**MogoDB && Mongoose ORM**

**2021113490이현서**

일단 MongoDB는 NoSQL(비관계형) 데이터베이스로서, 데이터의 액세스 및 관리를 위해 다양한 데이터 모델을 사용합니다. 이러한 데이터베이스 유형은 큰 데이터 볼륨, 짧은 지연 시간과 유연한 데이터 모델이 필요한 애플리케이션에 최적화 되어 있습니다.

또한 Node에서 MongoDB를 사용하기 위해 mongose ORM을 사용할 수 있습니다.

MongoDB를 사용하기 위한 Mongose의 공식문서를 간단히 서술해 보도록 하겠습니다.

먼저 Defining your schema 저희의 스키마를 구축해야 합니다.

텍스트이(가) 표시된 사진

자동 생성된 설명

스키마에는 Db에 들어갈 속성의 타입을 지정해 줍니다.



그 후에는 다음과 같이 저희가 스키마에 해당하는 모델을 만들어야 합니다. 이는 저희의 MongoDB database에 대한 권한을 가지는 즉 query문을 작성할 수 있도록 도와줍니다.

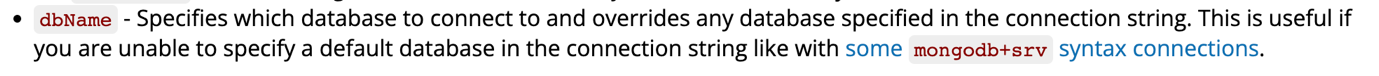
텍스트이(가) 표시된 사진

자동 생성된 설명

그 후에는 진짜로 MongoDB에 직접 접속해야 합니다.



기본적인 접속방법도 있지만 위와 같이 보다 정확하게 상세 정보를 적어줄 수도 있습니다.



그리고 옵션중에는 기본적으로 들어갈 db이름을 적을 속성도 있습니다.

mongoose.connect(

"mongodb://hyunseo:gustj486!!@localhost:27017",

{

dbName: "awp",

}

);

다음과 같이 저는 mongodb의 local에 접속하는 코드를 작성할 수 있었습니다.

그 다음에는 쿼리문인데

텍스트이(가) 표시된 사진

자동 생성된 설명

다음과 같이 매우 많은 쿼리문들이 있는 것을 볼 수 있는데, 대부분 각각의 사용방법이 비슷비슷하며 두번쨰 인자로는 결괏값을 받을 수 있는 콜백함수를 넣을 수 있습니다. 그 이유는 비동기 함수이기 때문입니다.

간단히 저는 쿼리문을 다음과 같이 적을 수 있었습니다.

(async () => {

console.log("Strat Querying ...");

*await* BlogPost.create(

{

title: "What is Lorem Ipsum",

body: "Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book. It has survived not only five centuries, but also the leap into electronic typesetting, remaining essentially unchanged. It was popularised in the 1960s with the release of Letraset sheets containing Lorem Ipsum passages, and more recently with desktop publishing software like Aldus PageMaker including versions of Lorem Ipsum.",

},

(error, blogPost) => {

console.log(error ? error : "", blogPost);

}

);

BlogPost.find(

{

title: "What is Lorem Ipsum",

},

(error, blogpost) => {

console.log(error ? error : "", blogpost);

}

);

BlogPost.find({}, (error, blogpost) => {

console.log(error ? error : "", blogpost);

});

BlogPost.find(

{

title: /Lorem/,

},

(error, blogpost) => {

console.log(error ? error : "", blogpost);

}

);

const id = "6282fab9c5222f2a2ac682a5";

BlogPost.findById(id, (error, blogpost) => {

console.log(error ? error : "", blogpost);

});

BlogPost.findByIdAndUpdate(

id,

{

title: "Updated title Lorem v2",

},

(error, blogpost) => {

console.log(error ? error : "", blogpost);

}

);

BlogPost.findByIdAndDelete(id, (error, blogpost) => {

console.log(error ? error : "", blogpost);

});

console.log("Done Querying");

})();

https://mongoosejs.com/docs/

**Ejs 기본 태그**

Ejs는 Embedded JavaScript의 약자로, 쉽게 말하면 자바스크립트가 내장되어 있는 html파일입니다. Html에도 자바스크립트를 넣을 수 있다고 의문이 생길 수 있지만, html안에 내장되어 있는 것과 바깥에서 건드리는 것의 차이다.

Ejs는 html의 태그처럼 자바스크립트의 내용을 삽입할 수 있게 됩니다. 매우 큰 장점인데, 이것을 통해 페이지를 동적으로 짜는 것이 수월해 집니다.

Ejs에는 자바스크립트를 내장 시킬 수 있는 2가지의 태그가 있다. 가장 기본이 되는 것은 <%%>입니다. 저 사이에 원하는 자바스크립트 내용을 넣으면 됩니다.

텍스트이(가) 표시된 사진

자동 생성된 설명

또한 변수값을 내장시킬 수 있습니다. <%=%> or <%-%> 이 태그의 역할은 해당 태그 안의 변수 값을 그대로 코드처럼 옮겨주는 것입니다. 동적으로 페이지를 만들 때 아주아주 편하게 해주는 기능입니다. 이 태그를 썼을 때는 변수의 자료형 이런거 생각하지 말고 그냥 코드로 들어가 있다고 생각해야 합니다.

텍스트이(가) 표시된 사진

자동 생성된 설명

이를 활용하여 서버에서 넘겨준 변수를 사용할 수도 있습니다.

텍스트이(가) 표시된 사진

자동 생성된 설명

텍스트이(가) 표시된 사진

자동 생성된 설명

두번째 파일 이름이 test2.ejs인데 변수 one과 two가 넘어가게 됩니다.

텍스트이(가) 표시된 사진

자동 생성된 설명

다음과 같은 결과가 나타나게 됩니다.

또한 작업을 하다보면 header나 footer등의 내용이 중복되는 경우가 발생합니다. 중복되는 코드를 하나하나 다 써준다면 변화가 생길때마다 해당하는 모든 코드를 찾아 고쳐줘야 합니다. 따라서 템플릿을 이용해서 이렇게 중복되는 코드를 모듈화하여 사용할 수 있게 됩니다.

사용 방법은 <% include “템플릿 파일 경로” %>입니다.

텍스트, 화면이(가) 표시된 사진

자동 생성된 설명

다음과 같이 템플릿을 다른 ejs파일로 부터 불러올 수 있는데 navbar.ejs를 다음과 같이 작성해 주면 된다.

텍스트, 모니터, 화면, 스크린샷이(가) 표시된 사진

자동 생성된 설명

다음과 같이 꼭 필요한 부분만 삽입하면 된다. 매우 편리한 기능이 아닐 수 없다.

https://yahohococo.tistory.com/43