핵심 **REST** 구현

Astra Automation

NetApp
August 11, 2025

목차

핵심 REST 구현

REST 웹 서비스

REST(Representational State Transfer)는 분산된 웹 애플리케이션을 만드는 스타일입니다. 웹 서비스 API 설계에 적용할 경우 서버 기반 리소스를 노출하고 상태를 관리하기 위한 주요 기술 및 모범 사례 집합을 설정합니다. REST는 애플리케이션 개발을 위한 일관된 기반을 제공하지만 각 API의 세부 사항은 특정 설계 선택에 따라 달라질 수 있습니다. 라이브 구축과 함께 사용하기 전에 Astra Control REST API의 특성을 알고 있어야 합니다.

리소스 및 상태 표시

리소스는 웹 기반 시스템의 기본 구성 요소입니다. REST 웹 서비스 응용 프로그램을 만들 때 초기 설계 작업은 다음과 같습니다.

- 시스템 또는 서버 기반 리소스 식별

  모든 시스템은 리소스를 사용하고 유지합니다. 리소스는 파일, 비즈니스 트랜잭션, 프로세스 또는 관리 엔티티가 될 수 있습니다. REST 웹 서비스를 기반으로 애플리케이션을 설계하는 첫 번째 작업 중 하나는 리소스를 식별하는 것입니다.

- 자원 상태 및 연관된 상태 작업의 정의

  리소스는 항상 한정된 수의 상태 중 하나에 있습니다. 상태 변경에 영향을 주는 데 사용되는 상태 및 관련 작업을 명확하게 정의해야 합니다.

URI 끝점

모든 REST 리소스는 잘 정의된 주소 지정 체계를 사용하여 정의되고 사용 가능해야 합니다. 리소스가 있고 식별된 끝점은 URI(Uniform Resource Identifier)를 사용합니다. URI는 네트워크의 각 리소스에 대해 고유한 이름을 만들기 위한 일반 프레임워크를 제공합니다. URL(Uniform Resource Locator)은 리소스를 식별하고 액세스하기 위해 웹 서비스와 함께 사용되는 URI 유형입니다. 일반적으로 리소스는 파일 디렉터리와 비슷한 계층적 구조로 표시됩니다.

HTTP 메시지

HTTP(Hypertext Transfer Protocol)는 웹 서비스 클라이언트 및 서버가 리소스에 대한 요청 및 응답 메시지를 교환하기 위해 사용하는 프로토콜입니다. 웹 서비스 응용 프로그램 설계의 일환으로 HTTP 메서드는 리소스 및 해당 상태 관리 작업에 매핑됩니다. HTTP는 상태 비저장 입니다. 따라서 관련 요청 및 응답 집합을 하나의 트랜잭션으로 연결하려면 요청 및 응답 데이터 플로우와 함께 전달된 HTTP 헤더에 추가 정보가 포함되어야 합니다.

JSON 형식

정보는 여러 가지 방법으로 웹 서비스 클라이언트와 서버 간에 구조화되고 전송될 수 있지만 가장 일반적인 옵션은 JSON(JavaScript Object Notation)입니다. JSON은 단순 데이터 구조를 일반 텍스트로 나타내는 업계 표준이며 리소스를 설명하는 상태 정보를 전송하는 데 사용됩니다. Astra Control REST API는 JSON을 사용하여 각 HTTP 요청 및 응답의 본문으로 전송되는 데이터를 포맷합니다.

리소스 및 컬렉션

Astra Control REST API는 리소스 인스턴스 및 리소스 인스턴스 컬렉션에 대한 액세스를 제공합니다.

개념적으로 REST * 리소스 * 는 OOP(개체 지향 프로그래밍) 언어 및 시스템에 정의된 * 개체 * 와 유사합니다. 때때로 이러한 용어는 서로 바꿔서 사용할 수 있습니다. 그러나 일반적으로 "리소스"는 외부 REST API의 컨텍스트에서 사용하는 반면 "개체"는 서버에 저장된 해당 상태 저장 인스턴스 데이터에 사용됩니다.

Astra 리소스의 속성

Astra Control REST API는 RESTful 설계 원칙을 준수합니다. 각 Astra 리소스 인스턴스는 잘 정의된 리소스 유형을 기반으로 생성됩니다. 같은 형식의 리소스 인스턴스 집합을 * 컬렉션 * 이라고 합니다. API 호출은 개별 리소스 또는 리소스 모음에 대해 작동합니다.

리소스 유형

Astra Control REST API에 포함된 리소스 유형은 다음과 같은 특성을 갖습니다.

- 모든 리소스 유형은 스키마(일반적으로 JSON)를 사용하여 정의됩니다.
- 모든 리소스 스키마에는 리소스 유형과 버전이 포함됩니다
- 리소스 유형은 전역적으로 고유합니다

리소스 인스턴스

Astra Control REST API를 통해 사용 가능한 리소스 인스턴스의 특징은 다음과 같습니다.

- 리소스 인스턴스는 단일 리소스 유형에 따라 만들어집니다
- 리소스 유형은 미디어 유형 값을 사용하여 표시됩니다
- 인스턴스는 Astra 서비스에서 유지 관리하는 상태 저장 데이터로 구성됩니다
- 각 인스턴스는 고유하고 오래 지속되는 URL을 통해 액세스할 수 있습니다
- 리소스 인스턴스에 둘 이상의 표현이 있을 수 있는 경우 다양한 미디어 유형을 사용하여 원하는 표현을 요청할 수 있습니다

리소스 컬렉션

Astra Control REST API를 통해 사용 가능한 리소스 수집에는 다음과 같은 특성이 있습니다.

- 단일 리소스 형식의 리소스 인스턴스 집합을 컬렉션이라고 합니다
- 리소스 컬렉션에는 고유하고 오래 지속되는 URL이 있습니다

인스턴스 식별자

모든 리소스 인스턴스는 만들 때 식별자가 할당됩니다. 이 식별자는 128비트 UIDv4 값입니다. 할당된 UUIDv4 값은 전역적으로 고유하며 변경할 수 없습니다. 새 인스턴스를 만드는 API 호출을 실행하면 관련 ID가 있는 URL이 의 호출자에게 반환됩니다 `Location` HTTP 응답의 헤더입니다. 식별자를 추출하여 리소스 인스턴스를 참조할 때 후속 호출에 사용할 수 있습니다.

리소스 식별자는 컬렉션에 사용되는 기본 키입니다.

Astra 리소스의 공통 구조

모든 Astra Control 리소스는 공통 구조를 사용하여 정의됩니다.

공통 데이터

모든 Astra 리소스에는 다음 표에 나와 있는 키 값이 포함되어 있습니다.

키	설명
유형	자원 유형 * 이라고 하는 전역적으로 고유한 자원 유형입니다.
버전	리소스 버전*이라고 하는 버전 식별자입니다.
ID입니다	리소스 식별자 * 라고 하는 전역 고유 식별자입니다.
메타데이터	사용자 및 시스템 레이블을 비롯한 다양한 정보를 포함하는 JSON 개체입니다.

메타데이터 개체입니다

각 Astra 리소스에 포함된 메타데이터 JSON 개체에는 다음 표에 나와 있는 키 값이 포함됩니다.

키	설명
라벨	리소스와 연결된 클라이언트 지정 레이블의 JSON 배열입니다.
CreationTimestamp 를 클릭합니다	JSON 문자열에는 리소스가 생성된 시점을 나타내는 타임스탬프가 포함되어 있습니다.
modificationTimestamp	ISO-8601 형식의 타임 스탬프가 포함된 JSON 문자열로, 리소스가 마지막으로 변경된 시기를 나타냅니다.
생성 시	리소스를 생성한 사용자 ID의 UUIDv4 식별자가 포함된 JSON 문자열입니다. 내부 시스템 구성 요소에 의해 리소스가 생성되었고 생성 엔티티와 연관된 UUID가 없는 경우 * null * UUID가 사용됩니다.

리소스 상태입니다

선택한 자원 A `state` 수명 주기 전환을 조율하고 액세스를 제어하는 데 사용되는 값입니다.

HTTP 세부 정보

Astra Control REST API는 HTTP 및 관련 매개 변수를 사용하여 리소스 인스턴스 및 컬렉션에 대한 작업을 수행합니다. HTTP 구현에 대한 자세한 내용은 아래에 나와 있습니다.

API 트랜잭션 및 CRUD 모델

Astra Control REST API는 잘 정의된 작업과 상태 전환을 통해 트랜잭션 모델을 구현합니다.

요청 및 응답 **API** 트랜잭션

모든 REST API 호출은 Astra 서비스에 대한 HTTP 요청으로 수행됩니다. 각 요청은 연결된 응답을 클라이언트에 다시 생성합니다. 이 요청 응답 쌍은 API 트랜잭션으로 간주될 수 있습니다.

CRUD 운영 모델 지원

Astra Control REST API를 통해 사용 가능한 각 리소스 인스턴스 및 컬렉션은 * CRUD * 모델을 기반으로 액세스합니다. 4개의 작업이 있으며, 각 작업은 단일 HTTP 메서드에 매핑됩니다. 다음과 같은 작업이 포함됩니다.

- 생성
- 읽기
- 업데이트
- 삭제

일부 Astra 리소스의 경우 이러한 작업의 일부만 지원됩니다. 를 검토해야 합니다 "온라인 API 참조입니다" 특정 API 호출에 대한 자세한 내용은 를 참조하십시오.

HTTP 메서드

API에서 지원하는 HTTP 메서드 또는 동사는 아래 표에 나와 있습니다.

방법	CRUD	설명
가져오기	읽기	리소스 인스턴스 또는 컬렉션의 개체 속성을 검색합니다. 이 작업은 컬렉션과 함께 사용할 때 * list * 연산으로 간주됩니다.
게시	생성	입력 매개 변수를 기반으로 새 리소스 인스턴스를 만듭니다. 장기 URL은 에서 반환됩니다 `Location` 응답 헤더.
를 누릅니다	업데이트	제공된 JSON 요청 본문으로 전체 리소스 인스턴스를 업데이트합니다. 사용자가 수정할 수 없는 키 값은 보존됩니다.
삭제	삭제	기존 리소스 인스턴스를 삭제합니다.

요청 및 응답 헤더

다음 표는 Astra Control REST API와 함께 사용되는 HTTP 헤더를 요약한 것입니다.

을 참조하십시오 "RFC 7232" 및 "RFC 7233" 를 참조하십시오.

머리글	유형	사용 참고 사항
수락	요청하십시오	값이 " * / * "이거나 제공되지 않은 경우 `application/json` Content-Type 응답 헤더로 반환됩니다. 값이 Astra 리소스 미디어 유형으로 설정되어 있으면 Content-Type 헤더에서 동일한 미디어 유형이 반환됩니다.
권한 부여	요청하십시오	사용자에 대한 API 키가 있는 베어러 토큰.
Content-Type(콘텐츠 유형)	응답	에 따라 반환됩니다 `Accept` 요청 헤더.
ETag	응답	RFC 7232에 정의된 대로 에 포함됩니다. 값은 전체 JSON 리소스에 대한 MD5 값의 16진수 표현입니다.
일치하는 경우	요청하십시오	3.1 RFC 7232절에 설명된 대로 구현된 전제 조건 요청 헤더와 * PUT * 요청 지원.

머리글	유형	사용 참고 사항
If-Modified-Since	요청하십시오	섹션 3.4 RFC 7232에 설명된 대로 구현된 전제 조건 요청 헤더와 * PUT * 요청 지원.
수정되지 않은 경우 - 이후	요청하십시오	섹션 3.4 RFC 7232에 설명된 대로 구현된 전제 조건 요청 헤더와 * PUT * 요청 지원.
위치	응답	새로 만든 리소스의 전체 URL을 포함합니다.

쿼리 매개 변수

다음 쿼리 매개 변수를 리소스 모음과 함께 사용할 수 있습니다. 을 참조하십시오 "컬렉션 작업" 를 참조하십시오.

쿼리 매개 변수입니다	설명
포함	컬렉션을 읽을 때 반환되어야 하는 필드를 포함합니다.
필터	컬렉션을 읽을 때 반환할 리소스에 대해 일치해야 하는 필드를 나타냅니다.
주문	컬렉션을 읽을 때 반환되는 리소스의 정렬 순서를 결정합니다.
제한	컬렉션을 읽을 때 반환되는 최대 리소스 수를 제한합니다.
건너뛰기	컬렉션을 읽을 때 전달하고 건너뛸 리소스 수를 설정합니다.
카운트	메타데이터 개체에서 총 리소스 수를 반환해야 하는지 여부를 나타냅니다.

HTTP 상태 코드입니다

Astra Control REST API에서 사용하는 HTTP 상태 코드는 아래와 같다.

Astra Control REST API는 HTTP API * 표준에 대한 * Problem Details도 사용합니다. 을 참조하십시오 "진단 및 지원" 를 참조하십시오.

코드	의미	설명
200	좋습니다	새 리소스 인스턴스를 만들지 않는 호출의 성공 여부를 나타냅니다.
201	작성됨	객체가 성공적으로 생성되고 위치 응답 헤더에 객체의 고유 식별자가 포함됩니다.
204	콘텐츠가 없습니다	반환된 콘텐츠가 없지만 요청이 성공했습니다.
400	잘못된 요청입니다	요청 입력이 인식되지 않거나 부적절합니다.
401	권한이 없습니다	사용자에게 권한이 없으며 인증이 필요합니다.
403	금지됨	인증 오류로 인해 액세스가 거부되었습니다.
404	찾을 수 없습니다	요청에서 참조되는 리소스가 없습니다.
409	충돌	개체가 이미 있으므로 개체를 만들지 못했습니다.
500입니다	내부 오류입니다	서버에서 일반적인 내부 오류가 발생했습니다.
503	서비스를 사용할 수 없습니다	어떤 이유로 요청을 처리할 준비가 되지 않았습니다.

URL 형식

REST API를 통해 리소스 인스턴스 또는 컬렉션에 액세스하는 데 사용되는 URL의 일반 구조는 여러 값으로 구성됩니다. 이 구조체는 기본 개체 모델 및 시스템 설계를 반영합니다.

root로 계정을 지정합니다

모든 REST 끝점에 대한 리소스 경로의 루트는 Astra 계정입니다. 따라서 URL의 모든 경로는 로 시작합니다 `/account/{account_id}` 위치 `account_id` 계정에 대한 고유한 UUIDv4 값입니다. 내부 구조 모든 리소스 액세스가 특정 계정을 기반으로 하는 설계를 반영합니다.

끝점 리소스 범주입니다

Astra 리소스 끝점은 세 가지 범주로 나뉩니다.

- 코어 (`/core`)
- 관리형 애플리케이션 (`/k8s`)
- 토폴로지 (`/topology`)

을 참조하십시오 "리소스" 를 참조하십시오.

카테고리 버전

세 가지 리소스 범주 각각에는 액세스한 리소스의 버전을 제어하는 전역 버전이 있습니다. 규칙 및 정의에 따라 리소스 범주의 새로운 주 버전(예: 에서)으로 이동합니다 `/v1` 를 선택합니다 `/v2`)는 API의 변경 사항을 깨는 방법을 소개합니다.

리소스 인스턴스 또는 컬렉션입니다

리소스 인스턴스 또는 컬렉션 액세스 여부에 따라 경로에서 리소스 형식과 식별자의 조합을 사용할 수 있습니다.

예

- 리소스 경로입니다

  위에서 설명한 구조를 기반으로 엔드포인트에 대한 일반적인 경로는 다음과 같습니다. `/accounts/{account_id}/core/v1/users`.

- URL을 완료합니다

  해당 끝점의 전체 URL은 다음과 같습니다. `https://astra.netapp.io/accounts/{account_id}/core/v1/users`.