0001 - MongoDB 设置

MacOS Installation

```
$ brew install mongodb
```

Linux Installation

https://docs.mongodb.com/master/administration/install-on-linux/

Windows Installation

https://docs.mongodb.com/master/tutorial/install-mongodb-on-windows/

package.json

```
"devDependencies": {
....
...
  "ethereumjs-util": "5.1.2",
  "mongoose": "4.11.7"
}
```

```
$ npm install
```

0002 - Product Definition

product.js

```
var mongoose = require('mongoose');
mongoose.Promise = global.Promise;

var Schema = mongoose.Schema;

var ProductSchema = new Schema({
 blockchainId: Number,
 name: String,
 category: String,
 ipfsImageHash: String,
 ipfsDescHash: String,
 auctionStartTime: Number,
 auctionEndTime: Number,
 price: Number,
 condition: Number,
 productStatus: Number
});

var ProductModel = mongoose.model('ProductModel', ProductSchema);

module.exports = ProductModel;
```

0003 - NodeJS app 设置

package.json

```
"devDependencies": {
....
...
  "express": "4.15.4",
  "nodemon": "^1.11.0"
}
```

```
$ npm install
```

server.js

```
var express = require('express');
var app = express();

app.listen(3000, function() {
 console.log('Ebay Ethereum server listening on port 3000!');
});

app.get('/', function(req, res) {
 res.send("Hello, World!");
});
```

start nodemon

```
$ node_modules/.bin/nodemon server.js
```

访问： http://localhost:3000

0004 - Solidity Events

事件声明

```
event NewProduct(uint _productId, string _name, string _category, string _im
ageLink, string _descLink,
  uint _auctionStartTime, uint _auctionEndTime, uint _startPrice, uint _prod
uctCondition);
```

触发事件

```
NewProduct(productIndex, _name, _category, _imageLink, _descLink, _auctionSt
artTime, _auctionEndTime, _startPrice, _productCondition);
```

事件监听

```


 EcommerceStore.deployed().then(function(i) {
  productEvent = i.NewProduct({fromBlock: 0, toBlock: 'latest'});

  productEvent.watch(function(err, result) {
   if (err) {
    console.log(err)
    return;
   }
   saveProduct(result.args);
  });
```

0005 - Store Product

在合约中添加事件逻辑

```
.....
.....
function EcommerceStore() {
  productIndex = 0;
}
event NewProduct(uint _productId, string _name, string _category, string _imageLink, string _descLink, uint _auctionStartTime, uint _auctionEndTime, uint _startPrice, uint _productCondition);
.....
.....
function addProductToStore(string _name, string _category, string _imageLink, string _descLink, uint _auctionStartTime, uint _auctionEndTime, uint _startPrice, uint _productCondition) {
.....
.....
  NewProduct(productIndex, _name, _category, _imageLink, _descLink, _auctionStartTime, _auctionEndTime, _startPrice, _productCondition);
}
```

server.js

```
var ecommerce_store_artifacts = require('./build/contracts/EcommerceStore.json')
var contract = require('truffle-contract')
var Web3 = require('Web3')
var provider = new Web3.providers.HttpProvider("http://localhost:8545");
var EcommerceStore = contract(ecommerce_store_artifacts);
EcommerceStore.setProvider(provider);

//Mongoose setup to interact with the mongodb database
var mongoose = require('mongoose');
mongoose.Promise = global.Promise;
var ProductModel = require('./product');
mongoose.connect("mongodb://localhost:27017/ebay_dapp");
var db = mongoose.connection;
db.on('error', console.error.bind(console, 'MongoDB connection error:'));
```

```
// Express server which the frontend with interact with
var express = require('express');
var app = express();

app.use(function(req, res, next) {
 res.header("Access-Control-Allow-Origin", "*");
 res.header("Access-Control-Allow-Headers", "Origin, X-Requested-With, Conte
nt-Type, Accept");
 next();
});

app.listen(3000, function() {
 console.log('Ebay Ethereum server listening on port 3000!');
});

function setupProductEventListner() {
 let productEvent;
 EcommerceStore.deployed().then(function(i) {
  productEvent = i.NewProduct({fromBlock: 0, toBlock: 'latest'});

  productEvent.watch(function(err, result) {
   if (err) {
    console.log(err)
    return;
   }
   saveProduct(result.args);
  });
 })
}

setupProductEventListner();

function saveProduct(product) {
 ProductModel.findOne({ 'blockchainId': product._productId.toLocaleString()
}, function (err, dbProduct) {

  if (dbProduct != null) {
   return;
  }

  var p = new ProductModel({name: product._name, blockchainId: product._prod
uctId, category: product._category,
   ipfsImageHash: product._imageLink, ipfsDescHash: product._descLink, aucti
onStartTime: product._auctionStartTime,
   auctionEndTime: product._auctionEndTime, price: product._startPrice, cond
ition: product._productCondition,
   productStatus: 0});
```

```
    p.save(function (err) {
     if (err) {
      handleError(err);
     } else {
      ProductModel.count({}, function(err, count) {
       console.log("count is " + count);
      })
     }
    });
   })
  }
```

Testing

启动truffle控制台，提交数据到区块链，检查是否有数据插入到数据库中。

```
$ mongo
```

MongoDB Console

```
> show dbs;
> use ebay_dapp;
> db.ebay_dapp.find({})
>
```

0006 - View Products

app.js

```
function renderProducts(div, filters) {
 $.ajax({
  url: offchainServer + "/products",
  type: 'get',
  contentType: "application/json; charset=utf-8",
  data: filters
 }).done(function(data) {
  if (data.length == 0) {
   $("#" + div).html('No products found');
  } else {
   $("#" + div).html('');
  }
  while(data.length > 0) {
   let chunks = data.splice(0, 4);
   let row = $("<div/>");
   row.addClass("row");
   chunks.forEach(function(value) {
    let node = buildProduct(value);
    row.append(node);
   })
   $("#" + div).append(row);
  }
 })
}

function renderStore() {
 renderProducts("product-list", {});
 renderProducts("product-reveal-list", {productStatus: "reveal"});
 renderProducts("product-finalize-list", {productStatus: "finalize"});
 categories.forEach(function(value) {
  $("#categories").append("<div>" + value + "");
 })
}
```

server.js

```
app.get('/products', function(req, res) {
 current_time = Math.round(new Date() / 1000);
 query = { productStatus: {$eq: 0} }

 if (Object.keys(req.query).length === 0) {
  query['auctionEndTime'] = {$gt: current_time}
 } else if (req.query.category !== undefined) {
  query['auctionEndTime'] = {$gt: current_time}
  query['category'] = {$eq: req.query.category}
 } else if (req.query.productStatus !== undefined) {
  if (req.query.productStatus == "reveal") {
   query['auctionEndTime'] = {$lt: current_time, $gt: current_time - (60*60)
}
  } else if (req.query.productStatus == "finalize") {
   query['auctionEndTime'] = { $lt: current_time - (60*60) }
   query['productStatus'] = {$eq: 0}
  }
 }

 ProductModel.find(query, null, {sort: 'auctionEndTime'}, function (err, ite
ms) {
  console.log(items.length);
  res.send(items);
 })
});
```

0007 - 思考

下面有几个我们值得思考的问题：

1. 当交易成功后，给裁判1%的手续费。
2. 目前在我们系统中，任何人都可以成为仲裁者，我们增加一个限制，只有超哥5个eth以上的人才能成为仲裁者。
3. 如果你发现仲裁者和卖家或者买家之间存在勾结的行为，那么添加一个销毁仲裁者存款的功能。
4. 目前，卖家没有收到任何买家给他们评论的功能。实现买方给卖方评级的功能。