Java for your first Blockchain Brew

Blockchain is a powerful technology, disrupting business across the world. The revolution is similar to the way email disrupted traditional messaging systems.

This blog is your head start into world of blockchain covering the basics and also getting your hands dirty with a dive into coding to constructing your own blockchain. It wraps up with point-of-view on criteria to be considered when evaluating Blockchain as technology for your use-cases and solutions. With intent to keeping it simple, it is not covering more complex aspects in blockchain like consensus algorithms which makes a blockchain trustless.

First things first - what is blockchain?

In simple words, it is a technology that establishes trust among untrusted peers through secure, encrypted data, which data once recorded can never be modified. Transactions can be clustered in a block, which are chained together (hence the name Block-chain). Data in a block and order of a block in its chain remains fixed - each block stores the hash of its previous block. Blockchain brings to you the promise of secure, distributed and unmodifiable ledger. By storing data in a distributed network, Blockchain eliminates the need of centralized entity, which owns all data and operates on it.

When did you hear blockchain for first time – was it Bitcoin?

BitCoin – popular crypto-currency – was one of the first and by far the most popular use-case of Blockchain. It was proposed in 2008, to get around the transaction fees and delayed transactions of money across parties, which happens through trusted public institutions like banks.

Blockchain is the technology that powers Bitcoin.

Why would you use the most inefficient database?

Blockchain is the most inefficient (slowest and resource hungry) database to ever have been invented. The only benefit of blockchain and why you would want to deal with its inefficiency, is that it makes collaboration between mutually untrusting parties, possible, without the existence of a 3rd mutually trusted party. The final part is in the article but not a note of its inefficiency.

Is it so complex? No, let's code our block chain

To get started with a block chain, let's define a simple Block as follows:

```
public class Block {
    String hash;
    String previousHash;
    String data;
}
```

In the preceding code

- data - stores the actual data of a block (I used a String type to keep it simple).
- hash - stores hash for the current block
- previousHash – stores hash of the previous block.

Every block should store at least the preceding three data items. The blocks can include additional data as required. Let's add a method to calculate the hash of a block. Typically, a bock calculates its hash using its own data and hash of its previous hash code. This is to ensure that data of a block plays a crucial role in its placement in a blockchain. If a hacker tries to replace an existing block with the modified data, it would interfere in its hash; resulting in denial of its acceptance in the chain.

To calculate the hash of a block, let's use SHA256 (Secure Hashing Algorithm 256). Here's the code:

```
public static String getHashWithSHA256(String input){
    try {
        MessageDigest msgDigest = MessageDigest.getInstance("SHA-256");
        byte[] hash = msgDigest.digest(input.getBytes("UTF-8"));
        StringBuffer hexString = new StringBuffer();
        for (int i = 0; i < hash.length; i++) {
            String hex = Integer.toHexString(0xff & hash[i]);
            if(hex.length() == 1) hexString.append('0');
            hexString.append(hex);
        }
        return hexString.toString();
    }
    catch(Exception e) {
        throw new RuntimeException(e);
    }
}
```

By using `java.security.MessageDigest`, you can use the SHA256 algorithm in Java – which is one of the most secure and widely algorithm, for cryptography.

Let's modify the Block class, calling the preceding method to calculate hash of a Block on its creation:

```
public class Block {
    final private String hash;
    final private String previousHash;
    final private String data;
    final private LocalDateTime dateTime;

    public Block(String data,String previousHash ) {
        this.data = data;
        this.previousHash = previousHash;
        this.dateTime = LocalDateTime.now();
        this.hash = blockHash(this);
    }

    public String blockHash(Block block) {
        long ms = block
                .dateTime
                .atZone(ZoneId.systemDefault()).toInstant().toEpochMilli();

         String hash = HashUtils
                .getHashWithSHA256(block.previousHash + ms + block.data);

         return hash;
    }
}
```

Let's now create a bare-essential block chain with these blocks:

```
List<Block> chain = new ArrayList<>();
```

```
Block block1 = new Block("Genesis block is the first block in block chain", "0");
Block block2 = new Block("Block 2",block1.hash);
Block block3 = new Block("Block 3",block2.hash);

chain.add(block1);
chain.add(block2);
chain.add(block3);
```

Here's a simple method to validate this blockchain:

```
static boolean validateChain(List<Block> chain) {
    for (int i = chain.size() - 1; i > 0; i --) {
        if (chain.get(i).previousHash != chain.get(i-1).hash)
            return false;
    }
    return true;
}
```

The preceding code would traverse the chain starting at its last element and moving backwards, comparing the `hash` and `previousHash` of blocks along the way. Let's call this method to check the validity of our blockchain:

```
public class MyFirstBlockChain {
    public static void main(String[] args) {
        List<Block> chain = new ArrayList<>();

        Block block1 = new Block("Genesis block - first block in block chain", "0");
        Block block2 = new Block("Block 2",block1.hash);
        Block block3 = new Block("Block 3",block2.hash);

        chain.add(block1);
        chain.add(block2);
        chain.add(block3);

        System.out.println("Valid = " + validateChain(chain));  // VALIDATE CHAIN
    }

    static boolean validateChain(List<Block> chain) {
        for (int i = chain.size() - 1; i > 0; i --) {
            if (chain.get(i).previousHash != chain.get(i-1).hash)
                return false;
        }
        return true;
    }
}
```

The preceding code outputs `true`.

Let's now create a new block and replace an existing block in the chain with this block. Will the validate method return false then? Let's check it out:

```
Block blockNew = new Block("NewBlock",block1.hash);
chain.set(1, blockNew);

System.out.println("Valid = " + validateChain(chain));
```

The preceding code outputs `false`.
Yay! You just created your first (simplest) blockchain.

Before you consider creating or using a blockchain, you must understand whether your business needs it. I think the biggest problem these days is that people don't understand what blockchain is, and what it is not.

Challenges with Blockchain

There are several problems with the blockchain.

First is that every piece of data must be replicated by a majority of the nodes in the network. This is slow, and very inefficient.

Second is that there is a need to authenticate, who can and cannot write to the blockchain. The token is the authorization badge, which allows a user to write to a public blockchain. These tokens have been traded, and the first example was bitcoin. Note - Bitcoin is the network, bitcoin is the token. The token is pre-created in non-serious or scam implementations, as this gives instant money. In serious implementations they are generated by the network as the blockchain grows. Blockchains that use Proof of Work (PoW) consensus algorithms, like Bitcoin or Ethereum, don't need *bitcoin* or *ether* to add a new block. For blockchains that use Proof of Stake (PoS) consensus algorithm, you must have coins to earn more, for example, NEO Blockchain.

Final one is that it cannot scale, because of the need to create many copies of the data. And scaling is required for a blockchain that can encompass the world. There are two solutions to this problem, live in a very high value space, like Bitcoin, or have a second layer, like the lightening network.

Private blockchains can and should use Certificates as authorization badges, instead of tokens, as it is a much more secure way of controlling access. The privacy allows much simpler and efficient consensus algorithms, but it can still not begin to approach a simple database, to a couple of orders of magnitude. Although there are many other consensus algorithms that claim they are scalable with higher transaction throughput, they are yet to be tested by real world applications.

What next?

This blog just scratched the surface by enabling you to get started with a basic blockchain. Blockchains come in multiple flavors – global or public, like bitcoin, or private, limited to be used within an organization or community. Blockchain is transforming business across various industries like heath care, transportation and shipping, finance, education and much more.

Industry is still in the early stages of Blockchain adoption. It has the power to transform businesses – making them rethink how they work.