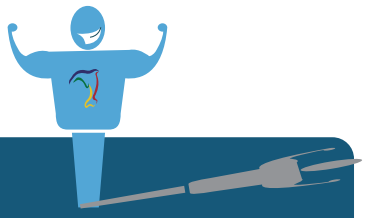
autentia

Avenida de Castilla,1 - Edificio Best Point - Oficina 21B
28830 San Fernando de Henares (Madrid)
tel./fax: +34 91 675 33 06
info@autentia.com - **www.autentia.com**

¿**Qué ofrece** Autentia Real Business Solutions S.L?

Somos su empresa de **Soporte a Desarrollo Informático**.
Ese apoyo que siempre quiso tener...

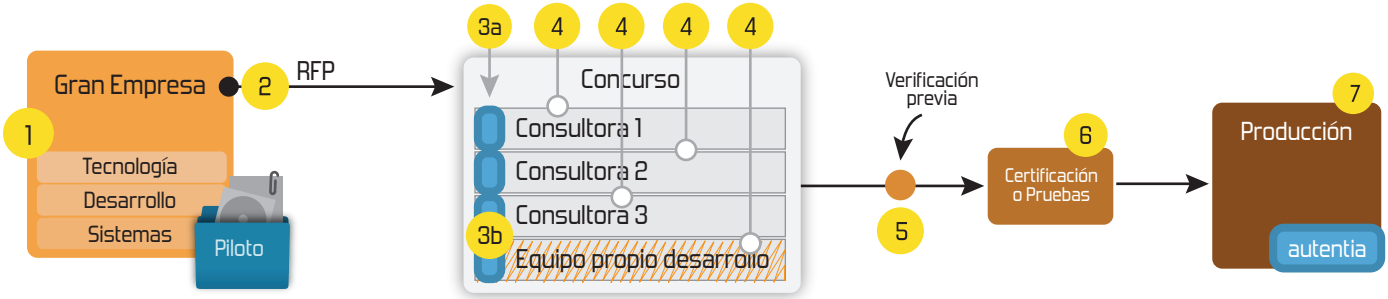
1. Desarrollo de componentes y proyectos a medida



2. Auditoría de código y recomendaciones de mejora

3. Arranque de proyectos basados en nuevas tecnologías

1. Definición de frameworks corporativos.
2. Transferencia de conocimiento de nuevas arquitecturas.
3. Soporte al arranque de proyectos.
4. Auditoría preventiva periódica de calidad.
5. Revisión previa a la certificación de proyectos.
6. Extensión de capacidad de equipos de calidad.
7. Identificación de problemas en producción.



4. Cursos de formación (impartidos por desarrolladores en activo)

Spring MVC, JSF-PrimeFaces /RichFaces, HTML5, CSS3, JavaScript-jQuery

Gestor portales (Liferay)
Gestor de contenidos (Alfresco)
Aplicaciones híbridas

Control de autenticación y acceso (Spring Security)
UDDI
Web Services
Rest Services
Social SSO
SSO (Cas)

JPA-Hibernate, MyBatis
Motor de búsqueda empresarial (Solr)
ETL (Talend)

Dirección de Proyectos Informáticos.
Metodologías ágiles
Patrones de diseño
TDD

Tareas programadas (Quartz)
Gestor documental (Alfresco)
Inversión de control (Spring)

BPM (jBPM o Bonita)
Generación de informes (JasperReport)
ESB (Open ESB)

Compartimos nuestro conocimiento en:
www.adictosaltrabajo.com

Para más información visítenos en:
www.autentia.com

AdictosAlTrabajo
Terrakas 1x04 ¡¡Ya está en la web!! ;-)
terrakas.com
autentia Soporte a desarrollo informático
Hosting patrocinado por enredados
Entra en Adictos a través de
E-mail
Contraseña
Entrar
Deseo registrarme
Olvidé mi contraseña
Inicio Quiénes somos Formación Comparador de salarios Nuestro libro Más
» Estás en: Inicio Tutoriales Node, Express y MongoDB: Crea un API REST en JavaScript server-side de form...



Paulino Corujo

Java Developer & Javascript geek. Passionate about development.

Contacta en: LinkedIn y en Twitter: @pcorujo

Ver todos los tutoriales del autor

MÁSTER INFOARQUITECTURA & INTERIOR 3D animum

Fecha de publicación del tutorial: 2009-02-26 Tutorial visitado 3 veces Descargar en PDF

Node, Express y MongoDB: Crea un API REST en JavaScript server-side de forma rápida, sencilla y eficiente

Índice

1. Introducción
2. Manos a la obra
3. Conclusiones

Introducción

¿Quieres construir un API REST de forma rápida, sencilla y con un alto rendimiento? Esta es tu combinación: NodeJS, Express y MongoDB. En este tutorial aprenderás los conceptos básicos de estas tecnologías y frameworks, que te ayudarán a construir una arquitectura robusta para desarrollar un API REST sobre JavaScript en la parte servidora.

¿Por qué JavaScript? A medida que los motores JavaScript se han hecho más y más eficientes, especialmente desde la aparición del motor V8 desarrollado por Google y su compilador JIT a código nativo, JavaScript se ha convertido en una alternativa cada vez más extendida de la nueva generación de aplicaciones web, no sólo en la parte cliente sino en la parte servidora, gracias también a los motores JavaScript tipo RingoJS o NodeJS, basados respectivamente en las implementaciones de ECMAScript Rhino, de Mozilla y V8 de Google.

Manos a la obra

Para seguir este tutorial necesitas:

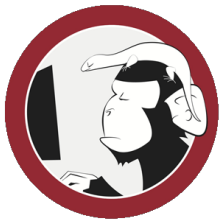
- NodeJS: Es una plataforma basada en el runtime JavaScript de Google Chrome, sigue un modelo de entrada/salida orientado a eventos y no bloqueante, de ahí su eficiencia. La arquitectura de NodeJS consiste en un hilo principal sirviendo a todos los usuarios y una serie de hilos para realizar operaciones de entrada/salida de forma asíncrona, con lo que la conexión del cliente (o el hilo principal de ejecución) no está asociado a los hilos de entrada/salida, optimizando los recursos del sistema que lo alberga (NodeJS).
- Express: Es un framework para construir aplicaciones web sobre NodeJS inspirado en el framework Sinatra de Ruby (Express).
- Mongoose: Se trata de un ODM (Object Data Mapper) en JavaScript diseñado para aplicaciones que usen MongoDB sobre NodeJS. Posee un API intuitivo para definir los modelos de datos y el almacenamiento asíncrono (Mongoose ODM).
- MongoDB: Es una base de datos opensource NoSQL, destaca por su alto rendimiento y escalabilidad. Es orientada a documentos de estilo JSON y posee características interesantes como el Auto-Sharding o alta disponibilidad y replicación entre otras (MongoDB).

Una vez instalado NodeJS y MongoDB (recomiendo usar una cliente con un interfaz gráfico para MongoDB), crearemos un directorio para el proyecto y en su raíz, el fichero package.json que definirá el paquete npm que crearemos para construir el proyecto. Dicho fichero contiene metadatos del propio proyecto, así como las dependencias con otras librerías:

```
view plain print ?
01. {
02.   "name": "node-express-mongoose-rest-sampleapp",
03.   "version": "0.0.1",
04.   "scripts": {
05.     "start": "node app"
06.   },
07.   "dependencies": {
08.     "express": "*",
```

Catálogo de servicios Autentia





Síguenos a través de:

Últimas Noticias

» Mi retrospectiva sobre la CAS 2012

» Autentia estuvo en el Duatlón de Torrejón de Ardoz

» Participamos en la Carrera de las Empresas 2012

» ¡¡¡Terrakas 1x04 recién salido del horno!!!

» Estreno Terrakas 1x04: "Terraka por un día"

Histórico de noticias

Últimos Tutoriales

» Manipulando Tablas de datos en IOS con StoryBoards usando UML y TDD

» Servicios REST documentados y probados con Swagger

» Insertando contenidos HTML en nuestra aplicación IOS

» Integrar Barcode Scanner en nuestra aplicación Android

» Tutorial de iniciación en

Impulsores Comunidad ¿Ayuda?

---- sin clicks

0 personas han traído clicks a esta página

+ + + + + + + +

powered by karmacracy

```
    "mongoose":"*"
  }
}
```

Mediante esta definición, declaramos las dependencias con las librerías Express y Mongoose que usaremos para construir nuestro API REST. Una vez creado el fichero ejecutaremos el comando **npm install** en el mismo directorio donde tenemos el fichero package.json, instalándose de forma local los paquetes Express y Mongoose para NodeJS.

```
express@3.0.2 node_modules/express
├── methods@0.0.1
├── fresh@0.1.0
├── cookie-signature@0.0.1
├── range-parser@0.0.4
├── cookie@0.0.4
├── crc@0.2.0
├── commander@0.6.1
├── debug@0.7.0
├── mkdirp@0.3.3
├── send@0.1.0 (mime@1.2.6)
└── connect@2.6.2 (pause@0.0.1, bytes@0.1.0, formidable@1.0.11, qs@0.5.1)

mongoose@3.3.1 node_modules/mongoose
├── sliced@0.0.3
├── hooks@0.2.1
├── ms@0.1.0
└── mongodb@1.1.11 (bson@0.1.5)
```

Con esto ya tenemos todo lo necesario para empezar nuestra aplicación. Comenzaremos creando en la raiz de nuestro proyecto el fichero **app.js**, que previamente hemos definido en el fichero package.json como el JavaScript principal a ejecutar por node.

view plain print ?

```
//Load app dependencies
var express = require('express'),
  mongoose = require('mongoose'),
  http = require('http');
var app = express();

//Configure: bodyParser to parse JSON data
//           methodOverride to implement custom HTTP methods
//           router to crete custom routes
app.configure(function(){
  app.use(express.bodyParser());
  app.use(express.methodOverride());
  app.use(app.router);
});

app.configure('development', function(){
  app.use(express.errorHandler());
});

//Sample routes are in a separate module, just for keep the code clean
routes = require('./routes/router')(app);

//Connect to the MongoDB test database
mongoose.connect('mongodb://localhost/test_database');

//Start the server
http.createServer(app).listen(8080);
```

La base de datos que usaremos (en el ejemplo test_database) se creará en MongoDB una vez iniciemos la aplicación.

Ahora crearemos una definión del recurso que expondremos en nuestro API REST, en el ejemplo crearemos un esquema **Person** en el fichero **models/person.js** y lo exportaremos como un modelo Mongoose en NodeJS:

view plain print ?

```
//Sample Mongoose Schema (Person class)
var mongoose = require('mongoose'),
    Schema = mongoose.Schema;

var personSchema = new Schema({
    name: String,
    lastName: String
});

//Export the schema
module.exports = mongoose.model('Person', personSchema);
```

Nuestro modelo **Person** contiene dos atributos, nombre y apellido definidos en un esquema Mongoose.

Por último, necesitamos enlazar las operaciones sobre el recurso **Person** con los métodos HTTP. Esto lo haremos creando un **router** de Express en el fichero **routes/router.js**:

view plain print ?

```
//App routes
module.exports = function(app){

    var Person = require('../models/person');

    //Create a new Person and save it
```

Framework ZK

Últimas ofertas de empleo

2011-09-08
Comercial - Ventas - MADRID.

2011-09-03
Comercial - Ventas - VALENCIA.

2011-08-19
Comercial - Compras - ALICANTE.

2011-07-12
Otras Sin catalogar - MADRID.

2011-07-06
Otras Sin catalogar - LUGO.

```
    person = function(req, res){
        var person = new Person({name: req.body.name, lastName: req.body.lastName});
        person.save();
        res.end();
    };

    //find all people
    list = function(req, res){
        Person.find(function(err, people) {
            res.send(people);
        });
    };

    //find person by id
    find = (function(req, res) {
        Person.findOne({_id: req.params.id}, function(error, person) {
            res.send(person);
        })
    });

    //Link routes and functions
    app.post('/person', person);
    app.get('/person', list);
    app.get('/person/:id', find);
}
```

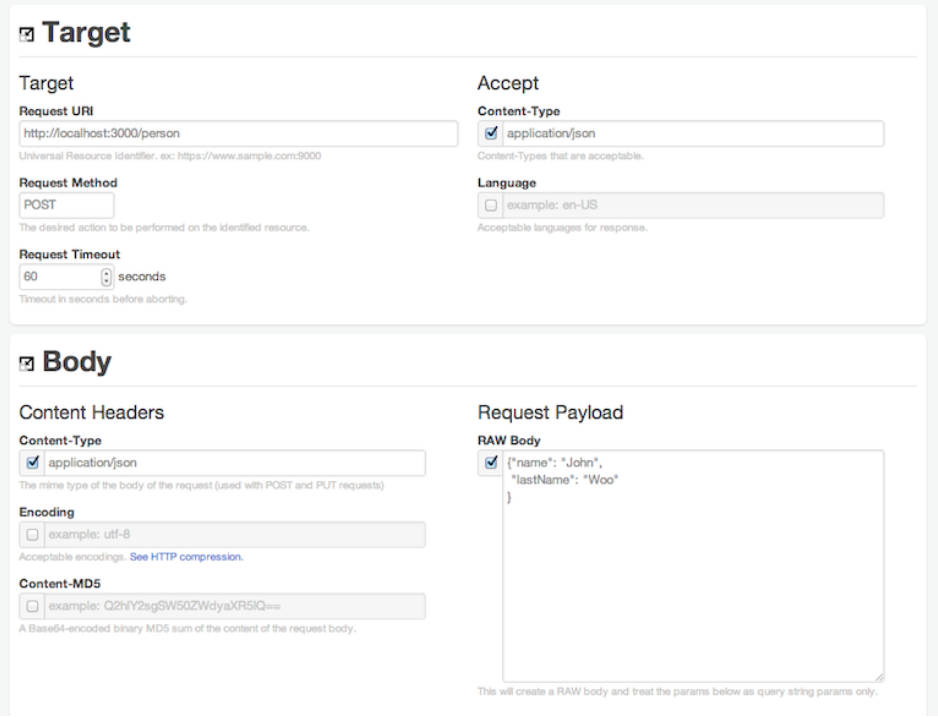
En este caso nuestro API tendrá tres operaciones sobre nuestro recurso **Persona**:

- Creación de nueva persona: POST /person
- Listado de personas: GET /person
- Lista de persona por Id: GET/person/:id

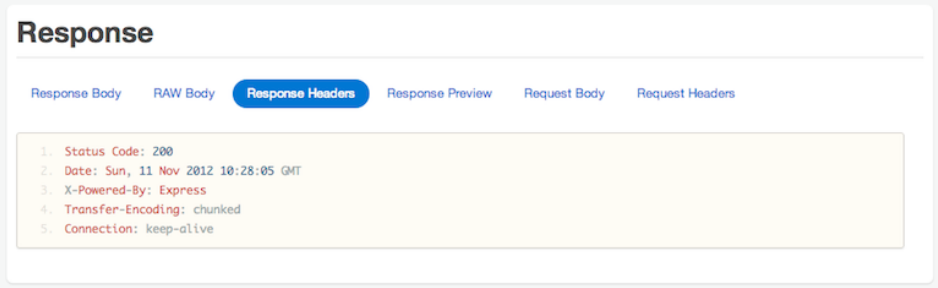
Una vez creadas las operaciones, podremos iniciar nuestra aplicación, para ello levantaremos MongoDB y cargaremos el fichero **app.js** en NodeJS usando el comando **node app.js**. Nuestra aplicación levanta un servidor HTTP escuchando en el puerto 8080 y que ejecutará las operaciones según nuestro router recien creado.

Para probar nuestro API REST usaremos REST Console, un plugin de Chrome muy completo para depurar las llamadas al API.

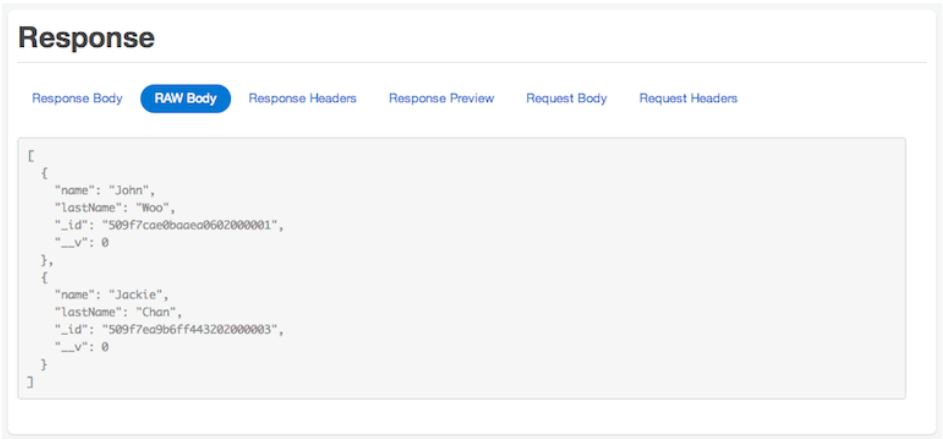
Para crear una nueva **Persona** haremos una llamada POST /person con los siguientes datos:



Si la creación del nuevo recurso **Persona** ha ido bien obtendremos un código HTTP 200 OK:

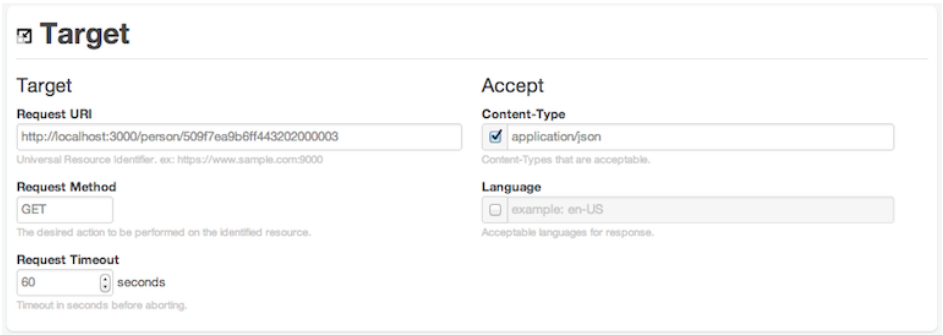


Para consultar los recursos creados usaremos GET /person:

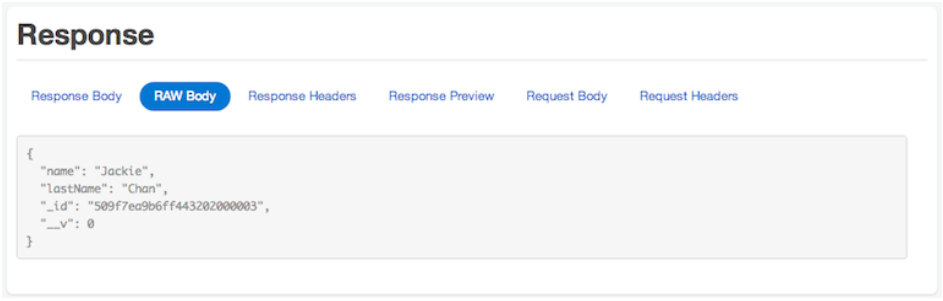


Esta consulta retornará una colección de personas, en este caso dos, con sus respectivos identificadores que usaremos en la siguiente consulta.

Para obtener un recurso por **id** haremos una llamada GET /person/:id. Por ejemplo: GET /person/509f7ea9b6ff443202000003



Obteniendo el recurso solicitado:



Si queremos consultar o realizar otras operaciones directamente en MongoDB sobre los recursos creados, podemos usar el shell usando el siguiente comando **mongo localhost/test_database** para después hacer, por ejemplo, una consulta a la **coleccion de Persona** creada:

```
tutorial$ mongo localhost/test_database
MongoDB shell version: 2.2.1
connecting to: localhost/test_database
> db.people.find();
{ "name" : "John", "lastName" : "Woo", "_id" : ObjectId("509f7cae0baaea0602000001"), "__v" : 0 }
{ "name" : "Jackie", "lastName" : "Chan", "_id" : ObjectId("509f7ea9b6ff443202000003"), "__v" : 0 }
>
```

Conclusiones

NodeJS, Express y MongoDB es una buena combinación para construir un API REST de forma sencilla, rápida y eficiente en un entorno escalable. La modularización del estándar CommonJS soportada por NodeJS permite construir código bien organizado. El framework Express nos facilita la tarea de construir un patrón MVC sobre NodeJS. MongoDB, mediante el ODM Mongoose, nos proporciona una forma muy intuitiva de almacenar y consultar los documentos JSON.

Si quieres descargar el código, puedes hacerlo en: https://bitbucket.org/pcorujo/node-express-mongoose-rest-sampleapp/pull-requests.

A continuación puedes evaluarlo:

Regístrate para evaluarlo

Por favor, vota +1 o compártelo si te pareció interesante

Share | 0

Anímate y coméntanos lo que pienses sobre este **TUTORIAL:**

» Registrate y accede a esta y otras ventajas **«**

SOME RIGHTS RESERVED Esta obra está licenciada bajo licencia Creative Commons de Reconocimiento-No comercial-Sin obras derivadas 2.5

Copyright 2003-2012 © All Rights Reserved | Texto legal y condiciones de uso | Banners | Powered by Autentia | Contacto

XHTML 1.0 CSS RSS ATOM