**Chapter 07. 익스프레스 프로젝트를 모듈화 하기**

* 사용자 정보 관련 기능을 모듈화
* [/database/user\_schema.js]

기존 DatabaseExample의 app5.js를 복사해 app.js를 생성한다.

그리고 그 내부의 스키마 생성 부분을 별도로 user\_schema.js로 생성해 분리한다. 기존에는 createUserSchema( ) 함수를 실행 시, 해당 함수 안에서 미리 선언했던 UserSchema 객체 안에 직접 스키마를 정의하고, virtual함수 및 기타 method를 선언했다.

* + - * 이 부분을 user\_schema.js 에서 실행하고 정의/선언된 스키마 관련 정보를 리턴하는 함수로 선언한다. app.js에서는 해당 user\_schema.js 내의 createSchema라는 함수를 실행함으로써 app.js 내부의 UserSchema 객체에 미리 선언된 스키마 객체가 할당되어, 종전과 같은 결과를 내놓는다.

[app.js]

function createUserSchema(){

// user\_schema.js 모듈 및 createSchema(mongoose) 함수 실행해 리턴값을

// 본 파일 UserSchema 객체에 할당

UserSchema = require('./database/user\_schema').createSchema(mongoose);

    // User 모델 정의

    UserModel = mongoose.model("users3", UserSchema);

    console.log('users3 정의함.');

}

* [/routes/user.js]

기존 app.js의 라우팅 관련 함수들을 분리한다. router.routes.post(….) 들을 별도 분리해준다.

router.route('/process/login').post(function(req, res){ … }  
router.route('/process/addUser').post(function(req, res){ … }  
router.route('/process/listuser').post(function(req, res){ … }

세 개의 기존 라우팅 함수를 /routes/user.js 내에 login, adduser, listuser 각각 익명 함수로서 별도로 정의해 준다.

var login = function(req, res){ … } 형태. login, adduser, listuser 익명함수 세 개 생성한 다음, 사용자 추가와 인증을 위한 authUser( ) 함수와 addUser( ) 함수도 추가해준다.

* + - * 마지막으로 module.exports.login / adduser / listuser 속성으로 login, adduser,listuser 를 할당해준다.
* **단, 이 파일 내에서는 데이터베이스 객체나 스키마 객체를 참조할 수 없다.**
  + - * 모듈 파일 안에서 ‘모듈을 불러오는 쪽’의 변수를 참조해야 할 필요성이 있다(데이터베이스, 스키마, 모델 객체는 user\_schema.js에 있다)
      * **익스프레스는 req(요청 객체)가 app 객체를 속성으로 갖고 있기 때문에** **메인 파일에서 모듈 파일로 객체를 전달 시에 app 객체에 속성으로 추가한 후 전달이 가능하다. 이 때 app.get / set ( ) 함수를 사용해 전달하고 불러온다.**