

Web ベース電子ログ及び画像ログシステムの開発

DEVELOPMENT OF WEB BASED ELECTRONIC LOG AND PICTURE LOG SYSTEMS

路川 徹也^{#, A)}, 帯名 崇^{B)}

Tetsuya Michikawa^{#, A)}, Takashi Obina^{B)}

^{A)} e-JAPAN IT Co.,Ltd.

^{B)} High Energy Accelerator Research Organization, KEK

Abstract

A paper log book has been used usually to record various operational and experimental results. On the other hand, an electronic log system based on web technology becomes popular in recent years. In KEK, some in-house electronic log system has been developed and used for years. The e-log system has a tight dependence on the old software framework, and the migration to the other system was difficult. A new e-log system has been developed to realize the easy installation and maintenance work. The system has been used for KEK PF, cERL and some other small-sized experiments.

1. 概要

従来、加速器の運転や実験を行う際にはログブック等の紙媒体に記述することが一般的であったが、近年では Web 技術を使用した電子ログシステムが使われるようになってきた。KEK 加速器でも独自の電子ログシステムが運用されてきたが、古いシステムへの依存が多く、別サーバーへの移植が困難という欠点があった。そこで、外部プログラムへの依存が少なく、インストールや運用が容易な電子ログシステムと画像ログシステムを新たに開発した。本稿では、その設計と実装について述べる。開発した電子ログシステムは PF-Ring および cERL のほか、いくつかの小規模実験にも使用されている。

2. はじめに

紙媒体の実験ノートや運転ログブックには改ざん防止の効果が高いことのほか、写真や図などを自由にレイアウトできて情報を追記できることなど、数多くのメリットがある。しかしながら、多くの人員が係わる運転や実験では情報のリアルタイム性や共有性、そして検索性という面では劣っていたのも事実である。そのため、近年はそれらの問題を解決するために、Web 技術を利用した電子ログシステムが使われるようになってきている。以前から、KEK 内でも、KEKB や Linac でそれぞれ独自の電子ログシステムを運用しており、PF でも長年 KEBK のシステムを利用してきた。しかし、KEKB の電子ログシステムは KEBK のサーバー上にあり、KEKB のネットワークに問題が発生すると、使用できなくなる問題が生じていたほか、古いシステムに依存している部分が多く別サーバーへの移植が難しいという欠点があった。

これらの問題を解決するため、外部プログラムへの依存関係が少なく、インストールが容易な電子ログシステムを開発した。このシステムは、電子ログシステム(botlog)と画像ログシステム(PrintAnyServer)の 2 システムで構成されており、連携することも、個別に運用することも可能となっている。botlog は通常の Web 電子ログシステムで必要な機能を実装するとともに、EPICS[1]レコードと連動し

て、自動的にログを登録することも可能となっている。また、PrintAnyServer は Web ブラウザから画像登録できるように構築しているため、クライアント OS に依存しない運用が可能となっている。Windows クライアントでは、専用のアプリケーションを用意することで、利便性を高めている。

両システムを設計する上での共通の方針としては、

- サーバー側の使用言語は、python2.7[2]
- Web クライアントは HTML + javascript(jQuery[3])で作成
- http サーバーは apache[4] 2.2 以降
- クライアントからデータへのアクセスは Web API(Web Application Programming Interface)を使用し、データフォーマットは JSON(JavaScript Object Notation)
- ファイヤーウォール外からも閲覧可能な設定が可能
- ログイン等のユーザー管理は行わない。

botlog の設定方針は以下の様に設定した。

- ログデータは追記のみとし、削除は行わない。
- データ保存にデータベース(RDB)を使用する。
- 使用可能な DB は、SQLite3[5]又は PostgreSQL[6]とするが、他の DB にも対応できるように設計する。
- ログに添付する画像情報は PrintAnyServer から取得。
- Web API を使用して、自動的にログを登録可能なシステムを提供する。

PrintAnyServer の設計方針は、

- 画像データは個別ファイルとして特定のディレクトリに保存。RDB には格納しない。
- ファイル名は特定の日付フォーマットで作成し、それ以外のファイル名は使用しない。
- ファイル情報の管理には、各ディレクトリ内にテキストのインデックスファイルを作成して行う。
- 各ファイルの追加情報としてキャプションを設定可能としてインデックスファイルに保存する。
- サムネイル画像を自動作成。
- 画像情報をキャッシュすることで画像表示を高速化。

[#] hig-mchi@post.kek.jp

- CUPS[7]を使用した印刷機能を提供する。

各プロセスの関係と使用するプロトコルを Fig. 1 に示す。

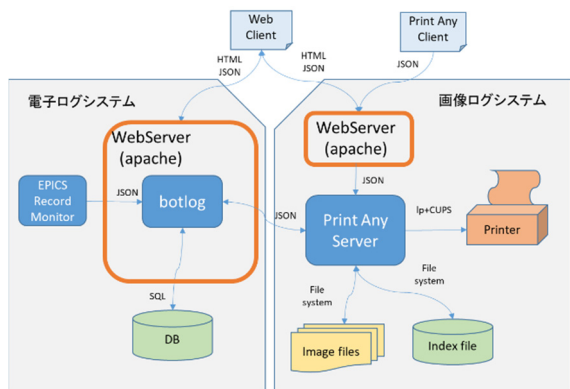


Figure 1: 電子ログシステム、画像ログシステム関連図

3. 電子ログシステム(botlog)

Web 技術を使ったシステムにおいて、電子ログシステムに限らず、システムの基盤となるフレームワークの選定は、開発方針を決定するにあたり非常に重要なものとなっている。実際の問題として、KEKB で開発された電子ログシステム Zlog[8]は、フレームワークとして Python で記述された Zope[9]を使用しているが、このフレームワーク自体の依存関係やバージョン間の互換性の問題が、Zlog の新規インストールや移植の妨げになっている。

そこで、今回の開発においては、依存関係をなるべく減らす観点から、bottle[10]という軽量フレームワークを使用することにした。他の候補としては、Django[11]、RubyOnRails[12]等もあるが、大規模で複雑な開発環境が必要なフレームワークを使用することは、Zlog と同じ問題を引き起こす可能性が高いので、候補から外した。

bottle の特徴としては、以下のようなものが挙げられる。

- Python で記述されている。
- シンプルで軽量(1 ファイル)。
- wsgi(Web Server Gateway Interface)が使用可能。
- テンプレート機能を内蔵。

今回開発した電子ログシステムの名称は、bottle を使用したログシステムなので、botlog とした。

3.1 ログ画面 ユーザーインターフェース

ログ画面には、大きく分けて編集モードと閲覧モードがあり、閲覧モードには印刷用画面表示機能がある。

3.1.1 編集モード画面

編集モード画面(Fig. 2)では、ログの記入と表示を行う。画面上部のログ入力部にある入力ボックスでログを記入し、画像を添付する際には、リストから選択した画像を設定する。ログの記入が終了したら、登録ボタンを押下することで、ログが登録され、画面中央のログ表示部のログリストに反映される。ログリストは 30 秒間隔で自動的に更新することで、他の人が記入した内容もほぼリアルタイムで見ることができる。画面下部には表示選択部の領域を設けており、ログの表示内容のうち自動登録や障害ログの表示・非表示を切り替えるチェックボタンや、表示するグ

ループの絞り込みを行うドロップダウンメニュー、表示する期間を変更するメニュー等を備えている。

また、過去に入力した内容を変更する場合には、ログリスト内の登録日付時刻へのリンクをクリックする。変更用ダイアログがポップアップ表示されるので、そこで編集作業や項目の削除を行うことができる。



Figure 2: 編集モード画面

3.1.2 閲覧モード画面

閲覧モード画面(Fig. 3)は、表示期間の変更や、文字列検索をするために使用する。文字列検索後にログリストの日付を選択することで、選択した時間を中心として前後の時間帯のログを表示することができる。これは例えば何らかのトラブルを検索したときに、その前後の時間帯でどのような操作を行ったかを簡単に表示できるためとても有効な機能である。

閲覧モードでは時間降順に表示するため主として表示後に紙媒体に印刷し・保存する用途に適している。



Figure 3: 閲覧モード画面

3.2 プログラム内部構成

3.2.1 初期設定

電子ログシステムを運用するにあたっては、実験や施設毎に別のサーバーを用意するのが簡単である。1 サーバーで複数の電子ログシステムを運用することも可能だ

が、その場合は全てのログシステムに共通の機能と、個別の電子ログシステムに必要な機能を分離し、他のログシステムに影響を及ぼさないようにシステム設計することが必須である。

現在、個別の電子ログシステム毎の設定は、botlog 内の設定用スクリプトを直接編集する必要があり、設定の変更後は、http サーバーの再起動を行う必要がある。http サーバーとの連携には wsgi を使用しており、http サーバー側でも wsgi に対応するためのモジュールのインストールや設定をする必要がある。

また、使用するDBの初期化やテーブル作成はbotlog 起動前に行う必要がある。利便性のため全てのテーブルを作成するスクリプトを用意してある。

3.2.2 Web API

データの登録/参照は、Web API を使用して行い、データ形式は JSON を使用する。現在、botlog で使用している Web API は次のようなものとなっている。

- /botlog/editLog ログ登録/変更/削除
- /botlog/getLog ログ取得
- /botlog/getImgList イメージリスト取得
- /botlog/getSection セクション名取得
- /botlog/getLogID/'idno' ID 指定ログ取得
- /botlog/getImgPAS 画像情報同期
- /botlog/getSvrTime サーバー時間取得
- /botlog/getAllImageDates 画像日付一覧取得
- /botlog/fixedphrase 定型文登録/変更/削除

送受信するデータ文字列は日本語を含んでいるため、そのまま送受信データとして扱うと文字化けして判別不能になる。このため、データ文字列を base64 変換して送受信することで、この問題を解決している。

3.3 DB

データの保存は DB を使用する。現時点で使用可能な DB システムは、SQLite3 と PostgreSQL である。DB のアクセスには、共通のインターフェイス関数を使用しているため、他の DB への対応も比較的容易である。

ログを保存するために必要なテーブルは、4 テーブルで補助データとして定型文テーブルがある。

各テーブルの関連を Fig. 4 に示す。

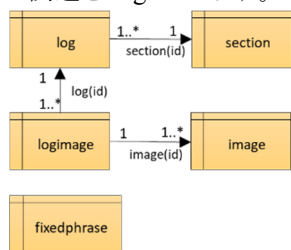


Figure 4: DB テーブル関係

3.3.1 log テーブル定義

log テーブルは、ログ情報を保存する。ログの更新を行う際には、まず新規にレコードを追加し、org_id に元データの ID を設定することで元データとの関連付けを行う。ログに表示するデータは、同じ org_id を持つデー

タのうち entrytime が最新のものを使用する。データの削除を行う際には、変更と同様に新規レコードを追加し、その際に disptime を NULL に設定することで、削除とみなす。Table 1 にテーブル構造を示す。

Table 1: Log Table

| 名前 | 型 | 制約 | 説明 |
|------------|-----------|-------------|----------|
| id | bigserial | index | ログ ID |
| entrytime | timestamp | index | 登録時間 |
| disptime | timestamp | | 表示時間 |
| section_id | smallint | section(id) | セクション ID |
| org_id | bigint | | 元 ID |
| log | text | | ログ文字列 |
| mode | smallint | | モード情報 |

モード情報は、データの種別を表すために使用し、bit で設定を行う。具体的なデータ種別を Table 2 に示す。

Table 2: Mode Bit

| Bit | 予約 | 説明 |
|-----|------|--|
| 1 | 予約 | ALL で使用するので設定不可 |
| 2 | 自動登録 | 1: 自動登録 |
| 3,4 | 予約 | 未使用 |
| 5,6 | アラーム | 00:normal, 01:warning, 10:alarm, 11:critical |
| 7,8 | 予約 | 未使用 |

3.3.2 image テーブル定義

image テーブルは、画像情報を保存する。通常は、PrintAnyServer から取得した画像情報を保存するために使用する。Table 3 にテーブル構造を示す。

Table 3: Image Table

| 名前 | 型 | 制約 | 説明 |
|-----------|-----------|----------|--------|
| id | bigserial | index | 画像 ID |
| entrytime | timestamp | index | 登録時間 |
| url | text | not null | 画像 URL |
| caption | text | | キャプション |

3.3.3 logimage テーブル定義

logimage テーブルは、ログに登録した内容(id)と、そのエントリに表示する画像データとを紐付けするために使用する。width,height はログに画像を表示する際のサイズを設定し、NULL ならば画像をオリジナルサイズで表示する。複数画像を表示する際の順序も指定可能である。Table 4 にテーブル構造を示す。

Table 4: Logimage Table

| 名前 | 型 | 制約 | 説明 |
|----------|---------|-----------|-------|
| log_id | bigint | log(id) | ログ ID |
| image_id | bigint | image(id) | 画像 ID |
| order_no | integer | not null | 表示順 |
| width | integer | | 画像幅 |
| height | integer | | 画像高さ |

3.3.4 section テーブル定義

section テーブルは、ログのグループ情報を保持する。id=0 は”ALL”、id=1 は”Undecided”という予約番号としており、それ以外のグループは使用するサイトによって、設定を変更する。例えば cERL では RF, Laser, Vacuum, Magnet などのグループ名称のほか、Study, Operation などの名称を使用している。Table 5 にテーブル構造を示す

Table 5: Section Table

| 名前 | 型 | 制約 | 説明 |
|------|-------------|---------|--------|
| id | integer | primary | ID |
| name | varchar(32) | UNIQUE | セクション名 |

3.3.5 fixedphrase テーブル定義

fixedphrase テーブルは、定型文を登録するために使用する。このテーブルは、ログ入力画面に入力する定型文を登録しているだけなので、他のテーブルとは直接関連がない。このテーブルだけは、データの変更/削除を行うように実装している。

Table 6 : Fixedphrase Table

| 名前 | 型 | 制約 | 説明 |
|-----------|-----------|---------|------|
| id | integer | primary | ID |
| entrytime | timestamp | | 登録時間 |
| phrase | text | | 定型文 |

3.4 EPICS Record Monitor

EPICS Record Monitor は、EPICS のレコード更新に伴って、設定された条件に合致した場合に自動的にログ登録を行うプログラムである。このプログラムは、botlog の Web API を使用した別プログラムであり、botlog と連携して使用する単独プログラムである。このプログラムは、各 EPICS レコードと条件、出力メッセージの設定を python で記述し、特定のディレクトリ内にある設定ファイルを起動時に読み込んで動作に使用する。

メッセージ登録処理は、編集画面からの登録と同様に Web API を使用して行い、mode に自動登録フラグを設定することで、ユーザーが登録したものと区別できるようにしてある。

4 画像ログシステム(PrintAnyServer)

今回の画像ログシステムを作成する前から同様のシステムは存在していた。従来の画像ログシステムでは、サーバープロセスが存在せず、サーバーの共有ディレクトリを各クライアントがマウントした状態にして、決められたファイル名の形式でディレクトリに保存し、インデックスファイルを更新する方法をとっていた。この方法では、違う種類のクライアント OS によってインデックスファイルやディレクトリの書込権限が取得できなくなることや、別クライアントから同時にファイルを登録した際にファイル名が重なり、上書きされてしまうことが何度か発生していた。

そこで、画像ファイルの上書き回避と、ファイルの書込権限の問題を解消するためにデータ保存用のサーバーを作成した。同時にファイルとディレクトリ情報をキャッシュすることで、複数クライアントからのアクセスによるディスクへの I/O を減らすことで応答の高速化にも対応した。

4.1 表示画面

画像ログシステムの Web 表示は、表示選択部のカレンダーで日付を選択すると、当該日時の画像一覧が表示される。画像の登録されていない日付はグレーアウトして選択できないようになっている。デフォルトでは時間順に表示されるが、逆順に表示することも可能とした。画像リストの自動更新機能は無く、画像が登録されたかを確認するためには、ブラウザをリロードするか”今日”ボタンを押下する必要がある。リストに表示されている画像をクリックすると、拡大表示される。

この Web 画面には画像登録の Client 機能がついている。クリップボードに画像をコピーしたのちに”Paste clipboard image or Drop image file”の部分で”貼り付け”をするか、画像ファイルをドラッグ&ドロップすると登録確認ダイアログが表示される。ダイアログ上でキャプションを入力し、登録ボタンを押下すると、画像が登録される。また、同等の機能を持つ Windows 版の Client プログラムも作成してあるので、ダウンロードして使用することも可能となっている。Fig.5 に実行例を示す。



Figure 5: 画像ログシステム Web 画面

4.2 内部構成

PrintAnyServer は、キャッシュデータをメモリ上に展開している関係で、独自のポート番号を使用した http サーバーとして運用している。そのままでは、firewall の外からアクセスすることができないため、http サーバー上に Web API の wrapper を作成し、PrintAnyServer に接続することで、問題を回避している。

4.2.1 設定ファイル

PrintAnyServer の設定ファイルは、botlog とは違い Windows ini ファイル形式を使用している。これは、PrintAnyServer が単独サーバーとして動作しているの、設定ファイルの読み込みは起動時にしか行わないためである。

4.2.2 Web API

データの登録/参照は、botlog と同様に Web API を使用して行い、データ形式は JSON を使用する。現在、PrintAnyServer で使用している Web API は次のようなものとなっている

- /api/addImage 画像登録
- /api/existDate 画像存在日付取得
- /api/images 画像情報取得
- /api/getPrinters プリンタ情報取得
- /api/getClientFileName 専用クライアントファイル名取得

PrintAnyServer も botlog と同様に、文字列と画像データを base64 変換してデータの送受信を行っている。

4.2.3 画像データファイル構造

画像データを保存するディレクトリとファイルの関係は Fig. 6 の様になっており、Zlog で使用している方式の後方互換となっている。



Figure 6: 画像データ保存ディレクトリ構成

ここで YYYY は西暦、MM は月、DD は日、HHMMSS は時分秒を表している。また、赤字になっている部分は、PrintAnyServer で独自に追加した仕様となっている。

各日付ディレクトリ内にある index.txt には、ファイル名とそれに対応するキャプション文字列が記述してある。また、index_list.txt には年月と日付ディレクトリの一覧が記述されていて、日付一覧を取得する際にディレクトリを再帰検索しなくてもディレクトリの日付一覧が取得できるようになっている。また、サムネイル画像も同時に作成する仕様とした。

4.2.4 画像登録

クライアントが送信する画像データはデフォルトでは png フォーマットの画像に変換される。ただし jpeg など画像ファイルを直接ドラッグ&ドロップでクライアントから送る場合には元データと同じフォーマットで保存する。

サーバー側での画像登録は、クライアントから渡される JSON データから画像データの部分を取り出し、サーバー時間からファイル名を作成後、画像データに変換して保存する。その際、同じ名前のサムネイル画像を作成し、.thumbnail ディレクトリに保存する。作成するサムネイル画像は、縦横比はそのまま、最大 180px になるように変換する。

画像ファイルが正常に作成されると、index.txt や index_list.txt の情報を更新し、同時に内部キャッシュにも登録を行う。

複数のユーザーが同時に画像登録を行った場合には、

1秒ずらした時間でファイル名を作成し、最大5枚までは対応可能とした。これは枝番をつけて対応することも可能であったが、過去のファイルフォーマットとの互換性を重視したためである。

4.2.5 画像データキャッシュ

画像データファイルのキャッシュは、独自の python 辞書形式でプログラム内に保持している。データキャッシュは、PrintAnyServer 起動時に index_list.txt と各ディレクトリ内の index.txt を読み込んでいる。そのため、画像ファイルが多くなってくると起動までに多少時間がかかることがある。

5 今後の課題

5.1 電子ログシステム

既知の問題として、複数の端末(クライアント)から同時に同じ ID のログ変更を行った場合、後から修正したログで上書きされてしまうことが判明している。この場合、後から保存しようとしたユーザーに対して編集を開始した時点から既にログが変更されたことを通知するのみにするか、あるいはいちど変更を受け付けてから、ユーザーにマージさせるかなど、いくつか対応策を検討している。

5.2 画像ログシステム

キャッシュデータの DB 化、画像ファイル名の枝番対応、画像ファイルやキャプションの変更/削除、簡単な画像編集機能等色々な要望が挙げられているので、順次対応していく予定である。

参考文献

- [1] Experimental Physics and Industrial Control System; <http://www.aps.anl.gov/epics/>
- [2] python 2.7; <https://www.python.org/>
- [3] jQuery <https://jquery.com/>
- [4] apache <https://httpd.apache.org/>
- [5] SQLite3; <https://sqlite.org/index.html>
- [6] PostgreSQL <https://www.postgresql.org/>
- [7] CUPS <https://www.cups.org/>
- [8] K.Yoshii *et al.*, "Zope Based Electronic Operation Log System - Zlog", Proceedings of the 1st Annual Meeting of Particle Accelerator Society of Japan and the 29th Linear Accelerator Meeting in Japan, Funabashi Japan, August 4 - 6, 2004; http://www.pasj.jp/web_publish/pasj1_lam29/WebPublish/5P28.pdf
- [9] Zope <http://www.zope.org/en/latest/>
- [10] bottle <https://bottlepy.org/docs/dev/>
- [11] Django <https://www.djangoproject.com/>
- [12] RubyOnRails <https://rubyonrails.org/>