**크래시웹(CrashWeb) 사용자 매뉴얼**

**목차:**

1. [1. 개요](#1_4444007745478302_68590635177_951373991323635)
   1. [1.1. 왜 필요한가?](#1_1_7079613858368248_120110687_2620270666666329)
   2. [1.2. 크래시웹이란?](#1_2_07361981412395835_22410415)
   3. [1.3. 구성 요소](#1_3_7710558692924678_272607447)
   4. [1.4. 시스템 구성도](#1_4_18258526106365025_71520739_9532319803256541)
2. [2. 설치](#2_27147433860227466_6055222167_18418093584477901)
   1. [2.1. 준비](#2_1_39966479898430407_15453212_4169343563262373)
      1. [2.1.1. 공개 서버](#2_1_1_3621180127374828_6978341_25963648594915867)
      2. [2.1.2. 개발 서버](#2_1_2_27263043471612036_582278_5609405583236367)
      3. [2.1.3. 빌드 서버 (for 심볼 서버)](#2_1_3_for_9724984613712877_990_5312192142009735)
   2. [2.2. 각 모듈 설치](#2_2_38727390859276056_09305069_7542330343276262)
      1. [2.2.1. Reception](#2_2_1_Reception_72491994500160)
      2. [2.2.2. DB](#2_2_2_DB_8188460222445428_3045_8367007416673005)
      3. [2.2.3. Browser](#2_2_3_Browser_5495834301691502_9797248486429453)
      4. [2.2.4. 심볼 서버](#2_2_4_9765760840382427_9401912)
      5. [2.2.5. CrashGatherer](#2_2_5_CrashGatherer_0186658585)
   3. [2.3. 클라이언트 모듈 설치](#2_4_8624673536978662_459980043_7279579383321106)
      1. [2.3.1. 필요한 오류 정보](#2_3_1_14112381218001246_416717)
      2. [2.3.2. 오류 정보 업로드](#2_3_2_35768580925650895_769253)
   4. [2.4. 최종 확인](#2_4_3793306879233569_446656550_23378722090274096)
3. [3. 추가 설치/설정 항목](#3_8910495825111866_74343909416_4221706048119813)
   1. [3.1. 공개 서버에서 개발 서버로 파일 긁어오는 방법](#3_1_9550949833355844_434503209)
   2. [3.2. Browser에서 보고된 zip 파일 원본 다운받기 기능을 설정하는 방법](#3_2_Browser_zip_75395650463178)
      1. [3.2.1. 프로젝트가 여러 개일 때, juction을 이용하는 방법](#3_2_1_juction_6254589275922626_9368979139253497)
   3. [3.3. 이슈 트래커(Redmine 또는 MANTIS BT)와 연동하는 방법](#3_3_MANTIS_1819571207743138_98)
   4. [3.4. 머큐리얼  등의 SCM 과 연동하는 방법](#3_4_SCM_7084588878788054_72638)
   5. [3.5. 새 프로젝트를 추가하는 방법](#3_5_8547644945792854_830283699)
   6. [3.6. 팀 사이트(게시판)과 연동하는 방법](#3_6_703849001089111_4163480727)
   7. [3.7. 이메일 (또는 야머)와 연동하는 방법](#3_7_9108924039173871_536254786)
      1. [3.7.1. 알람이 발송되는 조건을 수정하려면?](#3_7_1_6927773021161556_9683118)
      2. [3.7.2. 야머와 연동하는 방법](#3_7_2_555273212492466_47571083)
   8. [3.8. 서버 프로젝트에 적용하는 방법](#3_8_7545294566079974_635281632)
   9. [3.9. 윈도 심볼 등, 추가 심볼에 대하여 (중요)](#3_9_2011_3_4_35241820686496794)
   10. [3.10. 심볼이 원격 서버에 있을 때 발생하는, 심볼 로딩 속도 문제 해결하기](#3_10_3579597582574934_33400550_2101765633560717)
4. [4. FAQ](#4_FAQ_9308519365731627_1896619)
5. [5. 기타 정보](#5_712811884470284_743590855039_3002370735630393)
   1. [5.1. 소스 위치](#4_1_1073402059264481_307903906)
   2. [5.2. 적용한(혹은 예정인) 팀](#4_2_028726456686854362_1464341)
   3. [5.3. 연락처](#4_3_8279272457584739_325206136)
   4. [5.4. 문서 업데이트 기록](#4_4_4131406815722585_015997672_5148945772089064)

# 1. 개요

## 1.1. 왜 필요한가?

굉장히 많은 수의 DMP 파일이 쌓여 있을 때, 이것을 일일이 확인하기가 어렵기 때문입니다. 크래시웹 시스템은 보고된 DMP 파일을 데이터베이스화해서 보여주므로, 가장 많이 발생한 오류를 한번에 알아낼 수 있습니다.

## 1.2. 크래시웹이란?

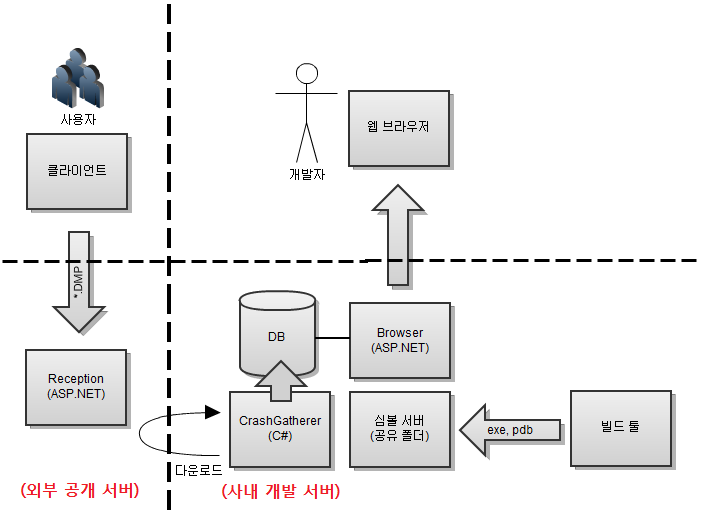
1. 클라이언트가 보낸 오류 보고를 수집하고
2. 분석해서 데이터베이스화한 결과를
3. 개발자가 웹 브라우저를 통해 조회할 수 있는 시스템

## 1.3. 구성 요소

크래시웹은 여러 프로그램으로 구성된 하나의 시스템입니다. 크래시웹을 구성하는 시스템은 다음과 같습니다:

|  |  |  |  |
| --- | --- | --- | --- |
| **프로젝트 명** | **항목** | **설명** | **비고** |
| - | 오류 정보 업로드 모듈 | 클라이언트가 크래시되면서 남긴 DMP 파일 등의 정보를 서버로 전송하는 모듈 | 위치:클라이언트 |
| Reception (ASP.NET) | 오류 수집 서버 | 클라이언트가 전송한 오류 정보 및 DMP 파일을 받는 서버 모듈 | 위치:공개 서버 |
| CrashGatherer (C#) | 오류 분석 도구 | DMP 파일을 분석해서 데이터베이스에 넣는 과정을 자동화한 툴 | 위치:개발 서버 |
| CDB.EXE | CDB 커맨드라인 툴 | DMP 파일에서 콜스택을 뽑을 때 사용 | Debugging Tools for Windows에 포함된 툴 |
| - | 오류 데이터베이스 | 분석이 끝난 오류 정보가 저장되는 곳 | SQL Server 2005 이상 권장 |
| SYMSTORE.EXE | 심볼 서버 | 심볼은 배포 시 사용하는 빌드 툴에서 항상 이곳에 동기화해야 합니다. | Debugging Tools for Windows에 포함된 툴 |
| Browser (ASP.NET) | 오류 조회 사이트 | 개발자가 이 사이트를 통해 DB의 내용을 조회합니다. |  |

## 1.4. 시스템 구성도



1. 클라이언트에서 오류가 발생하면 \*.DMP 파일을 포함한 오류 정보를 Reception 서버로 업로드합니다.
2. 주기적으로 Reception에 저장된 오류 정보를 개발 서버로 긁어 옵니다.
3. 주기적으로 CrashGatherer가 실행되어 오류 정보에서 콜스택 등을 뽑고 DB화합니다.
   1. CDB.EXE를 이용해서 콜스택을 뽑는 과정에서 심볼 파일(exe, pdb)이 필요합니다.
   2. 심볼 파일은 빌드 툴이 생성하는 것이고, 콜스택을 뽑기 전에 심볼 서버에 미리 동기화되어 있어야 합니다.
4. DB화된 오류 정보는 Browser를 통해서 개발자 PC에서 웹 브라우저로 조회할 수 있게 됩니다.

# 2. 설치

각 설치 과정에 누가 작업하는 것이 좋은지 노란색으로 표시해 두었습니다. 참고하세요!

## 2.1. 준비

각 모듈을 설치하기 전에, 기본적으로 필요한 프로그램(prerequisite) 설치에 관한 내용입니다.

(서버 프로그래머의 할 일)

### 2.1.1. 공개 서버

* Reception이 설치될 곳
* IIS 6.0 이상[1](#FOOTNOTE-1)
* .NET Framework 2.0 + 최신 서비스 팩 설치 필요

### 2.1.2. 개발 서버

* DB, Browser, CrashGatherer 등이 설치될 곳
  + 각 모듈을 별도의 각각 별도의 서버에 설치하는 것도 가능합니다.
  + 예를 들어, DB는 팀에 이미 존재하는 SQL Server 인스턴스를 활용한다거나...
* **DB**
  + SQL Server 2005 이상 + 가능한 최신 서비스 팩
* **Browser**
  + IIS 6.0 이상
  + .NET Framework 2.0 + 최신 서비스 팩 설치 필요
* **CrashGatherer**
  + Debugging Tools for Windows 최신 버전
    - cdb.exe
    - **(중요) 설치된 경로를 환경변수 %PATH%에 등록합니다.**
      * 제어판 -> 시스템 -> 고급 -> 환경변수 -> 시스템 변수 -> Path

### 2.1.3. 빌드 서버 (for 심볼 서버)

심볼 서버를 이용해야 하므로, 빌드 자동화 툴에서 심볼 서버에 심볼 파일(exe. pdb)을 등록하도록 설정해야 합니다.

* Debugging Tools for Windows 최신 버전
  + symstore.exe
  + **(중요) 설치된 경로를 환경변수 %PATH%에 등록합니다.**
    - 제어판 -> 시스템 -> 고급 -> 환경변수 -> 시스템 변수 -> Path

## 2.2. 각 모듈 설치

### 2.2.1. Reception

※ 기존에 클라이언트에서 DMP 파일을 서버로 업로드하는 일련의 과정이 구현되어 있으면, Reception 설치는 생략할 수도 있습니다.

※ 단, 서버에서 파일 명은 **[Optional Text]\_[IP ADDR]\_YYYYMMDD\_HHMMSS\_밀리초.zip** 형식에 맞추어 남겨야 합니다.

이 애플리케이션은 [**공개 서버**]에 설치합니다.

(서버 프로그래머의 할 일)

1. 설치할 폴더를 생성합니다. (예: d:\crashweb\reception)
2. 위에서 생성한 폴더에 crashweb/ASPX/Reception 프로젝트 파일들을 풀어 놓습니다.
3. IIS 설정을 통해 위에서 생성한 폴더에 접속할 수 있도록 설정합니다.
   1. ASP.NET이 Prohibit 상태가 아닌지 확인하시고
   2. Application Pool 설정해서 script가 실행될 수 있도록... 등의 통상적인 aspx 웹 애플리케이션 설치와 동일합니다.
4. Reception이 이벤트로 로그를 남길 수 있도록 ASPX/EnableEventLog.reg 파일을 실행해서 레지스트리 항목을 추가합니다.
5. 웹 애플리케이션이 설치된 폴더로 가서 Web.config 항목을 수정합니다.
   1. 수정할 곳은 클라이언트가 업로드한 파일을 저장할 곳을 설정하는 downloadPath 항목 하나입니다.
6. 브라우저 등을 통해 잘 동작하는지 테스트를 합니다.
   1. 파일 업로드가 되는지 확인하기는 어렵고, 접속만 확인하도록 합니다.
   2. 파일 업로드 동작 여부는 클라이언트에 적용한 후에 하면 됩니다.

### 2.2.2. DB

이 DB는 [**개발 서버**]에 생성합니다.

(DBA의 할 일)

1. 새 데이터베이스를 만듭니다.
   1. 데이터베이스 이름은 CRASHWEB\_DB로 합시다
2. crashweb/SQL/1.crashweb\_tbl.sql 실행해서 테이블을 생성합니다.
3. crashweb/SQL/2.crashweb\_sp.sql 실행해서 저장 프로시저를 생성합니다.
4. crashweb/SQL/3.crashweb\_init.sql 파일을 편집기로 열고
   1. 로그인 계정 정보에 대한 부분 확인 (아이디, 패스워드 등)
   2. 프로젝트 설정에 대한 부분을 확인하고 (프로젝트 명)
   3. 알맞게 수정한 후에 적용합니다.
   4. project\_uid를 기억해 둡니다(처음 설치라면 project\_uid는 1입니다).

### 2.2.3. Browser

이 애플리케이션은 [**개발 서버**]에 설치합니다.

(서버 프로그래머의 할 일)

1. 설치할 폴더를 생성합니다. (예: d:\crashweb\browser)
2. 위에서 생성한 폴더에 crashweb/ASPX/Browser 프로젝트 파일들을 풀어 놓습니다.
3. IIS 설정을 합니다(위의 Reception 설정과 동일. 단, 레지스트리 항목 추가는 필요 없음).
4. Web.config를 수정합니다.
   1. connectionString 항목을 이전 단계에서 생성한 DB에 맞도록 설정합니다.
   2. (해당 파일에 예제가 있으므로, 참고해서 수정하시면 됩니다)
5. 브라우저를 열어서 사이트가 잘 동작하는지 테스트합니다.
   1. 실제로 데이터는 없으므로, 접속 확인 등의 기본적인 테스트만 하도록 합니다.

### 2.2.4. 심볼 서버

이 공유 폴더는 [**개발 서버**]에 구성합니다.

(서버 프로그래머 또는 클라이언트 프로그래머의 할 일)

1. 기존에 심볼 서버를 사용하고 있다면, 이 부분은 건너뛰시면 됩니다.
2. 심볼 서버로 사용할 폴더를 하나 생성합니다.
   1. 오류를 분석할 서버가 심볼 서버와 다른 서버라면, 이 폴더를 읽기 전용으로 공유 설정을 하도록 합니다.
3. 자동화된 빌드 스크립트의 POST BUILD 이벤트에 symstore.exe를 이용해서 매번 심볼 서버에 심볼 파일을 등록 시키도록 설정합니다.[2](#FOOTNOTE-2)
   1. **자동화된 빌드 시스템을 사용하고 있지 않다면? 당장 사용하십시오.**
   2. [CruiseControl.NET](http://ccnet.thoughtworks.com/)을 추천합니다.

### 2.2.5. CrashGatherer

이 애플리케이션과 스크립트 세트는 [**개발 서버**]에 설치합니다.

(서버 프로그래머의 할 일)

1. 설치할 폴더를 생성합니다. (예: d:\crashweb\bin)
2. 위에서 생성한 폴더에 crashweb/scripts 폴더 내의 파일들을 풀어 놓습니다.
3. CWRUN.js 파일을 열고 local settings 섹션 내의 설정을 수정합니다.
   1. **reportPath**: 사용자가 리포트한 파일 원본이 있는 곳의 위치
   2. **symServer**: 심볼 서버의 위치
   3. **connString**: DB 연결 문자열
   4. **주의**:
      1. 예: **var reportPath = "\"D:\\crashweb\\reported files\" ";**
         * 경로에 띄어쓰기가 포함되어 있을 경우 따옴표(")를 넣도록 하고, 자바스크립트 문법에 맞추어 따옴표 앞에 슬래시(\)를 넣도록 합니다.
         * 그리고, 폴더를 구분하는 슬래시(\)를 입력할 경우에는 슬래시를 두 개(\\) 입력하도록 합니다.
         * 각 설정 항목은 구분되어 커맨드라인 인자로 전달되므로, 제일 뒤에 공백을 두도록 합니다.
4. DeleteOld.js 파일을 열고, 오래된 파일이 삭제되도록 대상 폴더를 설정합니다.
   1. main 함수에 대상 폴더의 목록이 있고, 파일의 제일 아래 쪽에 있습니다.
   2. 기본 리스트를 보시고 적당히 수정하시고
   3. 기간을 변경하고 싶으면 DeleteOld 함수 내의 값을 수정합니다(기본값은 60일 이전의 파일을 삭제).
5. 테스트가 가능한 환경이 되면, 테스트를 해 봅니다.
   1. 심볼 서버와 클라이언트 작업 등의 환경이 필요하므로
   2. 이 단계에서의 테스트는 어렵습니다.
6. 주기적으로 실행되도록 installtask.bat를 실행해서 스케줄러에 등록합니다.
   1. **테스트가 이상 없이 완료된 후에 해야 합니다!**

## 2.3. 클라이언트 모듈 설치

(클라이언트 프로그래머의 할 일)

### 2.3.1. 필요한 오류 정보

1. \*.dmp 파일
2. stack.log 파일에 아래 정보를 CSV 형태로 남김
   1. account uid
   2. login id
   3. user uid
   4. nickname
   5. client version
      1. string으로 정렬했을 때, 정렬이 잘 되는 형태여야 합니다.
      2. 예를 들어, 버전이 1, 2, 3, 4, 5, ..., 10, 11, 과 같이 증가한다면
         1. 문자열 정렬 결과는 1, 10, 11, ... 2, 20, 21, ... 처럼되면서 순서가 꼬입니다.
      3. 따라서, v00001, v00002, ... 등과 같이 문자열 정렬이 잘 되는 형식으로 남기도록 합니다.
      4. **기존에 사용하는 클라이언트 버전이 없다고요? 당장 추가하십시오.**
      5. **버전 번호가 빌드할 때마다 증가하지 않는다고요? 당장 빌드 스크립트를 수정해서 매번 다른 버전이 생성되도록 수정하십시오.**
   6. 사용자 코멘트
3. stack.log 파일의 예

|  |
| --- |
| 1: 12124012,LoginID,1997692,Nickname,v20768  2: 로그인하다가 죽었어요~  3:  4: 둘째 줄 이후는 유저 코멘트를 위한 공간입니다. |

### 2.3.2. 오류 정보 업로드

1. dmp 파일과 stack.log 파일을 묶어서 zip으로 압축합니다.
   1. zip 파일의 이름은 서버에서 정하므로, 어떤 형태라도 상관이 없습니다.
2. HTTPHelper 모듈 등을 이용해서 서버로 zip 파일을 전송합니다.
3. 테스트해서 서버 측에 파일이 잘 생성되는지 확인합니다.
   1. 서버 측에 남는 파일 명은 아래와 같은 형식이어야 합니다.
   2. [Optional Text]\_[IP ADDR]\_YYYYMMDD\_HHMMSS\_밀리초.zip
      1. 예: loginid\_192.168.11.14\_20110103\_192533\_333.zip

## 2.4. 최종 확인

* 클라이언트에서 고의로 crash를 발생[3](#FOOTNOTE-3)시키고, 오류 보고를 합니다.
* 공개 서버를 거쳐서 개발 서버로 보고된 오류가 잘 전달되는지 확인합니다.
* 웹 브라우저로 Browser 사이트에 접속해서 보고된 오류가 잘 보이는지 확인합니다.
  + 사이트의 각 기능들에 이상이 없는지 확인합니다.
* 잘 동작하지 않을 경우에는?
  + 시스템 구성도를 보고 순서대로 어디에 문제가 있는지 확인해 봅니다.
  + 모든 시스템은 로그를 남기고 있으니, 로그 파일을 열어서 오류 메시지를 확인하도록 합니다.
    - 단, Reception은 이벤트 로그를 확인해야 합니다.
  + 문제를 해결하고 다시 확인해서 잘되면 일단 축하!

# 3. 추가 설치/설정 항목

이 부분의 거의 (서버 프로그래머의 할 일)입니다. 한두 개의 예외는 따로 표시하고 이니다.

## 3.1. 공개 서버에서 개발 서버로 파일 긁어오는 방법

FTP를 이용해서 긁어오는 예제를 소개해 드립니다만, 다른 방법으로 하셔도 상관이 없습니다.

**공개 서버 측 설정**:

1. IIS 매니저에서, 보고된 파일이 위치한 디렉토리에 접근이 가능하도록 FTP 서버 설정을 합니다.
2. 긁어가는 쪽에서 삭제도 가능하도록 write 권한도 주도록 합니다.
   1. 서버 쪽에서 주기적으로 삭제하는 방법도 있습니다.
   2. 긁어가는 쪽에서의 삭제를 권장합니다.
3. 각종 보안 설정(포트, 방화벽, 아이디, 패스워드 등)을 확인하고, 사내 개발 서버에서 접속이 잘 되는지 확인합니다.

**개발 서버 측 설정**(NcFTP Client[4](#FOOTNOTE-4) 프로그램 이용):

1. NcFTP Client를 내려받아[5](#FOOTNOTE-5) 설치합니다.
   1. 설치 경로 예: C:\Program Files (x86)\NcFTP
2. 스케줄러에서 다음 명령을 주기적으로 수행하도록 합니다.
   1. "C:\Program Files (x86)\NcFTP\ncftpget.exe" -u username -p password -F -R -DD -P port ipaddr "d:/crashweb/CB/" "/crashweb/reported files"
      1. -u [username]
      2. -p [password]
      3. -F : Passive 모드 설정
      4. -R : 하위 폴더 포함(Recursive) 모드 설정
      5. -DD : 성공적으로 다운로드했으면 서버 파일은 삭제하기
      6. -P [port]
   2. 배치 파일로 만들어서 스케줄러에 등록하면 됩니다.
3. 또는, 위의 내용이 포함된 [ncftp.js 파일[6](#FOOTNOTE-6)](http://docs.google.com/a/ntreev.com/leaf?id=0B7TDRcaYJK2zNDMwMWY4YWEtMmRiMC00MDVmLThiMDYtZDRiM2MzODQ2NGY0&amp;hl=ko)을 다운로드해서 local settings의 항목을 자신의 환경에 맞게 설정한 후에 스케줄러에 등록합니다.
4. 스케줄러를 설정할 때는 CrashGatherer와 인터벌을 같거나 짧게 맞추어야 합니다.
   1. 또는 CrashGatherer를 실행하는 CWRUN.js 스크립트에 위의 내용을 포함하는 방법도 있습니다.
5. 파일 다운로드와 삭제가 정상적으로 이루어지는지 확인합니다.

## 3.2. Browser에서 보고된 zip 파일 원본 다운받기 기능을 설정하는 방법

공개 서버에서 긁어온 파일이 저장된 위치가, Browser 웹 애플리케이션이 설치된 경로와 맞지 않으면 원본 파일을 내려받지 못하게 됩니다. 보통은 다르게 설정이 되고, 여기서 추가 설정을 통해 다운로드 경로를 맞추어 주는 것이 정상적인 방법입니다.

1. 임의의 새 폴더를 하나 만듭니다.
   1. 예: d:\crashweb\reports
2. IIS 매니저를 열고 Browser 웹 사이트를 선택합니다.
3. 오른클릭 팝업 메뉴 -> New -> Virtual Directory...를 선택합니다.
   1. Alias 입력 창에 **reports**라고 입력합니다.
   2. Path는 위에서 생성한 임의의 새 폴더를 위치로 지정합니다.
4. 가상 디렉토리 reports 아래에 또 하나의 가상 디렉토리를 만듭니다.
   1. Alias에 project\_uid를 입력합니다.
      1. 프로젝트가 하나라면 project\_uid는 보통 1입니다.
   2. Path는 공개 서버에서 긁어온 파일이 저장된 곳으로 선택합니다.
   3. 권한은 읽기만으로도 충분
   4. 최종적으로 파일 명은 http://crashweb:8080/**reports/1**/filename.zip 이런 형태가 됩니다.
5. 다운로드가 정삭적으로 되는지 확인해 봅니다.

### 3.2.1. 프로젝트가 여러 개일 때, juction을 이용하는 방법

[juction](http://technet.microsoft.com/en-us/sysinternals/bb896768)을 이용해서 project\_uid별로 원본 폴더와 링크를 만들 수 있습니다.

* 예:
  + d:\crashweb\reports> junction 1 d:\crashweb\scheduled\GametreeQA\reports
    - d:\crashweb\reports\1 폴더가 생성되고
    - 실제로는 d:\crashweb\scheduled\GametreeQA\reports 폴더를 가리킴
  + d:\crashweb\reports> junction 2 d:\crashweb\scheduled\Gametree\reports
    - d:\crashweb\reports\2 폴더가 생성되고
    - 실제로는 d:\crashweb\scheduled\Gametree\reports 폴더를 가리킴
  + ...

## 3.3. 이슈 트래커(Redmine 또는 MANTIS BT)와 연동하는 방법

이 기능은 최초에 Mantis만 지원을 했었는데, Mantis에 로그인한 상태로 한두 번의 클릭만 해서 이슈를 등록할 수 있습니다. Redmine을 사용하는 경우엔 간단히 링크 정도만 지원하고 있습니다(즉, 다른 이슈 트래커도 링크만 제공하는 식으로 연결할 수 있습니다).

1. Browser 프로젝트의 site.js 파일을 엽니다.
2. 제일 위의 local config 섹션에서 create\_issue\_tracker\_link 함수를 수정합니다.
   1. 다른 팀에서 사용하는 코드가 함께 있으므로
   2. 참고하시면 쉽게 직접 추가하실 수 있을 것입니다.
3. Redmine 또는 Mantis 등에 로그인한 상태에서 테스트를 진행합니다.
   1. Browser에서 특정 오류를 클릭하고
   2. Report this to MANTIS! 링크를 클릭합니다.
   3. 오류 보고 화면이 바로 뜨고, 확인 버튼만 눌러서 이슈가 잘 등록이 되는지 확인합니다.
      1. Mantis의 경우 내용이 자동 입력되므로 확인 버튼만 누르면 됩니다.
      2. 다른 이슈 트래커는 내용을 직접 입력하셔야 해요.
         1. GET 방식을 이용해서 파라미터를 전달하는 방법이 있으면 Mantis처럼 내용을 자동 입력할 수 있습니다.

## 3.4. 머큐리얼[7](#FOOTNOTE-7) 등의 [SCM[8](#FOOTNOTE-8)](http://ko.wikipedia.org/wiki/%EB%B2%84%EC%A0%84_%EA%B4%80%EB%A6%AC)과 연동하는 방법

콜스택 상세 보기 화면에서, 소스 위치에 파일명이 출력됩니다. 이 정보를 이용해서, 기존에 SCM과 연동이 가능합니다.

**시작하기 전에**:

* HTTP 형태로 소스에 접근할 수 있는 경우에만 가능합니다.
* 머큐리얼은 보통 HTTP를 사용하므로, 이 기능을 적용하기에는 아주 적격.
* 만약 추가 설정이 필요하다면 (SCM 관리자의 할 일)입니다.

**설정하기**:

1. Browser 프로젝트의 site.js 파일을 엽니다.
2. local config 섹션에서 create\_source\_link 함수를 알맞게 고칩니다.
   1. SCM별로 리턴할 값을 만드는 작업이 달라서 자세한 설명은 생략하겠습니다.
   2. 다른 팀에서 사용하는 코드가 같은 곳에 있으니, 참조해서 수정해 주세요.
3. 콜스택 상세 보기 화면에서 소스위치의 파일 명을 클릭했을 때, 원하는 결과가 출력되는지 확인합니다.

## 3.5. 새 프로젝트를 추가하는 방법

새로운 프로젝트를 추가하거나, 기존 프로젝트의 배포 버전(사내 개발 버전, 사용자 배포 버전 분리 같은)에 따라 구분해서 사용할 수도 있습니다. 이때, 위의 설정을 다시 다 하는 것은 아니고, 설치 단계에서 생성한 데이터베이스(CRASHWEB\_DB)에 새로운 프로젝트를 추가하는 방법으로 비교적 쉽게 할 수 있습니다.

DB 관련된 부분은 (DBA의 할 일)입니다. 나머지는 (서버 프로그래머의 할 일)입니다.

1. DB에 프로젝트 추가
   1. INSERT INTO dbo.tbl\_project VALUES ("게임이름 DEV");
2. 위에서 추가한 프로젝트의 project\_uid 값을 적어 둡니다.
   1. SELECT \* FROM dbo.tbl\_project 로 조회
3. 새로 추가한 프로젝트가 보고하는 오류를 접수할 Reception 서버를 구성합니다.
4. 심볼 서버가 구성되지 않았다면, 먼저 설정하도록 합니다.
5. CrashGatherer 설정
   1. CWRUN.js 파일을 열고
   2. CrashGatherer.exe 실행 인자의 4번째로 새로 추가한 프로젝트의 project\_uid 값을 입력합니다.
6. 오래된 파일의 자동 삭제를 위해 DeleteOld.js 파일을 열어서 대상 폴더를 입력합니다.
7. Browser는 있던 것을 그대로 쓰면 되고, 위쪽의 '프로젝트 선택' 메뉴에서 프로젝트를 변경해 봅니다.
8. 이상 없이 기본적인 테스트가 끝나면, CWRUN.js와 DeleteOld.js를 스케줄러에 등록합니다.

## 3.6. 팀 사이트(게시판)과 연동하는 방법

팀원들이 자주 가는 팀 사이트에 '오늘의 크래시 발생 횟수' 같은 정보를 출력하는 방법입니다.

1. 해당 웹 서버에서 CRASHWEB\_DB에 접근할 수 있도록 설정합니다.
2. 다음 저장 프로시저를 통해 오늘의 크래시 발생 횟수를 구할 수 있습니다:
   1. dbo.usp\_select\_daily\_count [project\_uid], 0, 0

## 3.7. 이메일 (또는 [야머](http://ko.wikipedia.org/wiki/%EC%95%BC%EB%A8%B8))와 연동하는 방법

짧은 시간에 같은 오류가 반복적으로 발생하면, 매우 위험한 상황일 가능성이 높습니다. 이를 가능한 빨리 알아차리기 위해 이메일 알람을 설정하는 방법이 있습니다.

메일 발송 관련 DB 수정은 (DBA의 할 일)입니다. 하지만 다른 부분도 있으니 서버 프로그래머도 같이 설정하고 확인해 보시기 바랍니다.

1. 메일 발송은 DB에서 msdb.dbo.sp\_send\_dbmail 방식을 이용합니다.[9](#FOOTNOTE-9)
   1. 먼저 메일 발송이 가능하도록 설정합니다.
   2. 기본 계정: CrashwebUser
   3. 실제 메일을 발송할 SP: dbo.usp\_insert\_report\_info
2. SQL Server Management Studio를 열어서 dbo.usp\_insert\_report\_info를 편집합니다.
   1. 현재는 설정이 따로 빠져 있지 않습니다(직접 프로시저를 편집해야).
   2. msdb.dbo.sp\_send\_dbmail 호출하는 부분을 보시고
      1. recipients, copy\_recipients 등의 설정을 합니다.

### 3.7.1. 알람이 발송되는 조건을 수정하려면?

msdb.dbo.sp\_send\_dbmail 호출하는 부분 바로 위에 기본 조건이 몇 개 있습니다. 기본 조건이 마음에 들지 않으면, 적절하게 수정해서 원하는 때에 메일 알람을 받으실 수 있습니다.

**기본 조건**:

* 5분 안에 동일한 오류가 3개 이상 보고되었는가?
* 10분 안에 동일한 오류가 3개 이상 보고되었는가?
* 30분 안에 동일한 오류가 10개 이상 보고되었는가?
* 시간에 관계 없이 동일한 오류가 10개 이상 보고되었는가?

### 3.7.2. 야머와 연동하는 방법

어떤 팀에서는 야머의 이메일 포스팅 기능을 이용해서 야머와 연동하고 있습니다.

1. 도메인에 새 계정을 하나 만듭니다.
   1. 구글 앱스 사용자는 [구글 그룹스](http://groups.google.com/a/ntreev.com)를 통해서 전용 이메일 계정을 하나 만듭니다.
      1. 무료로 이메일과 비슷한 기능을 할 수 있으므로
      2. 그룹스는 비공개로 만드는 것을 추천
2. 이 계정으로 야머 계정을 하나 생성합니다.
3. 이 계정으로 야머에 로그인 한 후에 이메일 포스팅이 가능하도록 설정을 합니다.
   1. 비공개 그룹에 포스팅하려면 해당 그룹에 가입이 되어 있어야 하고
   2. 이메일 포스팅은 기본적으로 막혀 있으니, 설정에서 풀어야 하고 등
      1. Account - Notifications - Post by Email - Require me to confirm…
4. 이메일 포스팅이 잘 되는지 확인합니다.
   1. 이메일 포스팅 기능에 대해서는 야머 공식 사이트의 도움말을 참조
5. msdb.dbo.sp\_send\_dbmail 호출 시에 copy\_recipients에 포스팅 대상 이메일 주소를 추가합니다.
6. DB 메일을 보내는 사람을 새로 생성한 그룹스 계정으로 변경합니다.

## 3.8. 서버 프로젝트에 적용하는 방법

아직 서버 프로젝트에 적용된 경우는 없는 것 같습니디만, 매뉴얼의 '클라이언트'를 '서버 프로그램'으로 간주하고 보시면 동일하게 적용이 가능할 것입니다.

## 3.9. 윈도 심볼 등, 추가 심볼에 대하여 (중요)[10](#FOOTNOTE-10)

윈도 심볼을 포함해, 프로젝트에서 사용하는 3rd party 라이브러리의 심볼 등, **가능한 많은 심볼을 CDB에 제공해야 합니다.** 그렇지 않으면, 잘못된 정보를 보여줄 수도 있습니다.

1. 보유하고 있는 심볼은 모두 심볼 서버에 올립니다.
2. 윈도 심볼을 위해 환경변수 \_NT\_SYMBOL\_PATH를 설정합니다.[11](#FOOTNOTE-11)
   1. [내 컴퓨터] - [속성] 창을 열고 - [고급] 탭에서 - [환경 변수] 창을 엽니다.
   2. 해당 유저 또는 시스템 환경 변수에 \_NT\_SYMBOL\_PATH 항목을 새로 만듭니다.
      1. 값 symsrv\*symsrv.dll\*d:\localsymbols\*http://msdl.microsoft.com/download/symbols
      2. d:\localsymbols 부분은 시스템 상황에 맞게 수정하세요.

## 3.10. 심볼이 원격 서버에 있을 때 발생하는, 심볼 로딩 속도 문제 해결하기

클라이언트 심볼의 경우에는 한두 달 정도의 심볼 파일만 가지고도 수십 기가에 이릅니다. 이때, 심볼이 원격지에 있으면, 내부 네트워크라고 하더라도 **비정상적으로 속도가 느립니다**. 심볼을 로드하는 데에만 10분 이상이 걸릴 수가 있습니다(매번 참조할 때마다). 이때, CWRUN.js 파일의 symServer 설정에서 로컬 캐시를 사용하는 구문을 사용해서 해결할 수 있습니다.

* 캐시 없는 버전: var symServer = "\"\\\\orange\\shared\\@sym2\" ";
* 캐시 있는 버전: var symServer = "\"**SRV\*d:\\crashweb\\symbols\***\\\\orange\\shared\\@sym2\" ";

※ 주의: 오래된 캐시는 삭제하도록 DeleteOld.js 파일에도 관련 내용을 추가하시길 권장합니다.

# 4. FAQ

완벽한 매뉴얼이라 문의사항은 없을 것 같습니다만...

# 5. 기타 정보

## 5.1. 소스 위치

* <https://github.com/NtreevSoft/Crashweb>

## 5.2. 적용한(혹은 예정인) 팀

## 5.3. 연락처

* 이메일: [flow3r@gmail.com](mailto:flow3r@gmail.com)

notes

1 Windows Server 2003 이상의 운영체제에 IIS를 설치했으면 버전이 6.0 이상이실 겁니다.  
  
2 참고 문서: Using SymStore <http://msdn.microsoft.com/en-us/library/ms681417.aspx>  
  
3 고의로 crash를 발생시키려고 프로그램을 수정했다면, 심볼 서버에도 등록하는 것을 잊지 않도록 합니다.  
  
4 상업적인 용도로도 무료로 이용 가능합니다.  
  
5 NcFTPd Server를 받지 않도록 주의하세요. NcFTP Client를 받아야 합니다. <http://www.ncftp.com/download/>  
  
6 이 버전을 사용하면, 까만 콘솔 창이 주기적으로 뜨는 것을 방지할 수 있습니다. 설정도 약간 더 쉽습니다. 다만, cscript가 기본으로 설정되어 있으면 여전히 콘솔 창이 뜰 수도 있습니다.  
  
7 머큐리얼(Mercurial): Hg라고도 하며, 통상 분산 버전 관리 시스템(DVCS)으로 분류됩나다. 쉽게 설명하면, 서브버전이나 소스세이프와 비슷한 프로그램입니다.  
  
8 SCM: 버전 관리 시스템. 서브버전이나 소스세이프와 같은 프로그램을 통칭하는 용어입니다.  
  
9 이 과정에서 SMTP 서버가 필요합니다.  
  
10 이 항목은 2011년 3월 4일에 추가했습니다.  
  
11 마이크로소프트 지원 사이트의 관련 문서:   
<http://support.microsoft.com/kb/311503/en-us>