摘要

研究者都希望自己的論文能被出版，被期刊引用，一篇有公信力的論文往往需要領域內權威人士的認證。但根據學術界教授表示，現階段論文審稿多為無報酬、無紀錄的「公益行為」，所以我們希望可以藉由經濟市場的供需關係來促進學術圈的交流。因此我們利用JAVA Spring boot建立一個網頁平台應用，架設於區塊鏈上，利用乙太坊發行ERC20代幣來進行價值計算，並利用區塊鏈公開紀錄且不會被抹滅的特性，來記錄審稿者的貢獻，並把無形的價值變得能夠被計算。

在這平台內用戶能夠建立活動，來搜集期刊或會議需要的論文，進行徵稿與招募協助審稿者。透過參與活動，上傳論文或是協助審核論文，來獲得PRV代幣。平台會將參與活動之資料傳至區塊鏈進行備份，可作為對雙方保障的紀錄，未來也可作為個人經歷上的參考。以代幣做為誘因，也能吸引會員參加活動協助審文，最終達到促進學術圈交流。

最後我們也能藉由此機會了解區塊鏈的運作模式、乙太坊ERC20代幣的發行與流通運用、以及使用JAVA Spring boot 來進行網頁的前後端開發。

致謝

這次專題報告能夠順利完成，我們要感謝指導教授查士朝教授的指導，從三年級上學期便開始與我們多次開會，在我們對於專題題目感到迷惘時給我們方向，經過多次討論才得出這次專題的主題、架構，甚至特別加開教我們區塊鏈與JAVA Spring boot 的課程，提供我們書籍資源，到了三年級下學期，開始進行實作的部分時，當我們遇到實務上架構與技術的問題，教授也是有問必答，百忙中依然抽空與我們討論，提供我們問題的解決辦法，感謝查教授細心的教學。

除此之外，還要感謝曾經提供我們前端參考範本、區塊鏈端的學長姐們，特別是黃子嘉學長，在我們遇到技術上的問題時給予的協助，讓我們能減少撞牆期，在有限的時間內產出這次的專題，感謝各位學長姐及教授的幫忙。

1.緒論

# 研究動機

研究者都希望自己的研究能夠被出版，若一篇研究論文能被出版於國內或國際知名期刊，便可證明是篇可靠的研究。所以論文大多會透過藉由領域內的權威人士協助進行審稿與認證，使其更有說服力。但現階段論文審核制度多為無報酬、無紀錄的「公益行為」，所以我們希望可以藉由經濟市場的供需關係促進學術圈的交流。

將舉辦論壇、審核論文之行為紀錄於區塊鏈中。未來可將其貢獻列入考量。藉此提高多數人對於審核論文之意願，讓更多人願意為學術圈貢獻，藉由協助審核論文可獲得Token。而會員在未來有論文需要被審核時可透過Token或是其貢獻度加速論文被指派審核之優先權。

# 研究目標

1. 建立一個使用Java Spring boot的網頁平台應用專案
2. 前端使用Thymleaf模板套件與HTML5
3. 串接H2資料庫來儲存資料
4. 架設私有區塊鏈進行活動、資料儲存
5. 發行ERC20代幣當作平台應用的價值基準
6. 撰寫智能合約於區塊鏈平台上進行應用

# 成果簡介

本次專題主要是透過建立網頁平台，讓用戶間可以創辦活動研討會、上傳論文及審核論文。並將其行為記錄於區塊鏈中，藉由紀錄以及賺取Token讓使用者願意參與活動，無形中也增加對學術圈的貢獻。

我們利用了區塊鏈的特性以及運作模式，選用了乙太坊作為架構，發行ERC20代幣，創造了專屬的Paper Review Token(簡稱PRT)，在此平台中以PRT作為價值的基準。完成審稿後可獲得PRT，未來若需審核論文時可藉由PRT來加速論文審核之排序。在會員參與活動的部分我們建立了智能合約（solidity），將流程與轉帳資訊寫入智能合約中。透過合約判斷流程進行的進度，與確認是否符合轉帳之條件。我們運用Web3j將JAVA Spring Boot連接上區塊鏈，將參與活動之資料連接至區塊鏈驗證，並作為參與活動之紀錄。

使用了Java Spring Boot作為開發平台上的框架，連接H2資料庫作為後端。透過MVC架構的三大核心：「Model, View, Controller」來進行前後台串接，實現資料傳輸。 利用了JavaMail來實現寄送電子郵件，通知用戶有新訊息的事件。結合Thymeleaf來加強網頁的呈現，活用管理前端介面。

2.系統開發平台框架與使用套件

# Git

Git是一個分散式版本控制軟體，最初由Linus Toralds於2005年設計，用以管理Linux的核心開發，現在則被廣泛的運用於程式碼的維護。

# Web3j

Web3j是一個輕量級、高度模組化、響應式、型別安全的Java/Android API，用於處理乙太坊智慧合約及與以太坊網路上的客戶端(節點)進行整合。並可以通過它進行乙太坊區塊鏈的開發，而無需為Java應用平臺編寫整合程式碼。

# Java Spring Boot

Spring Framework是一個Java平台上的應用程式框架，其核心功能理論上能適合任何Java 應用，並在Java開發者社群中被廣泛使用。其特色是適應多種網頁應用程式套件，開發者可利用JSP、Thymleaf、Velocity等等控制介面，進行與前端的對接與開發。

# Thymleaf

在Web應用程式中，Thymeleaf旨在成為JavaServer Pages （JSP）的完全替代品，並實現自然模板的概念。使檔案可以直接在瀏覽器中開啟，並且仍然可以正確顯示為網頁。可用於XML，XTML與HTML5的JAVA模板引擎，完整且可延伸的國際通用化支援。基於稱為方言和模組化的特徵集，多使用Standard和SpringStandard（適用於Spring MVC應用程式，與標準語法相同）這兩種方言進行入門。可於離線環境使用，沒有硬依賴Servlet API。可自動使用DOCTYPE的轉換，且極易擴充，功能完整。

# H2 database

H2 datanase是一個純JAVA開發的關係型資料庫，伺服器上、網頁嵌入式或是在本機端上執行等模式，其優點是安裝操作簡便且內建資料庫管理介面，使用者可在Console端使用SQL語法操作資料庫，至於我們則是在JpaRepository介面上用Java寫好指令，讓使用者操作時可直接呼叫使用。

# IntelliJ

IntelliJ IDEA是一種商業化銷售的Java整合式開發環境（Integrated Development Environment，IDE）工具軟體，由JetBrains軟體公司（前稱為IntelliJ）開發，是市面上的主流Java整合開發環境，同時具備支援目前主流的技術與框架，擅長於企業應用、移動應用與Web應用的開發。

在整合的部分，IntelliJ IDEA支援 Spring、Git、Maven，我們使用的就是Spring Framework。

# MetaMask

MetaMask是一款Google Chrome的虛擬貨幣錢包插件，主要是用來保存ETH及ETH鏈上的ERC20代幣，其特色有能輕易的讓使用者操作智能合約。

# Remix

Remix是一款由乙太坊提供的線上版IDE，用來開發Solidity，其功能除了有最基本的編譯、執行、分析、除錯或與github連結外，能根據solidity的不同版本來編譯，並在編譯後產生ABI、BYTECODE，提供web3j編譯成java檔案來進行與專案的連接，最大特色是能夠連結metamask帳號，來對編譯好的合約進行操作，像是發布到指定的鏈上，進行合約內方法的測試等。

# HTML5

HTML5是HTML最新的修訂版本，廣義來說，HTML5實際指的是一套包括HTML、CSS與JavaScript的技術組合，並且在CSS包中以內建RWD功能，使其可適應在各裝置頁面呈現時的情況。最重要的是，HTML5清楚定義了個版本間的相容性問題，其語法的一致性讓開發變的相對直觀與簡單。

# 

# Solidity

Solidity是一種合約式導向語言，可被應用於不同的區塊鏈平台上，並屬於靜態語言，編譯後可在EVM（Ethereum Virtual Machine，乙太坊虛擬機）上執行，目前主要應用於乙太坊上各式合約的撰寫。其中Solidity的語言邏輯設計參考ECMAScript，所以對於寫過JavaScript的人來說上手容易。藉由Solidity，開發人員能夠編寫出可自動執行欲實現之商業邏輯程式，該程式並被視為一份具有權威性且永不可逆的合約，又稱智慧合約（Smart Contract）。

3.基礎區塊鏈知識

# 區塊鏈

區塊鏈（Blockchain）是一個特定類別的資料庫，專門用來儲存交易資料；也是一種分散式的帳本軟體。數字貨幣比特幣（Bitcoin），就是區塊鏈技術下衍生出的產物。一般而言，區塊鏈並非單一創新技術，而是結合過去數十年所累積的多項跨領域技術而來，包括：數學、密碼學、演算法與經濟模型等等。因為該技術應用於網路中，使得點對點之間的互動關係，可以建立一套彼此不需基於信任基礎、也不須仰賴單一中心化機構，就能安全運作的分散式系統。此系統上運作的交易皆為不可逆的，也需鏈上成員的認可並可被追蹤，具備安全、開放、透明與低成本等特性。

# 乙太坊

乙太坊是一個開源、有智慧合約功能的區塊鏈平台。最初由 Vitalik Buterin 在2013年受比特網路啟發後提出，提供去中心化的虛擬機EVM（Ethereum Virtual Machine）來處理鏈上執行的合約。目前來說，乙太坊的產物乙太幣（ETH）是市值第二高的加密貨幣，僅次於比特幣。

# Quorum鏈

Quorum是由J.P.Morgan推出的企業級區塊鏈平台，建構於乙太鏈之上，擁有別於乙太鏈私有化功能與多種共識協定。並且運行效能方面則有著更出色的表現，這也是我們後來選用Quorum Chain的主要原因。

# 智慧合約

前面提到以Solidity語言撰寫，在EVM上執行的程式，就叫做智慧合約。其實也可以將智慧合約想成Java Programming中的一個class，這個class被部署（deploy）到區塊鏈上時，這個程式就會運行在每個節點（node）之上，並取得一個地址，他就像記憶體一樣，取得這個位置後搭配正確的interface資訊就可以執行這個合約。

智慧合約相關的技術，現今已逐漸應用在各種商業模式上，包含食品安全認證、醫療追朔、身份文件追蹤、物流監管、休閒事業管理、智慧財產保護、資安等等領域，是將來不可忽視的趨勢技術。

# ERC20

ERC20是Ethereum平台上的的數據通訊協定，規定了所發行代幣變數、指令的命名機制。所以ERC20可以被稱為一個標準，若是一個token符合ERC20，即代表此代幣是一種具有完整交易功能的代幣。且由於ERC20是Ethereum的一部分，所有符合其協定的代幣都可以利用乙太鏈的工具去交易、追蹤或監測交易，不需自己再開發額外的工具或系統。

4.研究方法

# 4.1系統分析與設計

## 需求與分析

### 問題需求與描述

1. 論文平台可提供研討會／活動的舉辦，透過審核文章可獲得論文幣，以增加審核論文之動機，再加上智能合約來減少驗證手續。
2. 用戶可能有三種角色，活動創辦人、論文作者及審稿人。
3. 活動創辦人可透過舉辦研討會/活動，進行徵文以及募集審稿人。
4. 會員可以針對活動上傳論文，活動創辦人收到論文後指派參加活動的用戶審稿。
5. 收到審稿通知的會員同意參與審稿後即成為審稿人，需在時間內回覆論文。
6. 審稿的收發程序與是否審稿完成的驗證需經過智能合約來判斷。
7. 完成審稿後，活動創辦人透過合約發放論文幣給審稿人。

備註：

1. 參與上述活動皆需要成為會員以及登入。
2. 所有活動都需要登入後才能參與。
3. 會員需要提供姓名、身分證字號等。

### 需求分析

##### 審核論文平台的功能性需求

1. 平台會提供用戶登入系統
2. 平台會提供用戶註冊為會員
3. 平台會提供會員瀏覽活動列表
4. 平台會提供會員參加活動
5. 平台會提供會員創建活動
6. 平台會提供會員修改活動
7. 平台會提供會員刪除活動
8. 平台會提供會員上傳論文
9. 平台會提供會員修改論文
10. 平台會提供會員刪除論文
11. 平台會提供會員檢視活動內論文審核狀態
12. 平台會提供會員檢視論文回覆內容
13. 平台會提供會員通知有指派待審核的論文
14. 平台會提供會員檢視論文內容並新增回覆
15. 平台會提供會員檢視論文內容並修改回覆
16. 平台會提供會員檢視論文內容並刪除回覆
17. 平台會提供會員檢視帳戶及歷史參與資訊
18. 會員完成審核文章獲得ERC20代幣

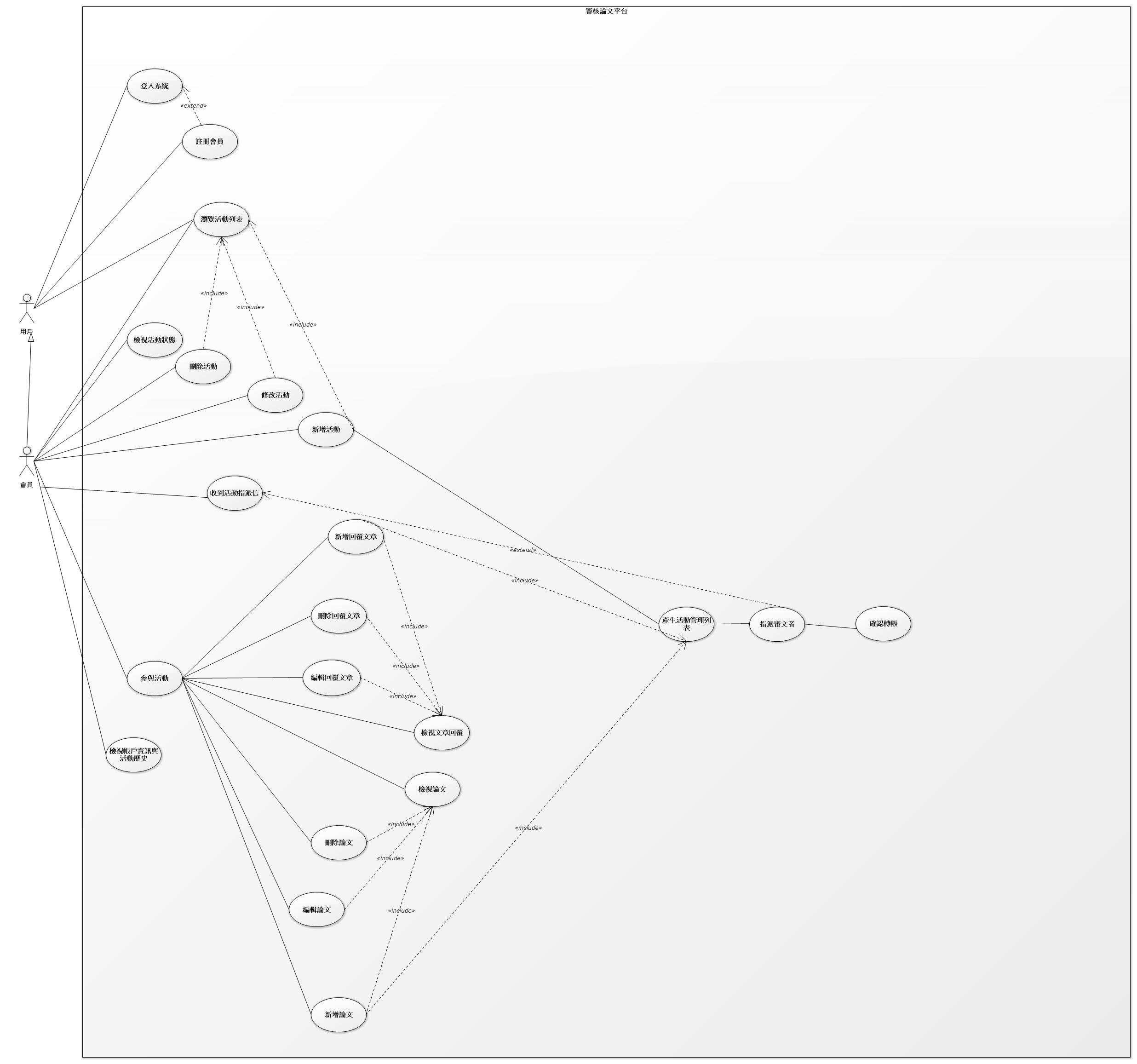
##### 審核論文平台的非功能性需求

1. 一位用戶只能註冊一名會員
2. 平台必須能夠至少讓500名會員同時登入系統
3. 平台登錄過程需要在5秒鐘之內完成
4. 平台能夠每秒處例1000筆活動
5. 平台當機次數必須少於每年2次

##### 事件表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 事件 | 觸發器 | 來源 | 活動/使用案例 | 回應 | 目的地 |
| 用戶登入系統 | 輸入使用者身份證和密碼 | 用戶 | 登入系統 | 成功登入或登入失敗 | 會員 |
| 用戶註冊會員 | 輸入會員資料 | 用戶 | 註冊會員 | 成功註冊或失敗 | 用戶 |
| 用戶瀏覽活動列表 | 準備參與活動 | 用戶 | 瀏覽活動 | 活動列表 | 用戶 |
| 會員創建活動 | 輸入活動資料 | 會員 | 創建活動 | 新增活動確認，活動列表 | 會員  （成為活動創辦人） |
| 會員修改活動 | 修改活動資料 | 會員(活動創辦人) | 修改活動 | 修改活動確認，活動列表 | 會員  （成為活動創辦人） |
| 會員刪除活動 | 刪除活動資料 | 會員(活動創辦人) | 刪除活動 | 刪除活動確認，活動列表 | 會員  （成為活動創辦人） |
| 會員參加活動 | 選擇參與活動 | 會員 | 參加活動 | 新增會員資料至活動名單 | 會員 |
| 會員上傳論文 | 輸入論文內容 | 會員 | 上傳論文 | 新增文章資訊至活動內的論文列表 | 會員  （成為論文作者） |
| 會員修改論文 | 修改論文資訊 | 會員（論文作者） | 修改論文 | 修改論文確認，活動內的文章列表 | 會員  （論文作者） |
| 會員刪除論文 | 刪除論文資訊 | 會員（論文作者） | 刪除論文 | 刪除論文確認，活動內的文章列表 | 會員  （論文作者） |
| 會員檢視活動內的論文審核狀態 | 請求檢視活動內的論文狀態 | 會員 | 檢視活動進度 | 活動內的論文審核狀態 | 會員 |
| 會員檢視論文回覆內容 | 請求檢視論文回覆內容 | 會員 | 檢視論文回覆 | 文章回覆列表內容 | 會員 |
| 通知會員有指派待審核的論文 | 活動創辦人指定參與活動之會員加入審文 | 會員 | 收到指派待審核的電子郵件 | 寄送電子郵件至會員註冊之電子信箱 | 會員 |
| 會員新增審核、回覆論文內容 | 輸入審核論文之評論 | 會員 | 新增論文回覆 | 新增文章回覆內容，文章回覆列表 | 會員  （成為審核人） |
| 會員修改審核、回覆論文內容 | 修改審核論文之評論 | 會員  （審核人） | 修改論文回覆 | 修改文章回覆內容，文章回覆列表 | 會員  （審核人） |
| 會員刪除審核、回覆論文內容 | 刪除審核論文之評論 | 會員  （審核人） | 刪除論文回覆 | 刪除文章回覆內容，文章回覆列表 | 會員  （審核人） |
| 會員檢視帳戶及歷史參與資訊 | 請求檢視帳戶資料 | 會員 | 檢視帳戶 | 帳戶資料 | 會員 |
| 會員完成審核文章得到代幣 | 會員（審核人）完成審核論文且經過會員（創辦人）認證 | 會員  （活動創辦人） | 完成回覆認證 | 獲得代幣 | 會員  （審核人） |

### 使用案例圖



### 流程圖

### 

# 4.2資料庫

## Member

|  |  |  |  |
| --- | --- | --- | --- |
| 變數名稱 | 變數 | 資料型態 |  |
| 會員編號 | member\_id | Long |  |
| 生日 | birthday | String |  |
| 區塊鏈私鑰 | blockchainPrivateKey | String |  |
| 教育程度 | education | Integer |  |
| 電子信箱 | email | String |  |
| 性別 | gender | Integer |  |
| 身分證字號 | idCardNumber | String |  |
| 姓名 | name | String |  |
| 密碼 | password | String |  |
| 電話號碼 | phoneNumber | String |  |
| 特殊事蹟 | special | String |  |
| 領域 | study | String |  |

## 

## Activity

|  |  |  |  |
| --- | --- | --- | --- |
| 變數名稱 | 變數 | 資料型態 |  |
| 活動ID | activity\_id | Long |  |
| 活動內容 | activityContent | String |  |
| 活動照片 | activityImg | String |  |
| 活動名稱 | activityName | String |  |
| 文章數量 | articleNumber | Integer |  |
| 活動舉辦人 | activityOrganizer | Long | Fk |
| 活動限制參與人數 | limitedParticipants | Integer |  |
| 活動領域 | activityStudy | String |  |
| 活動期限 | activityTime | String |  |

## 

## 

## Article

|  |  |  |  |
| --- | --- | --- | --- |
| 變數名稱 | 變數 | 資料型態 |  |
| 文章ID | article\_id | Long |  |
| 活動ID | activity\_Id | Long | Fk-activity |
| 文章地址 | articleAddress | String |  |
| 文章名稱 | articleName | String |  |
| 文章狀態 | articleState | String |  |
| 文章領域 | articleStudy | Integer |  |
| 文章價值 | articleValue | Double |  |
| 作者 | Author | Long | Fk-member |
| 繳交回覆期限 | deadLine | String |  |
| 檔案名稱 | fileName | String |  |
| 公式數量 | formulaNumber | Integer |  |
| 文章上傳時間 | postTime | String |  |
| 文章字數 | textNumber | Integer |  |
| 文章檔案 | uploadFile | String |  |
|  |  |  |  |

## 

## 

## articleReview

|  |  |  |  |
| --- | --- | --- | --- |
| 變數名稱 | 變數 | 資料型態 |  |
| 評論ID | articleReview\_id | Long |  |
| 評論接受任務 | acceptTask | Boolean |  |
| 文章ID | article\_Id | Long | Fk |
| 會員ID | member\_Id | Long | Fk |
| 評論地址 | reviewAddress | Long |  |
| 評論完成度 | reviewComplete | Boolean |  |
| 評論內容 | reviewText | String |  |
| 評論時間 | reviewTime | String |  |

## 

## 

## Member\_Role

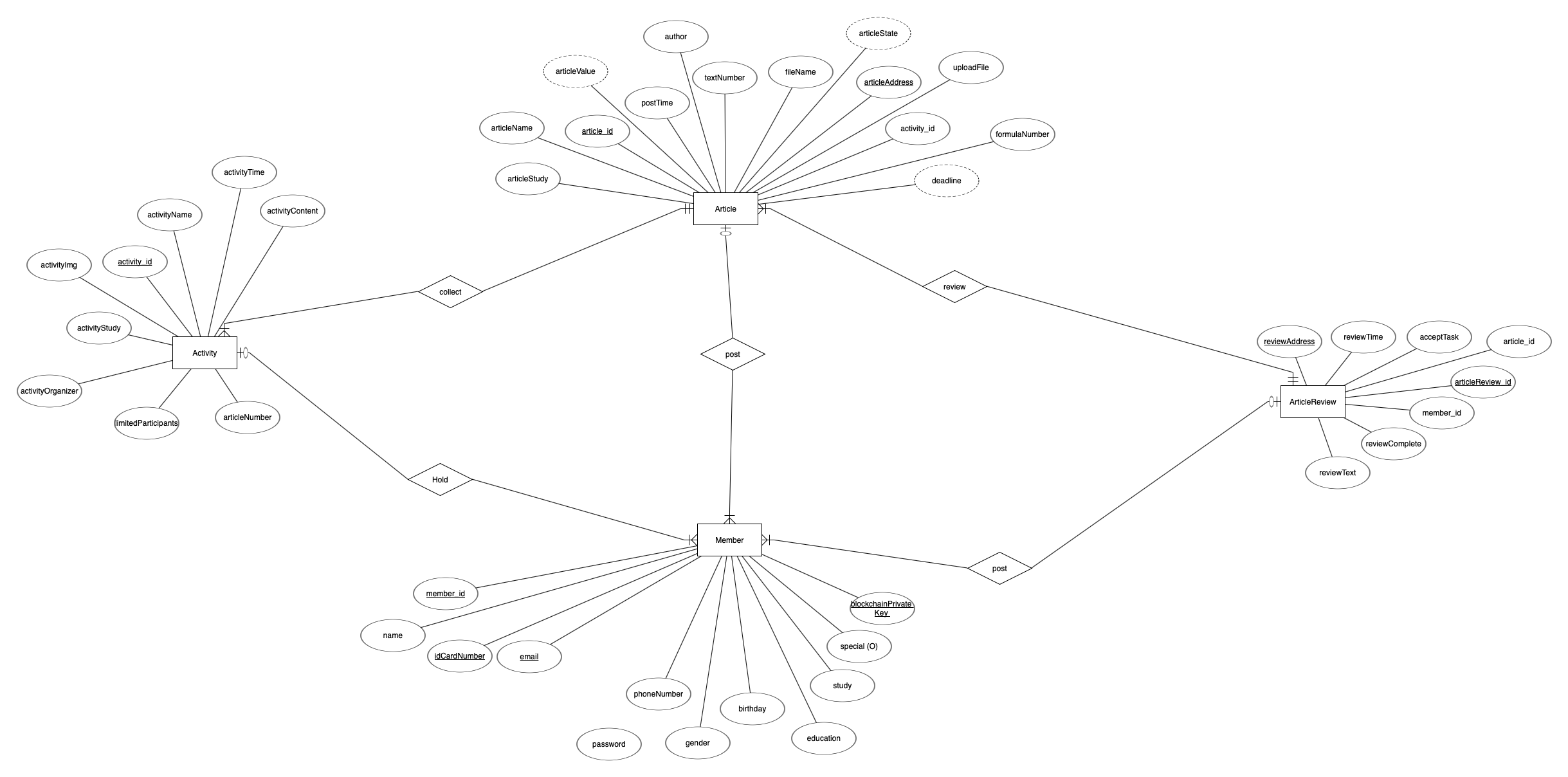
|  |  |  |  |
| --- | --- | --- | --- |
| 變數名稱 | 變數 | 資料型態 |  |
| 會員ID | ID | Long |  |
| 會員 | Name | String |  |

## 

## 

## ERD-Diagram

## 關聯圖



# 4.3研究步驟

## 區塊鏈

### 建立私鏈

前置動作

建立下載geth客戶端

安裝ethereum

#### 創世區塊

建立一個.json檔，此檔是自己架設的私有鏈，鏈上的第一塊區塊，稱為創世區塊，之後的區塊會從這塊開始往下一塊一塊接下去。

### 

參數說明：

* nonce：一個64為隨機數，用於挖礦。
* mixhash： 與nonce配合，用於挖礦，由上一個區塊的一部份產生的hash。
* difficulty: 設置當前區塊的難度，會影響到挖礦。
* coinbase：預設挖礦所得地址，此處可以先隨意設置，之後操作可以更改。
* timestamp: 設定創世區塊的時間戳。
* parentHash: 上一個區塊的hash值，因為是創世區塊，所以是0。
* extraData：附加訊息。
* gasLimit：設置gas的消耗總量限制，用來限制區塊能包含的交易訊息總和，私有鏈可先設置最大值。
* alloc：用來預設帳號及其乙太幣的數量。

#### 初始化私有鍊

在創世區塊建立完成後，要初始化我們的創世區塊

$ geth init ./genesis.json --datadir "./chain"

#### 啟動鏈

#### 在初始後，要啟動我們的私有鏈

#### 我們是輸入此指令：

geth --identity "test etherum" --rpc --rpccorsdomain "\*" --datadir "/Users/yihsiu/Desktop/passchain" --port "30303" --rpcapi "db,eth,net,web3" --networkid 20181013 console

#### 

#### 可調整參數說明

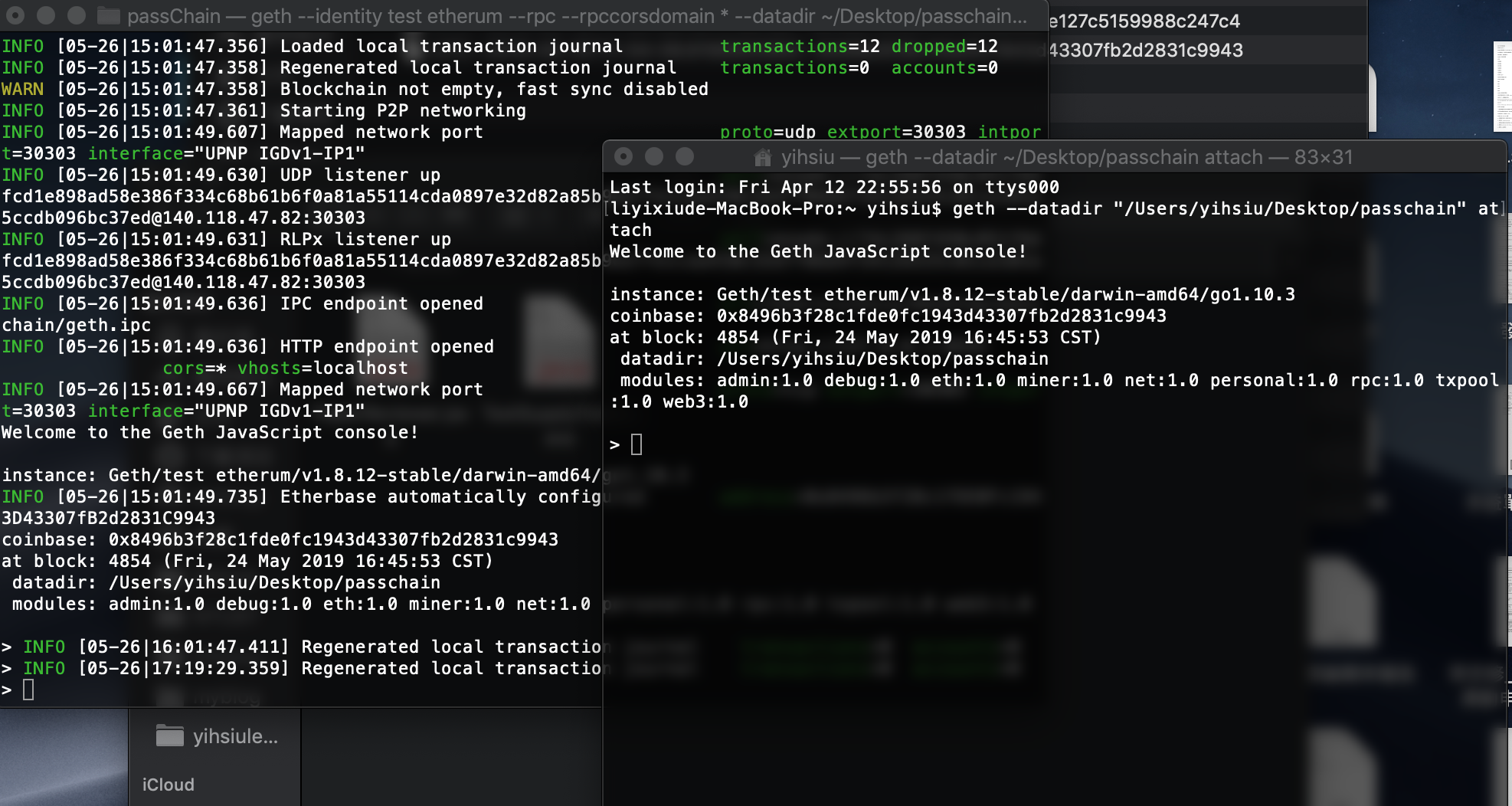
* datadir：指定當前區塊鍊網路數據存放的位置。
* networkid：設置當前區塊鏈的網路ID，用於區分不同的網路，是一個數字(1表示 Ethereum 主鏈，預設為1)。
* identity：區塊鏈的標示，隨便填寫，用於標示目前網路的名字。
* rpc：啟動rpc通信，可以進行智能合約的部署和調適。
* rpc port：rpc 埠， 預設為8545。
* port：P2P網路監聽端口, 預設為30303。
* rpcaddr：指定監聽地址，預設為127.0.0.1。設置為「0.0.0.0」，表示監聽所有的地址，允許外部進行RPC連接(從安全性上考量，建議 不要 指定該值)。
* rpccorsdomain：指定RPC可跨域連接的域名。設置為「\*」，表示允許所有的域名连接(從安全性上考量，建議 不要 指定該值)。
* rpcapi：指定允許連接的RPC客户端的API方法，預設為「db,eth,net,web3」。
* nodiscover：禁止被發現。添加個參數可以確保你的節點不會被非手動添加你的人發現，否則，你的節點可能因為你與其他的鏈有相同的創世區塊文件和網路ID而被陌生人的區塊鏈無意添加。
* console：啟動命令列模式，可以在Geth中執行指令。

### 客戶端連接私鏈

#### 啟動客戶端節點連結私鏈

使用geth attach方法，讓客戶端節點連上已啟動運行的區塊鏈。

geth --datadir "/Users/yihsiu/Desktop/passchain" attach



畫面後方終端機為運行中的鏈，前者是連上鏈的客戶端console。

### 客戶端功能測試

若客戶端成功連上乙太坊私有鏈，便可以開始下指令操作。

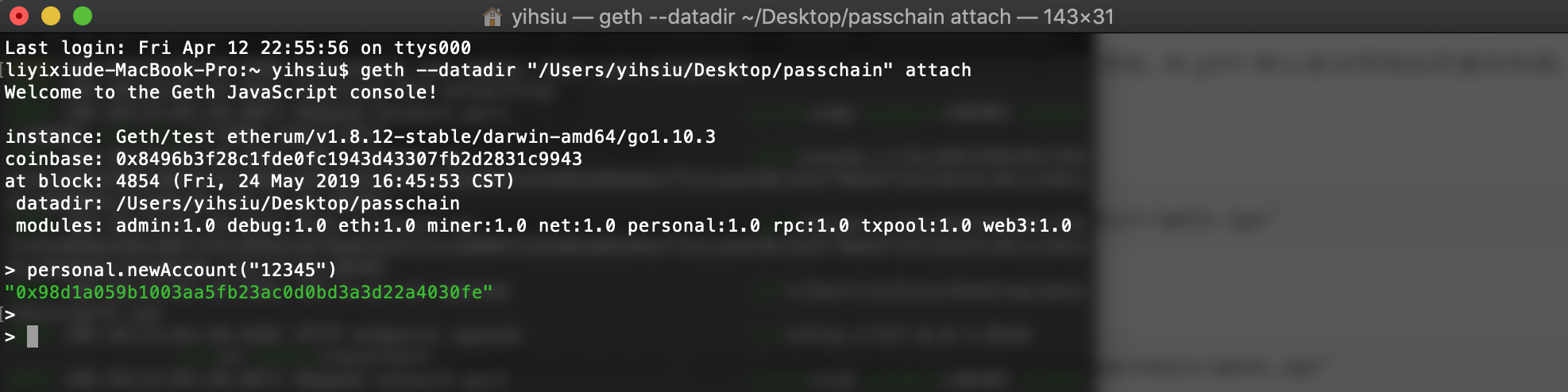
如前圖所示，出現“Welcome to the Geth JavaScript console!”

#### 帳戶建立

在要開始進行功能測試之前，需要先有最基本的錢包帳號。

以下是基本的錢包建立方法：

personal.newAccount("密碼")



若創建成功便會回傳一串地址，即為此新錢包的地址。

#### 餘額查詢

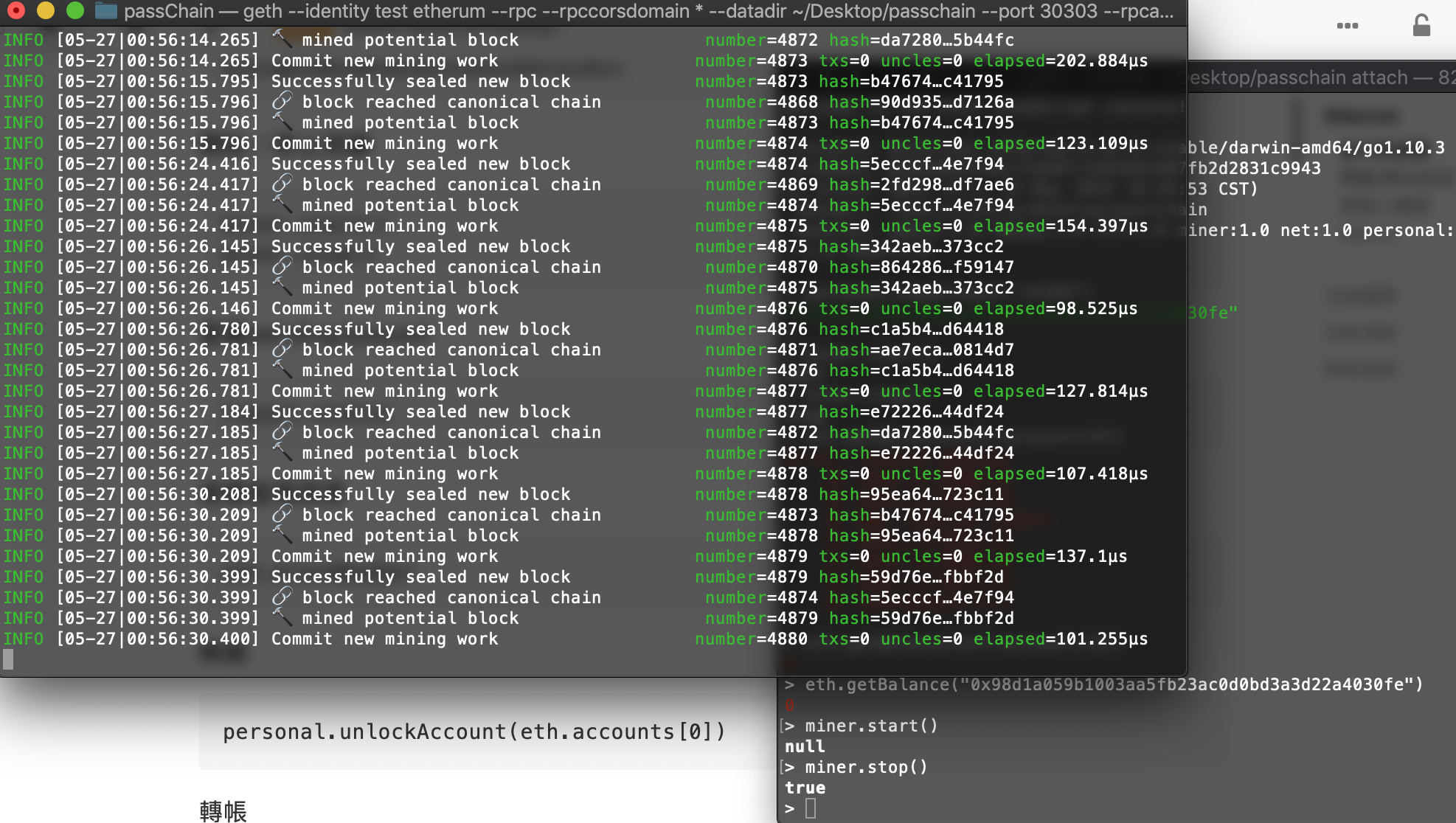
利用eth.getBalance(eth.accounts[序號])或者eth.getBalance(“錢包地址”)皆可查詢餘額



#### 挖礦啟動/停止

當輸入miner.start()時，挖礦會啟動，只要要運行把資料輸入到鏈上的相關操作時，挖礦保持在運行中的狀態，才會將數據寫入鏈中。

當輸入miner.stop()時，會停止挖礦。



上圖是運行中的挖礦可以看到區塊的高度、編號、hash等數值。

#### 轉帳功能測試

在轉出錢包金額前需要先進行將要轉出的錢包解鎖的動作。



輸入personal.unlockAccount(eth.accounts[“序號”])，接著輸入密碼便能將錢包解鎖。

解鎖後便能開始執行轉帳了。

eth.sendTransaction({from:eth.accounts[0],to:eth.accounts[1],value:web3.toWei(3,"ether")})

from 是轉出的錢包 to是轉入的錢包 value是金額，單位已經轉成乙太。

#### 

如圖，在後方的客戶端終端機輸入轉帳指令，在前方執行鏈的終端機上便會出現一條Submitted transaction等待轉帳，當客戶端輸入miner.start()啟動挖礦時，即可將交易轉出。

#### 其他常用指令

查看當前區塊高度。

eth.blockNumbers

查看pending/queued。

txpool.tatus

查看已連接的節點。

admin.peers

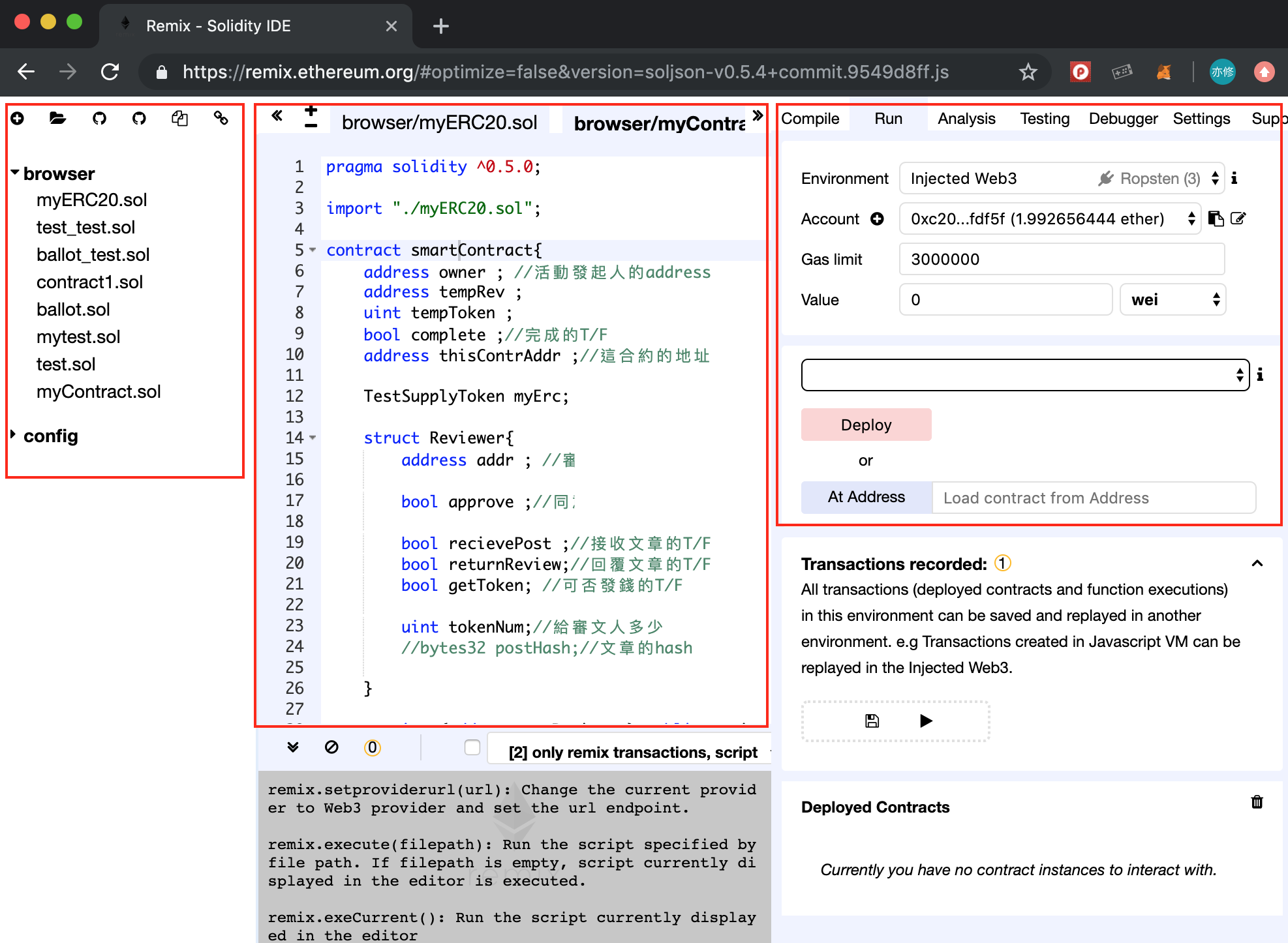
區塊鏈操作與測試暫時告一段落，之後要使用Web3j來當作客戶端連接私有鏈

## Solidity智能合約編寫

### REMIX IDE

在這裡我們一開始選用remix這個線上IDE來進行solidity程式碼的編譯，因為他能夠直接連上metamask錢包，方便錢包的建立與識別，再透過metamask錢包來連上測試鏈（之後會連接回私有鏈，為了方便測試先使用測試鏈），直接對合約的部署，對合約內方法進行測試。

不直接接上乙太坊的主鏈是因為每一個步驟，像是部署合約、執行合約功能等動作，都要消耗乙太幣給予礦工礦工費（也就是gas），現價來看，一顆乙太幣要價約250美元。



這邊挑幾個比較特殊的區塊來介紹。

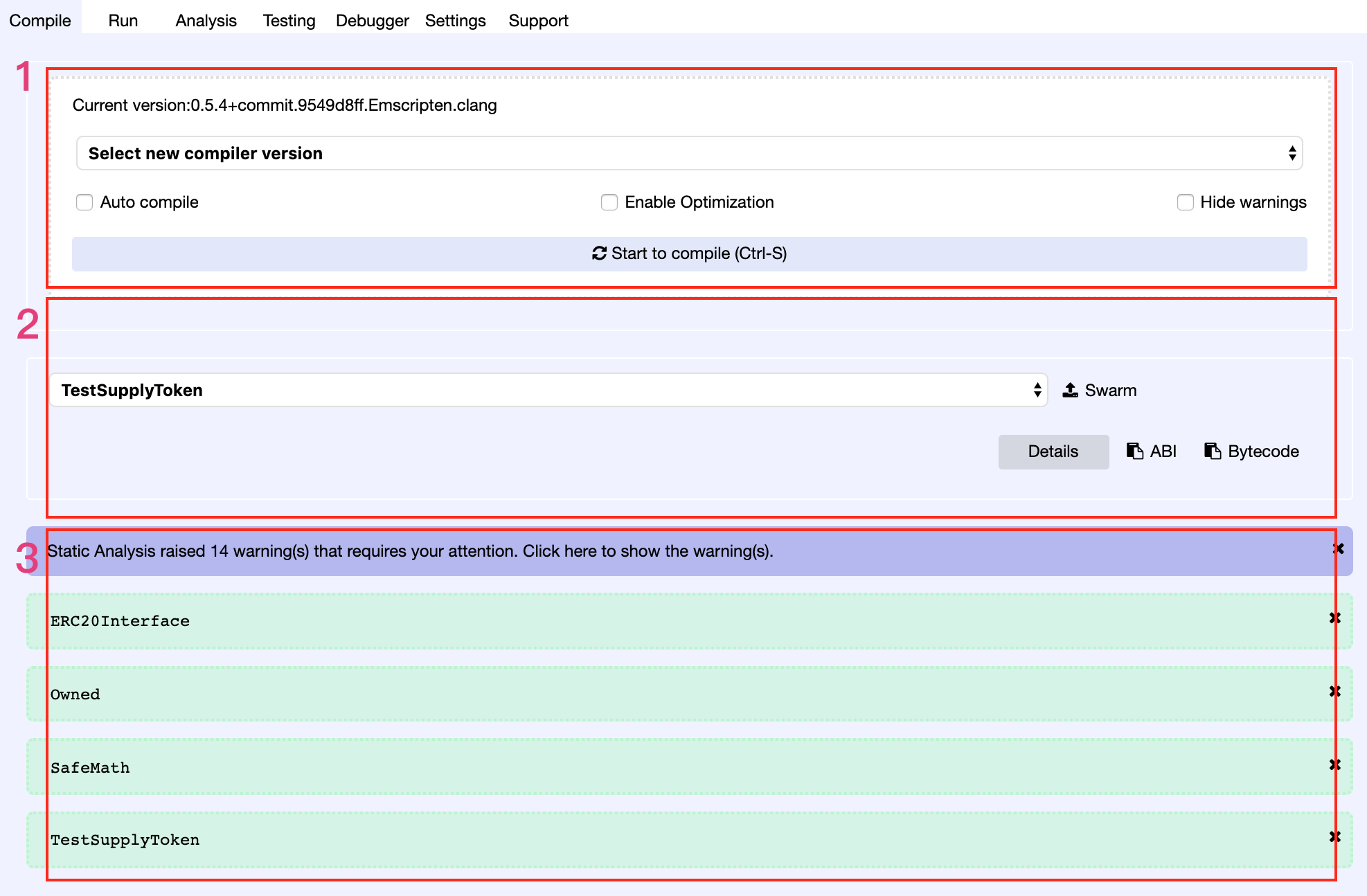
左邊區塊是目前存在此瀏覽器中的solidity檔案。

中間區塊則是目前撰寫中的程式碼。

右邊是此IDE的功能選項，接下來會針對有使用到的部分進行講解。

#### remix IDE功能選項

Compile



第一區塊

由於solidity此語言仍然不停的在更新，每一版本存在些許的差異，在這裡可以選取版本與進行Compile。

第二區塊

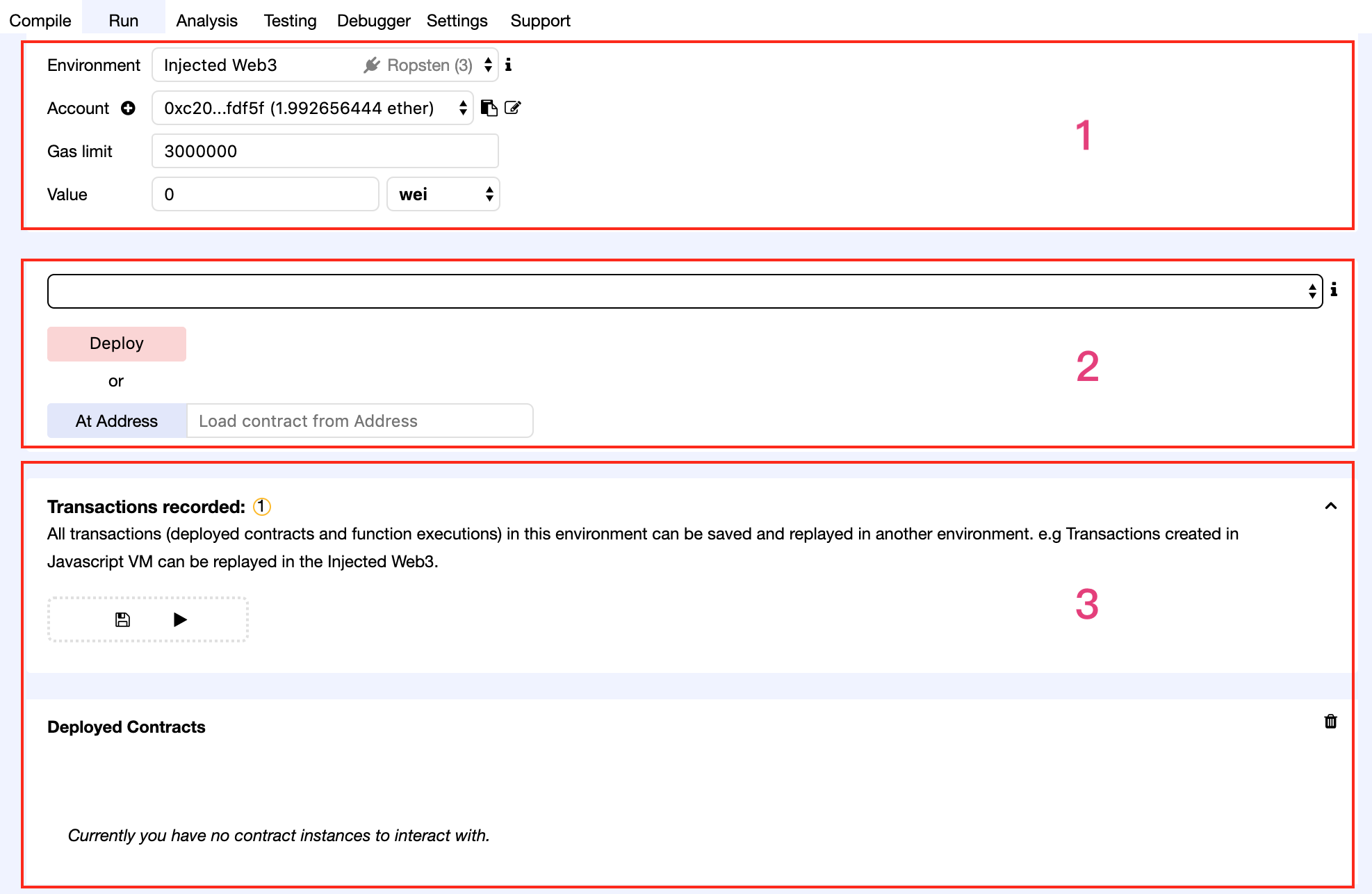
這裡則是選擇要編譯合約，一個.sol檔案內可以同時有許多合約，在選擇的合約編譯完成之後，可以複製出ABI與Bytecode來讓web3j將solidity語言編譯成需要使用的JAVA檔案。

此步驟其實也能透過本機安裝solc來執行，取得ABI與Bytecode。

第三區塊

如果編譯過程有出現錯誤，就會把錯誤或者warning顯示在這裡。

Run



第一區塊

Environment：可以選擇要執行的環境，私有鏈、測試鏈、公鏈等。

Account：執行合約的錢包，這裡已經連接到MetaMask錢包。

Gas limit：Gas上限。

value：多少乙太幣。

第二區塊

這裡選擇要Deploy到鏈上的合約，或者把已經在鏈上的合約載入進來使用，如要載入合約則把當初佈上區塊鏈時回傳的合約地址輸入到裡面即可。

若發佈的合約是要用ERC20代幣來交易，通常deploy後要輸入ERC20 token的合約地址。

第三區塊

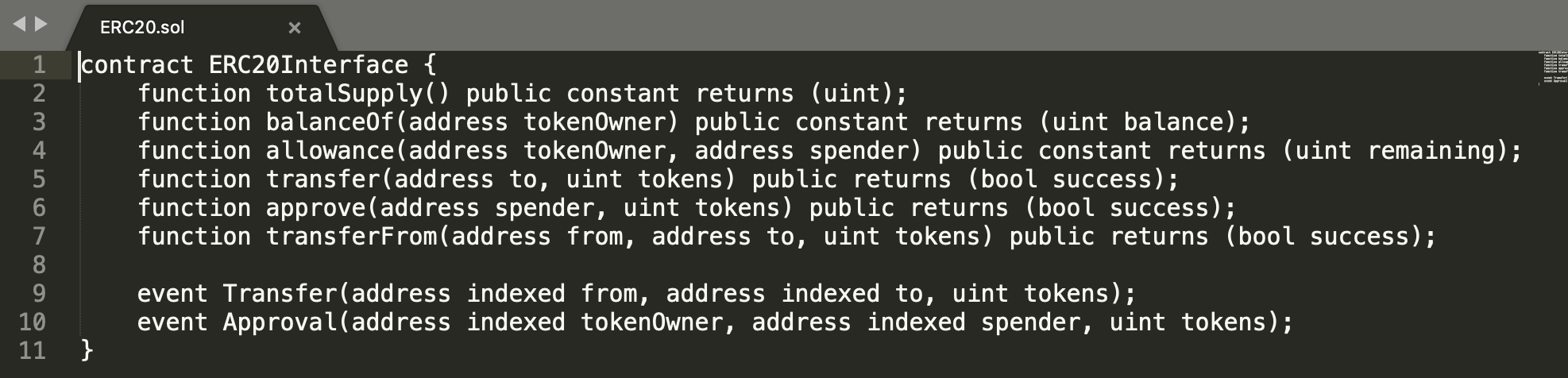
當發布合約或者載入合約後，能夠使用的方法會顯示在這裡，接下來只要根據自己寫的方法要輸入的參數輸入參數等，便能呼叫方法來使用。

### MetaMask錢包

一個google chrome的錢包插件，可以輕易的建立一個錢包與加入自己的ERC20代幣，來看交易的狀況，也能任意切換到要使用的鏈。

### ERC20合約編寫

ERC20合約主要是根據ERC20 Token Standard的標準來撰寫，這個標準定義了一個代幣需要擁有的貨幣機制，凡是符合此標準、根據此標準開發的的代幣，便能稱作為ERC20代幣，他是一種乙太坊上的應用，會將誰持有多少代幣統計在一張智能合約上，並當有ERC20代幣交易發生時，在那張智能合約上進行加減運算，來作為價值媒介的傳遞使用。



上圖為ERC20標準代幣所要求的功能，包含6個function與2個event

function totalSupply() 查看總發行量

function balanceOf() 查看帳號餘額

function allowance() 查看approve()中允許被提款的數量

function transfer() 轉帳交易

function approve() 允許某一帳號從一帳號中提款、與設定提款數量

function transferFrom() 從某一帳號中提款，最大數量是approve()中所限定的數量

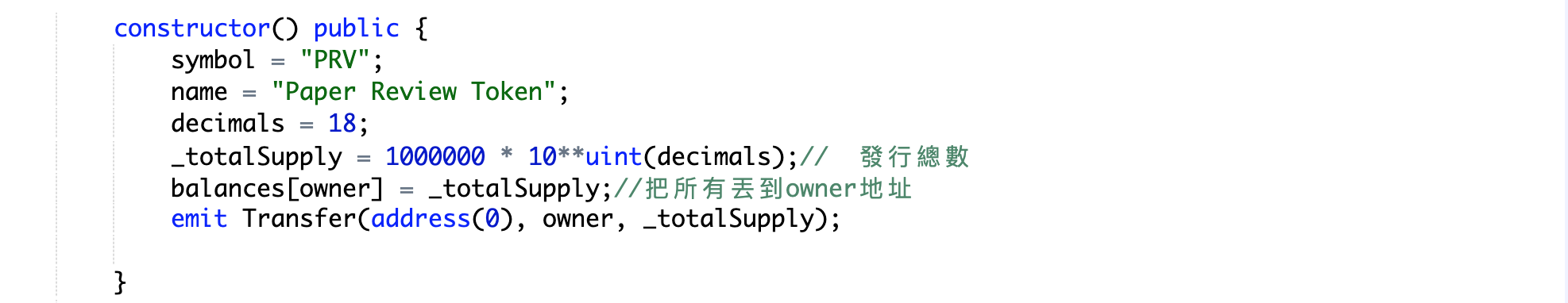
event Transfer()

event Approval()

event在這裡只是單純作為紀錄使用，不會影響合約的運作。

#### 主要合約方法：

除了上述的基本方法，在建構時也需要定義ERC20內的Token資訊。



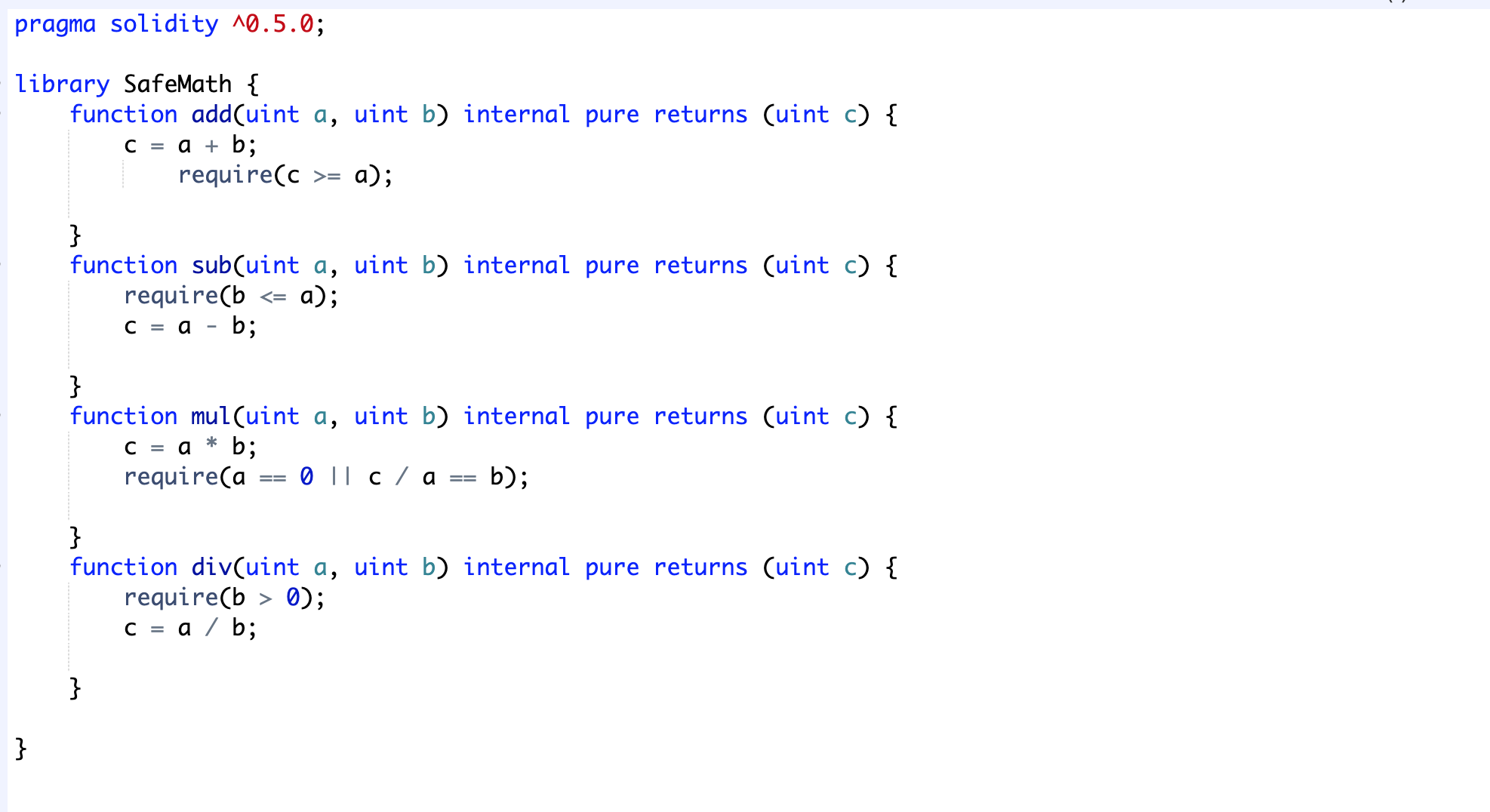
symbol 此Token所使用的代稱，如同Ethereum就稱之為ETH。

name 此Token的完整名稱，我們命名為Paper Review Token。

decimals 此Token最小會有幾位數，通常設定成18，與乙太幣相同。

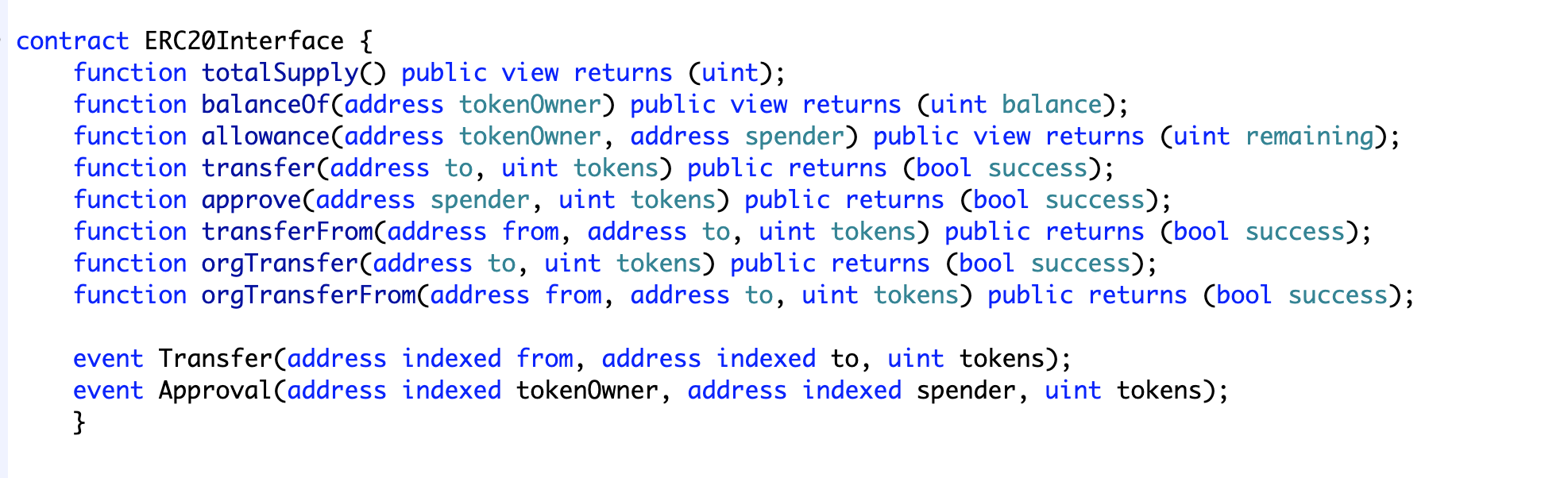
\_totalSupply 此Token的總發行量。

本ERC20合約內其他方法。



上圖：一開始先設定solidity版本，要求為0.5.0版本以上。

SafeMath方法：為了避免出現負數在ERC20合約內。



上圖：我們的ERC20介面，會根據這裡來進行功能的實作，新增了兩個方法，會於下方進行介紹。



為了轉換ERC20合約的擁有者而實作，可以經過此方法來讓合約擁有者進行轉換，並且實作modifier()方法，只有此合約擁有者能來呼叫此方法，以提升安全性。

ERC20功能實作：

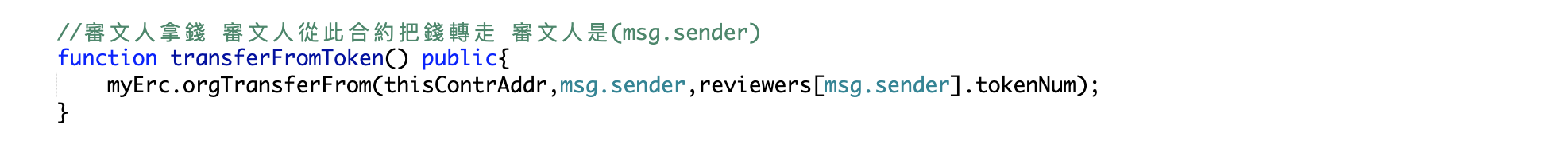


上圖：

在這裡我們特別多加了兩個方法orgTransfer()、orgTransferFrom()，基本上是根據原本的兩個方法來修改的，因為此兩個方法的呼叫者，在呼叫此2方法時會隔著一張智能合約，此時本ERC20合約的msg.sender會是中間那張智能合約的地址，不是我們所要的，因此將這兩的方法中的msg.sender改成tx.origin，便會留著呼叫者的地址，而不是中間那張智能合約。

### 智能合約編寫（主要的驗證功能）

#### 主要合約方法：



isApprove()

當審稿者同意審稿時會呼叫這個方法，使reviewers陣列內approve登記為true

sendArticle()

得知審稿者同意審稿後，發文者呼叫此方法，且在reviewers陣列中登記要給予的token數量

isRecievePost()

當審稿者接收到文章後呼叫此方法，使reviewers陣列內recievePost登記為true

isReturnReview()

當審稿者回覆審稿內容時，呼叫此方法，使reviewers陣列內returnReview登記為true

isGiveToken()

接收到回覆內容時，確認內容可以接受，呼叫此方法，使reviewers陣列內getToken登記為true

approveToken()

驗證上述reviewers陣列內的布林值都已經登記成true，便可以將錢轉到此合約中

transferFromToken()

完成上述程序後，審文者便可以從合約中提出當初約定的金額

## Solidity使用Web3j轉換成Java可編譯檔案

由於區塊鏈是讀取Bytecode在運行，我們在remix將編譯好的solidity合約可以轉換成ABI和Bytecode，將其複製下來，透過web3j將合約轉換成Java檔連接上我們的專案。

首先要先將他們裝成 .abi和 .bin的兩個檔案，再透過Web3j.exe轉換，如下：web3j-4.2.0/bin/web3j solidity generate -b 合約.bin -a 合約.abi -p"裡面資料夾name" -o"外面資料夾name"

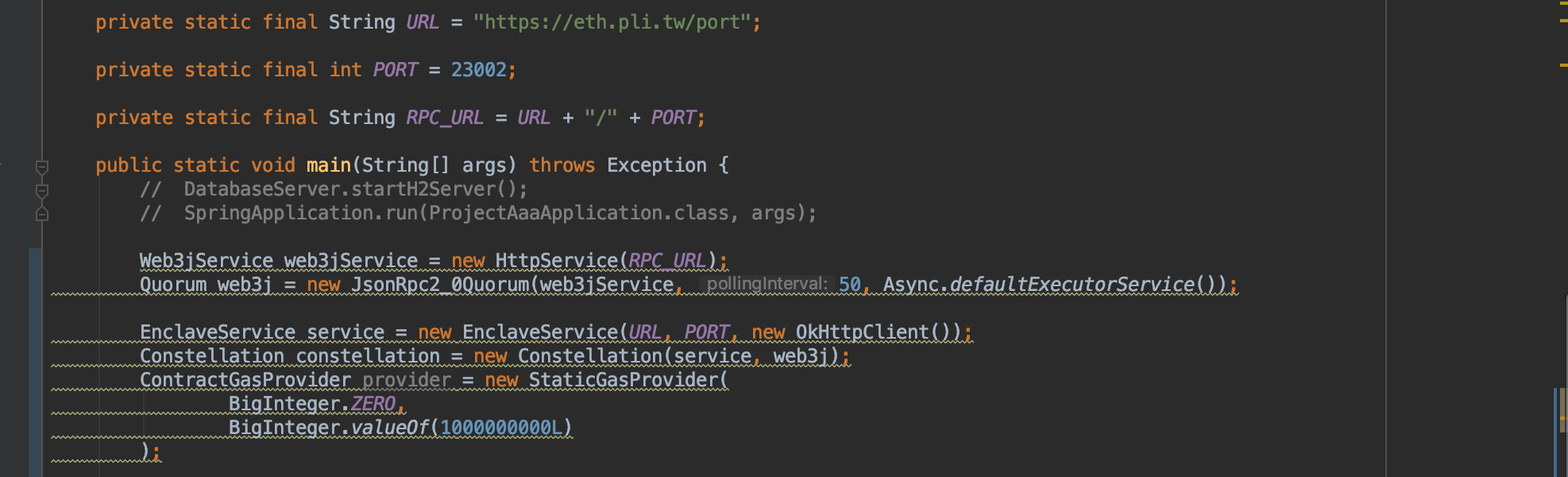
最後會產出JAVA檔案，直接加入專案中，再透過web3j來使用。

## 使用web3j連接區塊鏈與JAVA專案

在基本的幾個區塊鏈架構部分都測試完、solidity程式也撰寫完轉成java檔案後，接下來就是要藉由一個專案架構來將他們全部組裝起來，在網頁應用部分，我們使用的是JAVA Spring boot來撰寫後端，並連接h2資料庫，然後再透過web3j來將網頁應用連接上區塊鏈。

主要方法介紹：

Java專案連上Quorum鏈

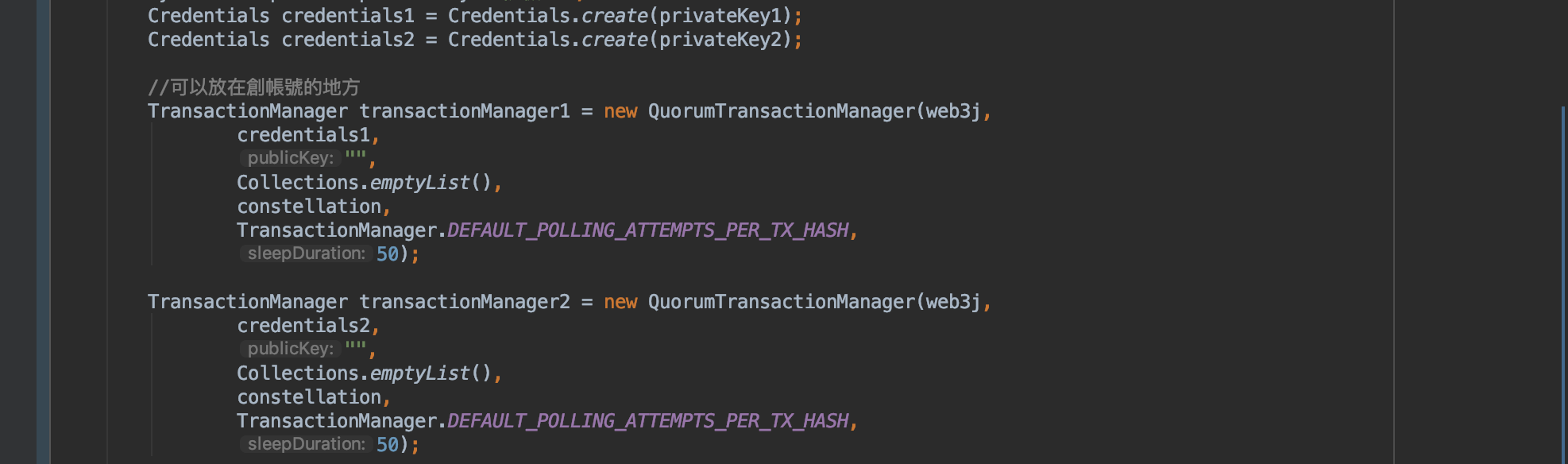


在最一開始串接，首先就是要把專案連接到鏈上，透過上圖的方法便能將其連接到Quorum鏈上，之後才能繼續發布合約、呼叫合約應用的操作。

產生PrivateKey

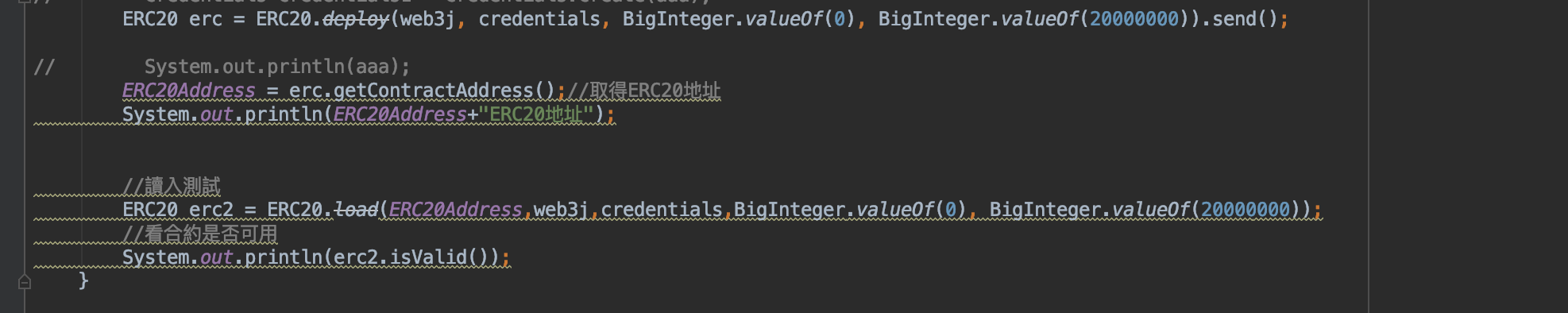
在我們的網頁應用中創辦帳號時，每個人都會獲得一把PrivateKey，以便後續發佈與執行合約時的身份驗證。

Credential TransactionManager



在這裡使用者需要利用先前所產生的privateKey來產生cerdential，credential在之後載入合約時是必須設定的參數，說明呼叫此合約者的身份。再來是TransactionManager，比credential更多了些參數，是Quorum所要求，用法與credential相似。

Java實作出發佈ERC20合約與代幣、讀入合約



首先我們發佈了一張ERC20合約，成功後他會使合約發佈者獲得1000000顆PRV Token，來成為我們網頁應用中價值的基準。

在成功deploy()合約上到區塊鏈中之後，再利用Web3j中的load()方法能實現把合約載入Java專案中，等到載入完成之後便可以直接調用先前使用web3j.exe來轉換ABI/Bytecode後所產生的JAVA檔案中的方法。

Java實作出ERC20合約中的getBalance方法



利用已經載入的合約來呼叫方法，這裡呼叫的是ERC20.java中的getBalance方法，他能夠取得ERC20合約中所登記，此合約呼叫者的Token餘額。

Java實作出智能合約方法



這裡是利用JAVA實作出我們用來驗證的智能合約的方法，根據每個方法函式所需的參數填入對應項目，最後資料都會記錄在區塊鏈中。

## QUORUM

因為在進行測試時遇到了合約在佈上我們建立的私有鏈後，會有無法被調用的狀況，我們懷疑是因為我們自行架設的私有鏈只有一個客戶端在進行挖礦驗證，導致驗證速度不夠快，在合約還沒被完全部署完成的情況下就被調用了，造成無法成功調用合約內方法，於是我們替換本機私有鏈成部署在學校內伺服器的quorum鏈上。

## 資料庫程式碼 實作說明

MVC架構的三大核心：Model、View、Controller來進行前後台串接，實現資料傳輸。

料庫相關連的兩大方法：

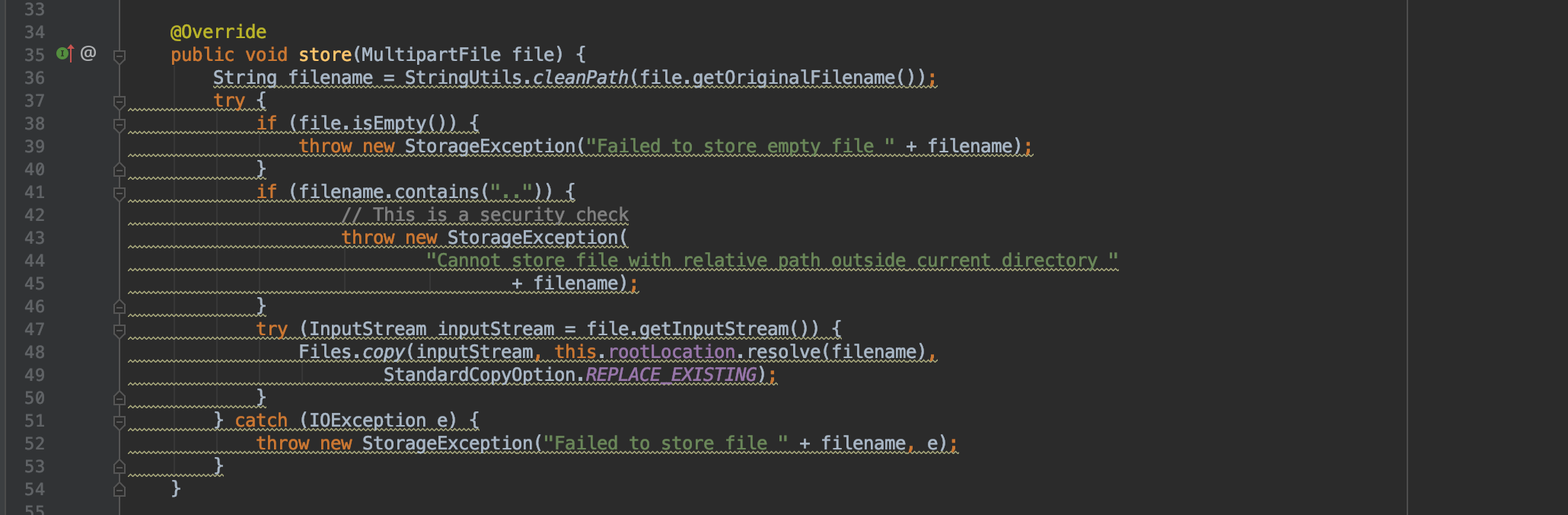
1. Service()：藉由此方法與後端資料庫進行串接取值的動作
2. Dao()：透過此方法去針對資料庫進行CRUD的動作

### Service

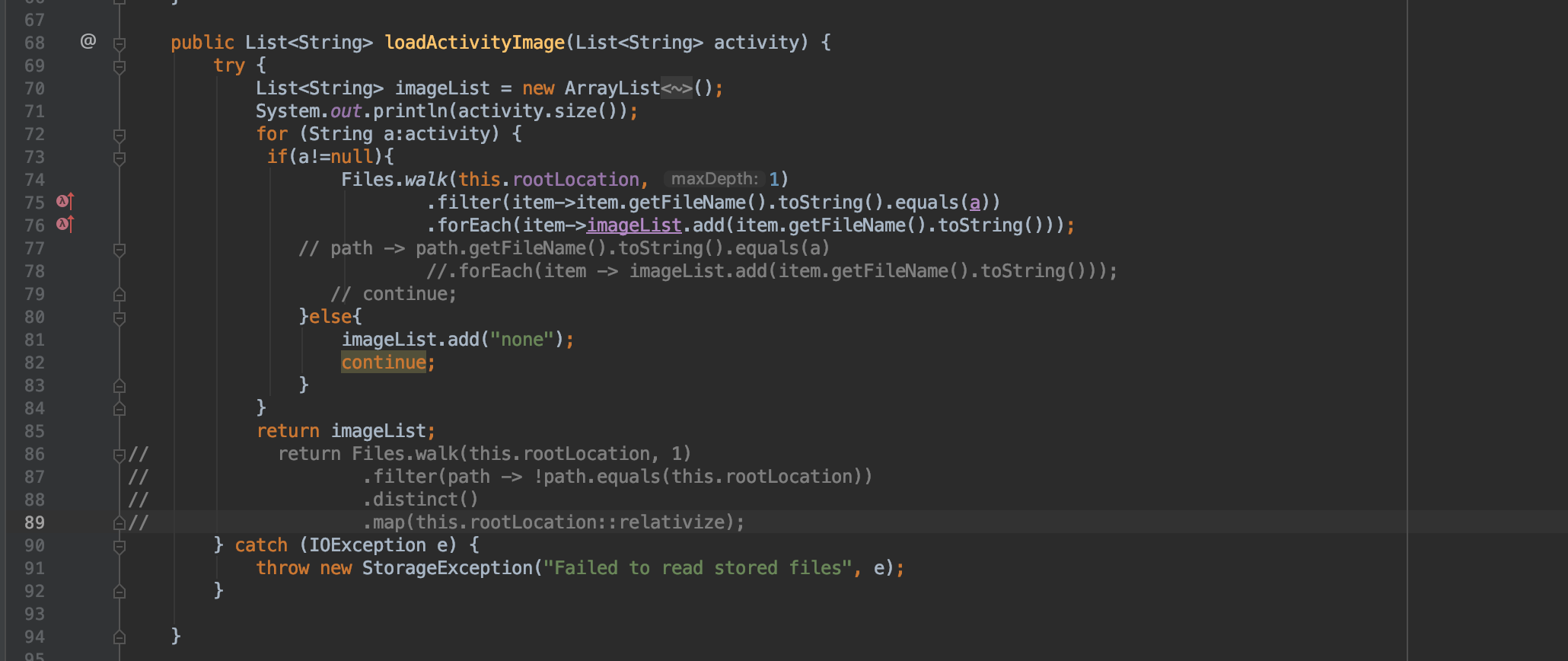
我們建構了六個Service來處理平台上需要從資料庫取值的動作：

1. MemberService：註冊時將會員資訊存進資料庫中
2. ActivityService：建立活動時將活動資訊存進資料庫中
3. ActivityImageFileService：儲存建立活動時所上傳的照片
4. ArticleService：上傳論文時將論文資訊存進資料庫中
5. ArticleFileService：儲存上傳論文時的論文檔案
6. ArticleReviewService：儲存審核論文的資訊進資料庫中

我們將針對需要儲存檔案的ActivityImageFileService以及ArticleFileService上傳與抓取檔案進行說明。以ActivityImageFileService以範例進行說明：



上圖ActivityImageFileService，建構儲存檔案，並抓到檔案名稱與檔案位置。並針對上傳失誤進行確認與測試。

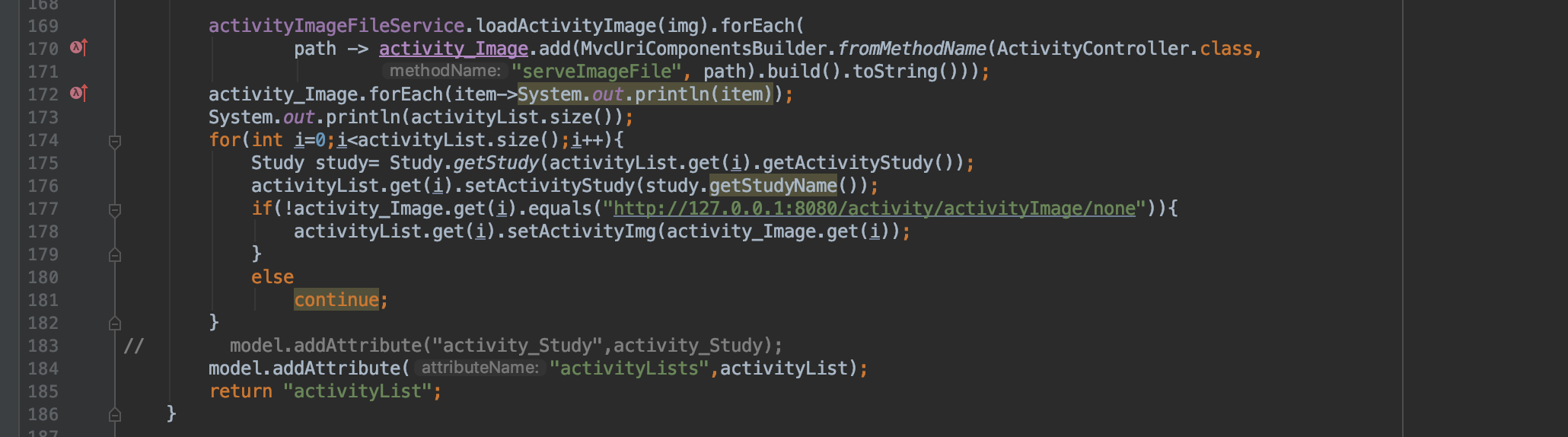


上圖為ActivityImageFileService，當下載檔案時根據指定路徑與檔案名稱。透過loadActivity()抓取資料名稱與路徑。



上圖為ActivityController，透過serveImageFile()的方法指定在抓取檔案時

的路徑以及檔名。完成抓取後建構一個網址，並將檔案上傳至網頁body的部分。



上圖為ActivityController，在完成serveImageFile()後將得到的網址回傳並透過連結的方式顯示於頁面。

## 寄件系統說明

當系統收到有關於指派事件、退稿事件時，我們將會利用電子郵件台通知使用者接收到任務或審核回覆被退件等訊息。

下列四項事件會觸發以下四種方法，產生email寄出：

1. sendSimpleMessageAssign()：

當被活動創辦人指派審核文章時，審稿人會收到指派通知

1. sendSimpleMessageConfirm()：

當審稿人同意參與該審核活動時，活動創辦人收到審稿人同意參加的通知

1. sendSimpleMessageRefused()：

當審稿人拒絕參與該審核活動時，活動創辦人收到審稿人拒絕參加的通知

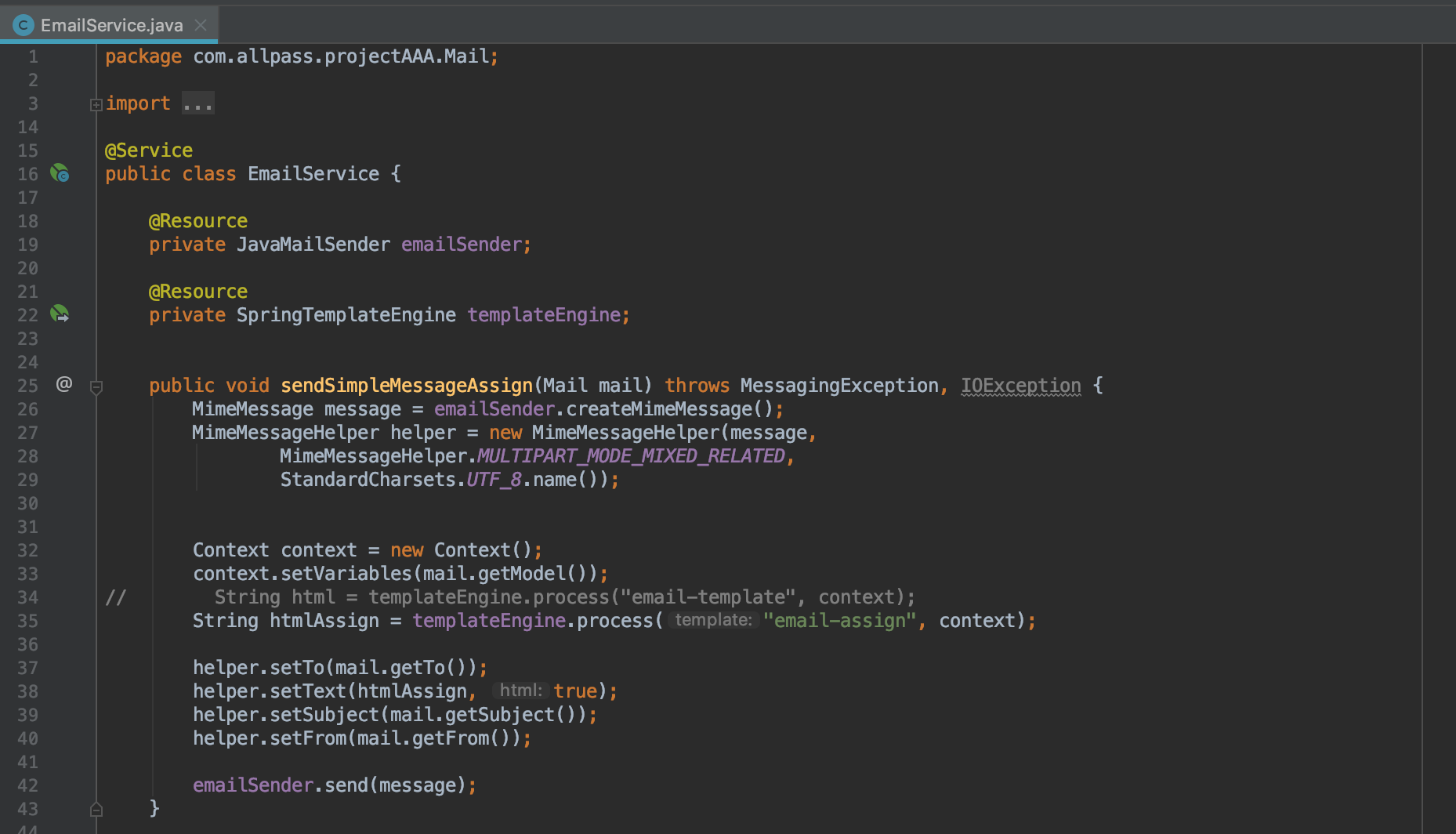
1. sendSimpleMessageReturn()：

當活動創辦人退件審稿人的回覆時，審稿人會收到退件通知

接下來我們將以sendSimpleMessageAssign()為範例介紹：

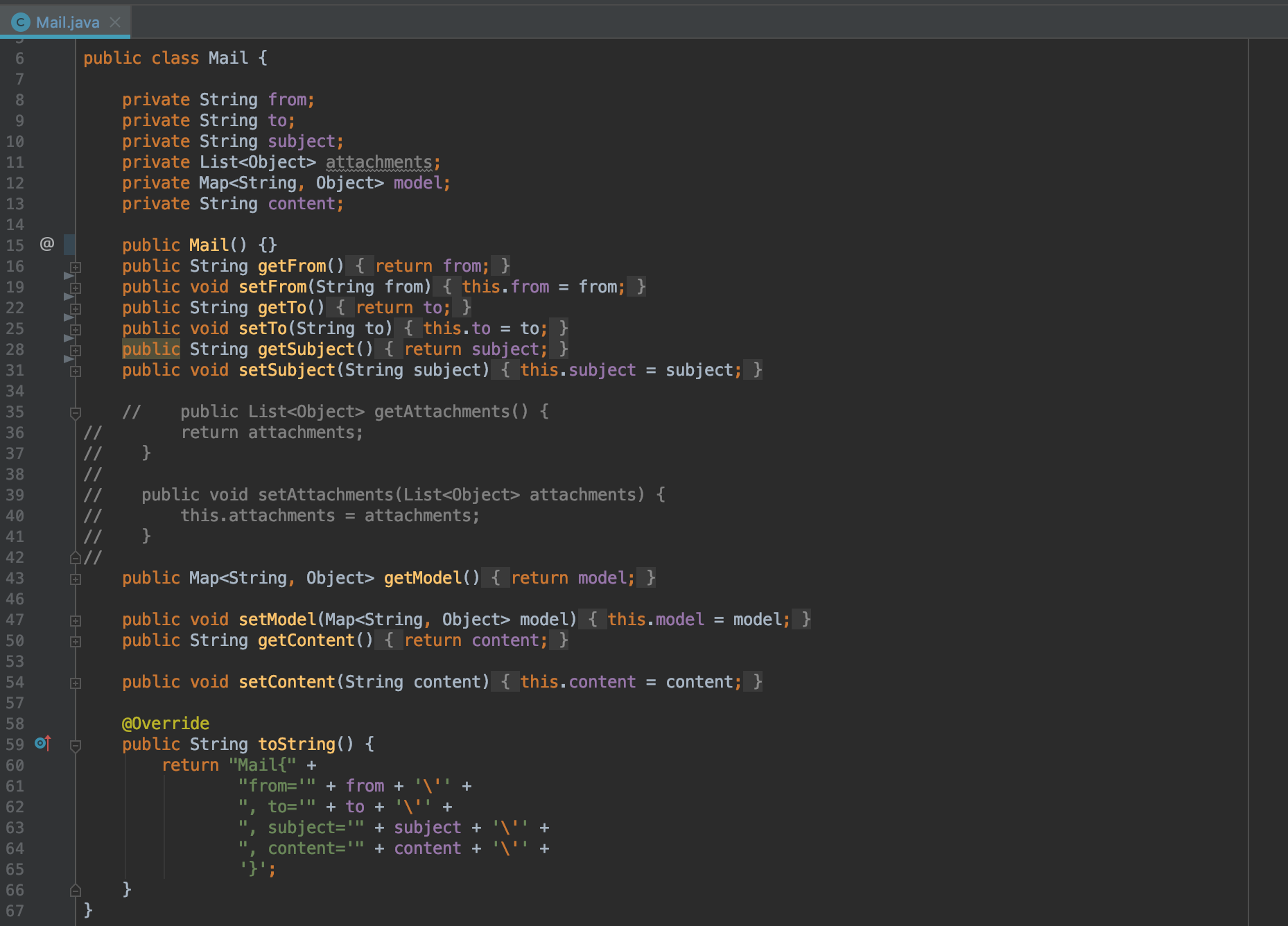
#### EmailService

透過EmailService.java建構出寄送email的方法以及model

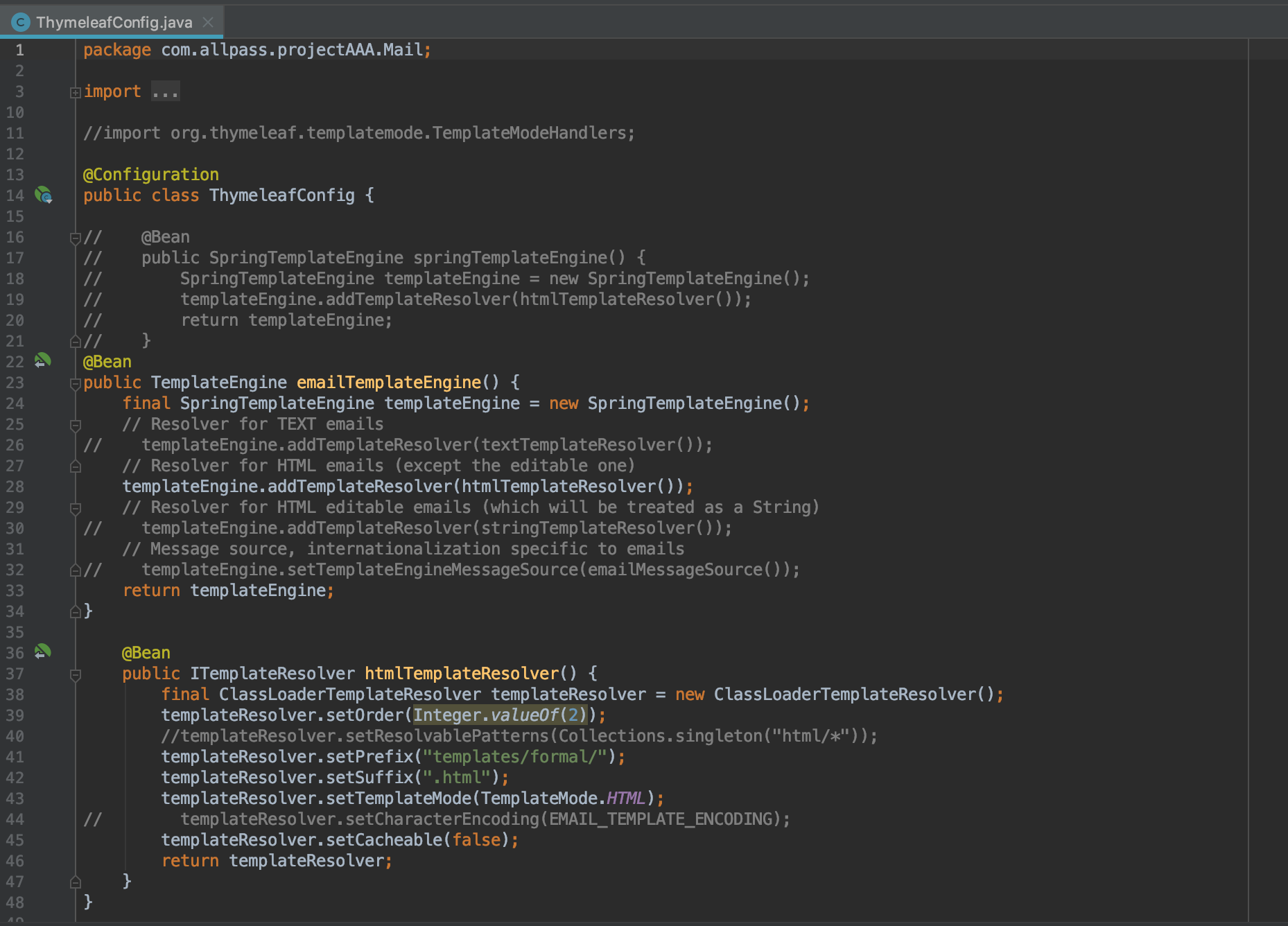
上圖:透過SpringTemplateEngine來支援讀取Html5的模板。

建構sendSimpleMessageAssign()的方法來指定，當寄出此封郵件時需要需要讀取email-assign.html。並指定郵件中具備寄件人、內文、標題以及收件人等基本欄位。

#### Mail

上圖，此方法去get與set信件的寄件人、內文、標題以及收件人。想要附上附件時亦透過此方法設定。

#### ThymeleafConfig



上圖，透過emailTemplateEngine()的方法新增Html的模板。再藉由htmlTemplateResolver()指定.html 檔所存在之路徑，並指定檔案類型。

5.內容展示（html截圖）

### 首頁

### 登入

### 創辦活動

### 管理活動

### 活動列表

### 參加活動

### 上傳文章

### 派發審核

### email頁面

### 參加與否

### 審核文章

### 查看審核

### QA

### （資料庫）

### （區塊鏈管理端）

6.結論

# 工作分配

鄭翔：整理資料表、需求與分析、

李亦修：定色

桑傳雲：

王璟元：

# 未來展望

1. 未來可以增加儲值的功能，增加市場競爭。讓對於審核論文有急需的作者有加速審查的機會。
2. 建立評價系統，可針對活動活動內容進行評價；針對審核人給予的回覆進行評價，未來活動創辦人在指派時可透過評價考慮指派對象
3. 未來可將平台應用至不同領域

# HTML定色

### 背景顏色

KUROTSURUBAMI 黑橡 #0B1013

GOFUN 胡粉 #FFFFFB

QA伸縮 #2f474b

info伸縮 #4F726C

### Button顏色

button special #8cc9d1

button special hover #91b9bf

button #25383B

button hover #2f474b

button alt #transparent

### 狀態鈕顏色

BENIKABA 紅樺 #B54434

UMEZOME 梅染 #E9A368

WASURENAGUSA 勿忘草 #7DB9DE

YANAGIZOME 柳染 #91AD70

GINNEZUMI 銀鼠 #91989F

7.參考資料

remix：

<https://blockcast.it/2017/06/17/eth-and-eth-token-wallet-series-metamask/>

web3j與智能合約、ERC20：

<https://www.itread01.com/content/1547721013.html>

<https://blog.csdn.net/kzcming/article/details/81510730>

<https://blog.csdn.net/kzcming/article/details/81735520>

<https://mifengcha.com/news/5c63e750babcc754b64e6ce9>

<https://web3j.readthedocs.io/en/latest/smart_contracts.html>

<https://stevenocean.github.io/2018/04/06/web3j-ethereum-token.html>

<https://stevenocean.github.io/2018/04/06/web3j-ethereum-token.html>

<https://ethereum.stackexchange.com/questions/32407/how-to-get-balanceof-of-a-deployed-erc20-contract-using-web3j>

<https://www.weaf.top/posts/75cd4959/>

<https://www.twblogs.net/a/5baabfe82b7177781a0e856f>

<https://blog.csdn.net/kzcming/article/details/81511679>

<https://zhuanlan.zhihu.com/p/36144853>

<https://blog.csdn.net/loy_184548/article/details/78668180>

<https://www.jianshu.com/p/e7e7a4efb9a6>

<https://www.jianshu.com/p/3671b65462aa>

[https://medium.com/hackoin-taiwan/ethereum-erc20-token-standard-以太坊代幣標準介紹-b7bc58171021](https://medium.com/hackoin-taiwan/ethereum-erc20-token-standard-%E4%BB%A5%E5%A4%AA%E5%9D%8A%E4%BB%A3%E5%B9%A3%E6%A8%99%E6%BA%96%E4%BB%8B%E7%B4%B9-b7bc58171021)

私有鏈：

<https://ithelp.ithome.com.tw/articles/10203964>

<https://ethereum.stackexchange.com/questions/28703/full-list-of-geth-terminal-commands>

<https://www.ldsun.com/da-jian-ethsi-lian-bing-chuang-jian-token/>

<https://zhuanlan.zhihu.com/p/36142350>

<https://g2ex.github.io/2017/09/12/ethereum-guidance/>

架構：

<https://www.jianshu.com/p/f045861a18e4>

JavaMail & Thymeleaf Mail

[Sending email in Spring with Thymeleaf - Thymeleafhttps://www.thymeleaf.org/doc/articles/springmail.html](https://www.thymeleaf.org/doc/articles/springmail.html)

<https://memorynotfound.com/spring-mail-sending-email-thymeleaf-html-template-example/>

DateRangerPicker

<http://www.daterangepicker.com>

前端介面：

<https://www.w3schools.com/default.asp>

其他問題：

<https://blog.csdn.net/sinat_34093604/article/details/53763448>

<https://blog.csdn.net/qq_24300625/article/details/54891502>

顏色：

<http://nipponcolors.com>