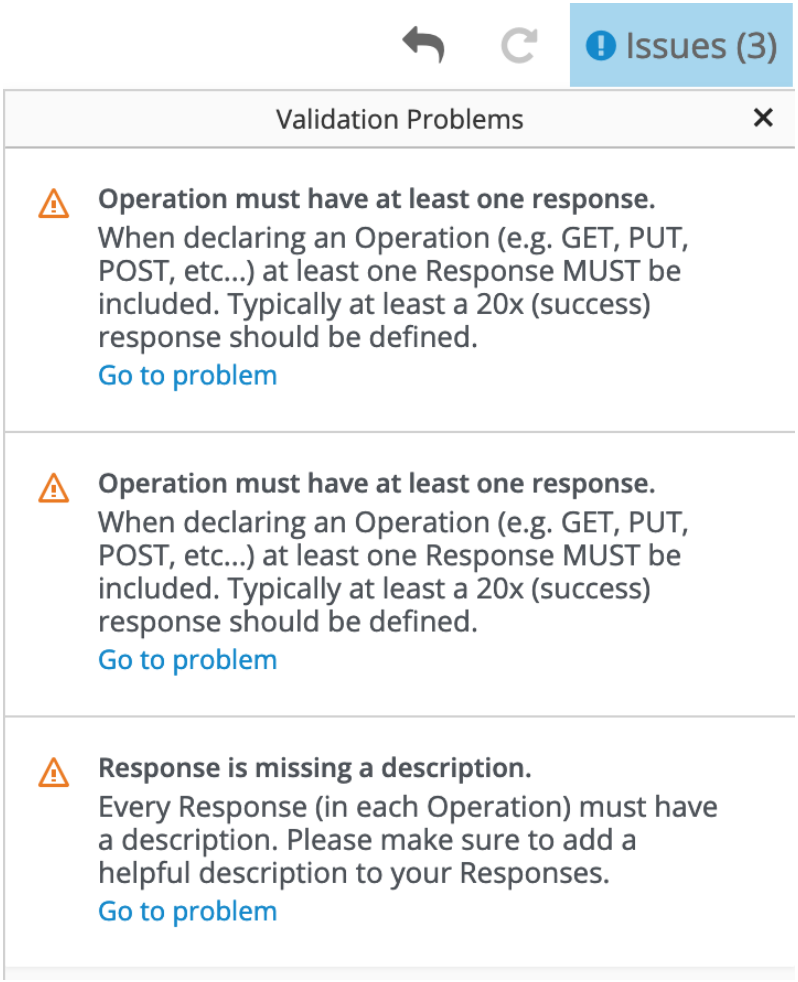
응답에 대한 설명을 입력하면 문제가 해결되고 느낌식 아이콘이 사라집니다.



4. 모든 문제에 대한 요약은 다음과 같습니다.

   a. 오른쪽 상단에 있는 **문제** 링크를 클릭합니다.

   b. **특정 문제의** 문제로 이동하여 문제를 해결할 수 있도록 문제 위치로 이동합니다.



4장. REST API를 기반으로 FUSE 애플리케이션 구현, 빌드 및 배포

Red Hat Fuse API Splunk를 사용하여 REST API 정의를 기반으로 Camel Fuse 프로젝트를 생성할 수 있습니다. Fuse 개발 환경에서는 Camel 경로와 Rest DSL API를 완료할 수 있습니다. 마지막으로 프로젝트를 빌드하고 OpenShift의 Fuse에 결과 애플리케이션을 배포할 수 있습니다.

사전 요구 사항

- 기존 API 정의가 있으며 OpenAPI 3(또는 2) 사양을 준수합니다. 예를 들어 API Policy로 생성한 **openapi-spec.json** 파일이 있습니다.
- API Cryostat가 로컬 OpenShift 클러스터에 설치되어 실행 중입니다.
- API Creator가 서비스로 추가된 기존 OpenShift 프로젝트가 있습니다.
- Maven 및 Red Hat Fuse를 설치했습니다.

다음 주제에서는 REST API를 기반으로 Fuse 애플리케이션을 구현, 빌드 및 배포하는 방법을 설명합니다.

- 4.1절. "API Cryostat에 API 정의 업로드"
- 4.2절. "API Cryostat에서 Fuse Camel 프로젝트 생성"
- 4.3절. "API window-generated Camel 프로젝트 완료"
- 4.4절. "REST 서비스 빌드 및 배포"

4.1. API CRYOSTAT에 API 정의 업로드

기존 API 정의를 API Cryostat에 업로드할 수 있습니다.

사전 요구 사항

- 기존 API 정의가 있으며 OpenAPI 3(또는 2) 사양을 준수합니다. 예를 들어 APIFilter로 생성한 **openapi.json** 파일이 있습니다.
- API Cryostat가 로컬 OpenShift 클러스터에 설치되어 실행 중입니다.
- API Creator가 애플리케이션으로 추가된 기존 OpenShift 프로젝트가 있습니다.

프로세스

1. OpenShift 웹 콘솔에서 API owner가 포함된 프로젝트를 엽니다.
2. API developers console을 엽니다. 프로젝트 애플리케이션 목록에서 apidesigner 아래의 URL을 클릭합니다. 예: **https://apidesigner-myproject.192.168.64.38.nip.io**
   API Splunk 콘솔이 별도의 웹 브라우저 탭 또는 창에서 열립니다.
3. **Open API**를 클릭합니다.
   파일 관리자 창이 열립니다.
4. 파일 관리자 창에서 다음을 수행합니다.
   a. 기존 OpenAPI 정의 파일이 포함된 폴더로 이동합니다(예: **openapi.json**).
   b. OpenAPI 정의 파일을 선택한 다음 **열기**를 클릭합니다.

APIPublish 콘솔에서 OpenAPI 정의가 열립니다.

4.2. API CRYOSTAT에서 FUSE CAMEL 프로젝트 생성

API window를 사용하여 API 정의를 기반으로 Fuse Camel 프로젝트를 생성할 수 있습니다.

사전 요구 사항

- API Cryostat가 로컬 OpenShift 클러스터에 설치되어 실행 중입니다.
- API Creator가 애플리케이션으로 추가된 기존 OpenShift 프로젝트가 있습니다.
- API Policy Console에서 API 정의 파일을 만들거나 열었습니다.

프로세스

API window console에서 다음을 수행합니다.

1. **생성** 을 클릭합니다.
2. 드롭다운 목록에서 **Fuse Camel 프로젝트를** 선택합니다.

APIchunk은 **camel-project.zip** 파일을 생성하여 로컬 기본 다운로드 폴더에 다운로드합니다.

zip 파일에는 Camel의 Rest DSL을 사용하여 API 정의의 기본 스keleton 구현을 제공하고 모든 리소스 작업이 포함된 Fuse Camel 프로젝트가 포함되어 있습니다. 프로젝트에는 프로젝트를 생성하는 데 사용한 원본 OpenAPI 정의 파일도 포함됩니다.

4.3. API WINDOW-GENERATED CAMEL 프로젝트 완료

API Cryostat는 Camel의 Rest DSL을 사용하여 API 정의의 기본 스쿼리톤 구현을 제공하고 모든 리소스 작업을 다루는 Fuse 프로젝트를 생성합니다. Fuse 개발 환경에서는 프로젝트를 완료합니다.

사전 요구 사항

- API Splunk에서 생성된 **camel-project.zip** 파일이 있습니다.
- (선택 사항) Fuse Tooling을 사용하여 Red Hat Developer Studio를 설치했습니다.

프로세스

1. API window가 생성한 **camel-project.zip** 파일의 압축을 임시 폴더에 압축을 풉니다.
2. Red Hat Developer Studio를 엽니다.
3. Developer Studio에서 **파일 → 가져오기** 를 선택합니다.
4. **가져오기** 대화 상자에서 **Maven → Existing Maven 프로젝트를** 선택합니다.
5. 편집기 보기에서 프로젝트의 **camel-context.xml** 파일을 엽니다.
6. **REST** 탭을 클릭하여 Rest DSL 구성 요소를 편집합니다.
   REST 서비스 정의에 대한 자세한 내용은 Apache Camel 개발 가이드의 " REST 서비스 정의" 섹션을 참조하십시오.

Swagger 지원을 사용하여 Cryostat-RS 끝점을 확장하는 방법에 대한 자세한 내용은 Apache CXF 개발 가이드를 참조하십시오.

Fuse Tooling REST 편집기를 사용하는 방법에 대한 자세한 내용은 도구 사용자 가이드의 "Rest DSL 구성 요소 보기 및 편집" 섹션을 참조하십시오.

7. 디자인 탭에서 Camel 경로를 편집합니다.
Camel 경로 편집에 대한 자세한 내용은 도구 사용자 가이드의 "경로 편집기에서 라우팅 컨텍스트 편집" 섹션을 참조하십시오.

4.4. REST 서비스 빌드 및 배포

Fuse 프로젝트를 완료한 후에는 OpenShift에서 프로젝트를 빌드하고 배포할 수 있습니다.

사전 요구 사항

- REST 서비스를 정의하는 오류가 없는 완전한 Fuse 프로젝트가 있습니다.
- Java 8 JDK(이상) 및 Maven 3.3.x(또는 이후 버전)를 설치했습니다.

프로세스

Minishift 또는 Red Hat Container Development Kit와 같은 단일 노드 OpenShift 클러스터 가 있는 경우 여기에서 프로젝트를 배포할 수 있습니다.

이 프로젝트를 실행 중인 단일 노드 OpenShift 클러스터에 배포하려면 다음을 수행합니다.

1. OpenShift 클러스터에 로그인합니다.

```
$ oc login -u developer -p developer
```

2. 프로젝트에 대한 새 OpenShift 프로젝트를 생성합니다. 예를 들어 다음 명령은 **test-deploy** 라는 새 프로젝트를 생성합니다.

```
$ oc new-project test-deploy
```

3. Fuse Camel 프로젝트가 포함된 폴더(예: **myworkspace/camel-project**)로 디렉터리를 변경합니다.

```
$ cd myworkspace/camel-project
```

4. OpenShift 클러스터에 프로젝트를 빌드하고 배포합니다.

```
$ mvn clean fabric8:deploy -Popenshift
```

5. 브라우저에서 OpenShift 콘솔을 열고 프로젝트(예: **test-deploy**)로 이동합니다. **camel-project** 애플리케이션의 Pod가 시작될 때까지 기다립니다.

6. 프로젝트의 **개요** 페이지에서 **camel-project** 애플리케이션의 URL을 찾습니다. URL은 이 양식을 사용합니다. **http://camel-project-MY_PROJECT_NAME.OPENSHIFT_IP_ADDR.nip.io**.

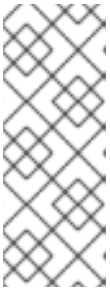
7. URL을 클릭하여 서비스에 액세스합니다.

5장. 3SCALE 검색을 위한 API 서비스 준비

Red Hat 3scale API Management는 공용 인터넷의 API 서비스에 대한 액세스를 규제할 수 있는 Red Hat의 서비스입니다. 3scale 기능에는 SLA(서비스 수준 계약)를 적용하고, API 버전을 관리하고, 보안 및 인증 서비스를 제공하는 기능이 포함됩니다. Fuse는 3scale의 *서비스 검색* 기능을 지원하므로 3scale 관리 포털 UI에서 Fuse 서비스를 쉽게 검색할 수 있습니다. 서비스 검색을 사용하면 동일한 OpenShift 클러스터에서 실행 중인 Fuse 애플리케이션을 스캔하고 관련 API 정의를 3scale로 자동으로 가져올 수 있습니다.

사전 요구 사항

- API 서비스를 제공하는 Fuse 애플리케이션이 OpenShift에서 배포 및 실행됩니다.
- Fuse 애플리케이션에는 3scale에서 검색할 수 있도록 필요한 주석이 포함되어 있습니다.



참고

API Cryostat에서 생성된 Fuse 프로젝트는 필수 주석을 자동으로 제공하도록 사전 구성됩니다.

API Cryostat에서 생성되지 않는 Fuse 프로젝트의 경우 API Cryostat에서 생성되지 않는 Fuse 프로젝트에 대한 주석 추가에 설명된 대로 프로젝트를 구성해야 합니다.

- 3scale API Management 시스템은 검색할 API 서비스와 *동일한* OpenShift 클러스터에 배포됩니다.

3scale에서 API 서비스를 검색하는 절차에 대한 자세한 내용은 Red Hat 3scale API Management 관리 포털 가이드의 서비스 검색 섹션을 참조하십시오.

추가 리소스

- Red Hat 3scale API Management 제품 페이지
- Red Hat 3scale API Management 문서

5.1. API CRYOSTAT에서 생성되지 않은 FUSE 프로젝트에 대한 주석 추가

3scale이 API 서비스를 검색하려면 API 서비스를 제공하는 Fuse 애플리케이션에는 검색할 수 있도록 Kubernetes Service Annotations가 포함되어야 합니다. 이러한 주석은 OpenShift Maven 플러그인의 일부인 Service Discovery Enricher에서 제공합니다.

Apache Camel Rest DSL 프로젝트의 경우 OpenShift Maven 플러그인은 기본적으로 Service Discovery Enricher를 실행합니다.

API Cryostat에서 생성된 Fuse 프로젝트는 필요한 주석을 자동으로 제공하도록 사전 구성됩니다.

프로세스

API Splunk에서 생성되지 않는 Fuse Rest DSL 프로젝트의 경우 다음과 같이 프로젝트를 구성합니다.

1. 다음 예와 같이 Fuse 프로젝트의 **pom.xml** 파일을 편집하여 **openshift-maven-plugin** 종속성을 포함합니다.

```
<plugin>
    <groupId>org.jboss.redhat-fuse</groupId>
```

```
      <artifactId>openshift-maven-plugin</artifactId>
      <version>${fuse.version}</version>
      <executions>
        <execution>
          <goals>
            <goal>resource</goal>
            <goal>build</goal>
          </goals>
        </execution>
      </executions>
    </plugin>
```

OpenShift Maven 플러그인은 특정 프로젝트 수준 조건이 충족되면 Service Discovery Enricher를 실행합니다(예: 프로젝트가 Camel Rest DSL 프로젝트여야 함). 향상된 동작을 사용자 정의하려는 경우(API 서비스 주석 값 사용자 지정에 설명된 대로) **pom.xml** 파일의 종속성으로 Service Discovery Enricher를 지정할 필요가 없습니다.

2. Fuse Rest DSL 프로젝트의 **camel-context.xml** 파일에서 **restConfiguration** 요소에 다음 속성을 지정합니다.

   - **scheme**: 서비스가 호스팅되는 URL의 스키마 부분입니다. "http" 또는 "https"를 지정할 수 있습니다.
   - **contextPath**: API 서비스가 호스팅되는 URL의 경로 부분입니다.
   - **apiContextPath**: API 서비스 설명 문서가 호스팅되는 위치 경로입니다. 문서가 자체 호스팅되는 경우 상대 경로 또는 문서가 외부적으로 호스팅되는 경우 전체 URL을 지정할 수 있습니다.
     예제 **camel-context.xml** 파일의 다음 발췌 내용은 **restConfiguration** 요소의 주석 특성 값을 보여줍니다.

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="
       http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd
       http://camel.apache.org/schema/spring
http://camel.apache.org/schema/spring/camel-spring.xsd">

    <camelContext xmlns="http://camel.apache.org/schema/spring">
        <restConfiguration component="servlet" scheme="https"
             contextPath="myapi" apiContextPath="myapi/openapi.json"/>
...
```

번영자는 이러한 **restConfiguration** 요소 속성 값에서 제공하는 정보를 사용하여 **discovery.3scale.net/scheme**,**discovery.3scale.net/path**, **discovery.3scale.net/description-path** 주석에 대한 값을 생성하므로 Red Hat 3scale API 관리 포털의 서비스 검색 섹션을 확인하는 데 설명된 대로 프로젝트의 배포된 OpenShift 서비스를 3scale 로 검색할 수 있습니다.

보강자는 다음 레이블 및 주석을 추가하여 3scale에서 서비스를 검색할 수 있도록 합니다.

- **discovery.3scale.net** 레이블: 기본적으로 richer는 이 값을 "true"로 설정합니다. 3scale은 선택기 정의를 실행하여 검색이 필요한 모든 서비스를 찾을 때 이 레이블을 사용합니다.
- 다음 주석입니다.

- **discovery.3scale.net/discovery-version**: (선택 사항) 3scale 검색 프로세스의 버전입니다. richer는 이 값을 기본적으로 "v1"로 설정합니다.
- **discovery.3scale.net/scheme**: 서비스가 호스팅되는 URL의 스키마 부분입니다. **restConfiguration** 요소의 **scheme** 속성에서 재정의하지 않는 한 기본 "http"를 사용합니다. 다른 가능한 값은 "https"입니다.
- **discovery.3scale.net/path**: 서비스가 호스팅되는 URL의 경로 부분입니다. 경로가 루트 "/"에 있을 때 이 주석은 생략됩니다. 번영자는 **restConfiguration** 요소의 **path** 속성에서 이 값을 가져옵니다.
- **discovery.3scale.net/port**: 서비스의 포트입니다. 보강자는 표시되는 서비스의 포트 번호를 포함하는 Kubernetes 서비스 정의에서 이 값을 가져옵니다. Kubernetes 서비스 정의가 두 개 이상의 서비스를 노출하는 경우 보강자는 나열된 첫 번째 포트를 사용합니다.
- **discovery.3scale.net/description-path**: (선택 사항) OpenAPI 서비스 설명 문서의 경로입니다. richer는 **restConfiguration** 요소의 **contextPath** 특성에서 이 값을 가져옵니다.

API 서비스 주석 값 사용자 지정 에 설명된 대로 Service Discovery Enricher의 동작을 사용자 지정할 수 있습니다.

5.2. API 서비스 주석 값 사용자 정의

Maven Fabric8 플러그인은 기본적으로 Fabric8 Service Discovery Enricher를 실행합니다. Red Hat 3scale API Management 관리 *포털 가이드 가이드* 의 Service Discovery 사용에 설명된 대로 Fuse Rest DSL 프로젝트의 API 서비스에 주석을 추가하여 API 서비스를 3scale에서 검색할 수 있습니다.

강화자는 일부 주석에 기본값을 사용하고 프로젝트의 **camel-context.xml** 파일에서 다른 주석의 값을 가져옵니다.

Fuse 프로젝트 **pom.xml** 파일 또는 **service.yml** 파일에서 값을 정의하여 **camel-context.xml** 파일에 정의된 기본값과 값을 재정의할 수 있습니다. (두 파일에 값을 정의하면 보강자는 **service.yml** 파일의 값을 사용합니다.) Fabric8 Service Discovery Enricher 에서 Fabric8 Service Discovery Enricher에 지정할 수 있는 요소에 대한 설명은 Fabric8 Service Discovery Enricher 요소를 참조하십시오.

프로세스

Fuse 프로젝트 **pom.xml** 파일에 주석 값을 지정하려면 다음을 수행합니다.

1. 선택한 편집기에서 Fuse 프로젝트의 **pom.xml** 파일을 엽니다.
2. 다음 예와 같이 **openshift-maven-plugin** 종속성을 찾습니다.

```
<plugin>
    <groupId>org.jboss.redhat-fuse</groupId>
    <artifactId>openshift-maven-plugin</artifactId>
    <version>${fuse.version}</version>
    <executions>
      <execution>
        <goals>
          <goal>resource</goal>
          <goal>build</goal>
        </goals>
      </execution>
    </executions>
</plugin>
```

3. 다음 예와 같이 Fabric8 Service Discovery Enricher를 openshift-maven 플러그인에 종속성으로 추가합니다.

```
<plugin>
  <groupId>org.jboss.redhat-fuse</groupId>
  <artifactId>openshift-maven-plugin</artifactId>
  <version>${fuse.version}</version>
  <executions>
    <execution>
      <goals>
        <goal>resource</goal>
        <goal>build</goal>
      </goals>
    </execution>
  </executions>
  <dependencies>
    <dependency>
      <groupId>io.acme</groupId>
      <artifactId>myenricher</artifactId>
      <version>1.0</version>
      <configuration>
        <enricher>
          <config>
            <f8-service-discovery>
              <scheme>https</scheme>
              <path>/api</path>
              <descriptionPath>/api/openapi.json</descriptionPath>
            </f8-service-discovery>
          </config>
        </enricher>
      </configuration>
      </dependency>
  </dependencies>
</plugin>
```

4. 변경 사항을 저장하십시오.

또는 다음 예와 같이 **src/main/fabric8/service.yml** 조각을 사용하여 주석 값을 덮어쓸 수 있습니다.

```
kind: Service
name:
metadata:
  labels:
    discovery.3scale.net/discoverable : "true"
  annotations:
    discovery.3scale.net/discovery-version : "v1"
    discovery.3scale.net/scheme : "https"
    discovery.3scale.net/path : "/api"
    discovery.3scale.net/port : "443"
    discovery.3scale.net/description-path : "/api/openapi.json"
spec:
  type: LoadBalancer
```

5.3. FABRIC8 SERVICE DISCOVERY ENRICHER 요소

다음 표에서는 기본값과 **camel-context.xml** 파일에 정의된 값을 재정의하려는 경우 Fabric8 Service Discovery Enricher에 지정할 수 있는 요소에 대해 설명합니다.

Fuse Rest DSL 프로젝트의 **pom.xml** 파일 또는 **src/main/fabric8/service.yml** 파일에 이러한 값을 정의할 수 있습니다. (두 파일에 값을 정의하면 보강자는 **service.yml** 파일의 값을 사용합니다.) 예를 들어 API 서비스 주석 값 사용자 지정을 참조하십시오.

표 5.1. Fabric8 Service Discovery Enricher 요소

element	설명	기본
springDir	**camel-context.xml** 파일이 포함된 Spring 구성 디렉토리의 경로입니다.	Camel Rest DSL 프로젝트를 인식하는 데 사용되는 **/src/main/resources/spring** 경로입니다.
scheme	서비스가 호스팅되는 URL의 스키마 부분입니다. "http" 또는 "https"를 지정할 수 있습니다.	http
path	API 서비스가 호스팅되는 URL의 경로 부분입니다.	
port	API 서비스가 호스팅되는 URL의 포트 부분입니다.	80
descriptionPath	API 서비스 설명 문서가 호스팅되는 위치 경로입니다. 문서가 자체 호스팅되는 경우 상대 경로 또는 문서가 외부적으로 호스팅되는 경우 전체 URL을 지정할 수 있습니다.	
discoveryVersion	3scale 검색 구현의 버전입니다.	v1
discoverable	**discovery.3scale.net** 레이블을 **true** 또는 **false** 로 설정하는 요소입니다. **true** 로 설정하면 3scale이 이 서비스를 검색하려고 합니다. **false** 로 설정하면 3scale이 이 서비스를 검색하지 않습니다. 이 요소를 스위치로 사용하여 3scale 검색 통합을 "false"로 설정하여 일시적으로 해제할 수 있습니다.	값을 지정하지 않으면 보강자가 서비스를 검색할 수 있는지 여부를 자동 탐지하려고 합니다. 번성자가 서비스를 검색할 수 없다고 결정하는 경우 3scale은 이 서비스를 검색하지 않습니다.