Lesson 5: HTTP and MongoDB Connectivity

Node.js can be used in database applications.

One of the most popular NoSQL database is MongoDB.

MongoDB

To be able to experiment with the code examples, you will need access to a MongoDB database.

You can download a free MongoDB database at https://www.mongodb.com.

Node.js can use this module to manipulate MongoDB databases:

```
var mongo = require('mongodb');
```

Creating a Database

To create a database in MongoDB, start by creating a MongoClient object, then specify a connection URL with the correct ip address and the name of the database you want to create.

MongoDB will create the database if it does not exist, and make a connection to it.

```
var MongoClient = require('mongodb').MongoClient;
var url = "mongodb://localhost:27017/pes";

MongoClient.connect(url, function(err, db) {
  if (err) throw err;
  console.log("Database created!");
  db.close();
});
```

In MongoDB, a database is not created until it gets content!

MongoDB waits until you have created a collection (table), with at least one document (record) before it actually creates the database (and collection).

A collection in MongoDB is the same as a table in MySQL

Creating a Collection

To create a collection in MongoDB, use the `Collection()` method:

```
var MongoClient = require('mongodb').MongoClient;
var url = "mongodb://localhost:27017/";

MongoClient.connect(url, function(err, db) {
```

```
  if (err) throw err;
  var dbo = db.db("pes");
  dbo.createCollection("students", function(err, res) {
    if (err) throw err;
    console.log("Collection created!");
    db.close();
  });
});
```

In MongoDB, a collection is not created until it gets content.

MongoDB waits until you have inserted a document before it actually creates the collection.

Insert a single document Into Collection

To insert a record, or *document* as it is called in MongoDB, into a collection, we use the insertOne() method.

A document in MongoDB is the same as a record in MySQL

The first parameter of the insertOne() method is an object containing the name(s) and value(s) of each field in the document you want to insert.

It also takes a callback function where you can work with any errors, or the result of the insertion:

```
var MongoClient = require('mongodb').MongoClient;
var url = "mongodb://localhost:27017/";

MongoClient.connect(url, function(err, db) {
  if (err) throw err;
  var dbo = db.db("pes");
  var myobj = { name: "Ria", id: "001" };
  dbo.collection("students").insertOne(myobj, function(err, res) {
    if (err) throw err;
    console.log("1 document inserted");
    db.close();
  });
});
```

Insert Multiple Documents into a collection

To insert multiple documents into a collection in MongoDB, we use the insertMany() method.

The first parameter of the insertMany() method is an array of objects, containing the data you want to insert.

It also takes a callback function where you can work with any errors, or the result of the insertion:

```
Insert multiple documents in the "customers" collection:
var MongoClient = require('mongodb').MongoClient;
var url = "mongodb://localhost:27017/";

MongoClient.connect(url, function(err, db) {
  if (err) throw err;
  var dbo = db.db("pes");
  var myobj = [
    { name: Ria, id: '001'},
    { name: 'Arun', address: '002'},
    { name: 'Vivek', address: '003'},

  ];
  dbo.collection("students").insertMany(myobj, function(err, res) {
    if (err) throw err;
    console.log("Number of documents inserted: " + res.insertedCount);
    db.close();
  });
});
```

The _id Field

If you do not specify an _id field, then MongoDB will add one for you and assign a unique id for each document.

MongoDB assigns a unique _id for each document. The value must be unique for each document:

Select the documents from collection:

In MongoDB we use the **find** and **findOne** methods to find data in a collection.

Just like the **SELECT** statement is used to find data in a table in a MySQL database.

Find:

To select data from a table in MongoDB, we can also use the find() method.

The find() method returns all occurrences in the selection.

The first parameter of the find() method is a query object.

```
Find all documents in the customers collection:

var MongoClient = require('mongodb').MongoClient;
var url = "mongodb://localhost:27017/";

MongoClient.connect(url, function(err, db) {
  if (err) throw err;
  var dbo = db.db("pes");
```

```
  dbo.collection("students").find({}).toArray(function(err, result) {
    if (err) throw err;
    console.log(result);
    db.close();
  });
});
```