



# オープン・ジャーナル・システム

バージョン 2.1

技術参考資料

改訂 3 版

最終更新日 (翻訳版) 2007 年 03 月 07 日



本資料は、クリエイティブ・コモンズの帰属-非営利-派生禁止ライセンスの下にライセンスされています。本ライセンスのコピーを閲覧するには、<http://creativecommons.org/licenses/by-nc-nd/2.0/ca/>にアクセスするか、以下にお問い合わせください。

Creative Commons,  
559 Nathan Abbott Way,  
Stanford, California 94305, USA.



## 目次

はじめに .....	3
Public Knowledge プロジェクトについて .....	3
オープン・ジャーナル・システムについて .....	3
この文書について .....	4
規約 .....	4
技術 .....	4
設計概要 .....	5
規約 .....	5
全般 .....	5
ユーザインターフェース .....	5
PHP コード .....	5
データベース .....	6
セキュリティ .....	6
はじめに .....	7
ファイル構成 .....	8
リクエストの処理 .....	10
URL .....	10
リクエストの処理例 .....	10
リクエスト処理コードの配置場所 .....	11
データベース設計 .....	13
クラスの解説 .....	17
クラス階層 .....	17
ページクラス .....	23
はじめに .....	23
アクションクラス .....	24
モデルクラス .....	25
データアクセスオブジェクト (DAO) .....	25
サポートクラス .....	26
メール送信 .....	26
国際化 .....	27
フォーム .....	28
設定 .....	29
コアクラス .....	29

データベースサポート .....	30
ファイル管理 .....	31
定期的タスク .....	31
セキュリティ .....	32
セッション管理 .....	32
テンプレートサポート .....	33
ページングクラス .....	33
プラグイン .....	34
共通タスク .....	34
メール送信 .....	34
DAO によるデータベースとの相互作用 .....	35
ユーザインターフェース .....	36
変数 .....	36
関数と修正子 .....	38
プラグイン .....	40
オブジェクトとクラス .....	41
サンプルプラグイン .....	41
ローダースタブ .....	42
プラグインオブジェクト .....	42
登録関数 .....	43
フックへの登録とコールバック関数 .....	43
プラグイン管理 .....	45
その他のプラグイン機能 .....	46
フックリスト .....	47
OJS の翻訳 .....	78
謝辞 .....	79
さらなる情報の入手 .....	80



## はじめに

### Public Knowledge プロジェクトについて

Public Knowledge プロジェクト (<http://pkp.sfu.ca>) は、学術研究の専門的・公共的価値を向上させるために新しい技術が利用できるか否か、できるとしたらどのように利用するかを探求しています。研究図書館員だけでなく様々な分野の研究者と共同で、持続的・世界的にアクセス可能な形態をしたこれら知識の学術的品質、アクセスの容易さ、および整合性を改善するために使用するオンライン基盤や知識管理戦略の社会的、経済的、技術的問題を研究しています。プロジェクトは、主題図や博士論文などの規格だけでなく、Open Archives や InterPARES などのデジタルライブラリへのアクセスや文書保存に関する新しい規格を統合することをめざしています。

### オープン・ジャーナル・システムについて

オープン・ジャーナル・システム (OJS) は、研究成果へのアクセスの拡大・改善を目的とする連邦政府の助成を受けた Public Knowledge プロジェクトにより開発された雑誌の管理・出版システムです。投稿からオンライン出版、インデックス作成にいたる査読付き雑誌を出版するプロセスのあらゆる側面を支援します。そして、その管理システム、研究成果に対する適切なインデックスの付与、関連情報の提供を通じて、査読を受けた研究成果の学術的・公共的品質を改善することをめざしています。OJS は、雑誌を世界中に公開するためのオープンソースソフトウェアであり、多くの雑誌をオープンアクセスで出版するための現実的なオプションを提供するものです。オープンアクセスは、世界規模の公益に貢献するだけでなく、雑誌の読者を増加させることもできるからです。

OJS バージョン 2.x は、OJS 1.x を全面的、あるいは部分的に使用した世界中の 250 誌の編集者の 2 年間の経験に基づいて、システムを完全に再設計して書き直したものです。OJS v2.0 の発表と同時に、Public Knowledge プロジェクトは、(オープン会議システムと PKP ハーベスタを含む)オープンソースソフトウェアの開発をサイモン・フレーザー大学図書館に移し、同大学にあるカナダ出版研究センターをパートナーに加えました。

OJS 2.x のユーザードキュメントは [http://pkp.sfu.ca/ojs\\_documentation](http://pkp.sfu.ca/ojs_documentation) で入手することができます。また、デモンストレーション用のサイトが [http://pkp.sfu.ca/ojs\\_demo](http://pkp.sfu.ca/ojs_demo) に用意されています。

## この文書について

### 規約

- コード例、ファイル名、URL、クラス名は、`courier` 書体で表します。
- コード例、ファイル名、URL、クラス名でサンプル値を示す場合は、角カッコ ([]) を使用します。例えば、`[anything]Handler.inc.php` は、`Handler.inc.php` で終わる任意のファイル名を表します。
- 多くの例で使用する URL の <http://www.mylibrary.com> は説明目的の架空のアドレスです。

### 技術

オープン・ジャーナル・システム 2.x は、オブジェクト指向 PHP (<http://www.php.net>) で書かれており、ユーザインターフェースの抽象化には Smarty テンプレートシステム (<http://smarty.php.net>) を使用しています。データは SQL データベースに格納されます。その際、ADODB データベース抽象化ライブラリ (<http://adodb.sourceforge.net>) を使用してデータベースの呼び出しを抽象化しています。

推奨するサーバ環境:

- PHP のサポート (4.2.x 以降)
- MySQL (3.23.23 以降) または、PostgreSQL (7.1 以降)
- Apache (1.3.2x 以降) または、Apache 2 (2.0.4x 以降) または、Microsoft IIS 6 (PHP 5.x が必要)
- Linux, BSD, Solaris, Mac OS X, Windows のいずれかのオペレーティングシステム

上記以外のバージョンやプラットフォームでも稼動する可能性はありますが、サポートはしていません。テストを行う予定もありません。ただし、上記の環境以外で OJS の稼動に成功したユーザからのフィードバックは歓迎します。

## 設計概要

### 規約

#### 全般

- ディレクトリ名は、lowerCamelCase スタイルの大文字使用法で命名します。
- OJS 2.x は複数の言語に翻訳されることを想定していますので、語順に関する仮定は一切行っていません。言語固有の文字列は該当するロケールファイルで定義し、必要に応じて変数置換を使用します。

#### ユーザインターフェース

- レイアウトは、カスケーディング・スタイル・シート (CSS) を使って、コンテンツから分離します。
- Smarty テンプレートは、XHTML 1.0 Transitional に適合させます (<http://validator.w3.org/> を参照)。

#### PHP コード

- 可能な限り、クラスの外に置くグローバル変数やグローバル関数は使用するべきではありません。
- 数値定数や文字列定数ではなく、PHP の define 関数を使って整数にマップしたシンボリック定数を使用します。
- ファイル名はクラス名に一致させます。例えば、SectionEditorAction クラスは、SectionEditorAction.inc.php ファイルに置きます。
- クラス名と変数の大文字使用法は次のようにします。クラス名は CamelCase を使用し、インスタンスは lowerCamelCase を使用します。例えば、MyClass クラスのインスタンスは \$myClass とします。
- 可能かつ論理的であれば常に、変数名はクラス名に一致させます。例えば、\$x のような適当な名前ではなく、\$myClass を使用します。
- クラス名とソースコードのファイル名は、記述的かつユニークな名前にします。
- 出力はできるだけ Smarty テンプレートに制限します。PHP コードによりレスポンスを出力しなければならない正当な状況とは、HTTP ヘッダが必要な場合です。
- パフォーマンスを上げ、サーバの負荷を下げるために、import(...) コールは、できるだけローカルで行います。
- 参照の振る舞いは、PHP のバージョンにより一定しないので、使用する場合は注意深く行います。パフォーマンスを上げるために、コンストラクタは通常参照で

呼び出し、オブジェクトを渡す際には可能であれば常に参照を使用します。

## データベース

- SQL テーブル名は複数形（例えば、users, journals）とし、小文字で記述します。
- SQL データベースの機能要件は、互換性を高めるためにできるだけ少なくします。例えば、日付の算術的扱いはデータベースにより互換性がないので、データベースレベルではなく、PHP コードレベルで処理を行います。
- SQL スキーマ情報はすべて、dbscripts/xml/ojs\_schema.xml で管理します（ただし、後で述べるように、プラグインのスキーマは除きます）。

## セキュリティ

- ユーザリクエストの妥当性は、ユーザインターフェースレベルと、関連する Page クラスの両方でチェックします。例えば、ユーザがあるボタンのクリックを許可されていない場合は、Smarty テンプレートを使って HTML レベルで無効とします。ユーザがこれを回避し、何らかの方法でボタンをクリックしても、そのフォームまたはリクエストを受け取る Page クラスでそれを確実に無視するようにします。
- 可能であれば常に、Smarty テンプレートエンジンの文字列エスケープ機能を使用し、HTML のセキュリティ上の穴やバグを避け、特殊文字が正しく表示されることを保証します。未チェックの HTML を入力できるのは、雑誌管理者とサイト管理者のみとし、かつ、特定のフィールド（雑誌設定の多言語フィールドなど）のみとします。例えば、ユーザ名を表示する場合は常に、`{ $user->getUsername() |escape }` を使用します。
- Smarty の `strip_unsafe_html` 修正子を使って HTML に制限を加えることができます。例えば、`{ $myVariable |strip_unsafe_html }`。

## はじめに

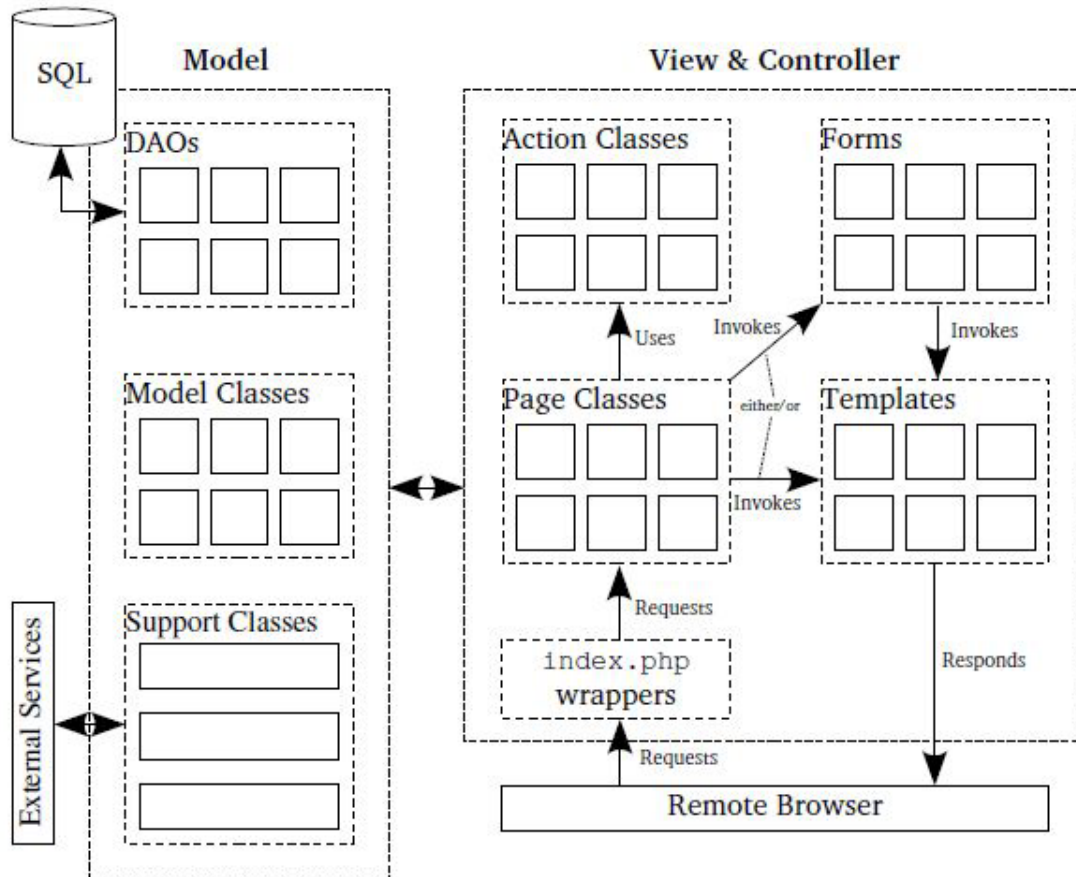
オープン・ジャーナル・システム 2.x の設計は、保守性、柔軟性、堅牢性を高めるために高度に構造化されたものです。そのため、初めて見た場合は複雑であると思うかもしれませんが、しかし、Sun のエンタープライズ Java ビーンズやモデル・ビュー・コントローラ (MVC) パターンになじみのある方は多くの類似点に気づくでしょう。

MVC 構造で見られるように、データの格納と表現、ユーザインターフェースの提供、制御の 3 つは異なる階層に分割されています。主要なカテゴリは、おおまかに「フロントエンド」から「バックエンド」の順に並べると以下の通りです。

- **Smarty テンプレート**。HTML ページを組み立てて、ユーザに表示することを担当します。
- **ページクラス**。ユーザの Web ブラウザからリクエストを受け取り、様々なクラスに委譲して必要な処理をさせ、適当な Smarty テンプレートを呼び出してレスポンスを作成します。
- **アクションクラス**。ページクラスにより使用され、ユーザリクエストの主要な処理を行います。
- **モデルクラス**。ユーザや論文、雑誌といったシステムの様々な実体を表す PHP オブジェクトを実装します。
- **データアクセスオブジェクト (DAO)**。一般に (何よりもまず) 関係するモデルクラスに更新、作成、削除関数を提供します。また、データベースとのすべてのやり取りを担当します。
- **サポートクラス**。コア機能、様々な共通クラス、関数などを提供します。

システムは継承を使用し、一貫したクラス命名規則を採用していますので、特定のクラスが所属するカテゴリを判別することは容易です。例えば、データアクセスオブジェクトクラスは常に DAO クラスを継承し、[Something]DAO という形のクラス名と、[Something]DAO.inc.php という形のファイル名を持っています。

下の図は、様々なコンポーネントとその相互作用を示しています。



## ファイル構成

OJS 2.x をインストールするとルートディレクトリには以下のファイルが置かれます。

ファイル/ディレクトリ	説明
cache	キャッシュ情報を格納するディレクトリ
classes	OJS 2.x のほとんどの PHP コード、すなわち、モデルクラス、データアクセスオブジェクト (DAO)、コアクラスなどを格納するディレクトリ
config.TEMPLATE.inc.php	設定ファイルの例
config.inc.php	システム全体の設定ファイル
dbscripts	XML データベーススキーマとメールテンプレートなどのデータを格納するディレクトリ

ファイル/ディレクトリ	説明
docs	システム文書を格納するディレクトリ
help	システムヘルプの XML 文書を格納するディレクトリ
includes	システム起動用の PHP コード、すなわち、クラスローダや様々なグローバル関数を格納するディレクトリ
index.php	すべてのリクエストはこの PHP スクリプトを通じて処理されます。その役割はシステムにある適切なコードを呼び出すことです。
js	クライアントサイド javascript ファイルを格納するディレクトリ
lib	ADODB(データベースの抽象化)クラスおよび Smarty(テンプレートシステム)クラスを格納するディレクトリ
locale	ロケールデータとキャッシュを格納するディレクトリ
pages	ページクラスを格納するディレクトリ
plugins	プラグインを格納するディレクトリ
public	外部のブラウザに公開するファイルを格納するディレクトリ。例えば、雑誌のロゴはシステムによりここに置かれます。
registry	システムで必要となる様々な XML データ、例えば、定期的に行うタスク、利用可能なロケール名、デフォルトの雑誌設定、コンテンツのインデックス作成の際に除外する単語などを格納するディレクトリ
rt	読書ツールで使用する XML データを格納するディレクトリ
styles	システムが使用する CSS スタイルシートを格納するディレクトリ
templates	すべての Smarty テンプレートを格納するディレクトリ
tools	システムの運用を支援するツール、例えば、未使用のロケールキーファインダ、定期的に行うタスクのラッパー、SQL ジェネレータなどを格納するディレクトリ

## リクエストの処理

外部のブラウザからのリクエストをシステムが処理する方法は、コードを直接調べた場合、少しわかりにくいものになっています。それは、しかるべき PHP クラスを呼び出すことが唯一の役割であるスタブファイルを使用しているからです。例えば、標準的な `index.php` ファイルが多く見られますが、このファイルはそれ自身では実際の処理を何も行いません。

その代わりにしかるべきページクラスに処理を委譲します。このページクラスは、Handler クラスのサブクラスであり、ソースツリーの `pages` ディレクトリに置かれています。

### URL

通常、OJS の URL には `PATH_INFO` 変数を使用します。例えば、次の (仮想的な) URL を考えます。

<http://www.mylibrary.com/ojs2/index.php/myjournal/user/profile>

このリクエストを処理するために呼び出される PHP スクリプトは、URL の中ほどにある `index.php` です。URL 上でこれより後ろにある部分は、`PATH_INFO` と呼ばれる CGI 変数を介して `index.php` に渡されます。

この種のリクエストを正しく処理できないように設定されているサーバが存在します。そのようなサーバがこの種の URL を処理すると多くの場合 404 エラーとなります。これらのリクエストを正常に処理するようにサーバを再設定できない場合は、別の方法で URL を生成するように OJS を設定することができます。 `config.inc.php` ファイルの `disable_path_info` オプションを参照してください。この方法を使った場合、OJS は上の例で示したものと違う URL を生成します。例えば、上の URL は次のようになります。

<http://www.mylibrary.com/ojs2/index.php?journal=myjournal&page=user&op=profile>

## リクエストの処理例

予想されるように、リクエスト処理の委譲は、リクエスト URL に基づいて行われます。雑



誌に対する典型的な URL は次のようなものです。

<http://www.mylibrary.com/ojs2/index.php/myjournal/user/profile>

以下の段落では、システムがこの URL によるリクエストを処理する基本的な方法を説明します。プロセスをより完全に理解するには、ステップごとにソースコードを見て行くと役に立つでしょう。

この例では、<http://www.mylibrary.com/ojs2/index.php> はソースツリーのルート `index.php` ファイルへのパスであり、これがファイル名となります。すべてのリクエストはこの PHP スクリプトを経由します。このファイルの役割は、システムが正しく設定されていることを保証し、しかるべき場所に制御を渡すことです。

`index.php` の後ろには、URL を構成するいくつかの要素があります。最初の 2 つ（この例では、`myjournal` と `user`）の機能はあらかじめ定義されています。それ以外の要素が続いている場合、それらは該当する処理関数への引数となります。

オープン・ジャーナル・システム 2.x は、1 つのシステムで複数の雑誌を運営することができます。`myjournal` はこのリクエストが対象とする雑誌を特定します。特定の雑誌を対象としない状況も存在します。例えば、ユーザがサイト管理ページを見ている場合です。この場合、このフィールドは `index` という値を取ります。

URL 例の次のフィールドは、リクエストを処理するために使用されるページクラスを特定します。この例では、上記 URL によるリクエストを処理するために、システムは `pages/user/index.php` ファイルのロードを試みます。このファイルを見てみると、ページクラス名を識別する定数を定義し（この例では、`Userhandler`）、そのクラスを定義している PHP ファイルをロードしているだけであることがわかります。

いよいよ、最後のフィールド（この例では、`profile`）です。これは、リクエストを処理するために呼ばれるページクラスの関数を特定します。この例では（`pages/user/UserHandler.inc.php` ファイルで定義されている）`User` クラスの `profile` メソッドです。

### リクエスト処理コードの配置場所

リクエストを処理する枠組みが理解できたら、コードの修正や拡張のためにそのタスクを

実行するコードの配置場所を見つけることは極めて容易です。なお、しかるべきクラスに制御を委譲するコードは拡張性を考慮して書かれています。したがって、変更の必要はほとんどないはずで

特定のリクエストを処理するコードを見つけるには、次のステップに従います。

- リクエスト URL からページクラスの名前を見つけます。これは `index.php` の後ろに続く 2 番目のフィールドです。例えば、次の URL では、

<http://www.mylibrary.com/index.php/myjournal/user/profile>

ページクラスの名前は、`UserHandler` です。(ページクラスの名前は常に `Handler` で終わります。また、大文字の使い方に違いがある点にも注意してください。URL では `lowerCamelCase` スタイルが使用されていますが、クラス名は常に `CamelCase` スタイルです。)

- このページクラスのソースコードをソースツリーの `pages` ディレクトリで探します。この例では、ソースコードは `pages/user/UserHandler.inc.php` にあります。
- 調べている URL により呼び出される関数を決定します。これは、`index.php` の後ろに続く 3 番目のフィールドです。この例では、`profile` です。
- したがって、このリクエストを処理するコードは `pages/user/UserHandler.inc.php` ファイルにある `profile` 関数です。

## データベース設計

オープン・ジャーナル・システム 2.x のデータベース設計は、柔軟かつ包括的で一貫性のあるものです。しかし、システムが提供する機能やオプションの数が多いので、その対象範囲はきわめて広いものになっています。

さらに詳しい情報については、[dbscripts/xml/ojs\\_schema.xml](#) を参照してください。

テーブル名	主キー	説明
access_keys	access_key_id	査読者のワンクリックアクセス用のキーを格納
article_authors	author_id	論文ごとに論文著者を格納
article_comments	comment_id	論文の編集プロセスに関係する者のコメントを格納。読者のコメントには使用されないことに注意してください
article_email_log	log_id	特定の論文に関して送信されたメールを記述するログエントリを格納
article_event_log	log_id	特定の論文に関して行われたイベントを記述するログエントリを格納
article_files	file_id, revision	特定の論文に関係する様々なファイル(画像やゲラ、補足ファイルなど)に関する情報を格納
article_galleys	galley_id	特定の論文に関係する特定のレイアウト(「ゲラ」)に関する情報を格納
article_html_galley_images	galley_id, file_id	article_galleys テーブルに格納されたゲラに関する画像
article_notes	note_id	特定の論文を追跡するために編集者により作成された注記を格納
article_search_object_keywords	object_id, pos	キーワードに関するインデックスをその位置で提供。キーワードが含まれている検索オブジェクトも同時に提供。
article_search_objects	object_id	検索「オブジェクト」、すなわち、検索可能な実体をリストアップ
article_search_keyword_list	keyword_id	システムがインデックスを作成したアイテムに現れるすべてのキーワードを格納
article_supplementary_files	Supp_id	特定の論文に属する補足ファイルに関する情報を格納

テーブル名	主キー	説明
articles	article_id	システムに保管されているすべての投稿物の情報を格納
comments	comment_id	論文に関する読者のコメントを格納
copyed_assignments	copyed_id	コピーエディタの任命に関する情報を格納
currencies	Currency_id	購読サブシステムで利用できる通貨に関する情報を格納
custom_section_orders	issue_id, section_id	雑誌セクションの巻号固有の順序付けに関する情報を格納
edit_assignments	Edit_id	編集者の任命に関する情報を格納
edit_decisions	Edit_decision_id	特定の論文に対する編集者の判断を格納
email_templates	Email_id	雑誌管理者により変更されたメールテンプレートのリストを格納
email_templates_data	Email_id, locale, journal_id	雑誌管理者により変更された email_templates におけるロケール固有のメールテキストを格納
email_templates_default	Email_id	OJS 2.x のこのバージョンに添付されたデフォルトのメールテンプレートのリストを格納
email_templates_default_data	Email_id, locale, journal_id	OJS 2.x のこのバージョンに添付された email_templates_default におけるロケール固有のメールテキストを格納
group_memberships	user_id, group_id	グループのメンバ情報を格納
groups	group_id	グループに関する情報(すなわち、カスタムマストヘッド)を格納
issues	issue_id	提供している雑誌の特定の巻号に関する情報を格納
journal_settings	journal_id, setting_name	各雑誌に関する任意設定項目を格納する方法を提供
journals	journal_id	提供する雑誌のリストおよび若干のメタデータを格納(ほとんどのメタデータは journal_settings に格納されます)
layouted_assignments	layouted_id	レイアウトエディタの任命に関する情報を格納
notification_status	journal_id, user_id	ユーザが特定の雑誌に関する新着案内を希望する場合、このテーブルで雑誌 ID と関連付け

テーブル名	主キー	説明
		られます
oai_resumption_tokens	token	OAI プロトコルインターフェースのリサンプショントークンを保持
plugin_settings	plugin_name, journal_id, setting_name	個々のプラグインの設定を格納
proof_assignments	proof_id	校正者の任命に関する情報を格納
published_articles	pub_id	論文が出版される際、articles テーブルの情報を補強するためにこのテーブルにエントリが作成されます
review_assignments	review_id	査読者の任命に関する情報を格納
review_rounds	article_id, round	各査読ラウンドにおける査読ファイル改訂版と論文 ID を関係付けます
roles	journal_id, role_id, user_id	特定の雑誌におけるユーザの役割(管理者、編集者、査読者など)を定義します
rt_contexts	context_id	読書ツールのコンテキスト
rt_searches	search_id	読書ツールの検索式
rt_settings	journal_id	各雑誌における読書ツールの設定
rt_versions	version_id	読書ツールのバージョン
scheduled_tasks	class_name	このテーブルは、定期的タスクをサポートするシステムにおいて、タスク実行スクリプトが最後にタスクを実行した時間を格納するのに使用されます
section_editors	journal_id, section_id, user_id	セクションエディタと担当する雑誌セクションを関係付けます
sections	section_id	論文を出版するための雑誌セクションを定義します
sessions	session_id	現在システムを使用しているユーザのセッション情報を格納
site	title	サイト全体の設定情報を格納
subscription_types	type_id	購読サブシステムで利用可能な購読種別を定義します
subscriptions	subscription_id	システムのユーザが「所有する」購読を定義します
temporary_files	file_id	ユーザのリクエスト間でサーバにファイルを

テーブル名	主キー	説明
		一時的に格納する必要がある場合に使用されます
users	user_id	システムに登録された各ユーザに関する情報を格納
versions	major, minor, revision, build	現在使用している OJS 2.x に関する情報を格納

## クラスの解説

### クラス階層

主要なOJS 2.xオブジェクトに関するすべてのクラスおよびサブクラスを以下に示します。インデントは、クラスの継承関係を示します。例えば、AuthorAction は Action を継承しています。

```
AccessKeyManager
Action
    AuthorAction
    CopyeditorAction
    LayoutEditorAction
    ProofreaderAction
    ReviewerAction
    SectionEditorAction
        EditorAction
ArticleLog
ArticleSearch
ArticleSearchIndex
CacheManager
CommandLineTool
    dbXMLtoSQL
    importExport
    installTool
    migrate
    rebuildSearchIndex
    runScheduledTasks
    upgradeTool
Config
ConfigParser
Core
DAO
    AccessKeyDAO
    ArticleCommentDAO
    ArticleDAO
    ArticleEmailLogDAO
    ArticleEventLogDAO
    ArticleFileDAO
    ArticleGalleyDAO
    ArticleNoteDAO
    ArticleSearchDAO
    AuthSourceDAO
    AuthorDAO
    AuthorSubmissionDAO
    CommentDAO
    CopyAssignmentDAO
    CopyeditorSubmissionDAO
    CountryDAO
```



CurrencyDAO  
EditAssignmentDAO  
EditorSubmissionDAO  
EmailTemplateDAO  
GroupDAO  
GroupMembershipDAO  
IssueDAO  
JournalDAO  
JournalSettingsDAO  
JournalStatisticsDAO  
LayoutAssignmentDAO  
LayoutEditorSubmissionDAO  
NotificationStatusDAO  
OAIDAO  
PluginSettingsDAO  
ProofAssignmentDAO  
ProofreaderSubmissionDAO  
PublishedArticleDAO  
RTDAO  
ReviewAssignmentDAO  
ReviewerSubmissionDAO  
RoleDAO  
ScheduledTaskDAO  
SectionDAO  
SectionEditorSubmissionDAO  
SectionEditorsDAO  
SessionDAO  
SiteDAO  
SubscriptionDAO  
SubscriptionTypeDAO  
SuppFileDAO  
TemporaryFileDAO  
UserDAO  
VersionDAO  
DAORegistry  
DBConnection  
DBDataXMLParser  
DBResultRange  
DataObject  
    AccessKey  
    Article  
        AuthorSubmission  
        CopyeditorSubmission  
        LayoutEditorSubmission  
        ProofreaderSubmission  
        PublishedArticle  
        ReviewerSubmission  
        SectionEditorSubmission  
        EditorSubmission  
    ArticleComment  
    ArticleEmailLogEntry  
    ArticleEventLogEntry  
    ArticleFile  
        ArticleGalley



- ArticleHTMLGalley
  - ArticleNote
  - SuppFile
- AuthSource
- Author
- BaseEmailTemplate
  - EmailTemplate
  - LocaleEmailTemplate
- Comment
- CopyAssignment
- Currency
- EditAssignment
- Group
- GroupMembership
- HelpToc
- HelpTopic
- HelpTopicSection
- Issue
- Journal
- LayoutAssignment
- Mail
  - MailTemplate
  - ArticleMailTemplate
- ProofAssignment
- ReviewAssignment
- Role
- Section
- Site
- Subscription
- SubscriptionType
- TemporaryFile
- User
  - ImportedUser
- Version
- EruditExportDom
- FileManager
  - ArticleFileManager
  - PublicFileManager
  - TemporaryFileManager
- FileWrapper
  - FTPFileWrapper
  - HTTPFileWrapper
- Form
  - ArticleGalleyForm
  - AuthSourceSettingsForm
  - AuthorSubmitForm
    - AuthorSubmitStep1Form
    - AuthorSubmitStep2Form
    - AuthorSubmitStep3Form
    - AuthorSubmitStep4Form
    - AuthorSubmitStep5Form
  - AuthorSubmitSuppFileForm
  - ChangePasswordForm
  - CommentForm



- CopyeditCommentForm
- EditorDecisionCommentForm
- LayoutCommentForm
- PeerReviewCommentForm
- ProofreadCommentForm
- CommentForm
  - CopyeditCommentForm
  - EditorDecisionCommentForm
  - LayoutCommentForm
  - PeerReviewCommentForm
  - ProofreadCommentForm
- ContextForm
- CreateReviewerForm
- EditCommentForm
- EmailTemplateForm
- GoogleScholarSettingsForm
- GroupForm
- ImportOJS1Form
- InstallForm
- IssueForm
- JournalSetupForm
  - JournalSetupStep1Form
  - JournalSetupStep2Form
  - JournalSetupStep3Form
  - JournalSetupStep4Form
  - JournalSetupStep5Form
- JournalSiteSettingsForm
- LanguageSettingsForm
- MetadataForm
- ProfileForm
- RegistrationForm
- SearchForm
- SectionForm
- SiteSettingsForm
- SubscriptionForm
- SubscriptionTypeForm
- SuppFileForm
- UpgradeForm
- UserManagementForm
- VersionForm
- FormError
- FormValidator
  - FormValidatorArray
  - FormValidatorCustom
  - FormValidatorInSet
  - FormValidatorLength
  - FormValidatorRegExp
    - FormValidatorAlphaNum
    - FormValidatorEmail
- GenericCache
  - FileCache
  - MemcacheCache
- Handler
  - AboutHandler

- AdminHandler
  - AdminFunctionsHandler
  - AdminJournalHandler
  - AdminLanguagesHandler
  - AdminSettingsHandler
  - AuthSourcesHandler
- ArticleHandler
  - RTHandler
- AuthorHandler
  - SubmissionCommentsHandler
  - SubmitHandler
  - TrackSubmissionHandler
- CommentHandler
- CopyeditorHandler
  - SubmissionCommentsHandler
  - SubmissionCopyeditHandler
- GatewayHandler
- HelpHandler
- IndexHandler
- InformationHandler
- InstallHandler
- IssueHandler
- LayoutEditorHandler
  - SubmissionCommentsHandler
  - SubmissionLayoutHandler
- LoginHandler
- ManagerHandler
  - EmailHandler
  - FilesHandler
  - GroupHandler
  - ImportExportHandler
  - JournalLanguagesHandler
  - PeopleHandler
  - PluginHandler
  - SectionHandler
  - SetupHandler
  - StatisticsHandler
  - SubscriptionHandler
- OAIHandler
- ProofreaderHandler
  - SubmissionCommentsHandler
  - SubmissionProofreadHandler
- RTAdminHandler
  - RTContextHandler
  - RTSearchHandler
  - RTSetupHandler
  - RTVersionHandler
- ReviewerHandler
  - SubmissionCommentsHandler
  - SubmissionReviewHandler
- SearchHandler
- SectionEditorHandler
  - EditorHandler
  - IssueManagementHandler



- SubmissionCommentsHandler
- SubmissionEditHandler
- SubscriptionManagerHandler
- UserHandler
  - EmailHandler
  - ProfileHandler
  - RegistrationHandler
- Help
- HookRegistry
- ImportOJS1
- Installer
  - Install
  - Upgrade
- IssueAction
- ItemIterator
  - ArrayItemIterator
  - DAOResultFactory
  - DBRowIterator
  - JournalReportIterator
  - VirtualArrayIterator
- Locale
- NativeExportDom
- NativeImportDom
- OAI
  - JournalOAI
- OAIConfig
- OAIIdentifier
  - OAIRecord
- OAIMetadataFormat
  - OAIMetadataFormat\_DC
  - OAIMetadataFormat\_MARC
  - OAIMetadataFormat\_MARC21
  - OAIMetadataFormat\_RFC1807
- OAIRepository
- OAIResumptionToken
- OAISet
- Plugin
  - AuthPlugin
    - LDAPAuthPlugin
  - GatewayPlugin
    - GoogleScholarPlugin
  - GenericPlugin
  - ImportExportPlugin
    - EruditExportPlugin
    - NativeImportExportPlugin
    - SampleImportExportPlugin
    - UserImportExportPlugin
- PluginRegistry
- RT
  - JournalRT
- RTAdmin
  - JournalRTAdmin
- RTContext
- RTSearch



```
RTVersion
RTXMLParser
Registry
Request
SMTPMailer
SQLParser
ScheduledTask
    ReviewReminder
SearchFileParser
    SearchHTMLParser
    SearchHelperParser
SessionManager
String
TemplateManager
Transcoder
UserExportDom
UserXMLParser
Validation

VersionCheck
XMLCustomWriter
XMLDAO
    HelpTocDAO
    HelpTopicDAO
XMLNode
XMLParser
XMLParserHandler
    XMLParserDOMHandler
```

## ページクラス

### はじめに

ページクラスは、ユーザの Web ブラウザからリクエストを受け取り、様々なクラスに委譲して必要な処理をさせ、(必要であれば) しかるべき Smarty テンプレートを呼び出してレスポンスを作成します。すべてのページクラスは pages ディレクトリにあり、各クラスは Handler クラスを継承したものでなければなりません ( `classes/core/Handler.inc.php` を参照 )。

さらに、ページクラスはユーザのリクエストが妥当であり、認証要件を満たしていることを保証することも担当します。ユーザが入力したフォームパラメタや URL パラメタは、パラメタを処理するための Form クラスが別に用意されていない限りは、できるだけページクラスで処理されるべきです。

ページクラスが実行しなければならないタスクを理解する簡単な方法は、典型的な例を調べてみることです。 `pages/about/AboutHandler.inc.php` ファイルには、<http://www.mylibrary.com/ojs2/myjournal/about/siteMap> のようなリクエストを処理する `AboutHandler` クラスを実装するコードが含まれています。これは、閲覧される雑誌やサイトに関する様々なメタデータを取り出して表示することを担当しているきわめて簡単なページクラスです。

各ページクラスは、リクエスト URL にしかるべきページクラスや関数を指定することによりユーザが呼び出すことができる数多くの関数を実装しています (URL とページクラスのマッピングに関する詳しい情報は「リクエストの処理」の節を参照してください)。

通常、ページクラスはユーザが担う役割に基づいてリクエストを処理します。例えば、`AuthorHandler` という名のページクラス (`pages/author/AuthorHandler.inc.php`) があり、これは著者が実行することになる様々なタスクの処理をサブクラスに委譲して処理します。同様に、`LayoutEditorHandler` や `ManagerHandler` という名のページクラスがあります。

ページハンドラが実行しなければならないタスクの数は、しばしば大きなものになる場合があります。例えば、セクションエディタ関数へのすべてのリクエストが直接 `SectionEditorHandler` クラスで処理されるとしたら、このクラスは極めて大きなものになり、保守が難しくなるでしょう。実際は、関数はさらにいくつかのクラスに細分化され (`SubmissionEditHandler` や `SubmissionCommentsHandler` など)、`SectionEditorHandler` 自体には特定のサブクラスを呼び出す関数だけが残されています。

## アクションクラス

アクションクラスはページクラスにより使用され、ユーザリクエストの実際の処理を行います。例えば、`SectionEditorAction` クラスは `SectionEditorHandler` クラス、またはそのサブクラス (「ページクラス」の節を参照) により呼び出され、容易に任せることができるほとんどの作業を処理します。このクラスはページクラスの仕事 (ユーザリクエストの妥当性チェック、認証、テンプレートの設定) はページクラスに残し、実際の処理とは独立させています。

アクションクラスは、`classes/submission/[actionName]/[ActionName]Action.inc.php` で見つけることができます。例えば、セクションエディタのアクションクラスは、

classes/submission/sectionEditor/SectionEditorAction.inc.php です。

アクションクラスが行うことになる最も一般的なタスクは、メールの送信と（モデルクラスと DAO クラスを介した）データベースレコードの変更、（さらにしかるべきクラスを介した）アップロードファイルの処理です。モデル/ビュー/コントローラ（MVC）アーキテクチャに戻ると、アクションクラスはコントローラ要素のうち、インターフェースに関係しない機能を実行します。

著者やセクションエディタ、校正者など、複雑な役割については、独自のアクションクラスを持っています。アクションクラスの機能を検討するためのもう一つの方法は、ユーザインターフェースを無視して、役割という観点から見てみることです。例えば、著者が実行できるようにする必要のある主要な処理はすべて AuthorAction クラスで実装されているはずで、そして、必要に応じて、ユーザインターフェースがこれらの関数を呼び出しています。

## モデルクラス

モデルクラスは、データベースエンティティをメモリ上で表現することだけを担当する PHP クラスです。例えば、articles テーブルは論文情報をデータベースに格納しますが、そこには、対応する Article という名のモデルクラス（classes/article/Article.inc.php）と ArticleDAO という名の DAO クラス（「データアクセスオブジェクト（DAO）」の節を参照）が存在します。

モデルクラスが提供するメソッドはほとんどの場合、Article クラスの getTitle（）メソッドと setTitle（\$title）メソッドのように、情報の取り出しと格納を行う get/set メソッドだけです。モデルクラスはデータベースへの格納や更新は担当しません。これは関連する DAO クラスにより実行されます。

すべてのモデルクラスは、DataObject クラスを継承しています。

## データアクセスオブジェクト（DAO）

データアクセスオブジェクトは、データベースからデータをモデルクラスの形で抽出する、変更されたモデルクラスでデータベースを更新する、データベースからデータを削除するために使用されます。

各モデルクラスは関連するデータアクセスオブジェクトを持ちます。例えば、Article クラス (classes/article/Article.inc.php) は、ArticleDAO (classes/article/ArticleDAO.inc.php) という名の関連する DAO を持っており、これがモデルクラスとそのデータベースエンティティの間の相互変換を実装しています。

すべての DAO は DAO クラス (classes/db/DAO.inc.php) を継承しています。PHP とバックエンドデータベースとの通信機能はすべて DAO クラスに実装されています。できるだけ論理的かつ効率的であるために、各 DAO は相互作用を本来対象とするテーブルに限定するべきです。

DAO が使用される場合、直接インスタンス化されることは決してありません。代わりに、システムが使用する DAO のインスタンスを管理している DAORegistry クラスを使って、名前を指定して取り出します。例えば、論文用の DAO を取り出すには次のようにします。

```
$articleDao = &DAORegistry::getDAO('ArticleDAO');
```

次に、これを使って、\$articleId に格納されている ID を持つ論文を取り出します。

```
$article = &$articleDao->getArticle($articleId);
```

結果集合を取り出す DAO メソッドの多くは、通常の PHP 配列ではなく、ItemIterator クラスのサブクラスを返すことに注意してください。これは多くのアイテムを含むリストのページングを容易にし、すべての結果を配列に入れて事前にロードするより効率を良くすることができます。「サポートクラス」の節のページングクラスの説明を参照してください。

## サポートクラス

### メール送信

```
classes/mail/Mail.inc.php  
classes/mail/MailTemplate.inc.php  
classes/mail/ArticleMailTemplate.inc.php
```

これらのクラスは、EmailTemplate と MailTemplate の両モデルクラス、および EmailTemplateDAO DAO クラスと共に、システムで使用されるすべてのメール機能を提



供します。

Mail.inc.php は、メールの組み立て、アドレス指定、送信という基本的な機能を提供します。MailTemplate クラスはこのクラスを継承し、テンプレートを使ったメール作成機能を追加しています。ArticleMailTemplate クラスは、論文単位で閲覧可能なメールログ出力など、特定の論文に関するメールに便利な機能を追加しています。

典型的な使用例や実行コードについては、様々なアクションクラス、例えば、SectionEditorAction クラスの notifyReviewer メソッドを参照してください。システムにより作成されるほとんどすべてのメールはユーザに表示され、その後、ブラウザを通じて数回のリクエストとレスポンスのやり取りを行う間にメールを変更することができなければなりませんので、リクエスト間でシステムの状態を保持するために、これらのクラスが若干複雑になるのは避けられないことに注意してください。

## 国際化

システムの国際化は、OJS 2.x の重要な機能です。OJS 2.x は、使用される言語に関する前提条件を一切設けることなく設計されています。

各言語による表示用の XML 文書が、locale ディレクトリの下のロケール名を名前とするサブディレクトリに置かれています。例えば、en\_US ロケールの情報は、locale/en\_US/locale.xml ファイルに置かれています。

このファイルはユーザインターフェースで使用される多くの文字列を含んでいます（ほとんどすべては Smarty テンプレートで直接使用されますが、中には、ページクラスなどでコードされている文字列もあります）。

これらの文字列は、`{translate key="[keyName]"}` ディレクティブを持つ Smarty テンプレートから呼び出されます（詳細は「ユーザインターフェース」の節を参照）。変数置換がサポートされています。

システムで使用するロケールは、サイト設定の言語ページで設定、インストール、管理されます。利用可能なロケールリストは、registry/locales.xml レジストリファイルから作成されます。

言語毎の locale.xml ファイルの他に、dbscripts/xml/data/locale ディレクトリ

と `registry/locale` ディレクトリのサブディレクトリにもロケール固有のデータを見つけることができます。ここでもロケール名がサブディレクトリ名になっています。例えば、XML ファイルである `dbscripts/xml/data/locale/en_US/email_templates_data.xml` には、`en_US` (米語) ロケールによるすべてのメールテンプレート文が含まれています。

すべての XML データは UTF-8 エンコーディングを使用しています。バックエンドデータベースが特殊文字を正しく扱うよう設定されていれば、データは入力した通りに格納され、表示されます。

OJS 2.x は、限定的ではありますが、1つの雑誌に複数のロケールを使用する機能をサポートしています。例えば、論文は主となるロケールを持っていますが、タイトルと要旨には最高2つのロケールを追加して持つことができます。

国際化関数は、`classes/i18n/Locale.inc.php` で提供されています。テンプレートを使ったロケール翻訳関数の実装については、`classes/template/TemplateManager.inc.php` (ユーザインターフェースのサポートクラスの一部) も参照してください。

## フォーム

データの検証やエラー処理などフォーム処理に関係する共通タスクの実装は、`Form` クラス (`classes/form/Form.inc.php`) とその様々なサブクラス (例えば、雑誌管理者がセクションを変更する際に使用する `classes/manager/form/SectionForm.inc.php` など) にまとめられています。

`Form` クラスのサブクラスは、コンストラクタ、および、`initData`, `display`, `readInputData`, `execute` の各メソッドをオーバーライドして、実装すべき個別のフォームを定義します。各関数の役割は次のとおりです。

- クラスコンストラクタ: このフォーム独自の変数を初期化します。これは例えば、特定の論文に関連するフォームなどに便利です。この場合、`Article` オブジェクト、あるいは論文 ID をコンストラクタの引数として設定し、メンバ変数として保持することができます。
- `initData`: フォームクラスが表示できるように、フォームが表示される前に、現在値あるいは (もしあれば) デフォルト値を (メンバ変数である) `_data` 配列にロードしなければなりません。

- `display`: 追加情報を表示するために、フォームが表示される直前にフォームの Smarty テンプレートに追加の引数を割り当てることができるのと便利でしょう。そのような割り当てを実行するには、このメソッドをオーバーライドします。
- `readInputData`: このフォームで使用するフォームパラメータを親クラスに知らせるために、このメソッドをオーバーライドします。さらに、データ検証のようなタスクをここで実行することができます。
- `execute`: このメソッドは、フォームのデータが「コミット」される際に呼び出されます。このメソッドは、例えば、既存のデータベースレコードの更新や(しかるべきモデルクラスと DAO クラスを介した)新規レコードの挿入を担当します。

様々なフォームクラスを理解する最高の方法は、上の例で示した `SectionForm` クラス (`classes/manager/form/SectionForm.inc.php` で実装)などの典型的な例を調べてみることです。もっと複雑な例については、(`classes/manager/form/setup` ディレクトリにある) 雑誌管理者が行うさまざまな設定用のフォームを参照してください。

ブラウザとシステムとのフォームを介したすべての相互作用を `Form` クラスとそのサブクラスを使って実行するのは不便であり、理にかなっていません。一般的に言えば、このアプローチが有効であるのは、1 ページが 1 データベースレコードに密接に対応している場合だけです。例えば、`SectionForm` クラスで定義されているページは、`sections` テーブルのレイアウトと密接に対応しています。

## 設定

OJS 2.x では、雑誌の設定が `journal_settings` テーブルに格納されるように、ほとんどの設定がデータベースに格納され、しかるべき DAO クラスとモデルクラスを介してアクセスされます。しかし、ある種のシステム全般の設定は、`config.inc.php` という名のテキストファイルに格納されます(このファイルは実際には PHP スクリプトではありませんが、外部のブラウザに公開されないことがないようにこのように命名されています)。

この設定ファイルは、`ConfigParser` クラス (`classes/config/ConfigParser.inc.php`) により解析され、`Config` クラス (`classes/config/Config.inc.php`) のインスタンスに格納されます。

## コアクラス

(`classes/core` ディレクトリにある) コアクラスは、基本的で重要な関数を提供するも

ので、OJS 2.x のほとんどの機能はそのうちの少数のクラスに基づいています。これらのクラスは、それ自体はシンプルなものですが、これを継承したクラスを通じて提供される柔軟性を持っています。

- `Core.inc.php`: システム全般に関わる様々な関数を提供します
- `DataObject.inc.php`: すべてのモデルクラスの親クラスです
- `Handler.inc.php`: すべてのページクラスの親クラスです
- `Registry.inc.php`: システム起動時間などのグローバル値の格納・取出といったシステム全般に関わる関数を提供します
- `Request.inc.php`: HTTP リクエストのラッパーと関連の共通使用関数を提供します
- `String.inc.php`: ロケール独立な文字列操作関数と関連の共通使用関数を提供します

特に、`Request` クラス (`classes/core/Request.inc.php` で定義) は、リモートユーザに関する情報を入手したり、レスポンスを構築したりするための数多くの関数を含んでいます。OJS により生成される OJS 自身へのリンクのすべての URL は、`Request::url` 関数を使って作成されます。同様に、OJS へのリダイレクトはすべて `Request::redirect` 関数を使って作成されます。

## データベースサポート

基本的なデータベース機能は ADODB ライブラリ (<http://adodb.sourceforge.net>) により提供されます。ADODB ライブラリの上に、データアクセスオブジェクト (DAO) により提供される抽象化層が追加されています。これらは `classes/db` ディレクトリにある少数の基本クラスを使用し、これを継承することにより個別の機能を提供しています。

- `DAORegistry.inc.php`: データアクセスオブジェクトの中央レジストリを実装しています。DAO が必要となった場合、DAO レジストリを介して取り出されます。
- `DBConnection.inc.php`: すべてのデータベースコネクションは、このクラスを介して確立されます。
- `DAO.inc.php`: すべての DAO が継承する基本クラスを提供します。これは、`DBConnection` クラスを介してデータベースにアクセスする関数を提供しています。

さらに、XML の解析やデータベースへのロードを支援するいくつかのクラスが存在します。

- XMLDAO.inc.php: XML データソースからオブジェクトを抽出したり、変更したりする操作を提供します。
- DBDataXMLParser.inc.php: XML スキーマを解析して、SQL 文に変換します。

## ファイル管理

(ゲラや雑誌ロゴなどの) ファイルは、データベースではなくサーバのファイルシステムに格納されます。したがって、ファイルシステムやファイルシステムと OJS の相互作用を管理するためのクラスが必要になります。これらのクラスは、classes/file ディレクトリで見つけることができます。

- FileManager.inc.php: 以下の 3 つのファイル管理クラスはこのクラスを継承したものです。Web サーバとファイルシステムの間で相互作用を行うために必要な基本的な機能を提供します。
- FileWrapper.inc.php: 内蔵のアクセスメソッドよりも互換性の高いファイルアクセス関数を持つラッパークラスを実装しています。
- ArticleFileManager.inc.php: FileManager クラスを継承して、特定の論文に関連するファイルを管理するために必要な機能を追加しています。例えば、論文ファイルに関係するディレクトリ構造の管理を担当しています。ArticleFile と ArticleFileDAO も参照してください。
- PublicFileManager.inc.php: 雑誌ロゴなど多くのファイルは、認証を必要とせず誰でもアクセスができるという意味で「パブリック」なものです。これらのファイルは、FileManager クラスを継承したこのクラスで管理されます。
- TemporaryFileManager.inc.php: このクラスは、特定のユーザに関連する一時的ファイルを格納し、リクエストをまたいで保管することを可能にします。例えば、ユーザが添付ファイルを持つメールを作成している場合、添付ファイルはメールの作成が終わるまでサーバに保管する必要があります。そして、これが複数のリクエストに関係する場合があるからです。TemporaryFileManager クラスも FileManager クラスを継承しています。TemporaryFile と TemporaryFileDAO も参照してください。

## 定期的タスク

OJS 2.x はオペレーティングシステムの助けを借りることにより、定期的タスクを自動的

に実行することができます。OS は、UNIX の cron に当たる機能を使って、tools/runScheduledTasks.php スクリプトを実行します。定期的タスクは、config.inc.php 設定ファイルと雑誌設定で有効にする必要があります。

自動タスクは、registry/scheduledTasks.xml で設定し、タスクの最終実行日などの情報は scheduled\_tasks テーブルに格納されます。

ScheduledTask モデルクラスと関連の ScheduledTaskDAO クラスがこれらのデータベースエントリの管理を担当します。さらに、定期的タスク自体は、classes/tasks ディレクトリに置きます。現在のところ、ReviewReminder タスクのみが実装されています。これは、査読を終了していない、または任命依頼の受諾を行っていない査読者に督促メールを送信するものです。

ScheduledTask モデルクラスを継承し、runScheduledTasks ツールにより実行されるこれらのタスクは、タスクを処理する execute( )メソッドを実装しなければなりません。

## セキュリティ

OJS 2.x のセキュリティモデルは、役割概念に基づいています。システム上の役割はあらかじめ定義されており（例えば、著者、読者、セクションエディタ、校正者など）、ユーザは雑誌ごとに役割を与えられます。1 人のユーザが同一の雑誌で複数の役割を持つことができます。

役割は、roles テーブルの管理とセキュリティチェック機能を提供する Role モデルクラスと関連の RoleDAO クラスを介して管理されます。

Validation クラス (classes/security/Validation.inc.php) は、ユーザのブラウザと Web サーバの間の相互作用におけるセキュリティを担当します。このクラスは、ログイン・ログアウトリクエストを処理し、パスワードハッシュを生成し、セキュリティや検証に関する問題を処理する数多くの便利なショートカット関数を提供します。Validation クラスはこれらの機能にアクセスする望ましい方法です。

## セッション管理

セッション管理は、Session モデルクラス、SessionDAO クラス、SessionManager クラス (classes/session/SessionManager.inc.php) の 3 者により提供されます。

Session クラスと SessionDAO クラスは、個々のユーザのためのデータベース永続化セッションを管理し、SessionManager クラスは、PHP と Apache 用に実装されたセッションの技術的仕様に relationship します。

## テンプレートサポート

Smarty テンプレート (<http://smarty.php.net>) は TemplateManager クラス (classes/template/TemplateManager.inc.php) を介してアクセス・管理されます。このクラスは、地域化に使用される {translate...} のような Smarty 関数の追加登録や URL や日付フォーマットなどの共通に使用されるテンプレート変数の設定など、数多くの共通タスクを処理します。

## ページングクラス

以下のクラスは、投稿物などのアイテムリストのページ表示を容易にします。

```
ItemIterator  
ArrayItemIterator  
DAOResultFactory  
DBRowIterator  
VirtualArrayIterator
```

ItemIterator クラスはイテレータの抽象クラスであり、個々の実装は別のクラスで提供されます。ItemIterator のサブクラスを返すすべての DAO クラスは ItemIterator を返すものとして扱われるべきです。

各イテレータは結果の 1「ページ」を表します。例えば、SectionEditorSubmissionDAO により投稿物リストを取り出す場合、必要な行数の範囲を指定することができ、返された ItemIterator (厳密に言えば ArrayIterator) はこの範囲の情報を含んでいます。

ArrayItemIterator と VirtualArrayIterator は PHP の配列を使ったイテレータを提供します。VirtualArrayIterator の場合、必要とするページのエントリのみを提供する必要があり、ArrayItemIterator は結果全体を引数として取り、カレントページのエントリのみイテレートします。

最も普通に使用され好まれている ItemIterator のサブクラスである

DAOResultFactory は、指定された DAO とインスタンス化メソッドを使って結果に対応するモデルオブジェクトをインスタンス化する処理を行います。

DBRowIterator は ADODB 結果構造体をラップした ItemIterator です

## プラグイン

プラグインレジストリのサポートを支援するいくつかのクラスが OJS 2.x には含まれています。プラグインレジストリに関しては、「プラグイン」の節を参照してください。

## 共通タスク

以下の節ではサンプルコードを示し、様々なクラスがどのように相互作用するかをさらに詳しく説明します

## メール送信

各ロケールのメールテンプレートは、dbscripts/xml/data/locale/[localeName]/email\_templates\_data.xml と名づけられた XML ファイルに格納されています。各メールテンプレートは、SUBMISSION\_ACK といった識別子 (XML ファイルでは email\_key と呼ばれている) を持っています。PHP コードは、この識別子を使って、本文と件名を含む特定のメールテンプレートを取り出します。

以下のコードは、SUBMISSION\_ACK メールを取り出して送信するものです。このメールは、投稿を終えた著者に感謝の意を伝えるために送信されるものです。(このコード例では、対象となる論文の ID が \$articleId に格納されていることを仮定しています。)

```
// article DAO を使って論文オブジェクトを取り出す
$articleDao = &DAORegistry::getDAO('ArticleDAO');
$article = &$articleDao->getArticle($articleId);

// 必要な ArticleMailTemplate クラスをロードする
import('mail ArticleMailTemplate');

// 名前を指定してメールテンプレートを取り出す
$mail = &new ArticleMailTemplate($article, 'SUBMISSION_ACK');
```

```

if ( $mail->isEnabled ( ) ) {
    // カレントユーザオブジェクトを取り出し、メールの受信者に設定する
    $user = &Request::getUser ( ) ;
    $mail->addRecipient ( $user->getEmail ( ) , $user->getFullName ( ) ) ;

    // カレント雑誌オブジェクトを取り出す
    $journal = &Request::getJournal ( ) ;

    // このテンプレートは、 { $variableName } という形の変数名を含んでおり、
    // この変数は該当する値で置き換えられる。Smarty テンプレートで使用
    // されている構文と似ているが、メールテンプレートは Smarty テンプレ
    // ートではないことに注意。直接的な変数置換のみサポートされている
    $mail->assignParams ( array (
        'authorName' => $user->getFullName ( ) ,
        'authorUsername' => $user->getUsername ( ) ,
        'editorialContactSignature' =>
            journal->getSetting ( 'contactName' ) ."¥n".
            $journal->getTitle ( ) ,
        'submissionUrl' => Request::getPageUrl ( ) .
            '/author/submission/' . $article->getArticleId ( )
    ) ) ;

    $mail->send ( ) ;
}

```

## DAO によるデータベースとの相互作用

以下のコード例は、変数 \$articleId で指定された論文 ID を使って論文オブジェクトを取り出し、タイトルを変更し、新しい値でデータベースを更新するものです。

```

// article DAOA を使って論文オブジェクトを取り出す
$articleDao = &DAORegistry::getDAO ( 'ArticleDAO' ) ;
$article = &$articleDao->getArticle ( $articleId ) ;

$article->setTitle ( 'This is the new article title.' ) ;

// 変更した情報でデータベースを更新する
$articleDao->updateArticle ( $article ) ;

```

同様に、次のコードは論文をデータベースから削除します。

```

// article DAOA を使って論文オブジェクトを取り出す
$articleDao = &DAORegistry::getDAO ( 'ArticleDAO' ) ;
$article = &$articleDao->getArticle ( $articleId ) ;

// データベースから論文を削除する
$articleDao->deleteArticle ( $article ) ;

```

この処理は、次のようにもっと効率的に行うことができます。

```
// article DAOA を使って論文オブジェクトを取り出す
$articleDao = &DAORegistry::getDAO('ArticleDAO');
$articleDao->deleteArticleById($articleId);
```

一般的に言えば、DAO は従属するデータベースエントリの削除も担当します。例えば、論文を削除すると、その論文の著者もデータベースから削除します。データベースに対する要件をできるだけ低いものにするために、これは PHP コードで処理されており、データベーストリガーやその他のデータベースレベルで整合性を保つ機能は使用していないことに注意してください。

## ユーザインターフェース

ユーザインターフェースは、様々なページクラスから呼び出される大規模な Smarty テンプレートとして実装されています（「リクエストの処理」の節を参照）。

これらのテンプレートは各ページの HTML マークアップを担当します。テンプレートで 사용되는すべての内容は、テンプレート変数（論文タイトルなど）またはカスタム Smarty 関数による各言語の翻訳を通じて提供されます。

OJS 2.x のテンプレートを扱う前に Smarty テンプレートをよく理解しておく必要があります。Smarty に関するドキュメントは、<http://smarty.php.net> で入手することができます。

## 変数

テンプレート変数は、通常、そのテンプレートを呼び出す Page クラスまたは Form クラスで割り当てられます。しかし、それに加えて、多くの変数が TemplateManager クラスで割り当てられており、すべてのテンプレートで使用できます。

- defaultCharset: config.inc.php 設定ファイルの [i18n] セクションで設定されている client\_charset パラメタの値
- currentLocale: カレントロケールのシンボル名
- baseUrl: サイトのベース URL。例えば、http://www.mylibrary.com
- requestedPage: リクエストされたページのシンボル名

- pageTitle: ロケールキーで設定されたページタイトルのデフォルト名。これはテンプレートによる設定でより適切なものに置き換えられるべきです
- siteTitle: 現在、ユーザが雑誌に関係するページを見ている場合、これは雑誌タイトルです。それ以外の場合は、サイト設定で設定したサイトタイトルです
- publicFilesDir: 現在使用している Public Files ディレクトリの URL (「ファイル管理」の節を参照)
- pagePath: リクエストされたページと (該当する場合は) 処理のパス。先頭にスラッシュが付く。例えば、 /user/profile
- currentUrl: カレントページの完全 URL
- dateFormatTrunc: config.inc.php 設定ファイルの [general] セクションで設定されている date\_format\_trunc パラメタの値。Smarty 関数 date\_format で使用されます
- dateFormatShort: config.inc.php 設定ファイルの [general] セクションで設定されている date\_format\_short パラメタの値。Smarty 関数 date\_format で使用されます
- dateFormatLong: config.inc.php 設定ファイルの [general] セクションで設定されている date\_format\_long パラメタの値。Smarty 関数 date\_format で使用されます
- datetimeFormatShort: config.inc.php 設定ファイルの [general] セクションで設定されている datetime\_format\_short パラメタの値。Smarty 関数 date\_format で使用されます
- datetimeFormatLong: config.inc.php 設定ファイルの [general] セクションで設定されている datetime\_format\_long パラメタの値。Smarty 関数 date\_format で使用されます
- currentLocale: 現在使用しているロケール名。例えば、 en\_US
- articleSearchByOptions: サイドバーや検索ページにある検索機能で使われる検索可能なフィールド名
- userSession: カレントセッションオブジェクト
- isUserLoggedIn: ユーザがログインしているか否かを示す論理値
- loggedInUsername: 該当する場合は、カレントユーザのユーザ名
- page\_links: カレントジャーナルまたはサイトコンテキストのページリストに表示するページリンクの最大数
- items\_per\_page: カレントジャーナルまたはサイトコンテキストのページリストの各ページに表示するアイテムの最大数

さらに、ユーザが特定の雑誌に属するページを見ている場合は、次の変数も利用できます。

- `currentJournal`: 現在使用している (Journal クラスの) 雑誌オブジェクト
- `alternateLocale1`: 雑誌設定の第一代替ロケール (`alternateLocale1`)
- `alternateLocale2`: 雑誌設定の第二代替ロケール (`alternateLocale2`)
- `navMenuItems`: 雑誌設定のナビゲーションアイテム (`navItems`)
- `pageHeaderTitle`: 雑誌固有の情報を表示するために、`templates/common/header.tpl` で使用されます
- `pageHeaderLogo`: 雑誌固有の情報を表示するために、`templates/common/header.tpl` で使用されます
- `alternatePageHeader`: 雑誌固有の情報を表示するために、`templates/common/header.tpl` で使用されます
- `metaSearchDescription`: カレントジャーナルの説明。meta タグで使用されます
- `metaSearchKeywords`: カレントジャーナルのキーワード。meta タグで使用されます
- `metaCustomHeaders`: 定義されている場合、カレントジャーナルのカスタムヘッダ。meta タグで使用されます
- `stylesheets`: テンプレートに含めるスタイルシートの配列
- `pageFooter`: ページの最後に表示するカスタムフッタの内容

多言語機能が有効になると、次の変数がセットされます。

- `enableLanguageToggle`: この機能が有効な場合、`true` がセットされます
- `languageToggleLocales`: 選択されたロケールの配列

## 関数と修正子

国際化などの共通タスクを支援するために Smarty に内蔵されたテンプレート関数に多くの関数が追加されています。

- `translate` (例えば、`{translate key="my.locale.key" myVar="value"}`): この関数はロケールによる翻訳を提供します (「地域化」の節を参照)。Smarty スタイルの構文を使って変数置換を行うことができます。上の例を使うと、`locale.xml` ファイルが、

```
<message key="my.locale.key">myVar equals "{$myVar}" .</message>
```

を含んでいる場合、出力は次のようになります。

```
myVar equals "value".
```

(ロケールファイルで可能なのは直接変数置換だけであることに注意してください。オブジェクトのメソッドや Smarty 関数を呼び出すことはできません。)

- `assign` (例えば、`{translate|assign:"myVar" key="my.locale.key"}`) : テンプレート変数に値を設定します。この例は、結果がブラウザに表示されるのではなく、指定された Smarty 変数に設定されることを除いて、`{translate....}`と同じです。
- `html_options_translate` (例えば、`{html_options_translate values=$myValuesArray selected=$selectedOption}`) : `array('optionVal1' => 'locale.key.option1', 'optionVal2' => 'locale.key.option2Convert)` のような形の配列を、次のように HTML の `<option>...</option>` タグの組に変換し、Select メニューで使えるようにします。

```
<option value="optionVal1">Translation of "locale key option1" here</option>
```

```
<option value="optionVal2">Translation of "locale key option2" here</option>
```

- `get_help_id` (例えば、`{get_help_id key="myHelpTopic" url="true"}`) : (`url` パラメタの値により) 指定した名前のヘルプページのトピック ID または完全 URL を表示します
- `icon` (例えば、`{icon name="mail" alt="..." url="http://link.url.com" disabled="true"}`) : 指定されたリンク URL を持つアイコンを表示し、指定により有効にしたり、無効にしたりします。`name` パラメタには、`comment`, `delete`, `edit`, `letter`, `mail`, `view` のいずれかの値を指定します。
- `help_topic` (例えば、`{help_topic key="(dir)*.page.topic" text="foo"}`) : 指定したヘルプトピックへのリンクを表示します。`text` パラメタにはリンク内容を指定します。
- `page_links` : (例えば、`{page_links iterator=$submissions}`) : `ItemIterator` のサブクラス (この例では、`$submissions`) に関するページ付けされたリストのページリンクを表示します。
- `page_info` : (例えば、`{page_info name="submissions" iterator=$submissions}`) : `ItemIterator` のサブクラス (この例では、`$submissions`) に関するページ付けされたリストのページ情報 (例えば、ページや総ページ数) を表示します。
- `iterate` : (例えば、`{iterate from=submissions item=submission}`) : 指定された `ItemIterator` のサブクラスのアイテムをイテレートします。その際、各アイテムは指定された名前を持つ Smarty 変数にセットされます。(この例では、`$submissions` イテレータのアイテムをイテレートします。その際、各アイテムは `$submission` というテンプレート変数にセットされます。) 変数名の頭には

\$記号が付かないことに注意してください。指定するパラメタは変数名であり、変数それ自体ではないからです。

- `strip_unsafe_html`: (例えば、`{ $myVar | strip_unsafe_html }`): 一般的な利用において「安全でない」と判断された HTML タグや属性を削除します。この修正子はある種の単純な HTML タグはリモートブラウザへそのまま渡しますが、XSS(クロスサイトスクリプティング) 攻撃に使用される恐れのある高度なものはすべてサニタイズします。
- `call_hook`: (例えば、`{ call_hook name="Templates::Manager::Index::ManagementPages" }`): 名前を指定してプラグインフックを呼び出します。指定したフックに登録されたすべてのプラグインが呼び出されます。

これらの関数を使った多くの例が OJS 2.x で提供されているテンプレートに存在します。

## プラグイン

OJS 2.1 は、本体コードを変更することなくシステムの振る舞いを拡張したり、変更したりするためのメカニズムを開発者に提供する本格的なプラグイン機構を持っています。この仕組みの鍵となる概念は、**カテゴリ**、**プラグイン**、**フック**です。

**プラグイン**は、OJS の拡張や変更を実装するコードとリソースからなる自己完結型のコレクションです。しかるべきディレクトリに置くと、所属する**カテゴリ**に従って、自動的にロードされ呼び出されます。

各プラグインは、その振る舞いを定義する単一の**カテゴリ**に属します。例えば、`importexport` カテゴリ (OJS データのインポート/エクスポートに使用される) に属するプラグインは、雑誌管理者が「データのインポート/エクスポート」インターフェースを使用した場合やコマンドラインツールが実行された場合にロードされます。インポート/エクスポートプラグインは、プラグインと OJS の間で制御を委譲するために使用されるある種のメソッドを実装しなければなりません。

プラグインは、所属するカテゴリがリクエストされた時にロードされます。例えば、`importexport` プラグインは、ページクラスである `ImportExportHandler` (`pages/manager/ImportExportHandler.inc.php` で実装) によりロードされます。リクエストは、このカテゴリに属する各プラグインが継承する `ImportExportPlugin` クラスで定義されているメソッドを介してプラグインに委譲されます。

**フック**は、プラグインに対する通知ツールであり、OJS に組み込まれている振る舞いを書き換えるために使用されます。OJS 実行中の多くの時点で、名前を指定してフックが呼び出されます。例えば、`index.php` における `LoadHandler` です。既にロードされており、該当のフックに登録されているプラグインは、フックが呼び出された時点における OJS のデフォルトの振る舞いを変更するコードを実行するチャンスを持つことになります。

OJS に組み込まれているプラグインカテゴリのほとんどは、認証やハーベスティングのような特定のタスクを定義していますが、他のどのカテゴリにもふさわしくないプラグインのために `generic` カテゴリが用意されています。これらのプラグインは、コードはより複雑になりますが、OJS に適用できる変更の種類に非常に大きな柔軟性を提供します。フックは通常、このカテゴリのプラグインで使用されることを想定しています。

## オブジェクトとクラス

OJS 2.x のプラグインはオブジェクト指向です。各プラグインはそのカテゴリの関数を定義しているクラスを継承し、その実装を担当します。

カテゴリ	ベースクラス
<code>generic</code>	<code>GenericPlugin (classes/plugins/GenericPlugin.inc.php)</code>
<code>importexport</code>	<code>ImportExportPlugin (classes/plugins/ImportExportPlugin.inc.php)</code>
<code>auth</code>	<code>AuthPlugin (classes/plugins/AuthPlugin.inc.php)</code>
<code>gateways</code>	<code>GatewayPlugin (classes/plugins/GatewayPlugin.inc.php)</code>

各ベースクラスは、そのカテゴリのプラグインが実装すべき関数の説明を含んでいます。プラグインは、`PluginRegistry` クラス (`classes/plugins/PluginRegistry.inc.php` で実装) により管理されます。また、プラグインは `HookRegistry` クラス (`classes/plugins/HookRegistry.inc.php` で実装) を使ってフックに登録することができます。

## サンプルプラグイン

以下のコードは、`generic` プラグインカテゴリに属する基本的なサンプルプラグインを示すものです。このプラグインは、すべてのファイルを `plugins/generic/example` という名のディレクトリに置くことによりインストールすることができます。

このプラグインは、ユーザホームの「雑誌管理者」リンクを辿ることにより利用できる雑

誌管理者の機能リストにエントリを追加します。

## ローダースタブ

プラグインは、OJS がプラグインディレクトリにある `index.php` と名づけられたファイルをロードすることによりロードされます。このファイルは、プラグインオブジェクトをインスタンス化して返すローダースタブです。

```
<?php
require ( 'ExamplePlugin.inc.php' ) ;
return new ExamplePlugin ( ) ;
?>
```

## プラグインオブジェクト

プラグインオブジェクトはプラグインをカプセル化したもので、通常、ほとんどの処理を行います。この例では、プラグインは `generic` カテゴリに属することになるので、オブジェクトは `GenericPlugin` クラスを継承しなければなりません。

```
<?php
import ( 'classes.plugins.GenericPlugin' ) ;
class ExamplePlugin extends GenericPlugin {
    function register ( $category, $path ) {
        if ( parent::register ( $category, $path ) ) {
            HookRegistry::register (
                'Templates::Manager::Index::ManagementPages',
                array ( &$this, 'callback' )
            ) ;
            return true ;
        }
        return false ;
    }
    function getName ( ) {
        return 'ExamplePlugin' ;
    }
    function getDisplayName ( ) {
        return 'Example Plugin' ;
    }
    function getDescription ( ) {
        return 'A description of this plugin' ;
    }
}
```

```
function callback ($hookName, $args) {
    $params =& $args[0];
    $smarty =& $args[1];
    $output =& $args[2];
    $output = '<li>&#187; <a href="http://pkp.sfu.ca">My New
Link</a></li>';
    return false;
}
?>
```

このコードは、プラグインにおける最も重要な部分、すなわち、register 関数、フックへの登録とコールバック関数、プラグイン管理を示したものです。

## 登録関数

OJS がプラグインをロード・登録すると常に、プラグインの register (...) 関数が呼び出されます。これは、必要とする任意のフックへの登録や設定のロード、データ構造の初期化などを行う機会をプラグインに与えます。上の例では、プラグインがしなければならないことは（親クラスの register 関数を呼び出すことを除けば）Templates::Manager::Index::ManagementPages フックに登録することだけです。

登録関数で行うべきもう一つの一般的な処理は、ロケールデータのロードです。ロケールデータは、プラグインディレクトリのサブディレクトリに locale/[localeName]/locale.xml という名前で置く必要があります。ここで、[localeName] は該当するロケールの標準的なシンボル名、例えば、米語では en\_US です。これらのデータファイルをロードするために、プラグインは登録関数において親の登録関数を呼び出した後に、&this->addLocaleData (); を呼び出す必要があります。

## フックへの登録とコールバック関数

上のコードは、フックへの登録とコールバック関数の明確な例を示しています。後で示すフックリストに従い、フックを使って OJS を拡張するために必要なすべての情報を提供する必要があります。ここで、さらに調査する必要がある重要な点がいくつかあります。

フックへプラグインを登録する処理は次のとおりです。

```
HookRegistry::register (
    'Templates::Manager::Index::ManagementPages',
    array (&$this, 'callback')
);
```

この例で、HookRegistry::register への引数は次のとおりです。

1. フック名。後で示す全フックリストを参照してください。
2. フックされた際に制御を渡すべきコールバック関数。これは、PHP の call\_user\_func 関数で使用されているコールバック形式と同じ形式です。詳しくは、<http://php.net> にある資料を参照してください。配列に \$this をリファレンスで含めることが重要です。そうしないと、プラグインオブジェクトを複数インスタンス化する場合に問題が生じる可能性があります。

コールバック関数の定義は（上の例では）以下のとおりです。

```
function callback ($hookName, $args) {
    $params =& $args[0];
    $smarty =& $args[1];
    $output =& $args[2];
    ...
}
```

コールバック関数の引数リストは常に同じです。

1. コールバック関数に制御を渡したフック名（同一のコールバック関数を複数のフックに登録する場合に役に立ちます）。
2. コールバック関数に渡される追加引数からなる配列。この配列の内容は、登録するフックに依存します。上の例ではテンプレートフックですので、コールバック関数は上に示した 3 つの引数を期待することができます。

引数を配列で渡すことは少し面倒ですが、フック呼び出しにおいて必要がある場合にリファレンスを引数として矛盾なく渡すことを可能にします。そうでないと、例えば上のコードでは、実際の Smarty オブジェクトではなく複製した Smarty オブジェクトを受け取ることになり、\$smarty オブジェクトの属性に加えた変更はすべて、関数から戻る際に消えてしまうことになります。

最後に、フックコールバック関数からの戻り値は非常に重要です。フックコールバック関数が `true` を返すと、フックレジストリはこのコールバック関数が完全にフックを「処理した」と判断し、それ以上、そのフックに登録されている別のコールバック関数を呼び出さなくなります。コールバック関数が `false` を返すと、現在のコールバック関数の後に同じフックに登録された別のコールバック関数が、フックコールを処理する機会を持つこととなります。

上の例では、雑誌管理者の管理機能リストにリンクを追加しています。別のプラグイン(あるいは同一のプラグイン)が同じリストに別のリンクを追加するよう登録されている場合、このプラグインが `true` を返すと、フックに登録した他のプラグインは呼び出されないこととなります。

## プラグイン管理

サンプルプラグインにおいて、プラグインに関するメタデータを提供する 3 つの関数: `getName()`, `getDisplayName()`, `getDescription()` がありました。これらは、雑誌管理者が「システムプラグイン」で利用できる管理用インターフェースの一部です。

`getName()` を呼び出した結果は、プラグインをシンボルで参照する際に使用されるものであり、人間が理解できるものである必要はありません。しかし、`getDisplayName()` 関数と `getDescription()` 関数は地域化した値を返す必要があります。簡潔さのために、上の例ではこれを行っていません。

管理インターフェースは、雑誌管理者が実行できる様々な管理用関数を、`getManagementVerbs()` 関数と `manage($verb, $args)` 関数を使ってプラグインが指定できるようにしています。`getManagementVerbs()` 関数は、次のように 2 つの要素を持つ配列群の 1 つの配列を返す必要があります。

```
$verbs = parent::getManagementVerbs();  
$verbs[] = array('func1', Locale::translate('my.localization.key.for.func1'));  
$verbs[] = array('func2', Locale::translate('my.localization.key.for.func2'));
```

上の例のように、必ず親クラスの呼び出しをすることに注意してください。管理用コマンドを自動的に提供するプラグインカテゴリがあるからです。

上のサンプルコードを使った場合、プラグインは次のように管理用コマンド `func1` と

func2 を受け取る準備をする必要があります(ここでも親クラスで提供される管理用コマンドを考慮しています)。

```
function manage($verb, $args) {
    if (!parent::manage($verb, $args)) switch ($verb) {

        case 'func1':
            // Handle func1 here
            break;
        case 'func2':
            // Handle func2 here
            break;
        default:
            return false;
    }
    return true;
}
```

## その他のプラグイン機能

その他にも便利なプラグイン機能があります。

- **プラグインの設定:** プラグインは雑誌設定と同様なメカニズムで設定を格納・抽出することができます。これには、プラグインクラスの `getSetting` 関数と `updateSetting` 関数を使用します。
- **テンプレート:** プラグインは自身のプラグインディレクトリにテンプレートを持ち、次のように呼び出すことによりそれを表示することができます。

```
$templateMgr->display($this->getTemplatePath(). 'templateName.tpl');
```

提供されている `import/expoort` プラグインを参考にしてください。

- **スキーマ管理:** `getInstallSchemaFile()` をオーバーライドし、プラグインディレクトに指定した名前のスキーマファイルを置くことにより、`generic` プラグインは OJS のスキーマ管理機能を使うことができます。この関数は、OJS のインストールまたはバージョンアップ時に呼び出されます。
- **データ管理:** `getInstallDataFile()` をオーバーライドし、プラグインディレクトに指定した名前のデータファイルを置くことにより、`generic` プラグインは OJS のデータインストール管理機能を使うことができます。この関数は、OJS のインストールまたはバージョンアップ時に呼び出されます。
- **ヘルパーコード:** プラグインのディレクトに置いたヘルパーコードは、次のようにしてインポートすることができます。

```
$this->import('HelperCode'); // imports HelperCode.inc.php
```

## フックリスト

以下のリストは、リリース 2.1 の時点で OJS に内蔵されている全てのフックを説明したものです。頭に&記号が付いた変数 (&\$sourceFile など) は、フックのコールバック関数に渡される引数配列においてその引数がリファレンスとして渡されることを示しています。フックのコールバック関数の戻り値の効果は、該当する場合のみ説明しています。これとは別に、フックのコールバック関数の戻り値は常に、同一のフックに登録されている別のコールバック関数の実行を省略するか否かを決定します。

名前	引数	説明
LoadHandler	&\$page, &\$op, &\$sourceFile	ページ (&\$page)、操作 (&\$op)、handlerコードファイル (&\$sourceFile) の名前が決まった後、\$sourceFileがロードされる前に、OJSの主たるindex.phpスクリプトから呼び出されます。ブラウザからのリクエストをインターセプトしてプラグインで処理するために使用できます。コールバックがtrueを返すと、OJSが\$sourceFileにあるハンドラスタブをロードするのを抑制します。
ArticleEmailLogDAO::_returnLogEntryFromRow	&\$entry, &\$row	ArticleEmailLogDAOがデータベース行 (&\$row) からArticleEmailLogEntry (&\$entry) オブジェクトを作成した後、そのオブジェクトを呼び出し関数に返す前に呼び出されます。
ArticleEventLogDAO::_returnLogEntryFromRow	&\$entry, &\$row	ArticleEventLogDAOがデータベース行 (&\$row) からArticleEventLogEntry (&\$entry) オブジェクトを作成した後、そのオブジェクトを呼び出し関数に返す前に呼び出されます。
ArticleCommentDAO::_returnArticleCommentFromRow	&\$articleComment, &\$row	ArticleCommentDAOがデータベース行 (&\$row) からArticleComment (&\$articleComment) オブジェクトを作成した後、そのオブジェクトを呼び出し関数に返す前に呼び出されます。
ArticleDAO::_returnArticleFromRow	&\$article, &\$row	ArticleDAOがデータベース行 (&\$row) からArticle (&\$article) オブジェクトを作成した後、そのオブジェクトを呼び出し関数に返す前に

名前	引数	説明
		呼び出されます。
ArticleFileDAO::_returnArticleFileFromRow	&\$articleFile, &\$row	ArticleFileDAOがデータベース行 (&\$row) からArticleFile (&\$articleFile) オブジェクトを作成した後、そのオブジェクトを呼び出し関数に返す前に呼び出されます。
ArticleGalleyDAO::_returnGalleyFromRow	&\$galley, &\$row	ArticleGalleyDAOがデータベース行 (&\$row) からArticleGalley (&\$galley) オブジェクトを作成した後、そのオブジェクトを呼び出し関数に返す前に呼び出されます。
ArticleNoteDAO::_returnArticleNoteFromRow	&\$articleNote, &\$row	ArticleNoteDAOがデータベース行 (&\$row) からArticleNote (&\$articleNote) オブジェクトを作成した後、そのオブジェクトを呼び出し関数に返す前に呼び出されます。
AuthorDAO::_returnAuthorFromRow	&\$author, &\$row	AuthorDAOがデータベース行 (&\$row) からAuthor (&\$author) オブジェクトを作成した後、そのオブジェクトを呼び出し関数に返す前に呼び出されます。
SuppFileDAO::_returnSuppFileFromRow	&\$suppFile, &\$row	SuppFileDAOがデータベース行 (&\$row) からSuppFile (&\$suppFile) オブジェクトを作成した後、そのオブジェクトを呼び出し関数に返す前に呼び出されます。
PublishedArticleDAO::_returnPublishedArticleFromRow	&\$publishedArticle, &\$row	PublishedArticleDAOがデータベース行 (&\$row) からPublishedArticle (&\$publishedArticle) オブジェクトを作成した後、そのオブジェクトを呼び出し関数に返す前に呼び出されます。
CommentDAO::_returnCommentFromRow	&\$comment, &\$row, &\$childLevels	CommentDAOがデータベース行 (&\$row) からComment (&\$comment) オブジェクトを作成した後で、&\$childLevelsの子のコメントを取り出し、呼び出し関数にそのコメントを返す前に呼び出されます。trueを返すと、OJSが子のコメントを取り出すことを抑制します。
Request::redirect	&\$url	Request::redirectが&\$urlへリダイレクトする前に呼び出されます。trueを返すとフックの終

名前	引数	説明
		了後に、OJSがリダイレクトを実行することを抑制します。リダイレクトをインターセプトして書き換えるのに使用できます。
Request::getBaseUrl	&\$baseUrl	Request::getBaseUrlが初めて呼び出され、ベースURLが決定された後、それを呼び出し元に返す前に呼び出されます。この値は、後続のすべての呼び出しで使用されます。
Request::getBasePath	&\$basePath	Request::getBasePathが初めて呼び出され、ベースパスが決定された後、それを呼び出し元に返す前に呼び出されます。この値は、後続のすべての呼び出しで使用されます。
Request::getIndexUrl	&\$indexUrl	Request::getIndexUrlが初めて呼び出され、インデックスURLが決定された後、それを呼び出し元に返す前に呼び出されます。この値は、後続のすべての呼び出しで使用されます。
Request::getCompleteUrl	&\$completeUrl	Request::getCompleteUrlが初めて呼び出され、完全URLが決定された後、それを呼び出し元に返す前に呼び出されます。この値は、後続のすべての呼び出しで使用されます。
Request::getRequestUrl	&\$requestUrl	Request::getRequestUrlが初めて呼び出され、リクエストURLが決定された後、それを呼び出し元に返す前に呼び出されます。この値は、後続のすべての呼び出しで使用されます。
Request::getQueryString	&\$queryString	Request::getQueryStringが初めて呼び出され、クエリ文字列が決定された後、それを呼び出し元に返す前に呼び出されます。この値は、後続のすべての呼び出しで使用されます。
Request::getRequestPath	&\$requestPath	Request::getRequestPathが初めて呼び出され、リクエストパスが決定された後、それを呼び出し元に返す前に呼び出されます。この値は、後続のすべての呼び出しで使用されます。
Request::getServerHost	&\$serverHost	Request::getServerHostが初めて呼び出され、サーバホストが決定された後、それを呼び出し元に返す前に呼び出されます。この値は、後続

名前	引数	説明
		のすべての呼び出しで使用されます。
Request::getProtocol	&\$protocol	Request::getProtocolが初めて呼び出され、プロトコル( httpまたはhttps )が決定された後、それを呼び出し元に返す前に呼び出されます。この値は、後続のすべての呼び出しで使用されます。
Request::getRemoteAddr	&\$remoteAddr	Request::getRemoteAddrが初めて呼び出され、リモートアドレスが決定された後、それを呼び出し元に返す前に呼び出されます。この値は、後続のすべての呼び出しで使用されます。
Request::getRemoteDomain	&\$remoteDomain	Request::getRemoteDomainが初めて呼び出され、リモートドメインが決定された後、それを呼び出し元に返す前に呼び出されます。この値は、後続のすべての呼び出しで使用されます。
Request::getUserAgent	&\$userAgent	Request::getUserAgentが初めて呼び出され、ユーザエージェントが決定された後、それを呼び出し元に返す前に呼び出されます。この値は、後続のすべての呼び出しで使用されます。
Request::getRequestedJournalPath	&\$journal	Request::getRequestedJournalPathが初めて呼び出され、リクエストされたジャーナルのパスが決定された後、それを呼び出し元に返す前に呼び出されます。この値は、後続のすべての呼び出しで使用されます。
[すべての]DAO::Constructor	&\$dataSource	該当する名前のDAOのコンストラクタが &\$dataSourceを指定して呼び出されるたびに呼び出されます。このフックはPHP>=4.3.0でのみ使用できます。
[すべての]DAO:: [DAO::retrieveを呼び出す任意の関数]	&\$sql, &\$params, &\$value	DAO::retrieveを呼び出す任意のDAO関数がフックを呼び出す引き金となります。&\$sqlのSQL文と&\$paramsのADODB引数は変更することができます。フックのコールバック関数でこの呼び出し関数の機能を完全に置き換える場合は、&\$valueにはretrieve関数が意図していた結果を設定して、フックはtrueを返す必要があります。このフックはPHP>=4.3.0でのみ使用できます。

名前	引数	説明
[すべての]DAO:: [DAO::retrieveCachedを呼び出す任意の関数]	&\$sql, &\$params, &\$secsToCache, &\$value	DAO::retrieveCachedを呼び出す任意のDAO関数がフックを呼び出す引き金となります。&\$sqlのSQL文と&\$paramsのADODB引数、&\$secsToCacheの秒単位のキャッシュ時間値は変更することができます。フックのコールバック関数でこの呼び出し関数の機能を完全に置き換える場合は、&\$valueにはretrieve関数が意図していた結果を設定して、フックはtrueを返す必要があります。このフックはPHP>=4.3.0でのみ使用できます。
[すべての]DAO:: [DAO::retrieveLimitを呼び出す任意の関数]	&\$sql, &\$params, &\$numRows, &\$offset, &\$value	DAO::retrieveLimitを呼び出す任意のDAO関数がフックを呼び出す引き金となります。&\$sqlのSQL文と&\$paramsのADODB引数、&\$offsetと&\$numRowsで指定されるフェッチの開始位置と上限値は変更することができます。フックのコールバック関数でこの呼び出し関数の機能を完全に置き換える場合は、&\$valueにはretrieve関数が意図していた結果を設定して、フックはtrueを返す必要があります。このフックはPHP>=4.3.0でのみ使用できます。
[すべての]DAO:: [DAO::retrieveRangeを呼び出す任意の関数]	&\$sql, &\$params, &\$dbResultRange, &\$value	DAO::retrieveRangeを呼び出す任意のDAO関数がフックを呼び出す引き金となります。&\$sqlのSQL文と&\$paramsのADODB引数、&\$dbResultRangeの範囲情報は変更することができます。フックのコールバック関数でこの呼び出し関数の機能を完全に置き換える場合は、&\$valueにはretrieve関数が意図していた結果を設定して、フックはtrueを返す必要があります。このフックはPHP>=4.3.0でのみ使用できます。
[すべての]DAO:: [DAO::updateを呼び出す任意の関数]	&\$sql, &\$params, &\$value	DAO::updateを呼び出す任意のDAO関数がフックを呼び出す引き金となります。&\$sqlのSQL文と&\$paramsのADODB引数は変更することができます。フックのコールバック関数でこの呼び出し関数の機能を完全に置き換える場合は、&\$valueにはretrieve関数が意図していた結果を設定して、

名前	引数	説明
		フックはtrueを返す必要があります。このフックはPHP>=4.3.0でのみ使用できます。
TemporaryFileDAO::_returnTemporaryFileFromRow	&\$temporaryFile, &\$row	TemporaryFileDAOがデータベース行 (&\$row) からTemporaryFile (&\$temporaryFile) オブジェクトを作成した後、そのオブジェクトが呼び出し関数に返される前に呼び出されます。
Locale::_cacheMiss	&\$id, &\$locale, &\$value	ロケールキーがロケールキャッシュで見つからない場合に呼び出されます。&\$idは見つからなかったロケール文字列のキーで、&\$localeはロケール名 (例えば、en_US) です。このフックがこのキャッシュミスを埋める結果を提供する場合は、その値を&\$valueに格納し、フックのコールバック関数はtrueを返す必要があります。
Installer::installer	&\$installer, &\$descriptor, &\$params	インストーラが、&\$descriptorの記述子パスと&\$paramsの引数でインスタンス化された時に呼び出されます。フックがtrueを返すと、Installer属性の通常の初期化が実行されなくなります。
Installer::destroy	&\$installer	インストーラのcleanupメソッドが呼ばれた時に呼び出されます。
Installer::preInstall	&\$installer, &\$result	インストーラによるインストールの事前タスクが終わり、成功/失敗の結果を&\$resultで返す前に呼び出されます。
Installer::postInstall	&\$installer, &\$result	インストーラによるインストールの事後タスクが終わり、成功/失敗の結果を&\$resultで返す前に呼び出されます。
Installer::parseInstaller	&\$installer, &\$result	インストーラによるインストールタスクの解析が終わり、成功/失敗の結果を&\$resultで返す前に呼び出されます。
Installer::executeInstaller	&\$installer, &\$result	インストーラの実行が終わり、成功/失敗の結果を&\$resultで返す前に呼び出されます。
Installer::updateVersion	&\$installer,	インストーラがシステムバージョン情報を更新し

名前	引数	説明
ersion	&\$result	た後、成功/失敗の結果を&\$resultで返す前に呼び出されます。
IssueAction::subscriptionRequired	&\$journal, &\$issue, &\$result	&\$journalの&\$issueを閲覧するのに購読が必要か否かをOJSが決定した後、真偽値を&\$resultで返す前に呼び出されます。
IssueAction::subscriptionRequiredUser	&\$journal, &\$result	カレントユーザが&\$journalを購読しているか否かをOJSが決定した後、真偽値を&\$resultで返す前に呼び出されます。
IssueAction::subscriptionRequiredDomain	&\$journal, &\$result	カレントユーザが&\$journalを購読しているドメインからアクセスしているか否かをOJSが決定し、成功/失敗の結果を&\$resultで返す前に呼び出されます。
IssueDAO::_returnIssueFromRow	&\$issue, &\$row	IssueDAOがデータベース行(&\$row)からIssue(&\$issue)オブジェクトを作成した後、そのオブジェクトを呼び出し関数に返す前に呼び出されます。
IssueDAO::_returnPublishedIssueFromRow	&\$issue, &\$row	IssueDAOがデータベース行(&\$row)から出版済みのIssue(&\$issue)オブジェクトを作成した後、そのオブジェクトを呼び出し関数に返す前に呼び出されます。
JournalDAO::_returnJournalFromRow	&\$journal, &\$row	JournalDAOがデータベース行(&\$row)からJournal(&\$journal)オブジェクトを作成した後、そのオブジェクトを呼び出し関数に返す前に呼び出されます。
SectionDAO::_returnSectionFromRow	&\$section, &\$row	SectionDAOがデータベース行(&\$row)からSection(&\$section)オブジェクトを作成した後、そのオブジェクトを呼び出し関数に返す前に呼び出されます。
EmailTemplateDAO::_returnBaseEmailTemplateFromRow	&\$emailTemplate, &\$row	EmailTemplateDAOがデータベース行(&\$row)からBaseEmailTemplate(&\$emailTemplate)オブジェクトを作成した後、そのオブジェクトを呼び出し関数に返す前に呼び出されます。
EmailTemplateDAO::_returnLocaleEmail	&\$emailTemplate, &\$row	EmailTemplateDAOがデータベース行(&\$row)から地域化したLocaleEmailTemplate

名前	引数	説明
TemplateFromRow		( <code>&amp;\$emailTemplate</code> ) オブジェクトを作成した後、そのオブジェクトを呼び出し関数に返す前に呼び出されます。
EmailTemplateDAO::_returnEmailTemplateFromRow	<code>&amp;\$emailTemplate,</code> <code>&amp;\$row</code>	EmailTemplateDAOがデータベース行 ( <code>&amp;\$row</code> ) からEmailTemplate ( <code>&amp;\$emailTemplate</code> ) オブジェクトを作成した後、そのオブジェクトを呼び出し関数に返す前に呼び出されます。
Mail::send	<code>&amp;\$mail,</code> <code>&amp;\$recipients,</code> <code>&amp;\$subject,</code> <code>&amp;\$mailBody,</code> <code>&amp;\$headers,</code> <code>&amp;\$additionalParameters</code>	引数で指定したメールが送信される直前に呼び出されます。このフックのコールバック関数がメール送信そのものを処理する場合は、コールバック関数はtrueを返し、OJSの送信機能を省略させる必要があります。
RTDAO::_returnJournalRTFromRow	<code>&amp;\$rt,</code> <code>&amp;\$row</code>	RTDAOがデータベース行 ( <code>&amp;\$row</code> ) から読書ツールのRT ( <code>&amp;\$rt</code> ) オブジェクトを作成した後、そのオブジェクトを呼び出し関数に返す前に呼び出されます。
RTDAO::_returnVersionFromRow	<code>&amp;\$version,</code> <code>&amp;\$row</code>	RTDAOがデータベース行 ( <code>&amp;\$row</code> ) から読書ツールのVersion ( <code>&amp;\$version</code> ) オブジェクトを作成した後、そのオブジェクトを呼び出し関数に返す前に呼び出されます。
RTDAO::_returnSearchFromRow	<code>&amp;\$search,</code> <code>&amp;\$row</code>	RTDAOがデータベース行 ( <code>&amp;\$row</code> ) から読書ツールのSearch ( <code>&amp;\$search</code> ) オブジェクトを作成した後、そのオブジェクトを呼び出し関数に返す前に呼び出されます。
RTDAO::_returnContextFromRow	<code>&amp;\$context,</code> <code>&amp;\$row</code>	RTDAOがデータベース行 ( <code>&amp;\$row</code> ) から読書ツールのContext ( <code>&amp;\$context</code> ) オブジェクトを作成した後、そのオブジェクトを呼び出し関数に返す前に呼び出されます。
AccessKeyDAO::_returnAccessKeyFromRow	<code>&amp;\$accessKey,</code> <code>&amp;\$row</code>	AccessKeyDAOがデータベース行 ( <code>&amp;\$row</code> ) からAccessKey ( <code>&amp;\$accessKey</code> ) オブジェクトを作成した後、そのオブジェクトを呼び出し関数に返す前に呼び出されます。

名前	引数	説明
RoleDAO::_returnRoleFromRow	&\$role, &\$row	RoleDAOがデータベース行 (&\$row) からRole (&\$role) オブジェクトを作成した後、そのオブジェクトを呼び出し関数に返す前に呼び出されます。
SiteDAO::_returnSiteFromRow	&\$site, &\$row	SiteDAOがデータベース行 (&\$row) からSite (&\$site) オブジェクトを作成した後、そのオブジェクトを呼び出し関数に返す前に呼び出されます。
VersionDAO::_returnVersionFromRow	&\$version, &\$row	VersionDAOがデータベース行 (&\$row) からVersion (&\$version) オブジェクトを作成した後、そのオブジェクトを呼び出し関数に返す前に呼び出されます。
AuthorAction::deleteArticleFile	&\$articleFile, &\$authorRevisions	OJSが著者の論文ファイル &\$articleFileを削除する前に呼び出されます。
AuthorAction::uploadRevisedVersion	&\$authorSubmission	OJSが&\$authorSubmissionの改訂版をアップロードする前に呼び出されます。
AuthorAction::completeAuthorCopyedit	&\$authorSubmission, &\$email	著者が原稿整理作業を完了した時、(有効な場合) OJSがメール (&\$email)を送付し、原稿整理作業の処理フラグを完了にする前に呼び出されます。
AuthorAction::copyeditUnderway	&\$authorSubmission	著者が原稿整理作業を始めたことを示した時、OJSが作業日を設定する前に呼び出されます。
AuthorAction::uploadCopyeditVersion	&\$authorSubmission, &\$copyeditStage	著者が指定された &\$copyeditStage に応じて &\$authorSubmission の原稿を整理したファイルをアップロードする時、OJSがファイルアップロードを受け付ける前に呼び出されます。
AuthorAction::viewLayoutComments	&\$article	著者が論文 (&\$article) に対するレイアウトコメントの閲覧をリクエストした時に呼び出されます。OJSがコメント入力フォームをインスタンス化して表示することを避けたい場合は、コールバック関数の戻り値をtrueにする必要があります。
AuthorAction::postLayoutComment	&\$article, &\$emailComment	著者が論文 (&\$article) に対するレイアウトコメントを投稿した時に呼び出されます。提供されたコメントをOJSが記録することを避けたい場合

名前	引数	説明
		は、コールバック関数の戻り値をtrueにする必要があります。
AuthorAction::view EditorDecisionComments	&\$article	著者が論文 (&\$article) に対する編集者の判断コメントの閲覧をリクエストした時に呼び出されます。OJSがコメント入力フォームをインスタンス化して表示することを避けたい場合は、コールバック関数の戻り値をtrueにする必要があります。
AuthorAction::emailEditorDecisionComment	&\$authorSubmission, &\$email	OJSが論文 (&\$authorSubmission) に対する編集者の判断コメントをメール (&\$email) で送信する前に呼び出されます。このフックは、2.1.0-1より新しいOJSでのみ使用できます。
AuthorAction::view CopyeditComments	&\$article	著者が論文 (&\$article) に対する原稿整理コメントの閲覧をリクエストした時に呼び出されます。OJSがコメント入力フォームをインスタンス化して表示することを避けたい場合は、コールバック関数の戻り値をtrueにする必要があります。
AuthorAction::post CopyeditComment	&\$article, &\$emailComment	著者が論文 (&\$article) に対する原稿整理コメントを投稿した時に呼び出されます。提供されたコメントをOJSが記録することを避けたい場合は、コールバック関数の戻り値をtrueにする必要があります。
AuthorAction::view ProofreadComments	&\$article	著者が論文 (&\$article) に対する校正コメントの閲覧をリクエストした時に呼び出されます。OJSがコメント入力フォームをインスタンス化して表示することを避けたい場合は、コールバック関数の戻り値をtrueにする必要があります。
AuthorAction::post ProofreadComment	&\$article, &\$emailComment	著者が論文 (&\$article) に対する校正コメントを投稿した時に呼び出されます。提供されたコメントをOJSが記録することを避けたい場合は、コールバック関数の戻り値をtrueにする必要があります。

名前	引数	説明
AuthorAction::downloadAuthorFile	&\$article, &\$fileId, &\$revision, &\$canDownload, &\$result	著者が論文ファイル (&\$article, &\$fileId, &\$revision) をダウンロードしようとした時、著者がそのファイルにアクセスできるか否かを OJS が決定 (変更可能な論理値フラグ &\$canDownload に設定) した後、ダウンロードが始まる前に呼び出されます。OJS のデフォルトのダウンロード処理を置き換えたい場合は、コールバック関数の戻り値を true にする必要があります。
AuthorSubmissionDAO::_returnAuthorSubmissionFromRow	&\$authorSubmission, &\$row	AuthorSubmissionDAO がデータベース行 (&\$row) から AuthorSubmission (&\$authorSubmission) オブジェクトを作成した後、そのオブジェクトを呼び出し関数に返す前に呼び出されます。
Action::viewMetadata	&\$article, &\$role	指定された役割 (&\$role) のユーザが指定された論文 (&\$article) のメタデータを閲覧しようとした時に呼び出されます。OJS が通常のメタデータ入力フォームをインスタンス化して表示することを避けたい場合は、コールバック関数の戻り値を true にする必要があります。
Action::saveMetadata	&\$article	指定した論文 (&\$article) のメタデータを OJS が更新する前に呼び出されます。OJS が更新を実行することを避けたい場合は、コールバック関数の戻り値を true にする必要があります。
Action::instructions	&\$type, &\$allowed	OJS がリクエストされた作業種別 &\$type (copy, layout, proof) の作業手順を表示する前に呼び出されます。許可された作業種別は &\$allowed にリストアップされています。OJS が作業手順を表示することを避けたい場合は、コールバック関数の戻り値を true にする必要があります。
Action::editComment	&\$article, &\$comment	OJS が指定された論文 (&\$article) のコメント (&\$comment) のためのコメント編集フォームをインスタンス化して表示する前に呼び出されます。OJS がこれを行うことを避けたい場合は、コールバック関数の戻り値を true にする必要があ

名前	引数	説明
		ります。
Action::saveComment	&\$article, &\$comment, &\$emailComment	ユーザが論文 (&\$article) に対するコメント (&\$comment) を保存しようとした時に呼び出されます。OJSが提供されたコメントを保存することを避けたい場合は、コールバック関数の戻り値をtrueにする必要があります。
Action::deleteComment	&\$comment	OJSが提供されたコメントを削除する前に呼び出されます。OJSが提供されたコメントを削除することを避けたい場合は、コールバック関数の戻り値をtrueにする必要があります。
CopyAssignmentDAO::_returnCopyAssignmentFromRow	&\$copyAssignment, &\$row	CopyAssignmentDAOがデータベース行 (&\$row) からCopyAssignment (&\$copyAssignment) オブジェクトを作成した後、そのオブジェクトを呼び出し関数に返す前に呼び出されます。
CopyeditorAction::completeCopyedit	&\$copyeditorSubmission, &\$editAssignments, &\$author, &\$email	OJSが(有効な場合) COPYEDIT_COMPLETEメールを送信し、コピーエディタの最初の原稿整理の作業フラグを完了にする前に呼び出されます。
CopyeditorAction::completeFinalCopyedit	&\$copyeditorSubmission, &\$editAssignments, &\$email	OJSが(有効な場合) COPYEDIT_FINAL_COMPLETEメールを送信し、コピーエディタの最終の原稿整理の作業フラグを完了にする前に呼び出されます。
CopyeditorAction::copyeditUnderway	&\$copyeditorSubmission	OJSが指定された投稿物 (&\$copyeditorSubmission) の原稿整理の作業フラグを処理中にする前に呼び出されます。OJSがこのフラグを立て、関連するログエントリを出力することを避けたい場合は、コールバック関数の戻り値をtrueにする必要があります。
CopyeditorAction::uploadCopyeditVersion	&\$copyeditorSubmission	OJSが&\$copyeditorSubmissionの改定版をアップロードする前に呼び出されます。

名前	引数	説明
ion		
CopyeditorAction::viewLayoutComments	&\$article	コピーエディタが論文 (&\$article) に対するレイアウトコメントの閲覧をリクエストした時に呼び出されます。OJSがコメント入力フォームをインスタンス化して表示することを避けたい場合は、コールバック関数の戻り値をtrueにする必要があります。
CopyeditorAction::postLayoutComment	&\$article, &\$emailComment	コピーエディタが論文 (&\$article) に対するレイアウトコメントを投稿しようとした時に呼び出されます。OJSが提供されたコメントを記録することを避けたい場合は、コールバック関数の戻り値をtrueにする必要があります。
CopyeditorAction::viewCopyeditComments	&\$article	コピーエディタが論文 (&\$article) に対する原稿整理コメントの閲覧をリクエストした時に呼び出されます。OJSがコメント入力フォームをインスタンス化して表示することを避けたい場合は、コールバック関数の戻り値をtrueにする必要があります。
CopyeditorAction::postCopyeditComment	&\$article, &\$emailComment	コピーエディタが論文 (&\$article) に対する原稿整理コメントを投稿しようとした時に呼び出されます。OJSが提供されたコメントを記録することを避けたい場合は、コールバック関数の戻り値をtrueにする必要があります。このフックは、2.1.0-1より新しいOJSでのみ使用できます。
CopyeditorAction::downloadCopyeditorFile	&\$submission, &\$fileId, &\$revision, &\$result	コピーエディタが論文ファイル (&\$article, &\$fileId, &\$revision) をダウンロードしようとした時、コピーエディタがそのファイルにアクセスできるか否かをOJSが決定(変更可能な論理値フラグ &\$canDownloadに設定)した後、ダウンロード自体が始まる前に呼び出されます。OJSのデフォルトのダウンロード処理を置き換えたい場合は、コールバック関数の戻り値をtrueにする必要があります。このフックは、2.1.0-1より新しいOJSでのみ使用できます。

名前	引数	説明
CopyeditorSubmissionDAO::_returnCopyeditorSubmissionFromRow	&\$copyeditorSubmission, &\$row	CopyeditorSubmissionDAOがデータベース行 (&\$row) からCopyeditorSubmission (&\$copyeditorSubmission) オブジェクトを作成した後、そのオブジェクトを呼び出し関数に返す前に呼び出されます。
EditAssignmentsDAO::_returnEditAssignmentFromRow	&\$editAssignment, &\$row	EditAssignmentsDAOがデータベース行 (&\$row) からEditAssignment (&\$editAssignment) オブジェクトを作成した後、そのオブジェクトを呼び出し関数に返す前に呼び出されます。このフックは、2.1.0-1より新しいOJSでのみ使用できます。
EditorAction::assignEditor	&\$editorSubmission, &\$sectionEditor, &\$email	OJSが指定された編集者またはセクションエディタ (&\$sectionEditor) を論文 (&\$editorSubmission) に割り当て、(有効な場合) 提供されたメール (&\$email) を送信する前に呼び出されます。
EditorSubmissionDAO::_returnEditorSubmissionFromRow	&\$editorSubmission, &\$row	EditorSubmissionDAOがデータベース行 (&\$row) からEditorSubmission (&\$editorSubmission) オブジェクトを作成した後、そのオブジェクトを呼び出し関数に返す前に呼び出されます。
LayoutAssignmentDAO::_returnLayoutAssignmentFromRow	&\$layoutAssignment, &\$row	LayoutAssignmentDAOがデータベース行 (&\$row) からLayoutAssignment (&\$layoutAssignment) オブジェクトを作成した後、そのオブジェクトを呼び出し関数に返す前に呼び出されます。
LayoutEditorAction::deleteGalley	&\$article, &\$galley	OJSが論文 (&\$article) の指定されたゲラ (&\$galley) を削除する前に呼び出されます。OJSがゲラを削除することを避けたい場合は、コールバック関数の戻り値をtrueにする必要があります。
LayoutEditorAction::deleteSuppFile	&\$article, &\$suppFile	OJSが論文 (&\$article) の指定された補足ファイル (&\$suppFile) を削除する前に呼び出されます。OJSが補足ファイルを削除することを避けた

名前	引数	説明
		い場合は、コールバック関数の戻り値をtrueにする必要があります。
LayoutEditorAction ::completeLayoutEditing	&\$submission, &\$layoutAssignment, &\$editAssignments, &\$email	OJSが論文 (&\$submission) に任命したレイアウトエディタ (&\$layoutAssignment) を記録し、(有効な場合) 提供されたメール (&\$email) を送信する前に呼び出されます。
LayoutEditorAction ::viewLayoutComments	&\$article	レイアウトエディタが論文 (&\$article) に対するレイアウトコメントの閲覧をリクエストした時に呼び出されます。OJSがコメント入力フォームをインスタンス化して表示することを避けたい場合は、コールバック関数の戻り値をtrueにする必要があります。
LayoutEditorAction ::postLayoutComment	&\$article, &\$emailComment	レイアウトエディタが論文 (&\$article) に対するレイアウトコメントを投稿しようとした時に呼び出されます。OJSが提供されたコメントを記録することを避けたい場合は、コールバック関数の戻り値をtrueにする必要があります。
LayoutEditorAction ::viewProofreadComments	&\$article	レイアウトエディタが論文 (&\$article) に対する校正コメントの閲覧をリクエストした時に呼び出されます。OJSがコメント入力フォームをインスタンス化して表示することを避けたい場合は、コールバック関数の戻り値をtrueにする必要があります。
LayoutEditorAction ::postProofreadComment	&\$article, &\$emailComment	レイアウトエディタが論文 (&\$article) に対する校正コメントを投稿しようとした時に呼び出されます。OJSが提供されたコメントを記録することを避けたい場合は、コールバック関数の戻り値をtrueにする必要があります。

名前	引数	説明
LayoutEditorAction ::downloadFile	&\$article, &\$fileId, &\$revision, &\$canDownload, &\$result	レイアウトエディタが論文ファイル (&\$article, &\$fileId, &\$revision) をダウンロードしようとした時、レイアウトエディタがそのファイルにアクセスできるか否かをOJSが決定 (変更可能な論理値フラグ &\$canDownload に設定) した後、ダウンロード自体が始まる前に呼び出されます。OJSのデフォルトのダウンロード処理を置き換えたい場合は、&\$resultに成功を示す論理値を設定し、コールバック関数の戻り値をtrueにする必要があります。
LayoutEditorSubmissionDAO::_returnLayoutEditorSubmissionFromRow	&\$submission, &\$row	LayoutEditorSubmissionDAOがデータベース行 (&\$row) からLayoutEditorSubmission (&\$submission) オブジェクトを作成した後、そのオブジェクトを呼び出し関数に返す前に呼び出されます。
ProofAssignmentDAO::_returnProofAssignmentFromRow	&\$proofAssignment, &\$row	ProofAssignmentDAOがデータベース行 (&\$row) からProofAssignment (&\$proofAssignment) オブジェクトを作成した後、そのオブジェクトを呼び出し関数に返す前に呼び出されます。
ProofreaderAction: :selectProofreader	&\$userId, &\$article, &\$proofAssignment	OJSが論文 (&\$article) の校正者として指定したユーザ (&\$userId) を任命して、指定した ProofAssignment オブジェクト (&\$proofAssignment) に設定する前に呼び出されます。OJSが通常の処理を実行することを避けたい場合は、コールバック関数の戻り値をtrueにする必要があります。
ProofreaderAction: :queueForScheduling	&\$article, &\$proofAssignment	指定された校正者 (&\$proofAssignment) が指定された論文 (&\$article) をスケジュールキューに入れる時に呼び出されます。OJSが通常の処理を実行することを避けたい場合は、コールバック関数の戻り値をtrueにする必要があります。
ProofreaderAction: :proofreadEmail	&\$proofAssignment, &\$email,	OJSが校正者に指定されたメール種類 (&\$mailType、例えば、PROOFREAD_LAYOUT_COMPLETE) を送信し、指

名前	引数	説明
	&\$mailType	定された&\$proofAssignmentを任命した日付を登録する前に呼び出されます。
ProofreaderAction: :authorProofreadingUnderway	&\$submission, &\$proofAssignment	OJSが論文( &\$submission )に対する著者による校正( &\$proofAssignment )の作業フラグを作業中にする前に呼び出されます。OJSがフラグを作業中にするこことを避けたい場合は、コールバック関数の戻り値をtrueにする必要があります。
ProofreaderAction: :proofreaderProofreadingUnderway	&\$submission, &\$proofAssignment	OJSが論文( &\$submission )に対する校正者による校正( &\$proofAssignment )の作業フラグを作業中にする前に呼び出されます。OJSがフラグを作業中にするこことを避けたい場合は、コールバック関数の戻り値をtrueにする必要があります。
ProofreaderAction: :layoutEditorProofreadingUnderway	&\$submission, &\$proofAssignment	OJSが論文( &\$submission )に対するレイアウトエディタによる校正( &\$proofAssignment )の作業フラグを作業中にする前に呼び出されます。OJSがフラグを作業中にするこことを避けたい場合は、コールバック関数の戻り値をtrueにする必要があります。
ProofreaderAction: :downloadProofreaderFile	&\$submission, &\$fileId, &\$revision, &\$canDownload, &\$result	校正者が論文ファイル( &\$article, &\$fileId, &\$revision )をダウンロードしようとした時、校正者がそのファイルにアクセスできるか否かをOJSが決定( 変更可能な論理値フラグ &\$canDownloadに設定 )した後、ダウンロード自体が始まる前に呼び出されます。OJSのデフォルトのダウンロード処理を置き換えたい場合は、&\$resultに成功を示す論理値を設定し、コールバック関数の戻り値をtrueにする必要があります。
ProofreaderAction: :viewProofreadingComments	&\$article	校正者が論文( &\$article )に対する校正コメントの閲覧をリクエストした時に呼び出されます。OJSがコメント入力フォームをインスタンス化して表示することを避けたい場合は、コールバック

名前	引数	説明
		関数の戻り値をtrueにする必要があります。
ProofreaderAction: :postProofreadComment	&\$article, &\$emailComment	校正者が論文 (&\$article) に対する校正コメントを投稿しようとした時に呼び出されます。OJSが提供されたコメントを記録することを避けたい場合は、コールバック関数の戻り値をtrueにする必要があります。
ProofreaderAction: :viewLayoutComments	&\$article	校正者が論文 (&\$article) に対するレイアウトコメントの閲覧をリクエストした時に呼び出されます。OJSがコメント入力フォームをインスタンス化して表示することを避けたい場合は、コールバック関数の戻り値をtrueにする必要があります。
ProofreaderAction: :postLayoutComment	&\$article, &\$emailComment	校正者が論文 (&\$article) に対するレイアウトコメントを投稿しようとした時に呼び出されます。OJSが提供されたコメントを記録することを避けたい場合は、コールバック関数の戻り値をtrueにする必要があります。
ProofreaderSubmissionDAO::_returnProofreaderSubmissionFromRow	&\$submission, &\$row	ProofreaderSubmissionDAOがデータベース行 (&\$row) からProofreaderSubmission (&\$submission) オブジェクトを作成した後、そのオブジェクトを呼び出し関数に返す前に呼び出されます。
ReviewAssignmentDAO::_returnReviewAssignmentFromRow	&\$reviewAssignment, &\$row	ReviewAssignmentDAOがデータベース行 (&\$row) からReviewAssignment (&\$reviewAssignment) オブジェクトを作成した後、そのオブジェクトを呼び出し関数に返す前に呼び出されます。
ReviewerAction::_confirmReview	&\$reviewerSubmission, &\$email, &\$decline	OJSが指定された査読者 (&\$reviewAssignment) の返事 (&\$decline: 受諾/謝絶) を記録し、(有効な場合) 編集者にメール (&\$email) を送信する前に呼び出されます。
ReviewerAction::_recordRecommendation	&\$reviewerSubmission, &\$email,	OJSが指定された査読者 (&\$reviewAssignment) の答申 (&\$recommendation) を記録し、(有効な場合) 編集者にメール (&\$email) を送信する

名前	引数	説明
	<code>&amp;\$recommendation</code>	前に呼び出されます。
<code>ReviewerAction::uploadReviewFile</code>	<code>&amp;\$reviewAssignment</code>	OJSが指定された査読者 ( <code>&amp;\$reviewAssignment</code> ) の査読ファイルをアップロードされたファイルで更新する前に呼び出されます。
<code>ReviewerAction::deleteReviewerVersion</code>	<code>&amp;\$reviewAssignment,</code> <code>&amp;\$fileId,</code> <code>&amp;\$revision</code>	OJSが指定された査読者のファイル ( <code>&amp;\$reviewAssignment,</code> <code>&amp;\$fileId,</code> <code>&amp;\$version</code> ) を削除する前に呼び出されます。OJSが削除することを避けたい場合は、コールバック関数の戻り値をtrueにする必要があります。
<code>ReviewerAction::viewPeerReviewComments</code>	<code>&amp;\$user,</code> <code>&amp;\$article,</code> <code>&amp;\$reviewId</code>	OJSが指定された論文 ( <code>&amp;\$article</code> ) の査読ID ( <code>&amp;\$reviewId</code> ) の査読者コメントを査読者 ( <code>&amp;\$user</code> ) に表示する前に呼び出されます。OJSがコメントを表示することを避けたい場合は、コールバック関数の戻り値をtrueにする必要があります。
<code>ReviewerAction::postPeerReviewComment</code>	<code>&amp;\$user,</code> <code>&amp;\$article,</code> <code>&amp;\$reviewId,</code> <code>&amp;\$emailComment</code>	OJS が指定された論文 ( <code>&amp;\$article</code> ) に対して査読者 ( <code>&amp;\$user</code> ) が与えた査読ID ( <code>&amp;\$reviewId</code> ) の新規コメントを記録する前に呼び出されます。OJSがコメントを記録することを避けたい場合は、コールバック関数の戻り値をtrueにする必要があります。
<code>ReviewerAction::downloadReviewerFile</code>	<code>&amp;\$article,</code> <code>&amp;\$fileId,</code> <code>&amp;\$revision,</code> <code>&amp;\$canDownload,</code> <code>&amp;\$result</code>	査読者が論文ファイル ( <code>&amp;\$article,</code> <code>&amp;\$fileId,</code> <code>&amp;\$revision</code> ) をダウンロードしようとした時、査読者がそのファイルにアクセスできるか否かをOJSが決定 (変更可能な論理値フラグ <code>&amp;\$canDownload</code> に設定) した後、ダウンロード自体が始まる前に呼び出されます。OJSのデフォルトのダウンロード処理を置き換えたい場合は、 <code>&amp;\$result</code> に成功を示す論理値を設定し、コールバック関数の戻り値をtrueにする必要があります。

名前	引数	説明
ReviewerAction::editComment	&\$article, &\$comment, &\$reviewId	OJSが指定された論文 (&\$article) のコメント (&\$comment) のために査読者用のコメント編集フォームをインスタンス化して表示する前に呼び出されます。OJSがこれを実行することを避けたい場合は、コールバック関数の戻り値をtrueにする必要があります。
ReviewerSubmissionDAO::_returnReviewerSubmissionFromRow	&\$reviewerSubmission, &\$row	ReviewerSubmissionDAOがデータベース行 (&\$row) からReviewerSubmission (&\$reviewerSubmission) オブジェクトを作成した後、そのオブジェクトを呼び出し関数に返す前に呼び出されます。
SectionEditorAction::designateReviewVersion	&\$sectionEditorSubmission	OJSが指定された論文 (&\$sectionEditorSubmission) の査読者版としてオリジナルファイルを指定する前に呼び出されます。OJSがこれを実行することを避けたい場合は、コールバック関数の戻り値をtrueにする必要があります。
SectionEditorAction::changeSection	&\$sectionEditorSubmission, &\$sectionId	OJSが投稿物 (&\$sectionEditorSubmission) のセクションを指定のセクションID (&\$sectionId) を持つセクションに変更する前に呼び出されます。OJSがこれを実行することを避けたい場合は、コールバック関数の戻り値をtrueにする必要があります。
SectionEditorAction::recordDecision	&\$sectionEditorSubmission, &\$editorDecision	OJSが投稿物 (&\$sectionEditorSubmission) に対する編集者の判断 (&\$editorDecision) を記録する前に呼び出されます。OJSがこれを実行することを避けたい場合は、コールバック関数の戻り値をtrueにする必要があります。
SectionEditorAction::addReviewer	&\$sectionEditorSubmission, &\$reviewerId	OJSが指定された査読者 (&\$reviewerId) を投稿物 (&\$sectionEditorSubmission) に対する新規査読者として任命する前に呼び出されます。OJSがこれを実行することを避けたい場合は、コールバック関数の戻り値をtrueにする必要があります。

名前	引数	説明
SectionEditorAction::clearReview	&\$sectionEditorSubmission, &\$reviewAssignment	OJSが投稿物 (&\$sectionEditorSubmission) に対する査読者の任命 (&\$reviewAssignment) をクリアする前に呼び出されます。 OJSがこれを実行することを避けたい場合は、コールバック関数の戻り値をtrueにする必要があります。
SectionEditorAction::notifyReviewer	&\$sectionEditorSubmission, &\$reviewAssignment, &\$email	(有効な場合) 査読者へ督促メールを送信することにより、投稿物 (&\$sectionEditorSubmission) の査読者任命に対する返事がない査読者にOJSが通知フラグを立てる前に呼び出されます。
SectionEditorAction::cancelReview	&\$sectionEditorSubmission, &\$reviewAssignment, &\$email	OJSが投稿物 (&\$sectionEditorSubmission) に対する査読者の任命 (&\$reviewAssignment) をキャンセルする前に呼び出されます。 OJSがこれを実行することを避けたい場合は、コールバック関数の戻り値をtrueにする必要があります。
SectionEditorAction::remindReviewer	&\$sectionEditorSubmission, &\$reviewAssignment, &\$email	(有効な場合) 査読者へ督促メール (&\$email) を送信することにより、OJSが査読未了の投稿物 (&\$sectionEditorSubmission) を査読者に督促する前に呼び出されます。
SectionEditorAction::thankReviewer	&\$sectionEditorSubmission, &\$reviewAssignment, &\$email	(有効な場合) 査読者へメール (&\$email) を送信することにより、OJSが投稿物 (&\$sectionEditorSubmission) の査読が完了したことを査読者に感謝する前に呼び出されます。
SectionEditorAction::rateReviewer	&\$reviewAssignment, &\$reviewer, &\$quality	OJSが査読者 (&\$reviewer) の査読作業 (&\$reviewAssignment) に対する品質評価 (&\$quality) を記録する前に呼び出されます。 OJSが評価を記録することを避けたい場合は、コールバック関数の戻り値をtrueにする必要があります。
SectionEditorAction::makeReviewerFileViewable	&\$reviewAssignment, &\$articleFile,	OJSが査読作業 (&\$reviewAssignment) に付随する査読者のファイル (&\$articleFile) に対する読み出し権限の新規設定を記録する前に呼び

名前	引数	説明
	<code>&amp;\$viewable</code>	出されます。 OJSが新規設定を記録することを避けたい場合は、コールバック関数の戻り値をtrueにする必要があります。
<code>SectionEditorAction::setDueDate</code>	<code>&amp;\$reviewAssignment,</code> <code>&amp;\$reviewer,</code> <code>&amp;\$dueDate,</code> <code>&amp;\$numWeeks</code>	OJSが査読者 ( <code>&amp;\$reviewer</code> ) の査読作業 ( <code>&amp;\$reviewAssignment</code> ) の期限日を <code>&amp;\$dueDate</code> に設定する前に 呼び出されます。 OJSが期限日を設定することを避けたい場合は、コールバック関数の戻り値をtrueにする必要があります。
<code>SectionEditorAction::unsuitableSubmission</code>	<code>&amp;\$sectionEditorSubmission,</code> <code>&amp;\$author,</code> <code>&amp;\$email</code>	(有効な場合) メール ( <code>&amp;\$email</code> ) を送信することにより、OJSが著者 ( <code>&amp;\$author</code> ) の投稿物 ( <code>&amp;\$sectionEditorSubmission</code> ) が雑誌の投稿規程に不適合であると記録する前に呼び出されます。
<code>SectionEditorAction::notifyAuthor</code>	<code>&amp;\$sectionEditorSubmission,</code> <code>&amp;\$author,</code> <code>&amp;\$email</code>	OJSが投稿物 ( <code>&amp;\$sectionEditorSubmission</code> ) に関して著者 ( <code>&amp;\$author</code> ) に通知メール ( <code>&amp;\$email</code> ) を送信する前に呼び出されます。
<code>SectionEditorAction::setReviewerRecommendation</code>	<code>&amp;\$reviewAssignment,</code> <code>&amp;\$reviewer,</code> <code>&amp;\$recommendation,</code> <code>&amp;\$acceptOption</code>	査読者 ( <code>&amp;\$reviewer</code> ) の査読論文 ( <code>&amp;\$reviewAssignment</code> ) に対する答申 ( <code>&amp;\$recommendation</code> ) が記録される前に呼び出されます。 答申が記録されることを避けたい場合は、コールバック関数の戻り値をtrueにする必要があります。
<code>SectionEditorAction::setCopyeditFile</code>	<code>&amp;\$sectionEditorSubmission,</code> <code>&amp;\$fileId,</code> <code>&amp;\$revision</code>	OJSが投稿物 ( <code>&amp;\$sectionEditorSubmission</code> ) のコピーエディタファイルを指定したファイル ( <code>&amp;\$fileId</code> ) の改訂版 ( <code>&amp;\$revision</code> ) に設定する前に呼び出されます。 この変更を避けたい場合は、コールバック関数の戻り値をtrueにする必要があります。
<code>SectionEditorAction::resubmitFile</code>	<code>&amp;\$sectionEditorSubmission,</code> <code>&amp;\$fileId,</code> <code>&amp;\$revision</code>	OJSが投稿物 ( <code>&amp;\$sectionEditorSubmission</code> ) に属するファイル ( <code>&amp;\$fileId</code> と <code>&amp;\$revision</code> ) を査読用に再投稿する前に呼び出されます。 これが実行されることを避けたい場合は、コールバック

名前	引数	説明
		ク関数の戻り値をtrueにする必要があります。
SectionEditorAction::selectCopyeditor	&\$sectionEditorSubmission, &\$copyeditorId	OJSがID (&\$copyeditorId) のユーザを投稿物 (&\$sectionEditorSubmission) のコピーエディタとして任命する前に呼び出されます。これが実行されることを避けたい場合は、コールバック関数の戻り値をtrueにする必要があります。
SectionEditorAction::notifyCopyeditor	&\$sectionEditorSubmission, &\$copyeditor, &\$email	(有効な場合) メール (&\$email) を送信することにより、OJSが投稿物 (&\$sectionEditorSubmission) に対する原稿整理作業の任命をコピーエディタ (&\$copyeditor) に通知する前に呼び出されます。
SectionEditorAction::initiateCopyedit	&\$sectionEditorSubmission	投稿物 (&\$sectionEditorSubmission) に対する編集者による原稿整理作業の開始フラグを立てる前に呼び出されます。これが実行されることを避けたい場合は、コールバック関数の戻り値をtrueにする必要があります。
SectionEditorAction::thankCopyeditor	&\$sectionEditorSubmission, &\$copyeditor, &\$email	(有効な場合) メール (&\$email) を送信することにより、OJSが投稿物 (&\$sectionEditorSubmission) への貢献をコピーエディタに感謝する前に呼び出されます。
SectionEditorAction::notifyAuthorCopyedit	&\$sectionEditorSubmission, &\$author, &\$email	(有効な場合) メール (&\$email) を送信することにより、OJSが投稿物 (&\$sectionEditorSubmission) に対する原稿整理作業を著者 (&\$author) に依頼したことを記録する前に呼び出されます。
SectionEditorAction::thankAuthorCopyedit	&\$sectionEditorSubmission, &\$author, &\$email	(有効な場合) メール (&\$email) を送信することにより、OJSが投稿物 (&\$sectionEditorSubmission) の原稿整理に対する著者 (&\$author) の貢献を感謝したことを記録する前に呼び出されます。

名前	引数	説明
SectionEditorAction::notifyFinalCopyedit	&\$sectionEditorSubmission, &\$copyeditor, &\$email	(有効な場合)メール(&\$email)を送信することにより、OJSが投稿物(&\$sectionEditorSubmission)に対する最終の原稿整理作業を行うようコピーエディタに通知したことを記録する前に呼び出されます。
SectionEditorAction::thankFinalCopyedit	&\$sectionEditorSubmission, &\$copyeditor, &\$email	(有効な場合)メール(&\$email)を送信することにより、OJSが投稿物(&\$sectionEditorSubmission)の最終原稿整理に対するコピーエディタ(&\$copyeditor)の貢献を感謝したことを記録する前に呼び出されます。
SectionEditorAction::uploadReviewVersion	&\$sectionEditorSubmission	OJSが投稿物(&\$sectionEditorSubmission)の新規査読者版を格納する前に呼び出されます。OJSがこれを実行することを避けたい場合は、コールバック関数の戻り値をtrueにする必要があります。
SectionEditorAction::uploadEditorVersion	&\$sectionEditorSubmission	OJSが投稿物(&\$sectionEditorSubmission)の新規編集者版を格納する前に呼び出されます。OJSがこれを実行することを避けたい場合は、コールバック関数の戻り値をtrueにする必要があります。
SectionEditorAction::uploadCopyeditVersion	&\$sectionEditorSubmission	OJSが投稿物(&\$sectionEditorSubmission)の新規原稿整理版を格納する前に呼び出されます。OJSがこれを実行することを避けたい場合は、コールバック関数の戻り値をtrueにする必要があります。
SectionEditorAction::completeCopyedit	&\$sectionEditorSubmission	OJSが投稿物(&\$sectionEditorSubmission)の原稿整理作業の完了日付を記録する前に呼び出されます。OJSがこれを記録することを避けたい場合は、コールバック関数の戻り値をtrueにする必要があります。
SectionEditorAction::completeFinalCopyedit	&\$sectionEditorSubmission	OJSが投稿物(&\$sectionEditorSubmission)の編集者による最終原稿整理作業の完了日付を記録する前に呼び出されます(コピーエディタを任

名前	引数	説明
		命していない場合)。OJSがこれを記録することを避けたい場合は、コールバック関数の戻り値をtrueにする必要があります。
SectionEditorAction::archiveSubmission	&\$sectionEditorSubmission	OJSが投稿物 (&\$sectionEditorSubmission) をアーカイブする前に呼び出されます。OJSがこれを実行することを避けたい場合は、コールバック関数の戻り値をtrueにする必要があります。
SectionEditorAction::restoreToQueue	&\$sectionEditorSubmission	OJSが投稿物 (&\$sectionEditorSubmission) をアーカイブから作業キューに戻す前に呼び出されます。OJSがこれを実行することを避けたい場合は、コールバック関数の戻り値をtrueにする必要があります。
SectionEditorAction::updateSection	&\$submission, &\$sectionId	OJSが論文 (&\$submission) をID (&\$sectionId) のセクションに移動させる前に呼び出されます。OJSがこれを実行することを避けたい場合は、コールバック関数の戻り値をtrueにする必要があります。
SectionEditorAction::uploadLayoutVersion	&\$submission, &\$layoutAssignment	OJSが指定されたレイアウトエディタ (&\$layoutAssignment) による投稿物 (&\$submission) の新規レイアウト版を格納する前に呼び出されます。OJSがこれを実行することを避けたい場合は、コールバック関数の戻り値をtrueにする必要があります。
SectionEditorAction::assignLayoutEditor	&\$submission, &\$editorId	OJSがID (&\$editorId) のユーザを投稿物 (&\$submission) のレイアウトエディタとして任命する前に呼び出されます。OJSがこれを実行することを避けたい場合は、コールバック関数の戻り値をtrueにする必要があります。
SectionEditorAction::notifyLayoutEditor	&\$submission, &\$layoutEditor, &\$layoutAssignment, &\$email	(有効な場合) メール (&\$email) を送信することにより、OJSが投稿物 (&\$submission) のレイアウトエディタ (&\$layoutEditor) に通知したことを示すフラグを立てる前に呼び出されます。

名前	引数	説明
SectionEditorAction::thankLayoutEditor	&\$submission, &\$layoutEditor, &\$layoutAssignment, &\$email	(有効な場合)メール(&\$email)を送信することにより、OJSが投稿物(&\$submission)に対するコピーエディタ(&\$layoutEditor)の作業に感謝したことを記録する前に呼び出されます。
SectionEditorAction::deleteArticleFile	&\$submission, &\$fileId, &\$revision	OJSが投稿物(&\$submission)の論文ファイル(&\$fileId, &\$revision)を削除する前に呼び出されます。OJSがこれを実行することを避けたい場合は、コールバック関数の戻り値をtrueにする必要があります。
SectionEditorAction::addSubmissionNote	&\$articleId, &\$articleNote	OJSが論文ID(&\$articleId)の投稿物に投稿注記(&\$articleNote)を追加する前に呼び出されます。OJSがこれを実行することを避けたい場合は、コールバック関数の戻り値をtrueにする必要があります。
SectionEditorAction::removeSubmissionNote	&\$articleId, &\$noteId, &\$fileId	OJSが論文ID(&\$articleId)の投稿物の投稿注記(&\$noteId)と(もしあれば)関連ファイル(&\$fileId)を削除する前に呼び出されます。OJSがこれを実行することを避けたい場合は、コールバック関数の戻り値をtrueにする必要があります。
SectionEditorAction::updateSubmissionNote	&\$articleId, &\$articleNote	OJSが論文ID(&\$articleId)の投稿物について、オブジェクト(&\$articleNote)には適用したが、まだデータベースにコミットしていない投稿注記の変更を保存する前に呼び出されます。新規補足ファイルがアップロードされていたとしても、まだ格納はされていません。OJSが変更を保存することを避けたい場合は、コールバック関数の戻り値をtrueにする必要があります。
SectionEditorAction::clearAllSubmissionNotes	&\$articleId	OJSが論文ID(&\$articleId)の投稿物の投稿注記と(もしあれば)関連ファイルをすべて削除する前に呼び出されます。OJSがこれを実行することを避けたい場合は、コールバック関数の戻り値をtrueにする必要があります。

名前	引数	説明
SectionEditorAction::viewPeerReviewComments	&\$article, &\$reviewId	OJSが指定された論文 (&\$article) に対するID (&\$reviewId) の査読者の査読コメントを編集者またはセクションエディタに表示する前に呼び出されます。OJSがコメントを表示することを避けたい場合は、コールバック関数の戻り値をtrueにする必要があります。
SectionEditorAction::postPeerReviewComment	&\$article, &\$reviewId, &\$emailComment	OJSが論文 (&\$article) に対するID (&\$reviewId) の編集者またはセクションエディタによる新規コメントを記録する前に呼び出されます。OJSがコメントを記録することを避けたい場合は、コールバック関数の戻り値をtrueにする必要があります。
SectionEditorAction::viewEditorDecisionComments	&\$article	編集者またはセクションエディタが論文 (&\$article) に対する編集者の判断コメントの閲覧をリクエストした時に呼び出されます。OJSがコメント用フォームをインスタンス化して表示することを避けたい場合は、コールバック関数の戻り値をtrueにする必要があります。
SectionEditorAction::postEditorDecisionComment	&\$article, &\$emailComment	OJSが投稿物 (&\$article) に対する編集者の新規コメントを記録する前に呼び出されます。OJSがこれを実行することを避けたい場合は、コールバック関数の戻り値をtrueにする必要があります。
SectionEditorAction::emailEditorDecisionComment	&\$sectionEditorSubmission, &\$send	OJSが投稿物に対する編集者の判断コメント (&\$sectionEditorDecision) を著者にメール送信する前に呼び出されます。
SectionEditorAction::blindCcReviewsToReviewers	&\$article, &\$reviewAssignments, &\$email	OJSが論文 (&\$article) の査読者に査読結果 (&\$reviewAssignments) のメール (&\$email) を匿名で送信する前に呼び出されます。
SectionEditorAction::viewCopyeditComments	&\$article	編集者またはセクションエディタが論文 (&\$article) に対する原稿整理コメントの閲覧をリクエストした時に呼び出されます。OJSがコメント用フォームをインスタンス化して表示することを避けたい場合は、コールバック関数の戻り値をtrueにする必要があります。

名前	引数	説明
SectionEditorAction::postCopyeditComment	&\$article, &\$emailComment	編集者またはセクションエディタが論文 (&\$article) に対する原稿整理コメントを投稿しようとした時に呼び出されます。OJSが提供されたコメントを記録することを避けたい場合は、コールバック関数の戻り値をtrueにする必要があります。
SectionEditorAction::viewLayoutComments	&\$article	編集者またはセクションエディタが論文 (&\$article) に対するレイアウトコメントの閲覧をリクエストした時に呼び出されます。OJSがコメント用フォームをインスタンス化して表示することを避けたい場合は、コールバック関数の戻り値をtrueにする必要があります。
SectionEditorAction::postLayoutComment	&\$article, &\$emailComment	編集者またはセクションエディタが論文 (&\$article) に対するレイアウトコメントを投稿しようとした時に呼び出されます。OJSが提供されたコメントを記録することを避けたい場合は、コールバック関数の戻り値をtrueにする必要があります。
SectionEditorAction::viewProofreadComments	&\$article	編集者またはセクションエディタが論文 (&\$article) に対する校正コメントの閲覧をリクエストした時に呼び出されます。OJSがコメント用フォームをインスタンス化して表示することを避けたい場合は、コールバック関数の戻り値をtrueにする必要があります。
SectionEditorAction::postProofreadComment	&\$article, &\$emailComment	編集者またはセクションエディタが論文 (&\$article) に対する校正コメントを投稿しようとした時に呼び出されます。OJSが提供されたコメントを記録することを避けたい場合は、コールバック関数の戻り値をtrueにする必要があります。
SectionEditorAction::acceptReviewForReviewer	&\$reviewAssignment, &\$reviewer	OJSが、編集者が査読者 (&\$reviewer) に代わって査読作業 (&\$reviewAssignment) を引き受けたことを記録する前に呼び出されます。OJSがこれを実行することを避けたい場合は、コールバック関数の戻り値をtrueにする必要があります。

名前	引数	説明
SectionEditorAction::uploadReviewForReviewer	&\$reviewAssignment, &\$reviewer	OJSが、編集者が査読者 (&\$reviewer) に代わって査読作業 (&\$reviewAssignment) を行い、アップロードした編集者の査読結果を格納する前に呼び出されます。OJSがこれを実行することを避けた場合は、コールバック関数の戻り値をtrueにする必要があります。
SectionEditorSubmissionDAO::_returnSectionEditorSubmissionFromRow	&\$sectionEditorSubmission, &\$row	SectionEditorSubmissionDAOがデータベース行 (&\$row) からSectionEditorSubmission (&\$sectionEditorSubmission) オブジェクトを作成した後、そのオブジェクトを呼び出し関数に返す前に呼び出されます。
SectionEditorSubmissionDAO::_returnReviewerUserFromRow	&\$user, &\$row	SectionEditorSubmissionDAOがデータベース行 (&\$row) からUser (&\$user) オブジェクトを作成した後、そのオブジェクトを呼び出し関数に返す前に呼び出されます。このフックは将来削除されるかもしれませんので、その使用は推奨されません。
CurrencyDAO::_returnCurrencyFromRow	&\$currency, &\$row	CurrencyDAOがデータベース行 (&\$row) からCurrency (&\$currency) オブジェクトを作成した後、そのオブジェクトを呼び出し関数に返す前に呼び出されます。
SubscriptionDAO::_returnSubscriptionFromRow	&\$subscription, &\$row	SubscriptionDAOがデータベース行 (&\$row) からSubscription (&\$subscription) オブジェクトを作成した後、そのオブジェクトを呼び出し関数に返す前に呼び出されます。
SubscriptionTypeDAO::_returnSubscriptionTypeFromRow	&\$subscriptionType, &\$row	SubscriptionTypeDAOがデータベース行 (&\$row) からSubscriptionType (&\$subscriptionType) オブジェクトを作成した後、そのオブジェクトを呼び出し関数に返す前に呼び出されます。
TemplateManager::display	&\$templateMgr, &\$template, &\$sendContentType,	テンプレートマネージャが指定されたContentType (&\$sendContentType) とcharset (&\$charset) を持つContentTypeヘッダを送信し、テンプレート (&\$template) を

名前	引数	説明
	<code>&amp;\$charset</code>	表示する前に呼び出されます。OJSがこれを実行することを避けたい場合は、コールバック関数の戻り値をtrueにする必要があります。
UserDAO::_returnUserFromRow	<code>&amp;\$user,</code> <code>&amp;\$row</code>	UserDAOがデータベース行(&\$row)からUser(&\$user)オブジェクトを作成した後、そのオブジェクトを呼び出し関数に返す前に呼び出されます。
GroupDAO::_returnGroupFromRow	<code>&amp;\$group,</code> <code>&amp;\$row</code>	GroupDAOがデータベース行(&\$row)からGroup(&\$group)オブジェクトを作成した後、そのオブジェクトを呼び出し関数に返す前に呼び出されます。
GroupMembershipDAO::_returnMemberFromRow	<code>&amp;\$membership,</code> <code>&amp;\$row</code>	GroupMembershipDAOがデータベース行(&\$row)からGroupMembership(&\$membership)オブジェクトを作成した後、そのオブジェクトを呼び出し関数に返す前に呼び出されます。
Templates::About::Index::People	<code>&amp;\$params,</code> <code>&amp;\$templateMgr,</code> <code>&amp;\$output</code>	「この雑誌について」ページの組織セクションの<ul class="plain">...</ul>タグによる箇条書きリストの最後に呼び出されます。
Templates::About::Index::Policies	<code>&amp;\$params,</code> <code>&amp;\$templateMgr,</code> <code>&amp;\$output</code>	「この雑誌について」ページのポリシーセクションの<ul class="plain">...</ul>タグによる箇条書きリストの最後に呼び出されます。
Templates::About::Index::Submissions	<code>&amp;\$params,</code> <code>&amp;\$templateMgr,</code> <code>&amp;\$output</code>	「この雑誌について」ページの投稿物セクションの<ul class="plain">...</ul>タグによる箇条書きリストの最後に呼び出されます。
Templates::About::Index::Other	<code>&amp;\$params,</code> <code>&amp;\$templateMgr,</code> <code>&amp;\$output</code>	「この雑誌について」ページのその他セクションの<ul class="plain">...</ul>タグによる箇条書きリストの最後に呼び出されます。
Templates::Admin::Index::SiteManagement	<code>&amp;\$params,</code> <code>&amp;\$templateMgr,</code> <code>&amp;\$output</code>	サイト管理ページのサイト管理セクションの<ul class="plain">...</ul>タグによる箇条書きリストの最後に呼び出されます。
Templates::Admin::Index::AdminFunctions	<code>&amp;\$params,</code> <code>&amp;\$templateMgr,</code> <code>&amp;\$output</code>	サイト管理ページの管理機能セクションの<ul class="plain">...</ul>タグによる箇条書きリストの最後に呼び出されます。

名前	引数	説明
Templates::Editor: :Index::Submissions	&\$params, &\$templateMgr, &\$output	編集者ページの投稿物セクションの<ul class="plain">...</ul>タグによる箇条書きリストの最後に呼び出されます。
Templates::Editor: :Index::Issues	&\$params, &\$templateMgr, &\$output	編集者ページの巻号セクションの<ul class="plain">...</ul>タグによる箇条書きリストの最後に呼び出されます。
Templates::Manager ::Index::ManagementPages	&\$params, &\$templateMgr, &\$output	雑誌管理者ページの管理ページセクションの<ul class="plain">...</ul>タグによる箇条書きリストの最後に呼び出されます。
Templates::Manager ::Index::Users	&\$params, &\$templateMgr, &\$output	雑誌管理者ページのユーザセクションの<ul class="plain">...</ul>タグによる箇条書きリストの最後に呼び出されます。
Templates::Manager ::Index::Roles	&\$params, &\$templateMgr, &\$output	雑誌管理者ページの役割セクションの<ul class="plain">...</ul>タグによる箇条書きリストの最後に呼び出されます。
Templates::User::I ndex::Site	&\$params, &\$templateMgr, &\$output	ユーザホームの<ul class="plain">...</ul>タグで（もしあれば）サイト管理者リンクが表示された後に呼び出されます。
Templates::User::I ndex::Journal	&\$params, &\$templateMgr, &\$output	ユーザホームの各雑誌の役割セクションの<ul class="plain">...</ul>タグによる箇条書きリストの最後に呼び出されます。
Templates::Admin:: Index::MyAccount	&\$params, &\$templateMgr, &\$output	ユーザホームのマイ・アカウントセクションの<ul class="plain">...</ul>タグによる箇条書きリストの最後に呼び出されます。

## OJS の翻訳

提供されていない言語をサポートするには、次のディレクトリにある XML ファイルを翻訳し、しかるべき名前 (ja\_JP のような ISO ロケールコードを使うことが推奨される) のディレクトリに置く必要があります。

- locale/en\_US: このディレクトリは、地域化された OJS テキストの大部分を持つ主たるロケールファイルを含んでいます。
- dbscripts/xml/data/locale/en\_US: このディレクトリは、メールテンプレートなど地域化されたデータベースデータを含んでいます。
- help/en\_US: このディレクトリは、OJS のヘルプファイルを含んでいます
- registry/locale/en\_US: このディレクトリは国リストなど、その他の地域化された情報を含んでいます。
- rt/en\_US: このディレクトリは、読書ツールを含んでいます
- plugins/[プラグインカテゴリ]/[プラグイン名]/locale (必要な場合): これらのディレクトリはプラグイン固有のロケール文字列を含んでいます。

システムを正常に機能させるために最低限翻訳をする必要があるのは、locale/en\_US、dbscripts/xml/data/locale/en\_US、registry/locale/en\_US にあるファイルだけです。

新規ロケールは、サイト管理用の Web インターフェースを使ってシステムにインストールできるようにするために、registry/locales.xml ファイルに追加する必要があります。

翻訳を PKP に提供していただければ、将来の OJS リリースで配布することができます。



## 謝辞

Public Knowledge プロジェクトは、次のコミュニティメンバーの貢献に感謝いたします。

- Ramón Fonseca: ポルトガル語 (pt\_BR) への翻訳
- Sergio Ruiz Pérez: スペイン語 (es\_ES) への翻訳



## さらなる情報の入手

さらなる情報については、PKP の Web サイト <http://pkp.sfu.ca> をご覧ください。  
<http://pkp.sfu.ca/support/forum> には OJS のサポートフォーラムがあります。OJS チームに連絡を取りたい場合は、ここでお願いします。質問をする前には、既に回答されていないかフォーラムのアーカイブを十分検索してください。

バグの報告は、<http://pkp.sfu.ca/bugzilla> のバグ追跡システムをお願いします。

開発チームにはメール ([pkpsupport@sfu.ca](mailto:pkpsupport@sfu.ca)) でも連絡を取ることができます。