**Chapter 06. 데이터베이스 사용하기**

* 비밀번호 암호화 하여 저장하기
* 기본적으로 사용자 정보를 DB 저장 시, 단방향 암호화하여 문자열을 알 수 없도록 한다. 복호화(Description)이 불가능 한 방식이 단방향으로 암호화(Encryption)된 것이다. 글자를 암호화 한 후 다시 원본 글자로 복호화 할 수 있는 것이 양방향 암호화.
* 우선 유저 추가 시에 id / name / 암호화된 password를 추가하고, 인증 시에 클라이언트가 보낸 패스워드를 암호화 한 후 DB에 저장된 암호화 패스워드와 비교한다.
* Mongoose의 virtual 함수로 패스워드를 암호화하게 된다
* 즉 사용자 추가 / 인증 과정에서 password를 입력하지만, 중간에 get/set 함수를 사용해 DB 컬렉션의 hashed\_password와 비교를 거치게 된다.
* [virtual\_test1.js]

DB를 연결하고나서, createUserSchema( ); 로 스키마와 모델을 생성하고, doTest( );로 유저 모델에 있는 컬렉션 문서의 id와 name을 출력한다.

* 우선 스키마 정의 시 id / name / age / created\_at / updated\_at 속성을 설정한다
* 그리고 해당 스키마에 UserSchema.virtual(‘info’).set(…).get(…) 을 통해, 스키마에 저장되지 않는 info 라는 속성을 만든다. 이 때, set( )에서는 해당 속성에 ‘test01 아이즈원’ 형태의 값이 들어오면 공백 기준으로 해당 속성의 id / name에 분리해 설정되도록 메소드를 사용한다.
* doTest( )에서는 유저모델 인스턴스를 생성하면서 ‘info’ 속성에 ‘test01 아이즈원’ 값만 할당한다. 이후 해당 인스턴스에 save( )로 저장하고 출력하면 해당 문서 객체의 id 와 name 에 각각의 값이 분리되어 자동 저장된다.
* [app5.js]

createUserSchema( ) 함수를 통해 스키마를 정의하는데, 이 때 password는 암호화된hashed\_password 속성으로 설정하고 암호화 과정에서 사용하는 String 타입 salt 속성을 추가한다.

* virtual 속성으로 password 속성을 추가해 \_password, salt, hashed\_password 값을 설정해준다. get메소드에서는 this.\_password를 리턴한다.