Aspect-Oriented Smart Contracts

Saifur Rahman Bhuiyan

TU Dresden

September 28, 2022

Table of Contents

Abstract Update

Activities in Abstract

- Dive into Hyper-ledger Fabric Framework
- Develop very basic, small Use Case of smart contract.
- Getting into AspectJ and familiarize myself with AspectJ framework
- Introduction of Few AOP feature to Sample Smart Contract

Sample Smart Contract

- The Chain Code is about to simulation of Home Transfer
- Here, there is one model class that will represent the entity "Home" and another Business Logic class that will simulate Home Transaction
- Logic class has four methods "InitLedger", "queryHomeById", "changeHomeOwnership" and "addNewHome"

addNewHome

```
@Transaction()
public Home addNewHome(final Context ctx, final String id, final String name, final String area,
                       final String ownername, final String value) {

    ChaincodeStub stub = ctx.getStub();
    String homeState = stub.getStringState(id); //read from ledger
    if (!homeState.isEmpty()) {
        String errorMessage = String.format("Home %s already exists", id);
        System.out.println(errorMessage);
        throw new ChaincodeException(errorMessage, HomeTransferErrors.HOME_ALREADY_EXISTS.toString());
    }
    Home home = new Home(id, name, area, ownername,value);
    homeState = genson.serialize(home); // serialization of java object to JSON format
    stub.putStringState(id, homeState);
    return home;
}
```

queryHomebyId

```java
@Transaction()
public Home queryHomeById(final Context ctx, final String id) {
    ChaincodeStub stub = ctx.getStub();
    String homeState = stub.getStringState(id); //Query from ledger

    if (homeState.isEmpty()) {
        String errorMessage = String.format("Home %s does not exist", id);
        System.out.println(errorMessage);
        throw new ChaincodeException(errorMessage, HomeTransferErrors.HOME_NOT_FOUND.toString());
    }

    Home home = genson.deserialize(homeState, Home.class); //deserializing JSON to java Object
    return home;
}
```

chnageHomeOwnership

```java
@Transaction()
public Home changeHomeOwnership(final Context ctx, final String id, final String newHomeOwner) {
    ChaincodeStub stub = ctx.getStub();

    String homeState = stub.getStringState(id);

    if (homeState.isEmpty()) {
        String errorMessage = String.format("Home %s does not exist", id);
        System.out.println(errorMessage);
        throw new ChaincodeException(errorMessage, HomeTransferErrors.HOME_NOT_FOUND.toString());
    }

    Home home = genson.deserialize(homeState, Home.class);

    Home newHome = new Home(home.getId(), home.getName(), home.getArea(), newHomeOwner, home.getValue());

    String newHomeState = genson.serialize(newHome);

    stub.putStringState(id, newHomeState);

    return newHome;
}
```

Introduction of AOP to Chaincode

AOP Concept

1. Annotation based Aspectj tested so far with Smart Contract
2. Tested successfully different types advice like @Before, @After, @AfterReturning, @AfterThrowing and @Around
3. Sample logging aspect has been integrated with our sample chaincode "Home Transfer"

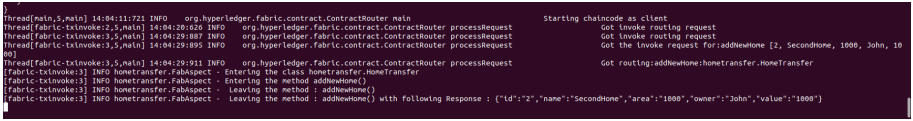
Logging Aspect

```java
@Around("myPointcut()")
public Object applicationLogger(ProceedingJoinPoint pjp) throws Throwable {

    //ProceedingJoinPoint is an extension of the JoinPoint which provides reflective access to the state available at a given jo
    ObjectMapper mapper = new ObjectMapper().configure(DeserializationFeature.FAIL_ON_UNKNOWN_PROPERTIES, state: false);
  // mapper.configure(SerializationFeature.FAIL_ON_SELF_REFERENCES, false);
    String methodName = pjp.getSignature().getName(); //this will give us the method name
    String className = pjp.getTarget().getClass().toString(); // this will give us the class name
    Object[] array = pjp.getArgs(); //this will capture the incoming argument to the method

    log.info("Entering the " + className);
    log.info("Entering the method "  + methodName + "()");
    log.info("Entering the method "  + methodName + "()" + " with arguments : " +mapper.writeValueAsString(array));
    Object object = pjp.proceed(); // after calling this proceed method, we can capture the response
    log.info(" Leaving the method : " + methodName + "()");
    log.info(" Leaving the method : " + methodName + "()" + " with following Response : " + mapper.writeValueAsString(object));
    return object;
}
```

Logs from Docker Container



Thank You!