KOOKMIN UNIVERSITY

Pydantic
ORM
SQLAlchemy

Pydantic

소프트웨어융합대학원
진혜진

Pydantic - BaseModel 상속과 타입 선언

- 선언형 데이터 모델
  - model_dump()
- 타입 안전과 오류 메시지
- 선택적 필드와 기본값
  - Optional[...] 또는 str | None = None
- Pydantic v2 변경점
  - .model_dump()

Request/Response 유효성 검사와 데이터 변환

- 입력 검증
- 기본 변환
  - Model(a=3.000, b='2.72', c=b'binary').model_dump()
  - {'a':3,'b':2.72,'c':'binary data'}
- 응답 모델
  - response_model

Field와 검증 제약조건, 커스텀 검증기

Field 제약조건

- Field()
- gt/ge/lt/le/multiple_of
- min_length, max_length, pattern

기본값·별칭·기타 옵션

- Field(default=..., alias=..., description=...)
- default_factory
- frozen

커스텀 검증기

- @validator
- @root_validator

중첩 모델과 컬렉션 타입

중첩 모델

리스트/딕셔너리 필드

- list[Model]
- dict[str, Model]

특수 타입

- URL, 이메일, UUID 등 공통 타입
- HttpUrl, EmailStr, UUID

FastAPI와 Pydantic 통합

- 요청 본문 처리
- 경로·쿼리 파라미터
  - Query(), Path()
- 응답 모델
  - response_model 파라미터
- 자동 문서화
  - Swagger UI

- ORM : SQL vs NoSQL
  - 데이터베이스
  - 확장성
  - 커뮤니티와 표준화

- 객체-관계 매핑(ORM)
  - 생산성 향상
  - 데이터베이스 독립성
  - 보안
  - 복잡한 쿼리의 한계
  - 성능 오버헤드
  - 학습 곡선

SQLAlchemy

- 엔진(Engine)
  - connection pool, dialect
  - create_engine()
- 세션(Session)
  - identity map, commit()
- 모델(Declarative 모델 클래스)
  - DeclarativeBase, __tablename__
  - mapped_column(primary_key=True), relationship()
- 쿼리
  - select(User)
  - Session.scalars(stmt), session.exec()

Response 모델과 ORM 모델 클래스 분리

- 보안과 데이터 노출 최소화
  - response_model
- 유효성 검증 및 변환 책임 분리
- 문서 자동화
- 확장성과 재사용성