MONASH University

FIT2095 e-Business software technologies S2 2019

Dashboard / My units / FIT2095_S2_2019 / Week 6 (2 Sep - 8 Sep) / Week 6: Workshop Quiz

Started on	Friday, 6 September 2019, 12:24 PM
State	Finished
Completed on	Friday, 6 September 2019, 12:47 PM
Time taken	22 mins 20 secs
Grade	**8.97** out of 10.00 (**90**%)

Print friendly format

Question **1**
Complete
Mark 8.97 out of 10.00

Question 1

(Q1)

1. MongoDB is a denormalized NoSQL database. This makes it inherently schema-less as documents have varying sets of fields with different data types. Mongoose defines a schema for your data models so your documents follow a specific structure with pre-defined data types.
2. Mongoose has built-in validation for schema definitions. This saves us from writing a bunch of validation code that we will have to otherwise write with the MongoDB driver. By simply including things like required: true in our schema definitions, Mongoose provides out-of-the-box validations for our collections (including data types).
3. Mongoose provides optional pre and post save operations for data models. This makes it easy to define hooks and custom functionality on successful reads/writes etc. We can also define custom methods that act on a particular instance (or document). While we can achieve similar functionality with the native MongoDB driver, Mongoose makes it easier to define and organize such methods within our schema definition.

(Q2)

(ORM) is a technique that lets us query and manipulates data from a database using an object-oriented paradigm. An ORM library is a completely ordinary library written in our language of choice that encapsulates the code needed to manipulate the data, so we don't use SQL anymore; we interact directly with an object in the same language we're using.

Comment:
Q1=69% Q2=90% Q3=100%

Question **2**

Complete

Not graded

Question 2

```
let authorSchema = mongoose.Schema({

    _id: mongoose.Schema.Types.ObjectId,

    from: {
        type: String,
        required: true,
        default: "Sydney"
    },

    to: {
        type: String,
        required: true,
    },

    airline: {
        type: String,
        required: true,
    },

    cost: {
        type: Number,
        validate:
        {
            validator: function (Value) {
                return Value >= 0;
            },
            message: 'Value should be positive.'
        }
    }
});
```

Question **3**
Complete
Not graded

Question 3

```
const mongoose = require('mongoose');
const Travel = require('./models/travelschema');

let url='mongodb://128.0.15.66:885566/Travel';

mongoose.connect(url, function (err) {

    if (err)
    {
        console.log('Error in Mongoose connection');
        throw err;
    }

    console.log('Successfully connected');

    let instance = new Travel({
        _id: new mongoose.Types.ObjectId(),
        from: "MEL",
        to: "JNB",
        airline: "VA",
        cost: 2500 });

    instance.save(function (err) {
        if (err) throw err;
        console.log("Instance added successfully !");
    });

});
```

Question **4**
Not answered
Not graded

Question 4

Question **5**
Not answered
Not graded

Question 5

Question **6**
Not answered
Not graded

Question 6

Question **7**
Not answered
Not graded

Question 7

Question **8**

Not answered

Not graded

Question 8

◀ Supplementary Material: Node.js +MongoDB on Multiple VMs

Jump to...

Week 6 Pre-Reading Quiz ▶