## 패스워드 해싱

〈CHAPTER 6 데이터베이스 연결〉에서는 사용자 패스워드를 일반 텍스트로 저장했다. 이 방법은 실제로 API를 구축할 때 지양해야 하는 매우 나쁜 습관이다. 패스워드는 적절한 라이브러리를 사용해서 반드시 암호화(해싱이라고도 한다)해야 한다. 여기서는 bcrypt를 사용해 패스워드를 암호화한다.

먼저 passlib 라이브러리를 설치한다. 이 라이브러리는 패스워드를 해싱하는 bcrypt 알고리즘을 제공한다.

```
(venv)$ pip install passlib[bcrypt]
```

그런 다음 hash_password.py 파일에 패스워드를 해싱하는 함수를 작성한다.

```
from passlib.context import CryptContext

pwd_context = CryptContext(schemes=["bcrypt"], deprecated="auto")


class HashPassword:
    def create_hash(self, password: str):
        return pwd_context.hash(password)

    def verify_hash(self, plain_password: str, hashed_password: str):
        return pwd_context.verify(plain_password, hashed_password)
```

bcrypt를 사용해 문자열을 해싱할 수 있도록 CryptContext를 임포트한다. 콘텍스트는 pwd_context 변수에 저장되며 이 변수를 사용해 해싱에 필요한 함수들을 호출한다.

이제 HashPassword 클래스를 정의하고 그 안에 create_hash()와 verify_hash()라는 두 개의 메서드를 작성해보자.

- **create_hash()**: 문자열을 해싱한 값을 반환한다.
- **verify_hash()**: 일반 텍스트 패스워드와 해싱한 패스워드를 인수로 받아 두 값이 일치하는지 비교한다. 일치 여부에 따라 불린boolean 값을 반환한다.