程式教育實戰 - DOMjudge使用攻略

程式教育實戰 - DOMjudge使用攻略

前言

DOMjudge是一個競賽評判系統，目前在各大比賽中皆有使用。除了作為競賽使用，其自動評判程式的特性也可以用於程式設計課程的作業及測驗等用途。
然而，設計給競賽用的工具在作業及考試等等需要的一些功能方面還是不太足夠(如Dump成績、程式碼)。本文將以儘量簡單的方式從頭對DOMjudge做說明，希望能夠對要將DOMjudge作教學用途的教師有所幫助

撰寫時DOMjudge版本：7.3.3

常用連結：

- Domjudge Manual (https://www.domjudge.org/docs/manual/7.3)
- API Documentation (https://www.domjudge.org/demoweb/api/doc)
- 範例Code on Github (https://github.com/johnson10024/DOMjudgeForEdu/tree/main/docker_dockerCompose)

  > domjudge_jury.tex, domjudge_jury.pdf: DOMjudge簡易說明(裁判版)
  > domjudge_student.tex, domjudge_student.pdf: DOMjudge簡易說明(學生版)

一、利用Docker及docker-compose部署DOMjudge

> 本章範例(Github)：docker_dockerCompose/docker-compose.yml

Docker 是一種軟體平台，透過將軟體封裝到名為容器的標準化單位中，可以快速地建立、測試和部署應用程式。

docker-compose 則可以透過指令檔案部署Docker
本文環境以 Ubuntu 20.04 為例

1. 安裝docker及docker-compose

```
$sudo apt install docker docker-compose
```

2. docker-compose.yml

將下面的檔案命名為「docker-compose.yml」，並決定備份用資料夾

```
version: '3'

services:
        dj-mariadb:
                image: mariadb
                volumes:
                        #「:」前面的部分是備份資料庫的路徑，可以避免重新部署後資料消失
                        - /absolute/path/to/backup/directory:/var/lib/mysql
                environment:
                        - MYSQL_ROOT_PASSWORD=rootpw
                        - MYSQL_DATABASE=domjudge
                        - MYSQL_USER=domjudge
                        - MYSQL_PASSWORD=djpw
                ports:
                        - 13306:3306
                command:
                        --max-connections=1000

        dj-domserver:
                image: domjudge/domserver:latest
                volumes:
                        - /sys/fs/cgroup:/sys/fs/cgroup:ro
                environment:
                        - CONTAINER_TIMEZONE=Asia/Taipei
                        - MYSQL_HOST=mariadb
                        - MYSQL_ROOT_PASSWORD=rootpw
                        - MYSQL_DATABASE=domjudge
                        - MYSQL_USER=domjudge
                        - MYSQL_PASSWORD=djpw
                ports:
                        #「:」前面的port number是外部連進去的port，可以自由設置
                        - 9487:80
                links:
                        - dj-mariadb:mariadb
        #judgehost可以依需要多設幾個，名字、hostname、DAEMON_ID記得改
        dj-judgehost:
                image: domjudge/judgehost:latest
                privileged: true
                hostname: judgedaemon-0
                volumes:
                        - /sys/fs/cgroup:/sys/fs/cgroup:ro
                environment:
                        - CONTAINER_TIMEZONE=Asia/Taipei
                        - DAEMON_ID=0
                        #DOMSERVER_BASEURL可以指定domserver的IP，從而在另一個終端開judgeh
                        #- DOMSERVER_BASEURL=127.0.0.1
                links:
                        - dj-domserver:domserver

        dj-judgehost_2:
                image: domjudge/judgehost:latest
                privileged: true
                hostname: judgedaemon-1
                volumes:
                        - /sys/fs/cgroup:/sys/fs/cgroup:ro
                environment:
                        - CONTAINER_TIMEZONE=Asia/Taipei
                        - DAEMON_ID=1
                links:
                        - dj-domserver:domserver
```

這個檔案是docker-compose用來configure用的，其實就是把參數放到檔案裡的概念。要修改的部分及功能概述都寫在註解

原本的指令可以參考 docker頁面 (https://hub.docker.com/r/domjudge/domserver/)

p.s. 之前踩過一個小坑是yml的縮排要全用空格 有tab混在裡面會出錯

3. 開始部署

後面大部分操作都需要sudo，就先省略，遇到問題先加上去試看看就對了

```
$docker-compose up -d
```

記得要cd到yml檔所在的目錄

如果要用的檔名不叫「docker-compose.yml」的話可以用 `-f` 參數:

```
$docker-compose -f file_name.yml up -d
```

第一次執行會需要先把Docker image載下來，會比較久

```
Creating network "documents_default" with the default driver
Pulling dj-mariadb (mariadb:)...
latest: Pulling from library/mariadb
405f018f9d1d: Pull complete
7a85079b8234: Pull complete
579c7ff691b1: Pull complete
4976663b5d6d: Pull complete
169024b1fb13: Pull complete
c0ffe8ce897f: Pull complete
b583c09d23c3: Pull complete
9b9f0c08d08f: Pull complete
9cd51f984586: Pull complete
d9f506bb8aca: Pull complete
24d689f79ba4: Pull complete
Digest: sha256:88fcb7d92c7f61cd885c4d309c98461f3607aa6dbd57a2474be86e1956b36d13
Status: Downloaded newer image for mariadb:latest
Pulling dj-domserver (domjudge/domserver:latest)...
latest: Pulling from domjudge/domserver
a9a32f3c5277: Pull complete
7338a3e49153: Pull complete
16b34b2e15d8: Pull complete
0a4951a0fa7d: Pull complete
1473d5109197: Pull complete
0955c3efe376: Pull complete
7fb6306e0620: Extracting [==================================================>]     291B/291B
1aa5c56983b2: Download complete
f73281d9f3e6: Download complete
```

看到這樣就是好了

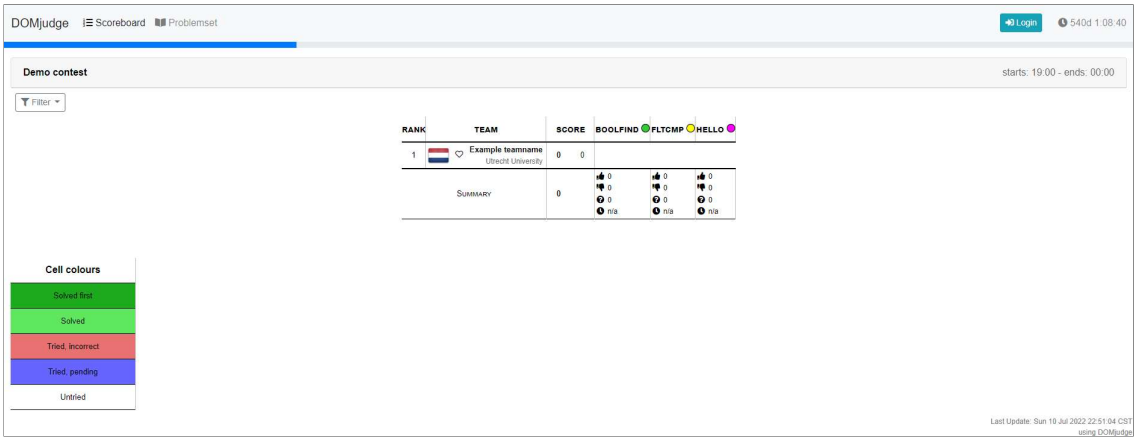
```
Creating domtest_dj-mariadb_1 ... done
Creating domtest_dj-domserver_1 ... done
Creating domtest_dj-judgehost_1    ... done
Creating domtest_dj-judgehost_2_1 ... done
```

4. admin帳號與judgehost

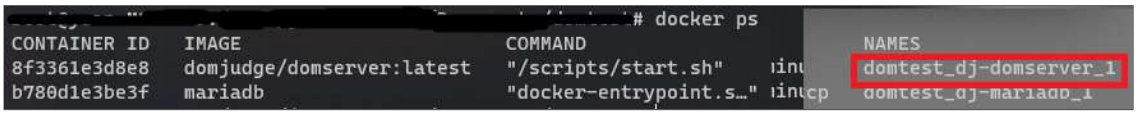
這時候可以用下面的指令確認一下已經跑起來的容器有哪些(想到就可以用一下)

```
$docker ps
```

這時候應該看不到judgehost有上去，先不用緊張
先到瀏灠器上打上 `localhost:9487` (剛剛在yml檔裡設定的port number)看看，應該就會看到登入畫面了(這時候已經可以透過網址連進來了，只要把port number指定好就行)



現在需要去找到admin帳號的密碼
先回到Command line輸入下面的指令 **(domserver_container_name替換成docker ps出來NAMES那欄domserver的container)**




```
$docker exec -it domserver_container_name cat /opt/domjudge/domserver/etc/initial_admin_pas
```

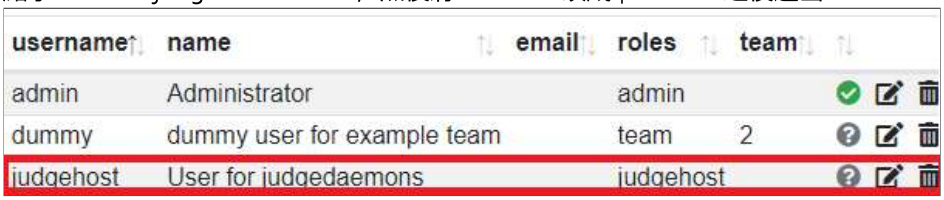
點擊右上角的Login按鈕，帳號輸入admin，密碼用剛才指令跑出來的東西

***initial_admin_password的密碼登不進去的情況**
用改密碼的指令重新設定一個

```
$docker exec -it domserver_container_name /opt/domjudge/domserver/webapp/bin/console domjud
```

接下來要設定judgehost，不然是沒辦法評測程式的

登入後(或是按左上角的DOMjudge)會看到 DOMjudge Jury Interface
點擊 User -> judgehost -> Edit，然後將Password改成 `password` 之後送出



接著再跑一次 docker-compose.yml

```
$docker-compose up -d
```

成功的話會有4(2+judgehost數)個containers，都寫up-to-date，如果沒有就多up幾次，再沒有可能就是前面有錯

(下 docker ps 指令應該要出現4個containers)

恭喜你！這樣就完成基本的架設了

二、Docker基本指令及DOMjudge jury Interface

> 本章範例(Github)：docker_dockerCompose/docker-compose.yml

Docker/docker-compose基本指令

- 部署DOMjudge

```
#docker-compose.yml
$docker-compose up -d
#其他檔案
$docker-compose -f file_name.yml up -d
```

如果有Container一直up不上去，可以把-d參數拿掉，會看得到錯誤訊息。

- 進入特定Container

```
$docker exec -it CONTAINER_NAME COMMAND
#範例:
$docker exec -it domtest-domserver_1 bash
```

CONTAINER_NAME一樣可以下 docker ps 看，COMMAND是一次性指令，如果要一次做多一點事可以先進 `bash` 。

- 下架Container

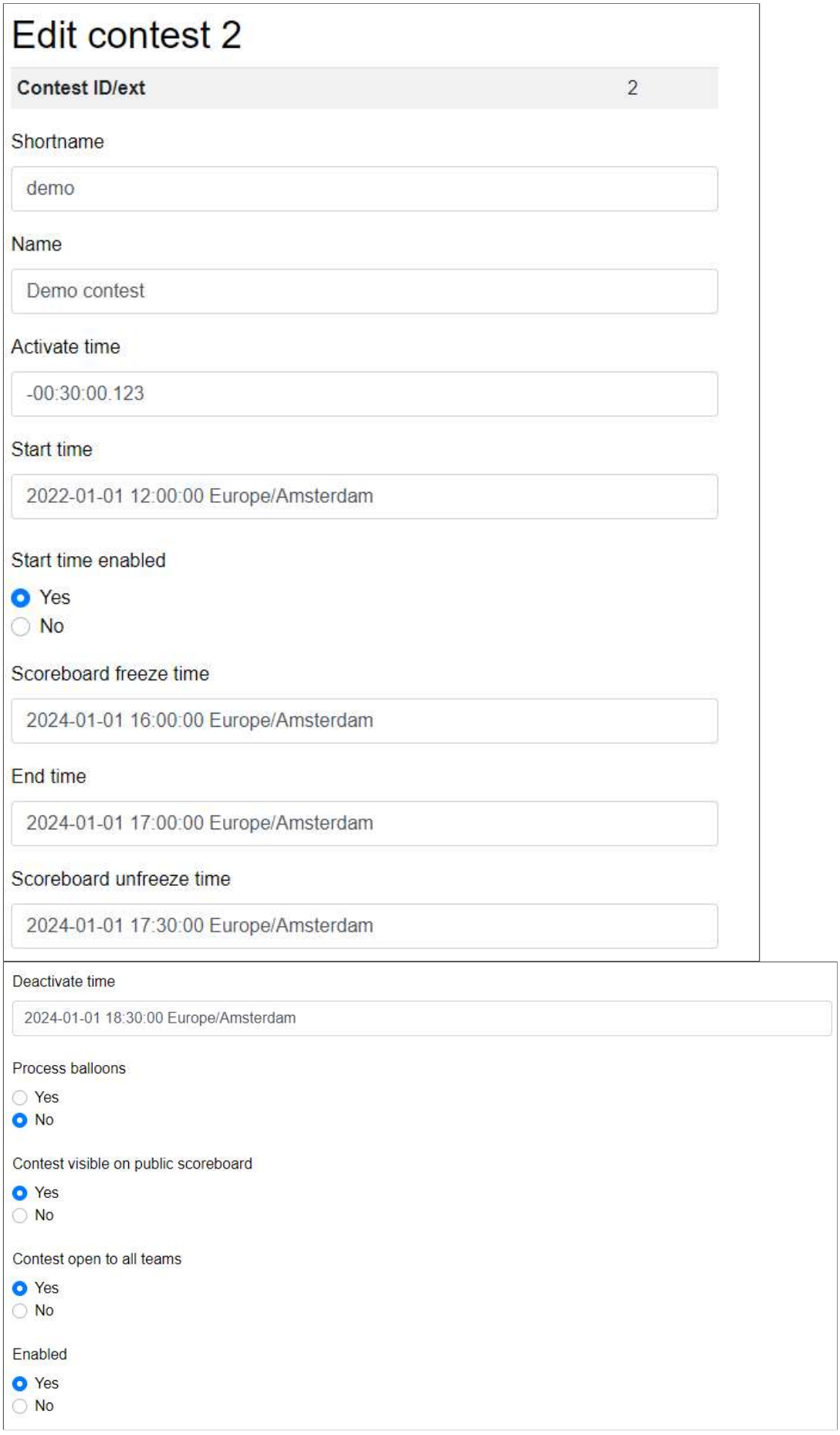
```
#docker-compose.yml
$docker-compose down
#其他檔案
$docker-compose -f file_name.yml down
```

DOMjudge介面

- *本區副標題有 #Contest、#Languages、#Problems、#Teams、#Team Categories、#Users、#Configurations、#Navigation Bar、#其他等等，可善用搜尋功能查詢

- 本文僅針對常用功能、欄位介紹，截圖也不一定完整

- #Contest表單


Edit contest 2
Contest ID/ext 2
Shortname
demo
Name
Demo contest
Activate time
-00:30:00.123
Start time
2022-01-01 12:00:00 Europe/Amsterdam
Start time enabled
Yes
No
Scoreboard freeze time
2024-01-01 16:00:00 Europe/Amsterdam
End time
2024-01-01 17:00:00 Europe/Amsterdam
Scoreboard unfreeze time
2024-01-01 17:30:00 Europe/Amsterdam
Deactivate time
2024-01-01 18:30:00 Europe/Amsterdam
Process balloons
Yes
No
Contest visible on public scoreboard
Yes
No
Contest open to all teams
Yes
No
Enabled
Yes
No

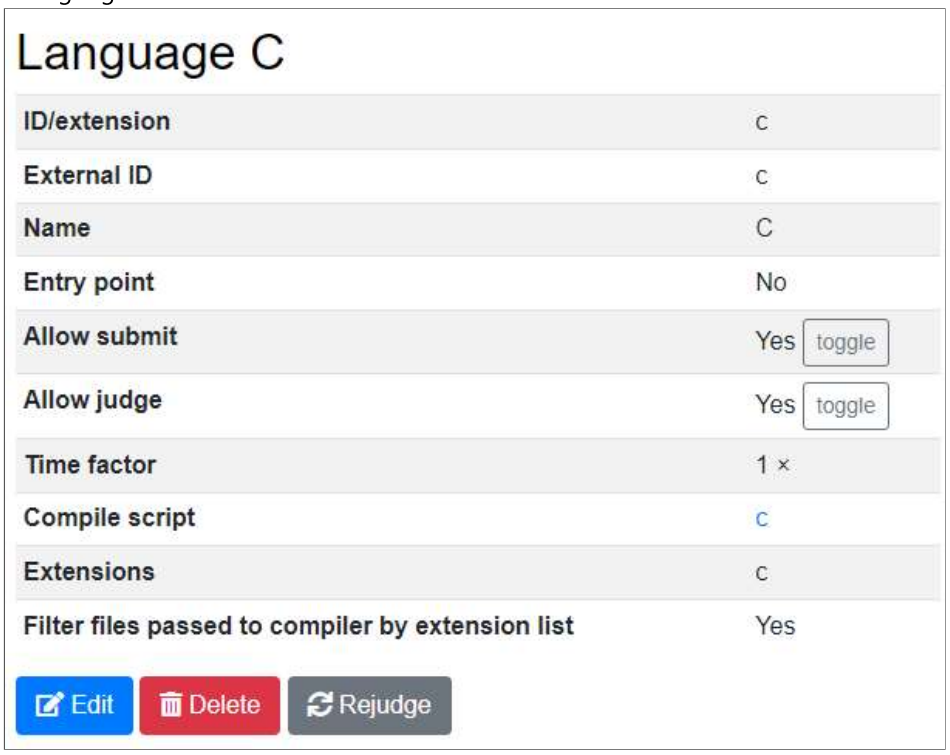
Problem	Short name	Points	Allow submit	Allow judge	Colour	Lazy eval
p3 - Boolean swi	boolfind	1	Yes	Yes	limegreen	Def
p2 - Float specia	fltcmp	1	Yes	Yes	yellow	Def
p1 - Hello World	hello	1	Yes	Yes	magenta	Def

- Short name: 會顯示在Navigation Bar右上角獎杯的名字
- Activate time: 簡單說是裁判(Jury)可以開始交程式的時間，可以用建立比賽的當下(或之前)
- Start time: 比賽開始的時間，學生(選手)在這個時間後才能開始繳交
- Scoreboard freeze time/Scoreboard unfreeze time: 在這段時間公開計分板會凍結，學生將看不到其他人的答題情況，**建議留空**
- End time: 比賽結束時間。超過這個時間再繳交會顯示 TOO LATE (但後台還是看得到結果)
- Deactivate time: 超過此時間便不可繳交程式，**可以留空**
- Process Balloons: 可以針對各題設定不同顏色，在選手答對題目後給選手對應氣球的功能(Queue在Jury Interface的Balloon Status)。教學使用可以選No
- Contest visible on public scoreboard: 是否不用登入就看得到解題狀況(題目也看得到，故考試用競賽建議關閉)
- Contest open to all teams: 如題。若選No可以指定隊伍或隊伍分類。應用範例：將學生的Category分為上午場/下午場。
- Enabled: 選No的話只會在Contests列表出現
- 題目區
  - Problem: 選擇已上傳的題目
  - Short Name: 顯示在計分板上的名稱
  - Points: 可以針對各題配分，計分板上會顯示總和

RANK	TEAM	SCORE	BOOLFIND [10 POINTS]	FLTCMP [20 POINTS]	HELLO [30 POINTS]
1	Example teamname Utrecht University	0 0			

  - Allow submit: 是否允許上傳。選擇No會暫時從計分板上消失
  - Allow judge: 是否評判。選擇No仍可繳交，但後台會暫停評判該題
  - Colour: 氣球顏色。
  - Lazy eval: 是否開啟Lazy evaluation。會在Configuration介紹。

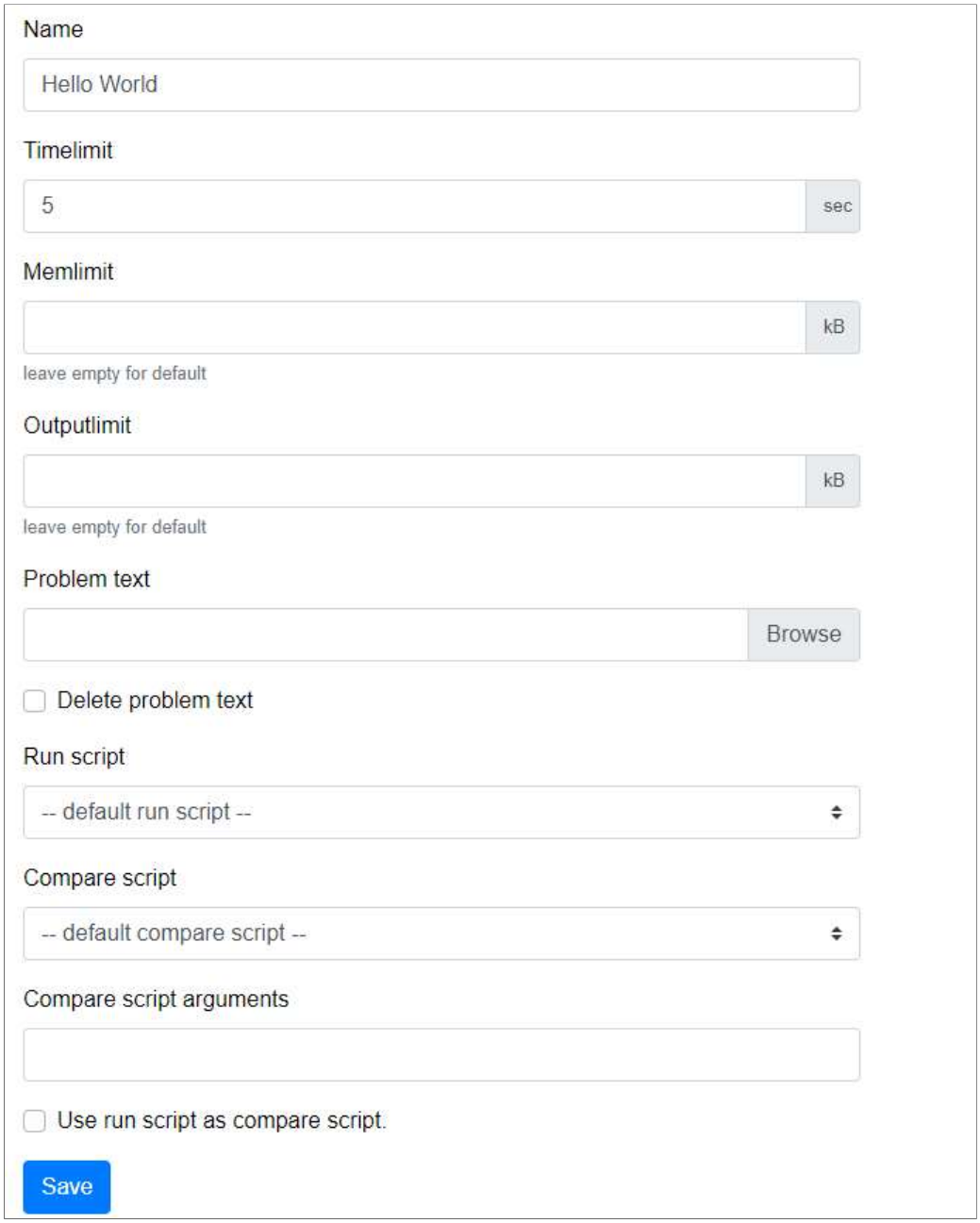
- #Languages



  - 按進各別語言後一般只會用到按toggle切換Allow submit跟Allow judge
  - Allow submit: 是否允許繳交。選No則在繳交時不會出現在語言列表中
  - Allow judge: 是否評判。選擇No仍可繳交，但後台會暫停評判以此語言繳交的submission。

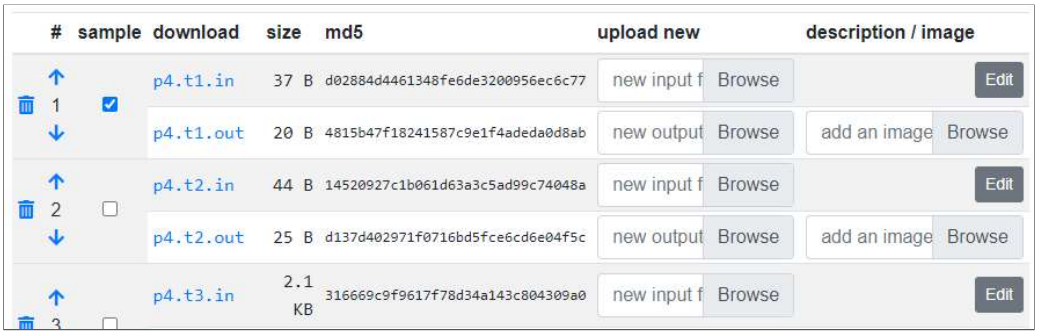
- #Problems

- 表單



  - Name: Problem名稱。所有Problem名稱皆不可重複。
  - Timelimit: 時間限制。指程式執行的時間限制(不包含編譯)。
  - Memlimit: 程式使用記憶體限制。可以留空使用預設值。
  - Outputlimit: 程式輸出大小限制。可以留空使用預設值。
  - Problem text: 題本PDF
  - Compare script arguments: 使用default compare script時有下列參數可使用，以空白隔開輸入即可:
    - `case_sensitive`：評判比對時區分大小寫(預設比對答案時不分大小寫)
    - `space_change_sensitive`：評判比對時比對所有空白字元(預設忽略space、\t、\n、\v、\f、\r)
    - `float_absolute_tolerance f`：判定浮點數時允許絕對誤差值 $abs(f_1 - f_2) < f$，推薦使用 `float_tolerance`
    - `float_relative_tolerance f`：判定浮點數時允許的相對誤差值 $abs(\frac{f_1 - f_2}{f2}) < f$，推薦使用 `float_tolerance`
    - `float_tolerance f`：判定浮點數時允許的誤差值 **(設定浮點數誤差推薦使用此參數，並將f設定為 `1E-6`，誤差 `1E-7` 可以將Compare script改成「default compare script for floats with prec 1E-7」)**

- testcase(從details/edit連結)



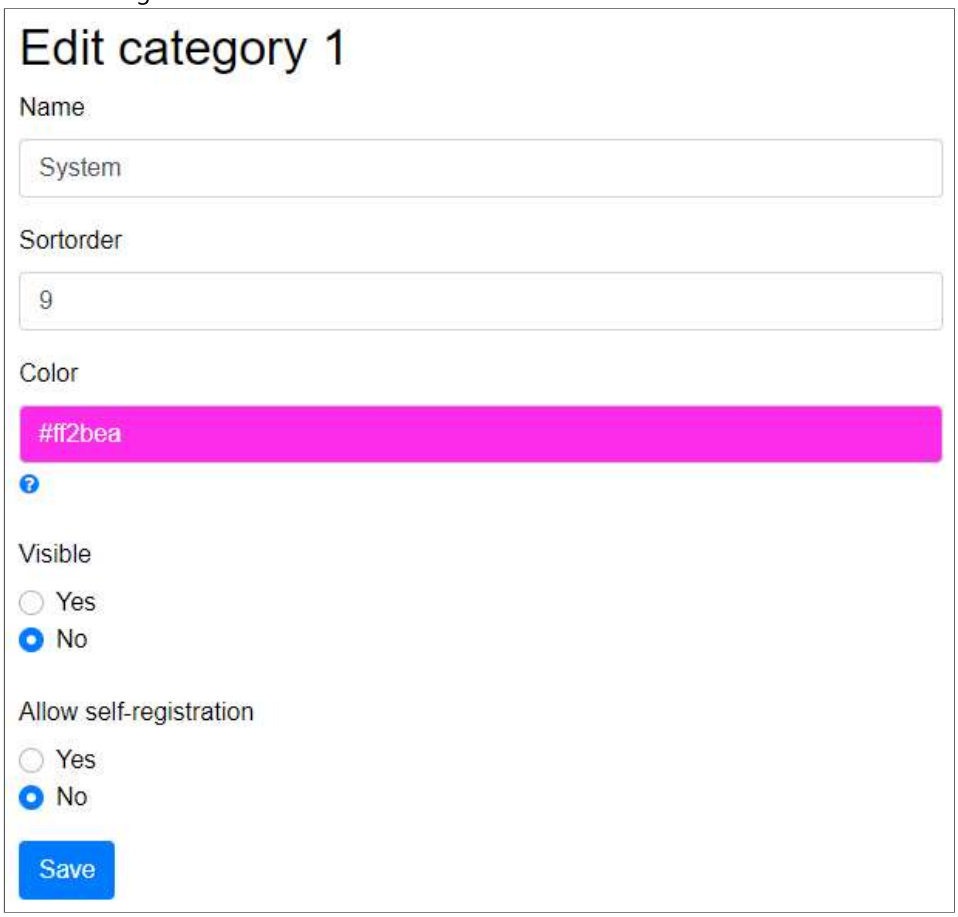
  - sample: 打勾的話會做為範例測資。隊伍在Navigation Bar的Problemset可以下載
  - upload new: 可以直接上傳替換版本(.in是輸入，.out是答案)
- Problem archives的部分會在後面出題建議的章節介紹

- #Teams

  - Team name: 必填，隊伍名稱
  - Display name: 在計分板上的顯示名稱，留空則會以Team name為準
  - Category: 隊伍分類，裁判隊伍可以選 `Observers` ，學生可以選 `Participants` ，詳細後文會介紹
  - Add new team時會多一個 Add user for this team的勾選框。若勾選則可以同時建立一個在此隊伍的使用者。Username推薦與Team name相同。

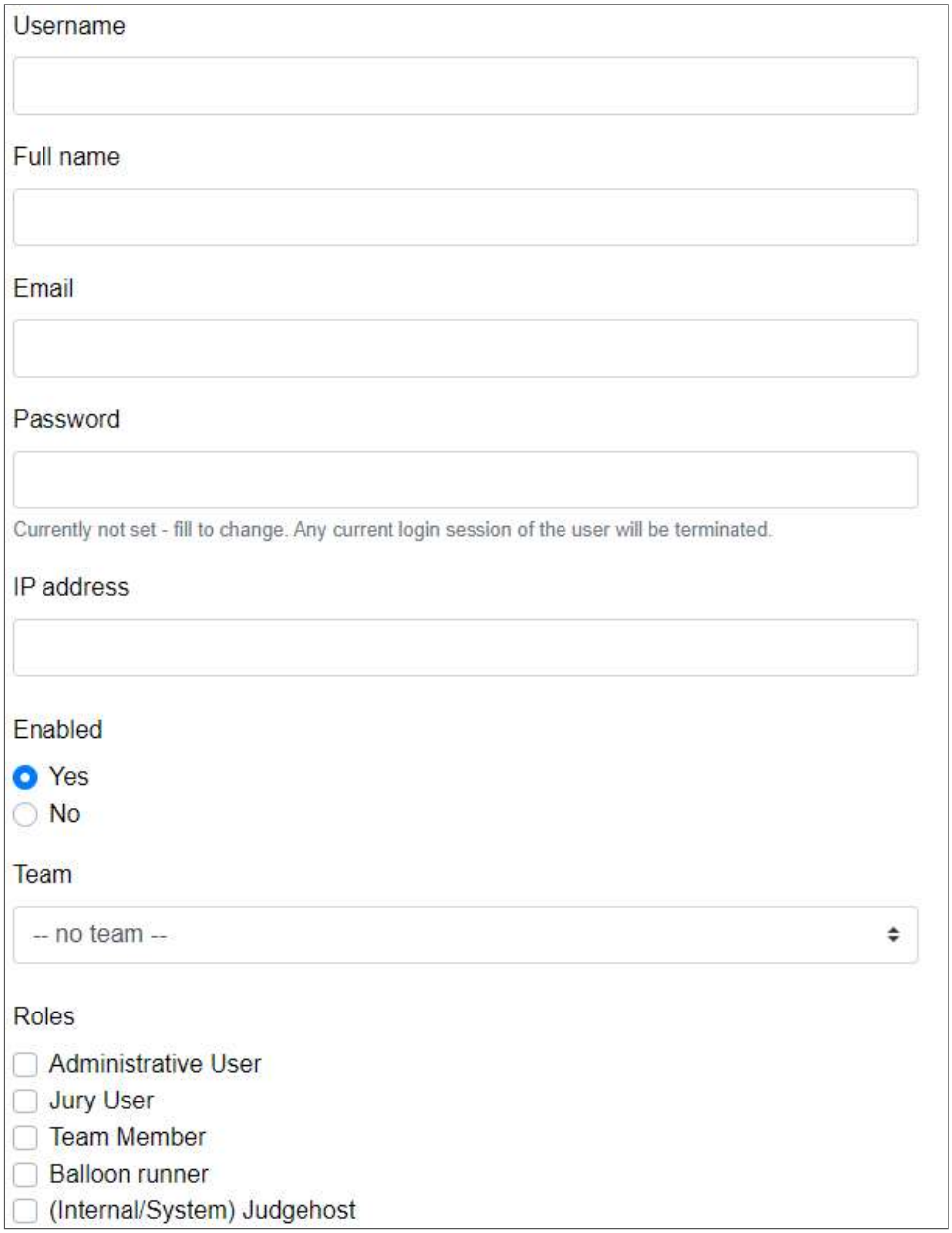
  - ***使用者須有隊伍才能繳交程式**

- #Team Categories



  - Name: 隊伍分類名稱
  - Sortorder: 數字不一樣的會被排在計分板的不同區塊，反之一樣就會混在一起排。可以做簡單分類，如正課生/旁聽生
  - Color: 在計分板上的背景色
  - Visible: 是否在公開計分板上可視
  - Allow self-registeration: 是否可以自行註冊。若有隊伍分類有打開，在登入頁面會出現前往自行註冊表單的連結

- #Users



  - Username: 使用者名稱。登入時使用
  - Full name: 計分板顯示名稱
  - Team: 使用者隊伍 **(使用者須有隊伍才能繳交程式)**
  - Password: 密碼 **(使用者必須有密碼才能登入)**
  - Roles: 使用者角色。裁判使用者建議勾選Administrative User、Jury User、Team Member，學生建議只勾選Team Member。

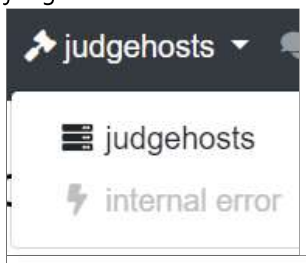
- #Configurations常用選項

  - compile_penalty: 是否因為編譯錯誤罰時
  - penalty_time: 答題正確後每次錯誤的罰時數
  - memory_limit/output_limit: 學生程式記憶體使用量/輸出限制預設值
  - lazy_eval_results: 是否開啟Lazy evaluation。若開啟，則評判時不一定會判完所有testcase，會在結果確定時便停止評判。
  - time_format: 時間顯示格式，主要會用在繳交時間上。可以參考php的strftime()函式。

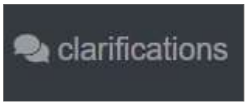
- #Navigation Bar

  - DOMjudge會根據身分不同會顯示不同的按鈕組合，本段以User role為「Administrative User、Jury User、Team Member」為例。

- judgehosts

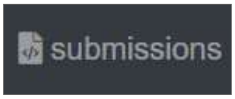


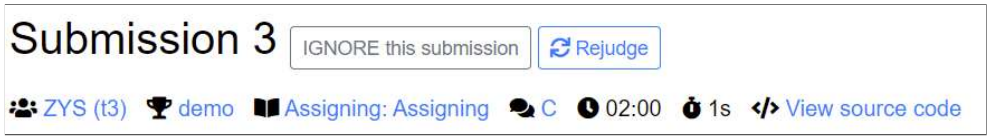
  - judgehost就是在docker上的image。如果status不是綠勾的話可能需要重新up或是用docker exec(參閱第一節) 尋找問題
  - internal error如果亮了 就是在評判過程在judgehost中有內部錯誤，需要用docker exec進入出錯的judgehost除錯
- Clarifications



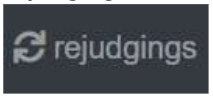
  - 學生可以透過繳交畫面的 

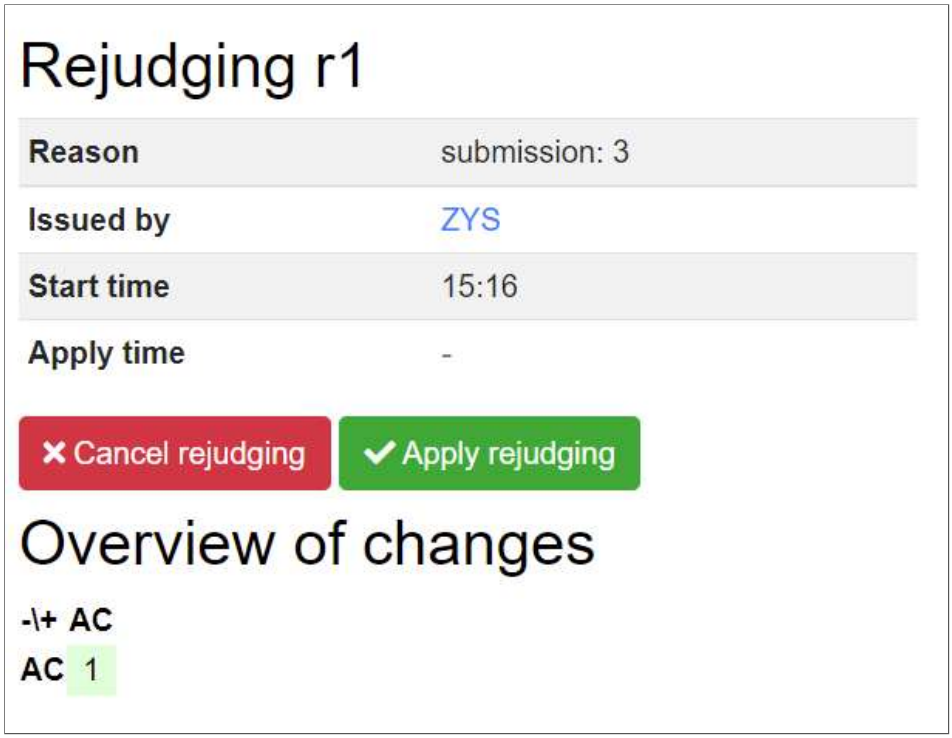
按鈕針對題目問問題，此頁面匯整所有的Clarifications並且可以各別回覆，也可以發公告。
每個Contest的Clarification都不同，需要用右上角的獎盃圖示切換
- Submissions



  - 可以看到當個Contest的所有繳交紀錄，按下各個submission可以看到詳細資料
  - 

    - IGNORE this submssion: 忽略此submission。若比賽中發現其中一個submission卡住，可能是internal error，可以先ignore造成問題的submission
    - Rejudge: 重新評測。一般會用在測資有新增或換過後， **會更新原本的結果**
    - View source code: 該submission的Code內容。可能會因為編碼問題有亂碼，所以**儘量建議學生不要在Code裡面寫中文**
      裡面的Edit按鈕可以編輯學生的程式碼重新繳交，幫學生除錯時可以使用。繳交身分是裁判，**不會影響到原本的結果**
- Rejudgings



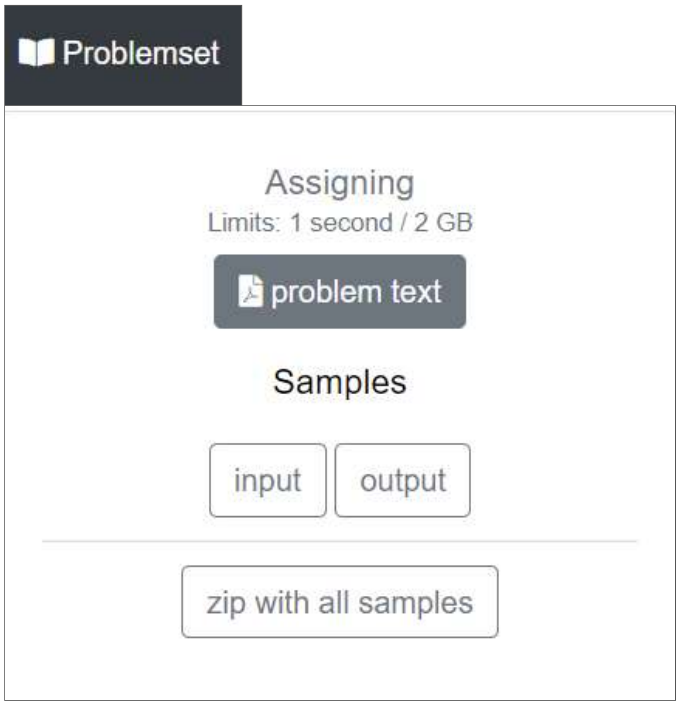


整理Rejudge的頁面，會顯示結果改變的概覽。在Apply前不會有變化。
預設已經CORRECT的submission不會再改變。

- team/Jury:

切換身分的按鈕。在team身分才能繳交程式，裁判看到的介面跟一般Team member基本相同，惟可以繳交程式的時間範圍較大。以下介紹的是team身分下的介面。

- Problemset



題目概覽。可以下載sample input/output檔案以及題目檔

- Scoreboard: 在team頁面可以點擊題目標題打開題本。在Jury頁面點擊題目標題可以連結到該Problem頁面，欄位可以連結到該學生的Submission統整。

- #其他
  - Submit時若先上傳檔案，judge會根據副檔名自動選擇語言。若檔名與ShortName(計分板上題目名)相同，也會自動選擇題目。
  - 在Contest中儲存Problems前，要注意Short Name欄位不要跟修改前內容衝突

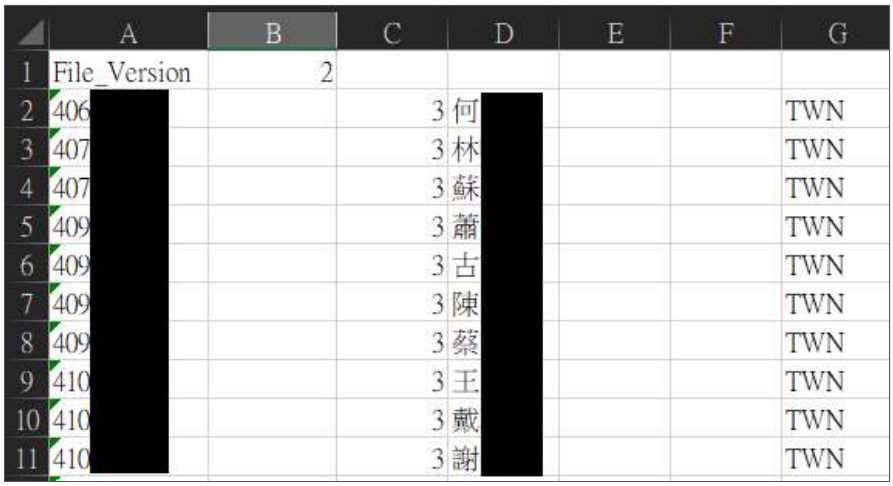
三、批次新增Teams

> 本章範例(Github)：importTeams/accounts.tsv, teams.tsv

本文使用Jury Interface的Import功能進行，另有JSON作法請參閱Manual (https://www.domjudge.org/docs/manual/8.0/import.html#id1)。

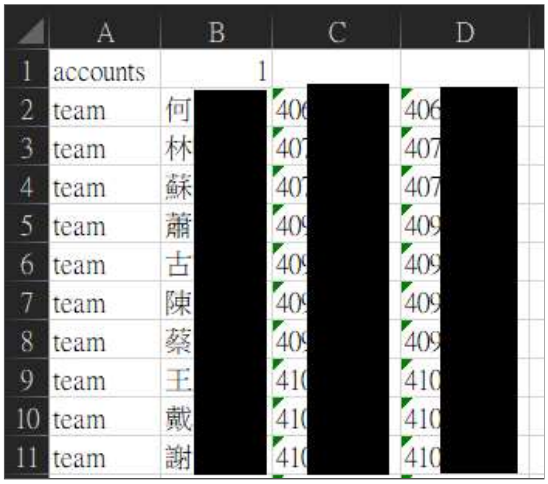
首先需要準備兩個檔案，皆**以tab分隔，不能有空行(注意最末行不能空)，注意編碼**(建議使用UTF-8)。

1. teams.tsv



	A	B	C	D	E	F	G
1	File_Version	2					
2	406[redacted]		3	何[redacted]			TWN
3	407[redacted]		3	林[redacted]			TWN
4	407[redacted]		3	蘇[redacted]			TWN
5	409[redacted]		3	蕭[redacted]			TWN
6	409[redacted]		3	古[redacted]			TWN
7	409[redacted]		3	陳[redacted]			TWN
8	409[redacted]		3	蔡[redacted]			TWN
9	410[redacted]		3	王[redacted]			TWN
10	410[redacted]		3	戴[redacted]			TWN
11	410[redacted]		3	謝[redacted]			TWN

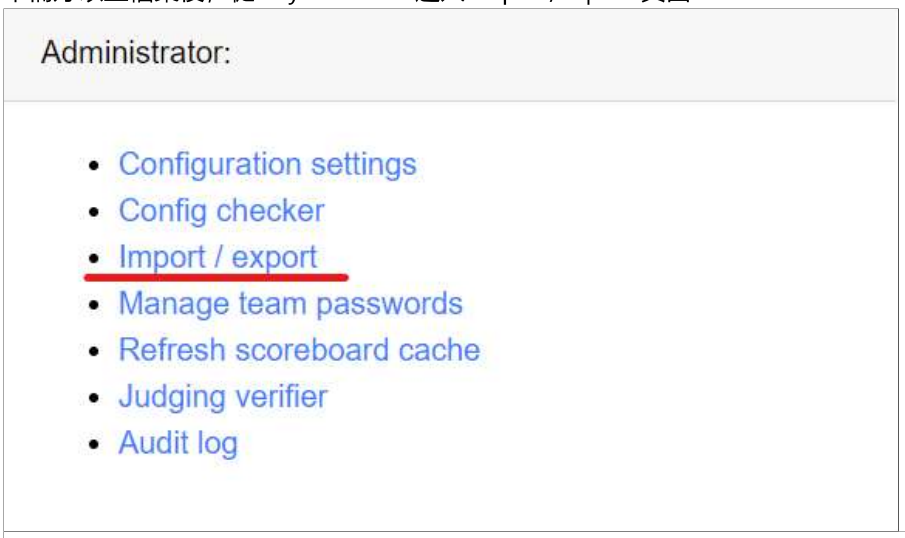
   - 第一列(row)： `File_Version 2`
   - 之後的列：
     - 第一欄(column)：Team ID，需使用純數字。**遇到已存在的會被覆蓋，請注意不要覆蓋到現有Teams**
     - 第三欄：Team category ID (預設3: Participants)，其他可以到Jury Interface/Team Categories看
     - 第四欄：Team Name
     - 第七欄：ISO 3166-1 alpha-3國家碼，可省略
     - 其他欄位可以參考Manual
2. accounts.tsv



	A	B	C	D
1	accounts	1		
2	team	何[redacted]	406[redacted]	406[redacted]
3	team	林	407	407
4	team	蘇	407	407
5	team	蕭	409	409
6	team	古	409	409
7	team	陳	409	409
8	team	蔡	409	409
9	team	王	410	410
10	team	戴	410	410
11	team	謝	410	410

   - 第一列： `accounts 1`
   - 之後的列：
     - 第一欄：User type， `team` 或 `judge` 。
     - 第二欄：User name。
     - 第三欄：Username，登入用的帳號，會對照teams.tsv的帳號Team ID，建議用「字母+teamID」的格式，如 `team100` 。
     - 第四欄：登入用的密碼。

準備好以上檔案後，從Jury Interface 進入 Import/export 頁面



首先選擇teams，匯入teams.tsv。然後再選擇accounts，匯入accounts.tsv。

四、題目結構與如何出題

> 本章範例(Github)：problems/*
> Collaboration.zip: 題目包裹範例
> problem.pdf: 題本範例
> ac.c, rand.cpp, gen_test.cpp, gen_ans.cpp: 出題用程式碼範例

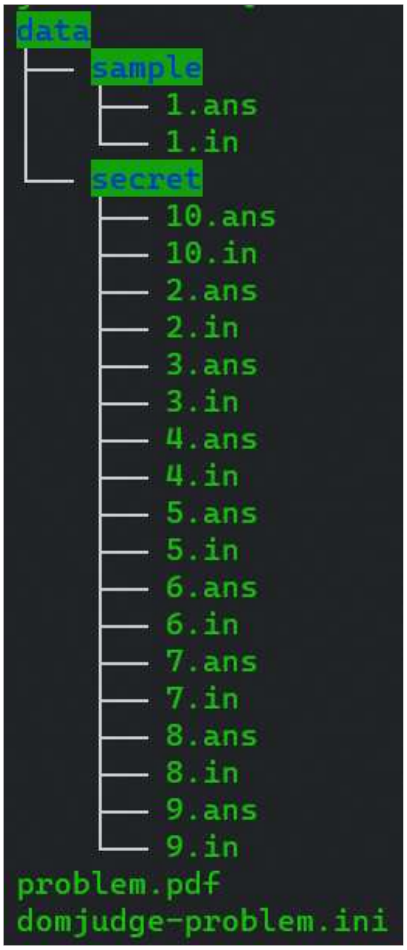
本節會從題目包裹(Problem Archive)介紹起，在熟悉其架構之後，將介紹一套程式化產生題目的流程以供參考。章節中的範例可以在Github上面找到。

提醒學生在正式繳交評測前，先**在本機確定可以編譯成功**，並**確認輸出與範例輸出相同**

1. 題目包裹架構與題目評判機制

題目包裹是可以在Problems表單中上傳的壓縮檔。比起新增Problem後再將內容一個個上傳，利用壓縮檔較為快速且方便管理。

其檔案樹如下圖：



- /problem.pdf: 題目敍述PDF檔
- /domjudge-problem.ini: 題目設定檔。常用參數為 `timelimit=1` (時間限制1秒)，其他參數可以參考Manual (https://www.domjudge.org/docs/manual/8.0/problem-format.html#)
- /data:
  - 資料夾中為放置測試資料的地方，分為範例測資與隱藏測資。範例測資目的是讓學生確認輸出格式正確，會出現在學生的Problemset區以供下載。隱藏測資是正式評判用的測資，學生**不會**知道檔案內容及各別的詳細判斷結果。
    檔案由{.in, .ans} 為一組，分別代表輸入及輸出。
  - /data/sample: 範例測資目錄
  - /data/secret: 隱藏測資目錄

將上述資料壓縮成zip後，就可以到Jury Interface/Problems的此表單進行上傳(可以一次上傳多個zip)

題目評判機制：

1. 將(範例/隱藏)輸入檔輸入以學生繳交的程式碼編譯成的程式
2. 以不同的方式將學生程式的輸出與答案檔案比對。預設是**不分大小寫、忽略空格**，可以透過compare script argument等等方式調整。
3. 可能結果
   - **CORRECT** : 正確

- **WRONG ANSWER**：比對後解題程式輸出錯誤
- **TIMELIMIT**： 解題程式執行時間過久
- **RUN-ERROR**： 解題程式遇到執行時期錯誤
- **COMPILER-ERROR**： 解題程式編譯錯誤
- **JUDGING**： 評判進行中

4. 罰時規則：第一次CORRECT的時間 + 罰時(預設為20分鐘) × (第一次CORRECT前的錯誤次數 − COMPILER-ERROR次數)。
   *若Configuration的compile_penalty有打開則不會扣除COMPILER-ERROR次數。

2. 出題指南

本節提供一個作者習慣的出題流程以供參考，請務必依自己的習慣調整。

1. **確定出題方向、主題、預設解法**：先想好題目的主軸，之後的步驟皆圍繞此主軸來撰寫題目，如範例中的主軸為「遞迴」。

2. **撰寫題本**：撰寫題本時，除了要敘述要清楚外，也可以依照需要包裝內文。一般來說題本分成以下部分：

   - 說明(Description)：
     題幹部分。應將期望解題程式達成的目標說明清楚，並考慮學生程度決定要放入多少引導。
   - 技術規格(Technical Specifications)：
     此部分可以寫在說明中的變數後，或是另外獨立一個區塊出來整理。此部分應參考預設解法的時間複雜度(建議以約 10^9次運算/秒估計)，訂定主要變數的大小以及題目時間限制。
   - 輸入/輸出格式(Input/Output Format)：
     解題程式應輸入/輸出的格式。所有測資皆應與此標準相符合，應儘量說明詳細(如空白、換行時機、小數點位數等等)以避免爭議。
   - 範例輸入/輸出(Sample Input/Output)：
     讓學生確認自己對題義理解的部分。一般情況下Problem archive的Sample input/output會與此處所寫的相同。若有特殊測資在說明中未能說明清楚或需避免，應在說明中補上或在此處增加該特殊情況的測資。
   - 備註(可選)：
     此處可以給學生提示、補充說明等等

3. **產生測資**： 先寫出AC Code並編譯，然後依據題目複雜程度，利用手動或者程式等不同方法產生測資，最後將產生的測資輸入AC Code編譯成的程式中取得解答。
   隨機產生測資時，建議以「**範例測資** → **小測資** → **大測資** → **極端測資**」的順序產生(依題目規模增減)。

   - 範例測資：如上所述，是讓學生確認題義理解的測資。以數量不大，可以手動驗算確認正確為準。
   - 小測資：以隨機方式產生的較小的測資。用以確認亂數函式的正確性，以可儘量手動驗算為準。
   - 大測資：以隨機方式產生的較大測資。建議除了主要變數的大小外，產生過程與小測資相同。
   - 極端測資：含有特殊情況或是數字儘量接近最大限制的測資。

   而產生測資的方式則可以選擇手動或者利用程式。一般來說範例測資，或是規模較小、較簡單的題目會選擇手動方式。

   - 手動方式：
     以手動方式產生測資需注意平台不同的問題。在Windows系統中，預設換行符號為「\r\n」，而DOMjudge所在的Linux系統預設換行符號則是「\n」。在學生輸入方式不一或開啓「space_change_sensitive」參數的情況下，可能造成誤判。

- 程式方式：
  範例程式包含 ac.cpp(AC Code), rand.cpp(產生單一隨機測資), gen_test.cpp(連續產生隨機測資), gen_ans.cpp(連續產生答案)，除AC Code外皆須在Linux環境下執行。
  rand.cpp中的亂數是利用C++ library中的random標頭檔，詳細可參考Reference 網站 (https://cplusplus.com/reference/random/)。並且為了可以在短時間內連續產生，random seed設為tv_usec。將產生亂數的部分撰寫完畢後，直接使用printf輸出。
  將AC Code及rand.cpp都準備好後，先編譯所有檔案。

  ```
  $g++ -o rand rand.cpp
  $g++ -o ac ac.cpp
  $g++ -o gen_ans gen_ans.cpp
  $g++ -o gen_test gen_test.cpp
  ```

  可以使用 `$./gen_ans` 或 `$./gen_test` 指令看一下使用方式。兩個程式皆會自動在指定的目錄下建立「data/secret/」目錄並將產生的檔案輸出至該處。
  指定完程式所在目錄及程式名稱後，若只有一個數字，代表產生編號 1 至指定編號的測資；兩個數字則是指定編號範圍。
  接著依序產生測資及解答。

  ```
  #Usage
  $./gen_test
  $./gen_ans
  #Example for generating 10 testcases
  $./gen_test . rand 1 10
  $./gen_ans . ac 10
  ```

4. **(Optional)Validators**: 若要保證題目嚴謹，可以撰寫Validators程式檢查測輸入及解答是否符合題本說明。

5. 最後再將包裹需要的檔案壓縮成zip檔後上傳即可。

五、輸出成績單及Submission Downloading

由於DOMjudge是提供競賽使用，故沒有實作輸出成績文字檔及批次下載Submission的功能。本章節搭配SQL及API實作此兩項功能。為了使用方便，滿多地方是使用複製貼上的方式，期待有志之士的自動化，欸嘿☆。

以下範例中Team Name為學生姓名，Team ID為學號，並且題目各自有設定Points。

1. 輸出成績單

本成績單列出各學生的學號、姓名、分數並輸出為.csv檔。
首先要連接資料庫：

```
$docker exec -it MARIADB_CONTAINER_NAME mysql -p
```

`MARIADB_CONTAINER_NAME` 須用 `$docker ps` 查詢並換成domserver或是mariadb的container。
輸入後會要求輸入密碼，預設密碼是 `rootpw` ，或是利用docker-compose.yml設定。

接著將下面的SQL根據需要調整WHERE條件：

- cid: Contest ID，可以到Jury Interface/Contests查看
- categoryid: Team Category ID，可以到Jury Interface/Team Categories查看

再貼到Command Line中執行。

```
SELECT rankcache.teamid, team.name, rankcache.points_public
FROM rankcache
LEFT JOIN team ON rankcache.teamid=team.teamid
WHERE rankcache.cid = 4 AND team.categoryid = 6
ORDER BY rankcache.teamid
INTO OUTFILE './Score.csv' CHARACTER SET 'utf8mb3' FIELDS TERMINATED BY ',';
```

注意若已經存在Score.csv檔案，需要改變輸出名稱或是先將其移除。

輸出後會出現在docker-compose.yml中設定備份用的volume中(見第一章)。

2. 下載Submissions

為了留存等目的(一開始是為了抓下來跑防抄襲程式)，有時候會需要下載一部分或所有的Submission Codes。
DOMjudge會將程式碼檔案以Base64編碼後存於資料庫中，本文先用SQL篩選出submission ID list後，使用Python呼叫API依序取得對應各個ID的程式碼檔案。範例以特定Contest Problem中所有結果判定為Correct的Submission為例。

首先進到DB的container，然後cd到資料庫的位置

```
$docker exec -it MARIADB_CONTAINER_NAME bash
$cd /var/lib/mysql
```

此處因為懶得裝vim，所以先上傳一個.sh檔「getlist.sh (http://getlist.sh)」如下(記得chmod成可執行檔)：

```
#getlist.sh
mysql --user=root --password=rootpw domjudge < script.sql > file_list
```

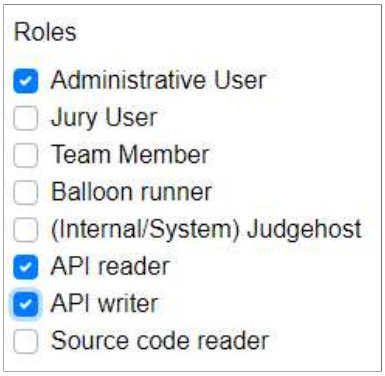
接著先將下面命令中SQL部分的WHERE條件依需要修改。

- cid: Contest ID，可以到Jury Interface/Contests查看
- probid: Problem ID，可以到Jury Interface/Problems查看
- categoryid: Team Category ID，可以到Jury Interface/Team Categories查看

```
$echo "SELECT submission.submitid, submission.cid, submission.teamid \
FROM submission \
LEFT JOIN (judging, team) \
ON (submission.cid=judging.cid AND submission.submitid=judging.submitid AND submission.team
WHERE submission.cid=3 AND submission.probid=18 AND judging.result='correct' AND team.categ
ORDER BY teamid;" > script.sql && ./getlist.sh
```

然後在command line貼上以上程式碼執行，產生file_list檔案。其中每一筆資料有三欄，分別為Submission ID, Contest ID及Team ID。

接下來要使用API下載程式碼檔案內容並且解碼。
首先需要為API存取建立一個帳號。前往Jury Interface/Users/Add new user，勾選「Administrative User、API reader、API writer」並設定密碼。



將前面步驟中得到的file_list放到要下載到的目錄中，並把下面的python檔案「get.py (http://get.py)」也放在同一個目錄中。需要更改的部分是Server的網址，以及API User的登入資訊。

```
import requests as req
import base64
import sys

inp = open("file_list", "r")

lines = inp.readlines()

inp.close()

# datas: subid, cid, teamid

for line in lines:

    datas = line.split()

    #Skip title
    if datas[0] == 'submitid' :
        continue

    #File name = teamid_subid.c
    fname= datas[2] + "_" + datas[0] + ".c"
    outf = open(fname, "w")

    #Get encoded file from API
    ###Replace the url to your domserver base url###
    tgt = "http://server.url:9487/api/v4/contests/" + datas[1] + "/submissions/" + dat
    ###Change auth to the right username and password###
    r = req.get(tgt, auth=('api', 'password'))

    #If error
    if r.status_code != 200:
        sys.stderr.write("Error on submission " + datas[0] + ", status code =" + str(r
        outf.close()
        break

    #Decode data
    data = r.json()
    enc = data[0]['source']
    dec = base64.b64decode(enc).decode("UTF-8", 'ignore')

    #Write file
    outf.write(dec)
    outf.close()

    sys.stderr.write("submission " + datas[0] + " done\n")
```

最後再執行Python Code即可

```
$python get.py
```

六、其他

1. 安全性建議

1. 初次部署完DOMjudge後，若要創建管理用/裁判帳號，建議在登入原始admin帳號後，分別為各裁判創建(擁有Administrator角色的)專屬帳號，以防止密碼遺失等問題發生。
2. 建議以防火牆(如ufw)對外關閉domjudge資料庫的port(範例yml檔設為13306)，以防止非經允許的直接存取

2. 評判編譯調整須知

調整評判時的編譯相關內容(如安裝 Pyhton package)時，需要在**每個**judgehost都做調整。

進入judgehost的container後執行chroot:

```
$docker exec -it JUDGEHOST_CONTAINER bash
$/opt/domjudge/judgehost/bin/dj_run_chroot
```

在這個root下做的改變才會反映在評判時的程式執行上。

```
#範例 - 安裝numpy:
$apt update
$apt install python3-numpy
```

3. 互動題

互動題是一種較特殊的題目類型，裁判程式與答題程式以標準輸入/輸出互動，通常由答題程式根據互動提供的線索來推測出正確輸出。

關於互動題的介紹及出題細節，可參考：Domjudge: 如何出互動題 (/W0JtwP5vQ2Gfk2HvFQkGYg)

後記

本文比起教學，比較偏向作者自己在這幾年利用DOMjudge出題的心得整理，因此內容偏向針對特定情況的一種解決方案(尤其是後半部分)。除了有許多踩過的坑沒有提及外，也有不少需要手動調整的地方。若有機會，也希望可以~~有人幫我~~自動化，讓其能夠更容易使用

作者: 張源宰
Email: johnson10024@csie.io (mailto:johnson10024@csie.io)
2022.07.29