

User Guide

Prerequisites

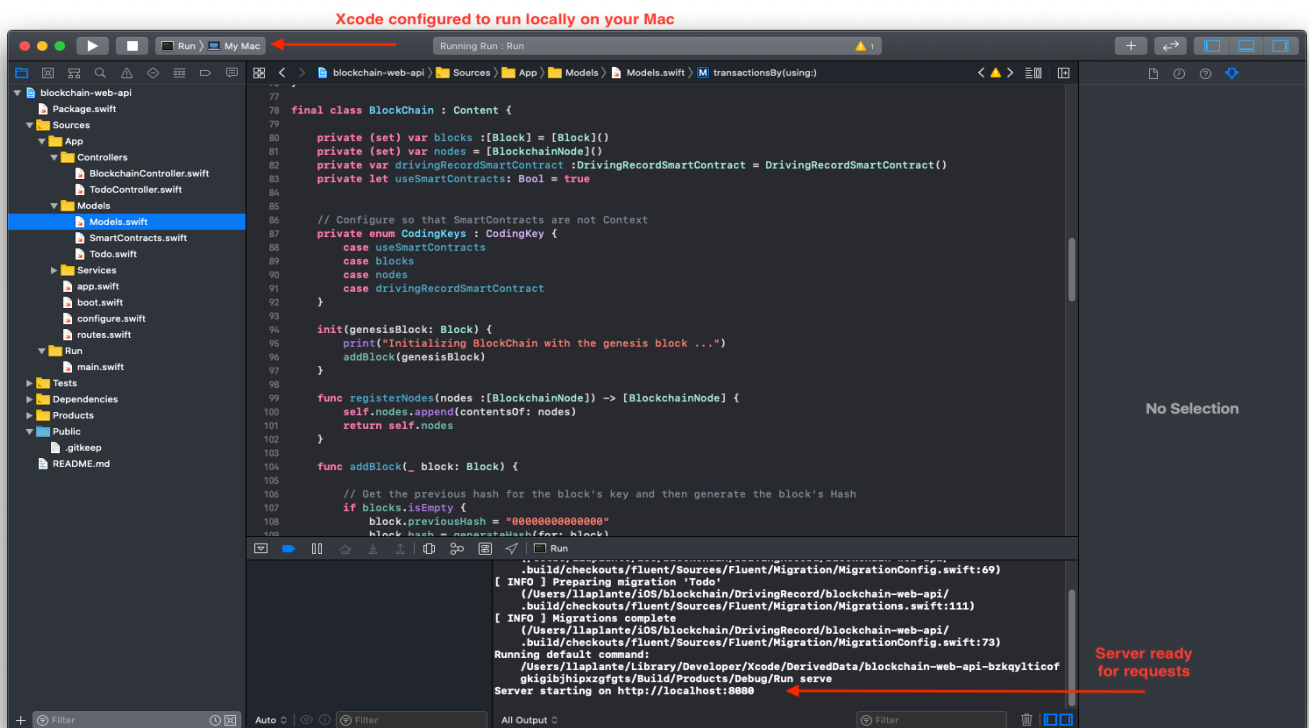
- Vapor Framework (version 3.x)
- Xcode 11.x
- A web client (such as **Postman**) for sending and receiving HTTP request with JSON body

NOTE: The examples in this guide use Postman

Setup and Running

1. Open the project in Xcode
2. Configure the active scheme for Run -> MyMac (See following figure)
3. Open the console window in Xcode
4. Build and run the project (See following figure)

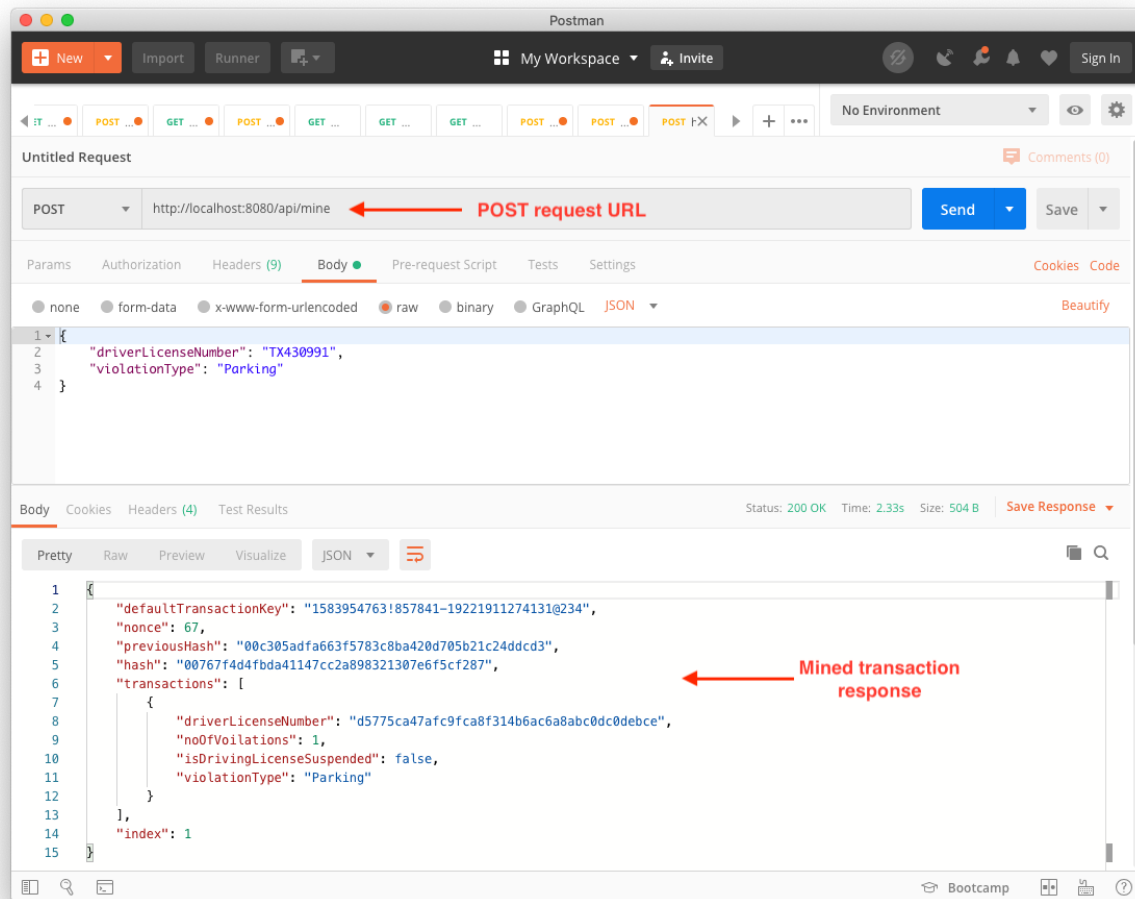
NOTE: The server is ready for requests when the 'Server starting ... ' message appears



Core Blockchain Operations

Add Transaction and Block Mining

Here we send a transaction to create/mine a new block. Refer to the following diagram for details for request URL and request JSON transaction. The JSON response is the newly mined block.



Using the same request URL as above, send two additional transactions to be mined. Transaction for TX, DWI

```
{
  "driverLicenseNumber": "TX430991",
  "violationType": "DUI"
}
```

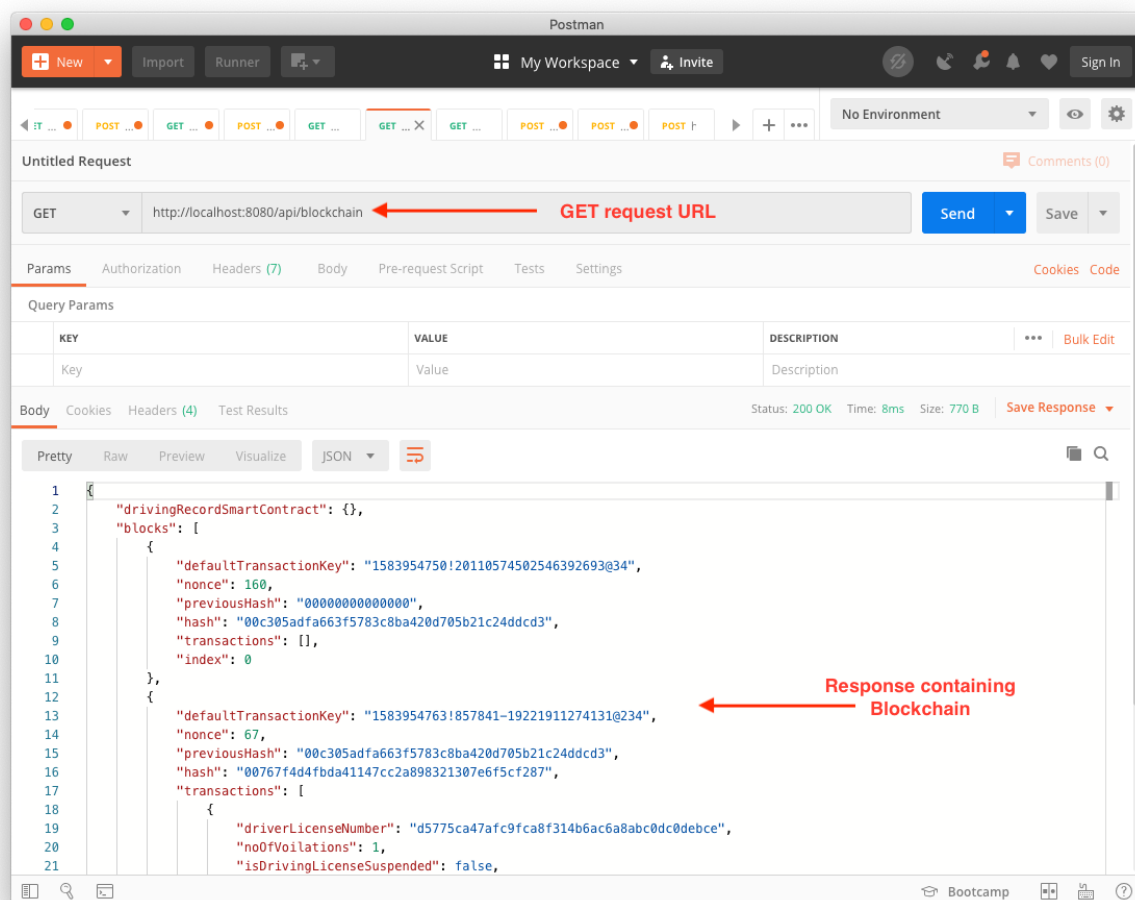
Transaction for CA, Speeding

```
{
  "driverLicenseNumber": "CA730981",
  "violationType": "Speeding"
}
```

Getting the Blockchain

Now let's query the server for the blockchain instance to review it. Refer to the following diagram for details for request URL. The JSON response is the complete blockchain instance showing all blocks recently mined based on the transactions submitted in the previous section.

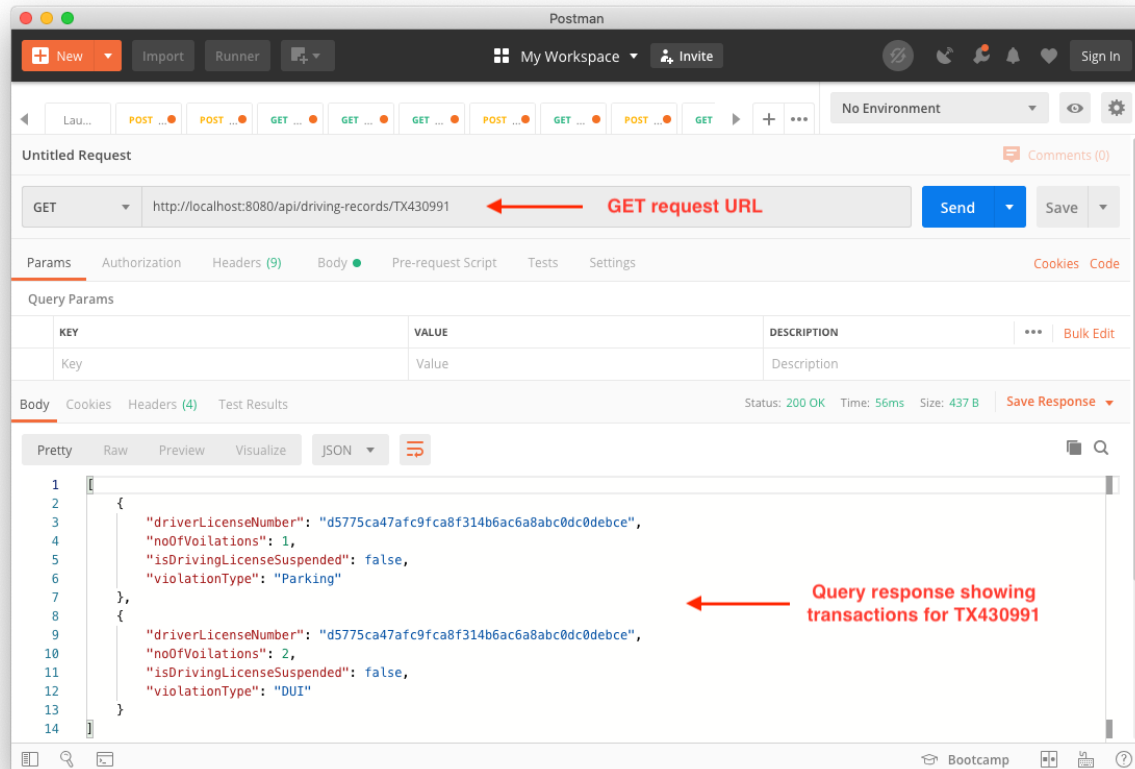
NOTE: The blockchain should contain all transactions submitted for mining in the 'Add Transaction and Block Mining' section.



Query the Blockchain

In this use case we will query the blockchain for all transactions for a specific driving license number. Refer to the following diagram for details for request URL. The JSON response is all transactions associated with the input driving license number.

NOTE: It's important that you completed the 'Add Transaction and Block Mining' section and that all transaction have been submitted for mining.

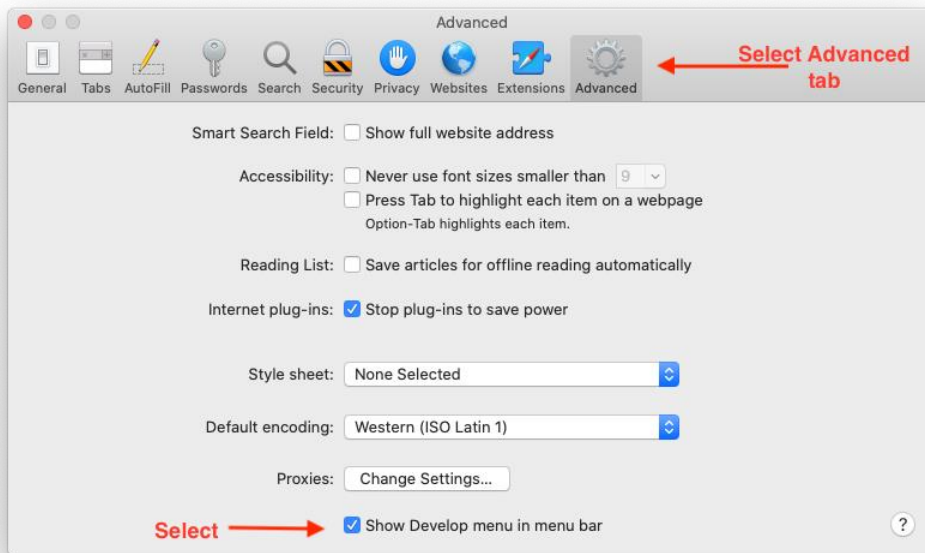


Query using the Web UI

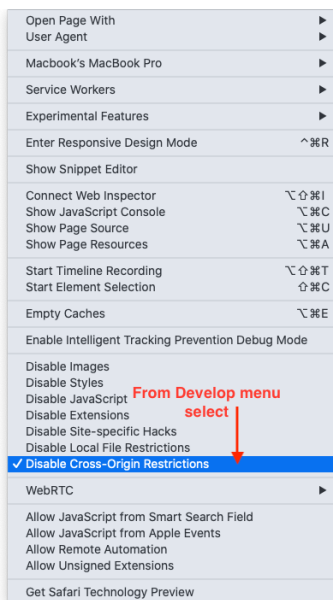
In this section we repeat the blockchain query using the provided Web UI.

NOTE: It is important that your browser of choice is configured to disable cross-origin restrictions. This example uses Safari which has been configured as per the following.

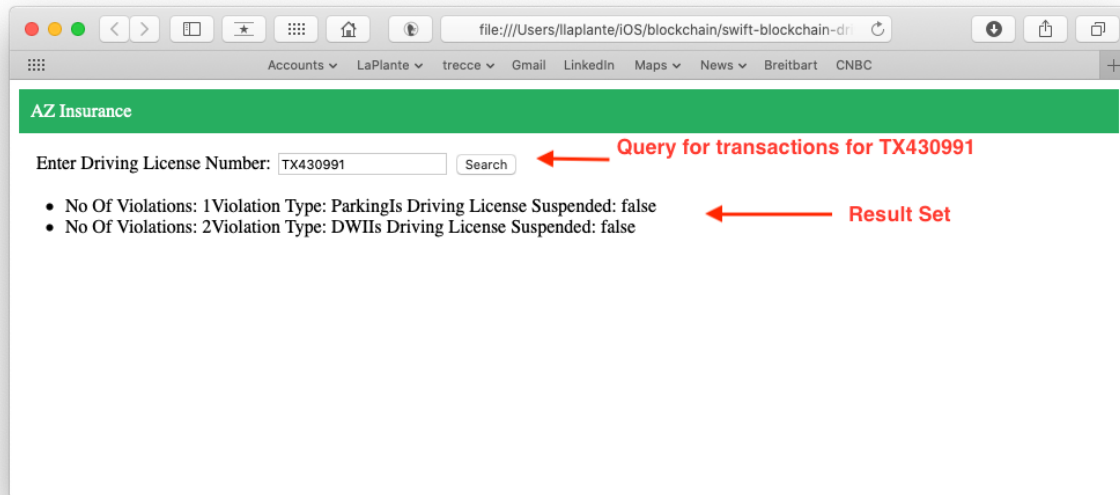
1. Enable the Develop menu in Safari Preferences



2. Disable Cross-Origin Restrictions in the Develop menu



Once your browser is configured, open the index.html in the project's **InsuranceWebSite** folder. Input the driver license number and Search. The following diagram shows the page with query result set with all transactions for the input driving license number.



Smart Contracts

In this use case we'll see the impact of SmartContracts as the number of driving violations exceeds the maximum allowed e.g 5. In this event smart contract logic will toggle the license to Suspended.

To get started, add additional transactions (as shown below) using the same request URL as shown in the 'Add Transaction and Mining' section above.

Using multiple POST requests to <http://localhost:8080/api/mine>, submit the following transactions:

```
{
  "driverLicenseNumber": "TX430991",
  "violationType": "Speeding"
}

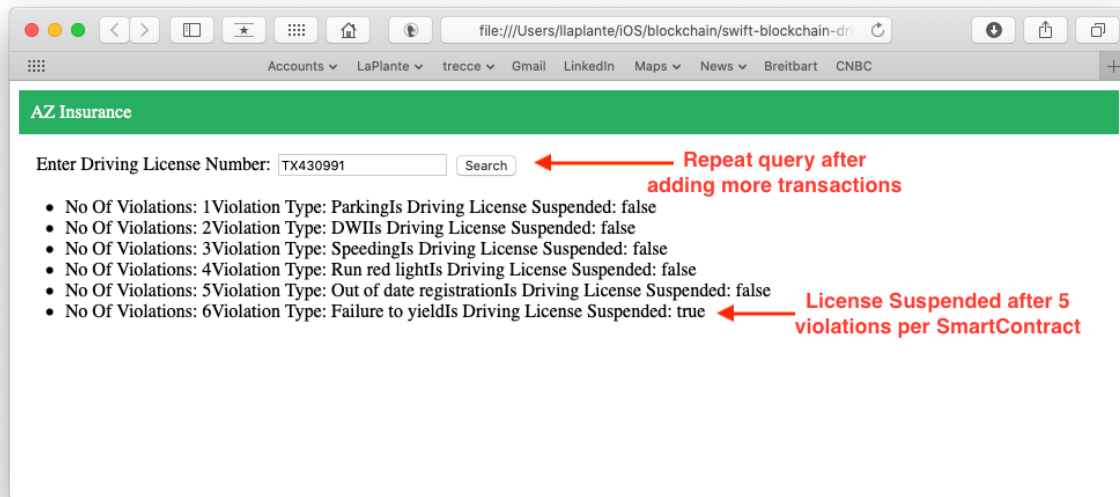
{
  "driverLicenseNumber": "TX430991",
  "violationType": "Run red light"
}

{
  "driverLicenseNumber": "TX430991",
  "violationType": "Out of date registration"
}

{
  "driverLicenseNumber": "TX430991",
  "violationType": "Failure to yield"
}
```

With the transactions added to the blockchain , return to the Web UI and repeat the Search for driving license number TX430991. The page is shown next.

NOTE: On the final transaction the Smart Contract logic has suspended the license.

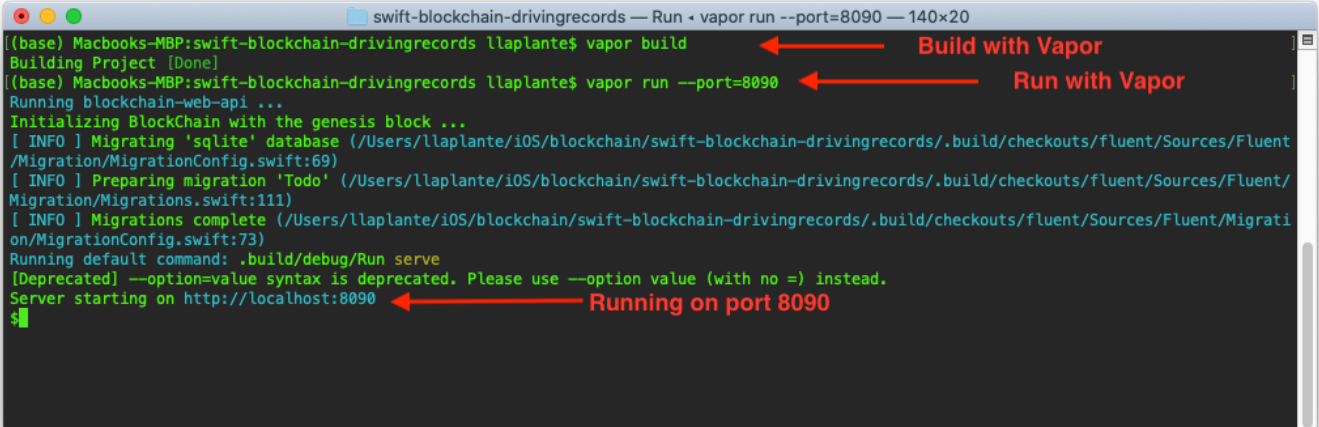


Distributed Blockchain Operations

In the previous sections we looked at core operations in a single node context. In this section we consider blockchain operations in a distributed environment. This would be more typical of any production blockchain environment composed of multiple nodes. A critical operation in any such distributed environment is determining which node holds the 'truth' blockchain and how other nodes are updated with this node's blockchain instance.

Setting Up a Distributed Environment

- Shutdown the server running on port 8080 in Xcode and restart
- Start another server instance on port 8090 (refer to the following figure)
 1. cd to the project directory
 2. Build the project with Vapor i.e. 'vapor build'
 3. Start the server with Vapor on port 8090 i.e. 'vapor run --port=8090'



```
swift-blockchain-drivingrecords — Run • vapor run --port=8090 — 140x20
(base) Macbooks-MBP:swift-blockchain-drivingrecords llaplante$ vapor build
Building Project [Done]
(base) Macbooks-MBP:swift-blockchain-drivingrecords llaplante$ vapor run --port=8090
Running blockchain-web-api ...
Initializing BlockChain with the genesis block ...
[ INFO ] Migrating 'sqlite' database (/Users/llaplante/iOS/blockchain/swift-blockchain-drivingrecords/.build/checkouts/fluent/Sources/Fluent/Migration/MigrationConfig.swift:69)
[ INFO ] Preparing migration 'Todo' (/Users/llaplante/iOS/blockchain/swift-blockchain-drivingrecords/.build/checkouts/fluent/Sources/Fluent/Migration/Migrations.swift:111)
[ INFO ] Migrations complete (/Users/llaplante/iOS/blockchain/swift-blockchain-drivingrecords/.build/checkouts/fluent/Sources/Fluent/Migration/MigrationConfig.swift:73)
Running default command: .build/debug/Run serve
[Deprecated] --option=value syntax is deprecated. Please use --option value (with no =) instead.
Server starting on http://localhost:8090
```

Build with Vapor

Run with Vapor

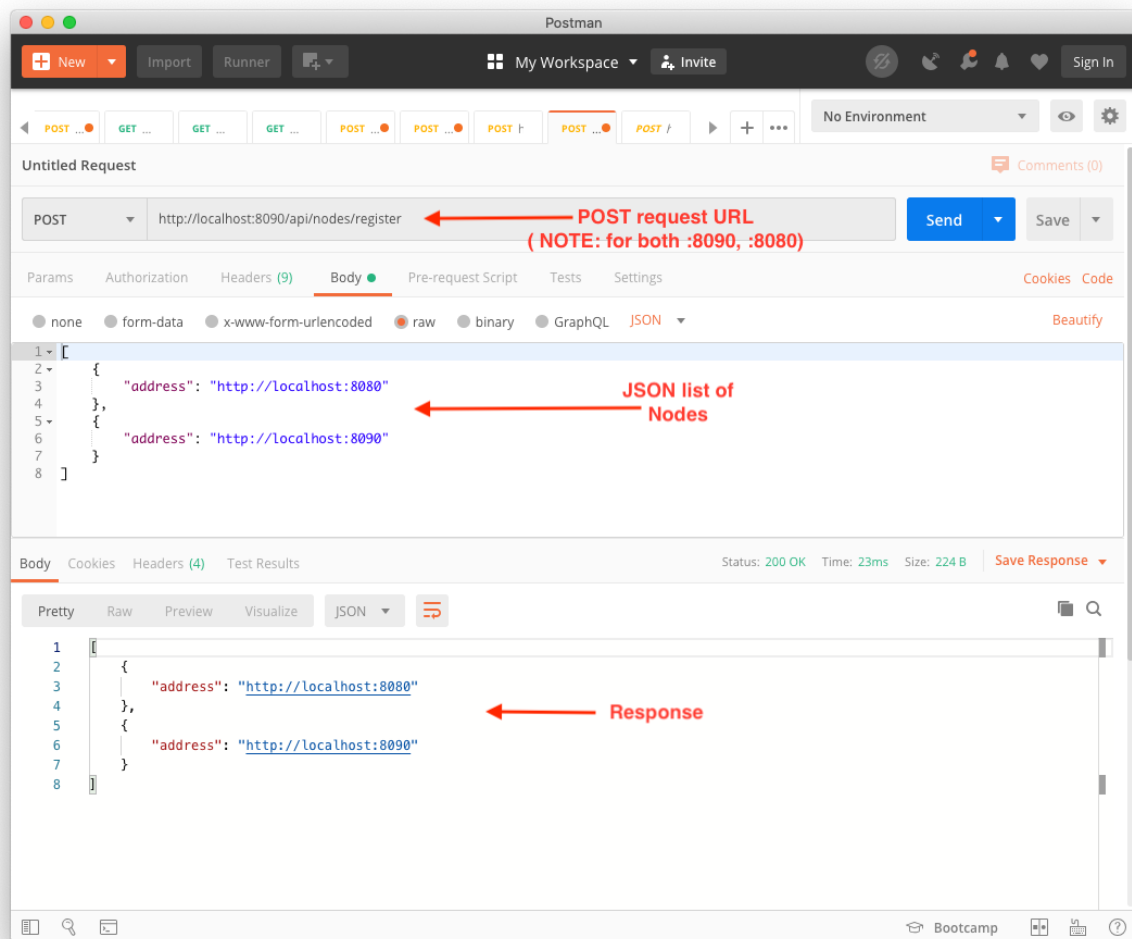
Running on port 8090

NOTE: After completing the above steps you should have servers on ports 8080 and 8090 running and ready to accept requests.

Registering Nodes

With both servers in the blockchain network running , the next step is to register these nodes with the servers. For both servers send a POST request with the JSON formatted node list as show in the following figure.

NOTE: Make sure that the node list is registered with both servers e.g. localhost:8080 and localhost:8090



Resolve Operation

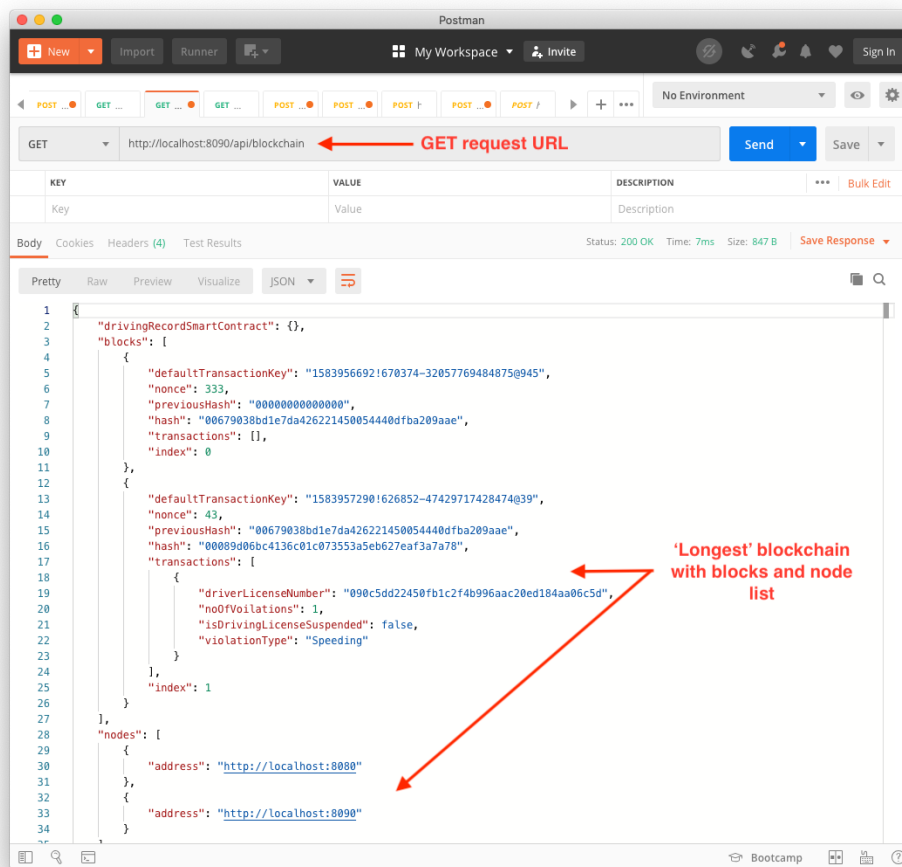
In this use case one of the nodes is sent a new transaction for mining a new block. At this point the remaining node's blockchain is now out of date and will be update with the 'truth' or longest blockchain. This process is referred to as resolve.

The first step is to send a transaction to one server. To do so, send a POST request to <http://localhost:8090/api/mine> with the following JSON request body:

```
{  
  
  "driverLicenseNumber": "TX430991",  
  "violationType": "Speeding"  
}
```

NOTE: Refer to the 'Add transaction and Block Mining' section if you need details on sending transactions.

Using the API, get the updated blockchain (with blocks and node list)for review.



Finally, let's update the outdated node using the resolve operations as shown in the next figure.

NOTE: The response shows the updated blockchain from the 8090 node. It should match the blockchain that you just reviewed directly above.

