Використання Mongoose

elegant [mongodb](https://www.mongodb.com/) object modeling for [node.js](https://nodejs.org/en/)

Посилання:

<https://mongoosejs.com/>

<https://mongoosejs.com/docs/guides.html>

<https://code.tutsplus.com/ru/articles/an-introduction-to-mongoose-for-mongodb-and-nodejs--cms-29527>

<https://metanit.com/web/nodejs/6.6.php>

Встановлення модуля:

npm install mongoose –save

Загальна схема роботи з mongoose

|  |  |
| --- | --- |
| Підключення модуля | const mongoose = require("mongoose"); |
| Підлючення до бази даних | mongoose.connect("<mongodb://localhost:27017/usersdb>", { useNewUrlParser: true }); |
| Маніпуляції з даними | * [Model.deleteMany()](https://mongoosejs.com/docs/api.html#model_Model.deleteMany) * [Model.deleteOne()](https://mongoosejs.com/docs/api.html#model_Model.deleteOne) * [Model.find()](https://mongoosejs.com/docs/api.html#model_Model.find) * [Model.findById()](https://mongoosejs.com/docs/api.html#model_Model.findById) * [Model.findByIdAndDelete()](https://mongoosejs.com/docs/api.html#model_Model.findByIdAndDelete) * [Model.findByIdAndRemove()](https://mongoosejs.com/docs/api.html#model_Model.findByIdAndRemove) * [Model.findByIdAndUpdate()](https://mongoosejs.com/docs/api.html#model_Model.findByIdAndUpdate) * [Model.findOne()](https://mongoosejs.com/docs/api.html#model_Model.findOne) * [Model.findOneAndDelete()](https://mongoosejs.com/docs/api.html#model_Model.findOneAndDelete) * [Model.findOneAndRemove()](https://mongoosejs.com/docs/api.html#model_Model.findOneAndRemove) * [Model.findOneAndUpdate()](https://mongoosejs.com/docs/api.html#model_Model.findOneAndUpdate) * [Model.replaceOne()](https://mongoosejs.com/docs/api.html#model_Model.replaceOne) * [Model.updateMany()](https://mongoosejs.com/docs/api.html#model_Model.updateMany) * [Model.updateOne()](https://mongoosejs.com/docs/api.html#model_Model.updateOne) |
| Від’єднання від бази даних | mongoose.disconnect(); |

Приклад.

|  |  |
| --- | --- |
| Підключення модуля | const mongoose = require("mongoose"); |
| Підключення до бази даних | mongoose.connect("<mongodb://localhost:27017/usersdb>", { useNewUrlParser: true }); |
| Операції з базою даних | const Schema = mongoose.Schema;  **// Створення схеми моделі**  const userScheme = new Schema({      name: String,      age: Number  });  **//Створення моделі**  const User = mongoose.model("User", userScheme);  **//Створення об’єкта моделі**  const user = new User({      name: "Bill",      age: 41  });  **//Збереження моделі і відключення від бази даних**  user.save(function(err){      mongoose.disconnect();  **// від’єднання від бази даних**        if(err) return console.log(err);      console.log("Збережено", user);  }); |
|  |  |

Задання схеми моделі

<https://mongoosejs.com/docs/guide.html>

|  |  |
| --- | --- |
| Загальна схема | Приклад |
| const схема = new Schema(  {  поле\_1 : специфікації\_поля ,  поле\_2 : специфікації\_поля ,  поле\_3 : специфікації\_поля ,  . . . . . . . . . . . .  }  ) | const Schema = mongoose.Schema;  const userScheme = new Schema({  name: String,  age: Number  }); |
| методи у схемах  схема.methods.метод=function(…) {…} | *// define a schema*  **var** animalSchema = **new** Schema({ name: String, type: String });  *// assign a function to the "methods" object of our animalSchema*  animalSchema.methods.findSimilarTypes = **function**(cb) {  **return** **this**.model('Animal').find({ type: **this**.type }, cb);  }; |
| віртуальні поля-властивості | personSchema.virtual('fullName').  get(**function**() { **return** **this**.name.first + ' ' + **this**.name.last; }).  set(**function**(v) {  **this**.name.first = v.substr(0, v.indexOf(' '));  **this**.name.last = v.substr(v.indexOf(' ') + 1);  });  axl.fullName = 'William Rose'; *// Now `axl.name.first` is "William"* |

Типи даних полів

|  |  |
| --- | --- |
| String |  |
| Number |  |
| Date |  |
| Buffer |  |
| Boolean |  |
| Mixed |  |
| Objectid |  |
| Array |  |
| Decimal128 |  |
| Map |  |
| Тип даних користувача | const userScheme = new Schema({  . . . . . .  company: {  name: String,  employee: [String], // тип – масив рядків  date: Date  }  }); |

Задання специфікації полів

При описі поля моделі можемо не тільки вказувати тип даних, а й додавати параметри валідації, зазначати обов’язковість поля та ін.

|  |  |  |
| --- | --- | --- |
| Тип даних | **type** | const userScheme = new Schema({      name: {          type: String      },      age: {          type: Number      }  }); |
| Значення за замовчуванням | **default** | const userScheme = new Schema({      name: {          type: String,          default: "NoName"      },      age: {          type: Number,          default: 22      }  }); |
| Валідація | * **required**: требует обязательного наличия значения для свойства | const userScheme = new Schema({      name: {          type: String,          required: true,          minlength:3,          maxlength:20      },      age: {          type: Number,          required: true,          min: 1,          max:100      }  }); |
|  | * **min** и **max**: задают минимальное и максимальное значения для числовых данных |
|  | * **minlength** и **maxlength**: задают минимальную и максимальную длину для строк |
|  | * **enum**: строка должна представлять одно из значений в указанном массиве строк | var RequirementSchema = new mongooseSchema({  status: {  type: String,  enum: ['NEW', 'STATUS'],  default: 'NEW'  },  }) |
|  | * **match**: строка должна соответствовать регулярному выражению |  |

Приклад.

|  |  |
| --- | --- |
|  |  |
|  | **var** breakfastSchema = **new** Schema({  eggs: {  type: Number,  min: [6, 'Too few eggs'],  max: 12  },  bacon: {  type: Number,  required: [true, 'Why no bacon?']  },  drink: {  type: String,  enum: ['Coffee', 'Tea'],  required: **function**() {  **return** **this**.bacon > 3;  }  },  phone: {  type: String,  validate: {  validator: **function**(v) {  **return** /\d{3}-\d{3}-\d{4}/.test(v);  },  message: props => `${props.value} is not a valid phone number!`  },  required: [true, 'User phone number required']  }  }); |

Маніпуляції з документами

|  |  |  |
| --- | --- | --- |
| Створення документу | об’єкт\_моделі . **save** | const user = new User({name: "Tom", age: 34});    user.save(function(err){      mongoose.disconnect();        if(err) return console.log(err);        console.log("Сохранен объект user", user);  }); |
|  | модель . **create** | const User = mongoose.model("User", userScheme);  User.create({name: "Tom", age: 34}, function(err, doc){. . .}) |
| Вибірка даних | * **find**: возвращает все объекты, которые соответствуют критерию фильтрации | User.find({name: "Tom"}, function(err, docs){      mongoose.disconnect();        if(err) return console.log(err);        console.log(docs);  }); |
| * **findById**: возвращает один объект по значению поля \_id | const id = "588a0f8c9151d23ce47bf98d";  User.findById(id, function(err, doc){      mongoose.disconnect();        if(err) return console.log(err);        console.log(doc);  }); |
| * **findOne**: возвращает один объект, который соответствует критерию фильтрации | User.findOne({name: "Tom"}, function(err, doc){      mongoose.disconnect();        if(err) return console.log(err);        console.log(doc);  }); |
| Видалення | **remove**  Видалення за шаблоном (усіх, які задовольняють) | const User = mongoose.model("User", userScheme);    User.remove({age:33}, function(err, result){      mongoose.disconnect();        if(err) return console.log(err);        console.log(result);  }); |
|  | **findOneAndDelete**  Видалення одного за шаблоном | User.findOneAndDelete({name:"Bill"}, function(err, doc){      mongoose.disconnect();        if(err) return console.log(err);        console.log("Удален пользователь ", doc);  }); |
|  | **findByIdAndDelete**  видадалення за id | var id = "588a0f8c9151d23ce47bf98d";  User.findByIdAndDelete(id, function(err, doc){      mongoose.disconnect();        if(err) return console.log(err);        console.log("Удален пользователь ", doc);  }); |
| Оновлення даних | **updateOne**  оновлення одного, що задовольняє шаблону | User.updateOne({name: "Tom"}, {name: "Tom Smith"}, function(err, result){        mongoose.disconnect();      if(err) return console.log(err);      console.log(result);  }); |
|  | **updateMany**  оновлення всіх, що задовольняють шаблону | User.updateMany({name: "Tom"}, {name: "Tom Smith"}, function(err, result){        mongoose.disconnect();      if(err) return console.log(err);      console.log(result);  }); |
|  | **findByIdAndUpdate**  оновлення документу, для якого знаємо id | const id = "588b2d6fc7d95503980ecf99";  User.findByIdAndUpdate(id, {name: "Sam", age: 25}, function(err, user){ // user – попереднє значення        mongoose.disconnect();      if(err) return console.log(err);      console.log("Обновленный объект", user);  });  //-----  User.findByIdAndUpdate(id, {name: "Bilbo Baggins", age: 111},  {**new: true**}, //у колбек передається оновлений документ  function(err, user){        mongoose.disconnect();      if(err) return console.log(err);      console.log("Обновленный объект", user);  }); |
|  |  | User.findOneAndUpdate({name: "Sam"}, {name: "Bill", age:24}, {new: true}, function(err, user){        mongoose.disconnect();      if(err) return console.log(err);      console.log("Обновленный объект", user);  }); |

Приклад.

|  |  |
| --- | --- |
|  |  |
| <!DOCTYPE html>  <html>  <head>      <meta charset="utf-8" />      <meta name="viewport" content="width=device-width" />      <title>Список пользователей</title>      <link href="<https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css>" rel="stylesheet" />      <script src="<https://code.jquery.com/jquery-2.2.4.min.js>"></script>  </head>  <body>      <h2>Список пользователей</h2>      <form name="userForm">          <input type="hidden" name="id" value="0" />          <div class="form-group">              <label for="name">Имя:</label>              <input class="form-control" name="name" />          </div>          <div class="form-group">              <label for="age">Возраст:</label>              <input class="form-control" name="age" />          </div>          <div class="panel-body">              <button type="submit" class="btn btn-sm btn-primary">Сохранить</button>              <a id="reset" class="btn btn-sm btn-primary">Сбросить</a>          </div>      </form>      <table class="table table-condensed table-striped table-bordered">          <thead><tr><th>Id</th><th>Имя</th><th>возраст</th><th></th></tr></thead>          <tbody>          </tbody>      </table>        <script>          // Получение всех пользователей          function GetUsers() {              $.ajax({                  url: "/api/users",                  type: "GET",                  contentType: "application/json",                  success: function (users) {                      var rows = "";                      $.each(users, function (index, user) {                          // добавляем полученные элементы в таблицу                          rows += row(user);                      })                      $("table tbody").append(rows);                   }              });          }          // Получение одного пользователя          function GetUser(id) {              $.ajax({                  url: "/api/users/"+id,                  type: "GET",                  contentType: "application/json",                  success: function (user) {                      var form = document.forms["userForm"];                      form.elements["id"].value = user.\_id;                      form.elements["name"].value = user.name;                      form.elements["age"].value = user.age;                  }              });          }          // Добавление пользователя          function CreateUser(userName, userAge) {              $.ajax({                  url: "api/users",                  contentType: "application/json",                  method: "POST",                  data: JSON.stringify({                      name: userName,                      age: userAge                  }),                  success: function (user) {                      reset();                      $("table tbody").append(row(user));                  }              })          }          // Изменение пользователя          function EditUser(userId, userName, userAge) {              $.ajax({                  url: "api/users",                  contentType: "application/json",                  method: "PUT",                  data: JSON.stringify({                      id: userId,                      name: userName,                      age: userAge                  }),                  success: function (user) {                      reset();                      console.log(user);                      $("tr[data-rowid='" + user.\_id + "']").replaceWith(row(user));                  }              })          }            // сброс формы          function reset() {              var form = document.forms["userForm"];              form.reset();              form.elements["id"].value = 0;          }            // Удаление пользователя          function DeleteUser(id) {              $.ajax({                  url: "api/users/"+id,                  contentType: "application/json",                  method: "DELETE",                  success: function (user) {                      console.log(user);                      $("tr[data-rowid='" + user.\_id + "']").remove();                  }              })          }          // создание строки для таблицы          var row = function (user) {              return "<tr data-rowid='" + user.\_id + "'><td>" + user.\_id + "</td>" +                     "<td>" + user.name + "</td> <td>" + user.age + "</td>" +                     "<td><a class='editLink' data-id='" + user.\_id + "'>Изменить</a> | " +                      "<a class='removeLink' data-id='" + user.\_id + "'>Удалить</a></td></tr>";          }          // сброс значений формы          $("#reset").click(function (e) {                e.preventDefault();              reset();          })            // отправка формы          $("form").submit(function (e) {              e.preventDefault();              var id = this.elements["id"].value;              var name = this.elements["name"].value;              var age = this.elements["age"].value;              if (id == 0)                  CreateUser(name, age);              else                  EditUser(id, name, age);          });            // нажимаем на ссылку Изменить          $("body").on("click", ".editLink", function () {              var id = $(this).data("id");              GetUser(id);          })          // нажимаем на ссылку Удалить          $("body").on("click", ".removeLink", function () {              var id = $(this).data("id");              DeleteUser(id);          })            // загрузка пользователей          GetUsers();      </script>  </body>  </html> | const mongoose = require("mongoose");  const express = require("express");  const Schema = mongoose.Schema;  const app = express();  const jsonParser = express.json();    const userScheme = new Schema({name: String, age: Number}, {versionKey: false});  const User = mongoose.model("User", userScheme);    app.use(express.static(\_\_dirname + "/public"));    mongoose.connect("<mongodb://localhost:27017/usersdb>", { useNewUrlParser: true }, function(err){      if(err) return console.log(err);      app.listen(3000, function(){          console.log("Сервер ожидает подключения...");      });  });    app.get("/api/users", function(req, res){        User.find({}, function(err, users){            if(err) return console.log(err);          res.send(users)      });  });    app.get("/api/users/:id", function(req, res){        const id = req.params.id;      User.findOne({\_id: id}, function(err, user){            if(err) return console.log(err);          res.send(user);      });  });    app.post("/api/users", jsonParser, function (req, res) {        if(!req.body) return res.sendStatus(400);        const userName = req.body.name;      const userAge = req.body.age;      const user = new User({name: userName, age: userAge});        user.save(function(err){          if(err) return console.log(err);          res.send(user);      });  });    app.delete("/api/users/:id", function(req, res){        const id = req.params.id;      User.findByIdAndDelete(id, function(err, user){            if(err) return console.log(err);          res.send(user);      });  });    app.put("/api/users", jsonParser, function(req, res){        if(!req.body) return res.sendStatus(400);      const id = req.body.id;      const userName = req.body.name;      const userAge = req.body.age;      const newUser = {age: userAge, name: userName};        User.findOneAndUpdate({\_id: id}, newUser, {new: true}, function(err, user){          if(err) return console.log(err);          res.send(user);      });  }); |