

プラグインよもやま話

2024.11.23

Yujiro Araki

全スライド共有OK!

2025.01.31 r2

- 自己紹介
- プラグイン基礎
- mt.cgiの仕組み
- プラグインが登録される仕組み
- 各種プラグインの定義と実装
 - コールバック
 - テンプレートタグ
 - グローバルモディファイア
 - テンプレート
 - トランスフォーマー
 - メニュー
 - リスティング・リストアクション
 - ウィジェット
 - パーミッション
 - スケジュールタスク
 - DataAPI
 - オーバーライド
 - ローカライズ
 - プラグインデータ・プラグイン設定
 - アプリケーション
 - データベースにテーブル・フィールド追加
- プラグイン資材
- デバッグ
- ノウハウ
- パフォーマンス
- 失敗談
- プラグイン・プラグイン開発関連サイト

自己紹介

2003年11月：ブログ「小粋空間」開始 (MT3.0D)
MTのプラグインやカスタイズ情報を公開中
<https://www.koikikukan.com/>

本業はIT企業 (通信事業系) に所属

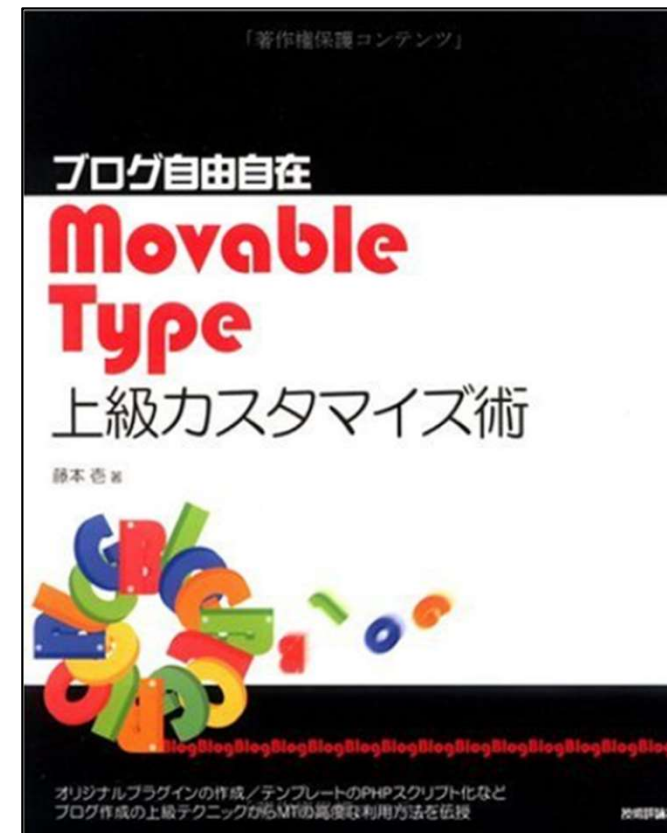


プラグインを作り始めたきっかけ

ブログ自由自在 Movable Type 上級カスタマイズ術

プラグイン作成前の状況

- テンプレートやカスタイズが主
- perl4が少し書ける程度



最初に作ったプラグイン

duplicateTBPingfilter（重複トラックバック防止） 2005年5月

- 同一エントリーに対する、同一URLからのトラックバックを受け付けない
- チェック方法：
トラックバック受信時、同一エントリーのトラックバック（の source_url ）を新着順に取り出し、受信したトラックバックの source_url と比較、同一URLであればフィルタリング
- 受信したトラックバックに含まれるタイトルまたは概要を含めたフィルタリングも可能
デフォルトではタイトル・概要が全く同じトラックバックのみをフィルタリング
- チェックするトラックバック数を指定可能
- トラックバックがフィルタリングされた場合、ログを出力

duplicateTBPingfilter.pl

```

package MT::Plugin::DuplicateTBPingFilter;
use strict;

our $title = 1;
our $excerpt = 1;
our $counter = 0;

require MT::Plugin;
my $plugin = new MT::Plugin({
    name => 'Duplicate Trackback Ping Filter Plugin 1.02',
    description => 'Filter duplicate trackback ping to the same entry.',
});

MT->add_plugin($plugin);
MT->add_callback('TBPingFilter', 11, 'duplicate trackback ping filter',
    \&duplicate_tbping_filter);

use MT::TBPing;
use MT::Trackback;

sub is_same_url {
    my ($new_url, $old_url) = @_;
    return ($new_url eq $old_url);
}

sub is_same_title {
    my ($new_title, $old_title) = @_;
    if (!$title) {
        return 1;
    } else {
        return ($new_title eq $old_title);
    }
}

```

```

sub is_same_excerpt {
    my ($new_excerpt, $old_excerpt) = @_;
    if (!$excerpt) {
        return 1;
    } else {
        return ($new_excerpt eq $old_excerpt);
    }
}

sub duplicate_tbping_filter {
    my ($eh, $app, $ping) = @_;
    my $ping_iter = MT::TBPing->load_iter({tb_id=>$ping->tb_id},
        {'sort' => 'created_on', direction =>
        'descend'});
    while (my $old_ping = $ping_iter->()) {
        if ($ping->id ne $old_ping->id) {
            if (&is_same_url($ping->source_url, $old_ping->source_url) &&
                &is_same_title($ping->title, $old_ping->title) &&
                &is_same_excerpt($ping->excerpt, $old_ping->excerpt)) {
                $app->log("Filter duplicate trackback ping from the following
                URL:" . $ping->source_url);
                return 0;
            }
        }
        if ($counter) {
            $counter--;
            if (!$counter) {
                return 1;
            }
        }
    }
    return 1;
}
1;

```

作ったプラグイン

約170

AdventCalendarNotifier
AlertNotifier
AllCategorySelector
AssetAppenderByUrl
AssetEntries
AssetExporter
AssetFolderViewer
AssetIdViewer
AssetPerPageLimitChanger
AssetRenameChnager
Assets
AssociationBlogDiscriminator
AuthorInformationEditController
AuthoredOnApdater
AvailableWidgetSorter
BlogIdViewer
BodyFieldCustomizer
BodyFieldEraser
BodyLengthListing
BrowserBackButtonEnabler
BulkSaveRevisioner
CacheExpireEventEnabler
CategoryArchiveCleaner
CategorySelectorFilter
CategoryTitleLinkChanger
ChangeCategoryCollectively
CharsetEncoder
CommentAuthorIdentity
CommentAutoReply
CommentCustomField
CommunityCommentEditor
CommunityEntryEditor
ConfigSetter

ContentFieldName
CreateBlogPermission
CSVAssetDataImporter
CSVAuthorDataImporter
CSVBlogDataImporter
CSVContenttypeImexporter
CSVDataImexporter
CurrentStyle
CustomFieldAssetDetailsHandler
CustomFieldDataChecker
CustomFieldDefinitionInexporter
CustomFieldEditor
CustomfieldExtensionTags
CustomizedCommentFilter
DataUpdateSuppressor
DateUsec
DefaultSortKeyChanger
DefaultValueSetter
DefinedSelectbox
DeleteAssetButton
DeleteCategoryRebuilder
DeleteExif
DeleteFileChecker
DeleteassetWithDeletecfasset
DispalyAuthorname
DraggableListTemplate
DuplicatePluginChecker
duplicateTBPingFilter
EmulateIe
EntryAndCommentDeleter
EntryAssetListing
EntryAssociaterWithUploadAsset
EntryBasenamelisting

EntryCategoryId
EntryConverter
EntryDeleter
EntryEditor
EntryEditorInExcel
EntryExporter
EntryIdViewer
EntryLinkByTemplateName
EntryRevision
EntryTextareabraker
ErrorlessRebuild
ExcelDataImporter
ExcerptRichtext
ExpandCategoryAreaCss
FfieldOptionResizer
ForceRecoverEntry
FuturePostRevisioner
GetArchives
GzipFilePublisher
ImagePreviewer
ImageRotator
ImageSizeGenerator
ImageSizeModifier
ImageTypeOptimizer
IndexTemplateIdentifier
InsertFileEnabler
InvalidateCodemirror
KeywordEntries
LastLoginDateTime
LimitPerPageChanger
ListCategoryFolder
ListingAuthoredon_changer
ListingCustomField

ListingFieldEditor
LogSupplementals
LoginTest
LogoChanger
MondayCalendar
MtTagBracketChanger
MultiPluginSwitcher
MultilineSearcher
Object
PageExporter
PagebuteCanonicalChanger
ParentCategoryRebuilder
ParentCategorySelector
PixenateTmpl
PostEntryLimiter
PowerEdit
PowerEditContentData
PowerEditEntrySorter
PowerListingFieldEditor
PreviewDraftEntries
PreviewFileSaver
PreviewTargetChanger
PublishButtonChanger
PublishDraft
PublishingPagesCounter
RebuildByChangeWidget
RebuildEntryById
RebuildIndexFilter
RebuildOrderChanger
RemoveTemporaryFilesStopper
ReplyCommenterAddressChanger
RevisableAsset
RevisableSaveEntries

RevisionDiff
RevisionRemove
SaveWithoutRebuild
ScreenChanger
SearchCategory
SearchConditionAdaptor
SearchConditionAppender
SearchConditionChanger
SearchDisabler
SearchResultsCfListing
SelectUsers
SelectboxBulkSetter
SessionTimeoutExtender
SettingExporter
ShowAssetImage
SiteStatsTermChanger
SortByTitle
SortCategories
SortFieldDisabler
Split
SystemCfAsset
TableOfContents
TemplatePreviewSelector
TextareaTabEnabler
TitleChanger
Trash
TwitComment
UnpublishdOnSaver
UploadImageResizer
Vote
Workflow
WorkflowDataapi

プラグイン基礎

プラグインの定義

config.yaml (YAML形式) を利用

- ID
 - 名前
 - バージョン
 - 説明
 - 作成者
 - 作成者のリンク
 - ドキュメントのリンク
- 等

config.yaml

```
id: Test
name: Test
version: 1.0
description: Test plugin
author_name: yujiro
author_link: https://www.koikikukan.com/
doc_link: https://www.koikikukan.com/plugins/Test/docs/
```

→ すべて書かなくてもそれなりに動く

簡単なプラグイン (1/3)

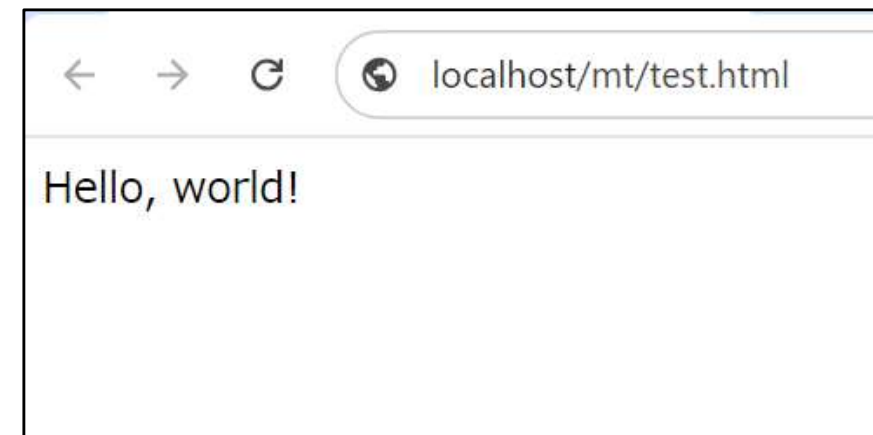
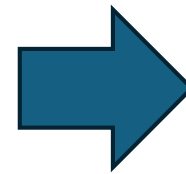
インデックステンプレートの作成

テンプレート名

テンプレートの内容

```
1 <$mt:HelloWorld$>
```

再構築

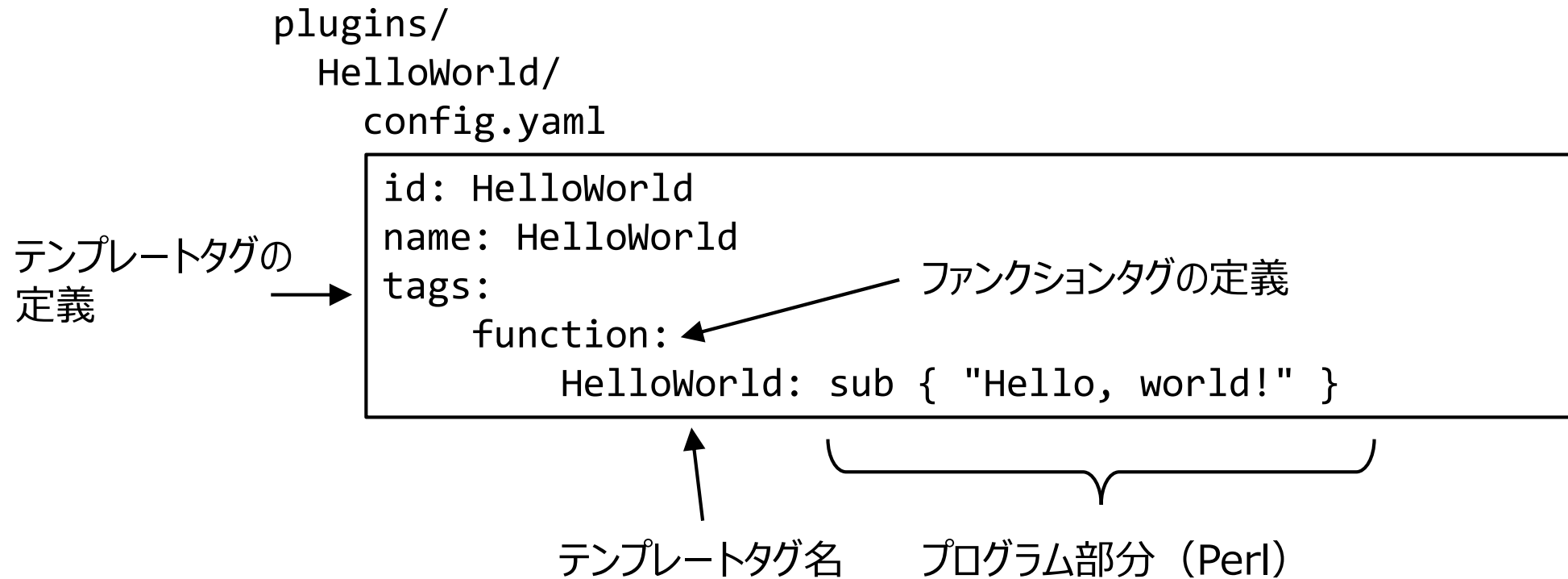


簡単なプラグイン (2/3)

```
plugins/  
  HelloWorld/  
    config.yaml
```

```
id: HelloWorld  
name: HelloWorld  
tags:  
  function:  
    HelloWorld: sub { "Hello, world!" }
```

簡単なプラグイン (3/3)



プラグイン構成を変更 (1/3)

```
plugins/  
  HelloWorld/  
    config.yaml
```

```
id: HelloWorld  
name: HelloWorld  
tags:  
  function:  
    HelloWorld: sub { "Hello, world!" }
```

プラグイン構成を変更 (2/3)

```
plugins/  
  HelloWorld/  
    config.yaml
```

```
id: HelloWorld  
name: HelloWorld  
tags:  
  function:  
    HelloWorld: $HelloWorld::HelloWorld::Tags::hdlr_hello_world
```

プラグイン構成を変更 (3/3)

```
plugins/  
  HelloWorld/  
    config.yaml
```

```
id: HelloWorld  
name: HelloWorld  
tags:  
  function:  
    HelloWorld: $HelloWorld::HelloWorld::Tags::hdlr_hello_world
```

```
lib/  
  HelloWorld/ ← パス  
    Tags.pm ← モジュール
```

lib配下を
パッケージ

```
package HelloWorld::Tags;  
sub hdlr_hello_world { ← メソッド  
  return "Hello, world!";  
}  
1;
```

\$HelloWorldが必要な理由 (1/2)

```
plugins/  
  HelloWorld/  
    config.yaml
```

```
id: HelloWorld  
name: HelloWorld  
tags:  
  function:  
    HelloWorld: $HelloWorld::HelloWorld::Tags::hdlr_hello_world
```



これ

\$HelloWorldが必要な理由 (2/2)

MT.pm

```
sub handler_to_coderef {
    my $pkg = shift;
    my ( $name, $delayed ) = @_;
    :

    my $code;
    if ( $name !~ m/->/ ) {
        no strict 'refs';
        $code = ¥&$name if defined &$name;
        return $code if $code;
    }
    my $component;
    if ( $name =~ m!^¥$! ) {
        if ( $name =~ s/^¥$(¥w+)::// ) {
            $component = $1;
        }
    }
    :
```

\$HelloWorldの「HelloWorld」を
コンポーネントとして取得

定義ファイルを“config.yaml”で配置する理由

MT.pm

```
sub _init_plugins_core {
    my $mt = shift;
    my ( $PluginSwitch, $use_plugins, $PluginPaths ) = @_;
    :
    foreach my $PluginPath ( @$PluginPaths ) {
        if ( opendir my $DH, $PluginPath ) {
            my @p = readdir $DH;
            closedir $DH;
            for my $plugin ( @p ) {
                :
                my $yaml = File::Spec->catdir( $plugin_full_path, 'config.yaml' );
                if ( -f $yaml ) {
                    my $obj = __load_plugin_with_yaml( $use_plugins,
                                                         $PluginSwitch,
                                                         $plugin_dir );

                    push @loaded_plugins, $obj if $obj;
                }
                next;
            }
        }
    }
    :
}
```

config.yamlを検索

「lib」ディレクトリが必要な理由 (1/3)

```
plugins/  
  HelloWorld/  
    config.yaml  
これ → lib/  
      HelloWorld/  
        Tags.pm
```

「lib」ディレクトリが必要な理由 (2/3)

MT.pm

```
sub _init_plugins_core {
    my $mt = shift;
    my ( $PluginSwitch, $use_plugins, $PluginPaths ) = @_;
        :
    my @loaded_plugins;
    foreach my $PluginPath ( @$PluginPaths ) {
        my $plugin_lastdir = $PluginPath;
        $plugin_lastdir =~ s![¥¥/]$!!;
        $plugin_lastdir =~ s!^.*[¥¥/]!!;

        if ( opendir my $DH, $PluginPath ) {
            my @p = readdir $DH;
            closedir $DH;
            for my $plugin ( @p ) {
                next if ( $plugin =~ /^¥.¥.?$ / || $plugin =~ /~$/ );
                $plugin_full_path = File::Spec->catfile( $PluginPath, $plugin );
                    :
                foreach my $lib ( qw(lib extlib) ) {
                    my $plib = File::Spec->catdir( $plugin_full_path, $lib );
                    unshift @INC, $plib if -d $plib;
                }
                    :
            }
        }
    }
}
```

libおよびextlibのパスを
@INC (Perlが読み込むパス) に追加

「lib」ディレクトリが必要な理由 (3/3)

@INCの中身

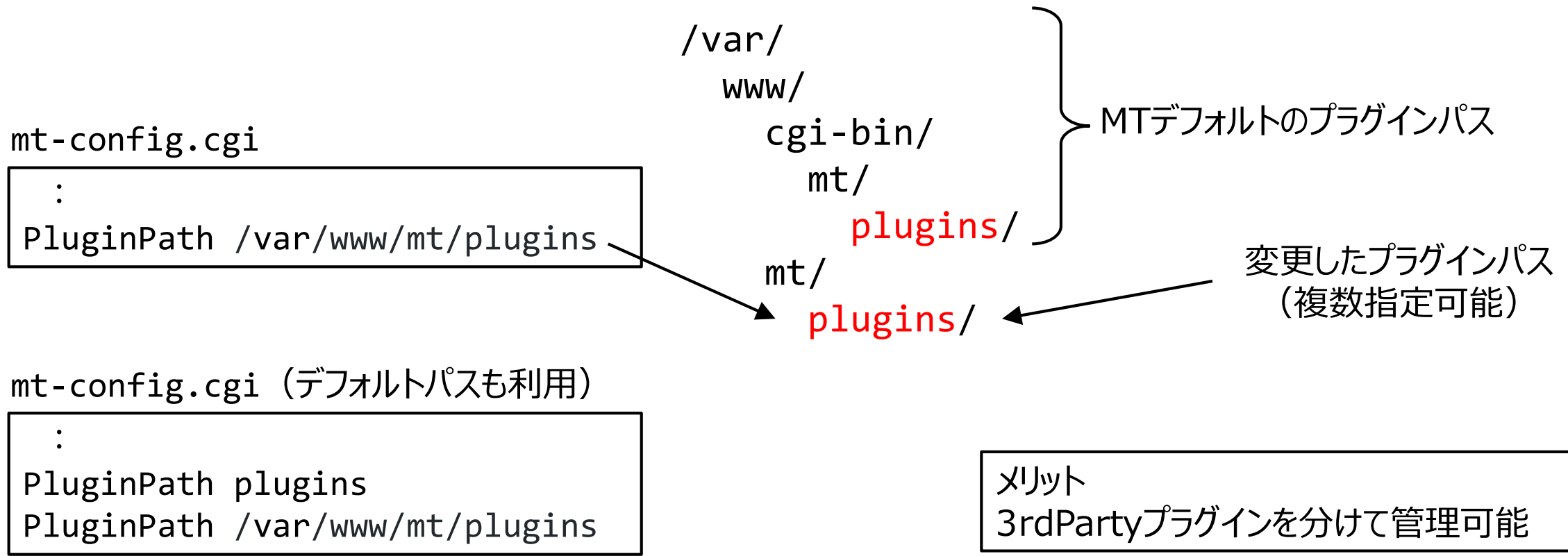
```
@INC = [  
  '/var/www/cgi-bin/mt/plugins/TinyMCE/lib',  
  '/var/www/cgi-bin/mt/plugins/FormattedTextForTinyMCE/lib',  
  '/var/www/cgi-bin/mt/plugins/TinyMCE5/lib',  
  '/var/www/cgi-bin/mt/plugins/HelloWorld/lib',  
  :  
  '/var/www/cgi-bin/mt/plugins/FormattedTextForTinyMCE5/lib',  
  '/var/www/cgi-bin/mt/addons/Commercial.pack/lib',  
  '/var/www/cgi-bin/mt/extlib',  
  'lib',  
  '/usr/local/lib64/perl5',  
  '/usr/local/share/perl5',  
  '/usr/lib64/perl5/vendor_perl',  
  '/usr/share/perl5/vendor_perl',  
  '/usr/lib64/perl5',  
  '/usr/share/perl5'  
];
```

extlibについて

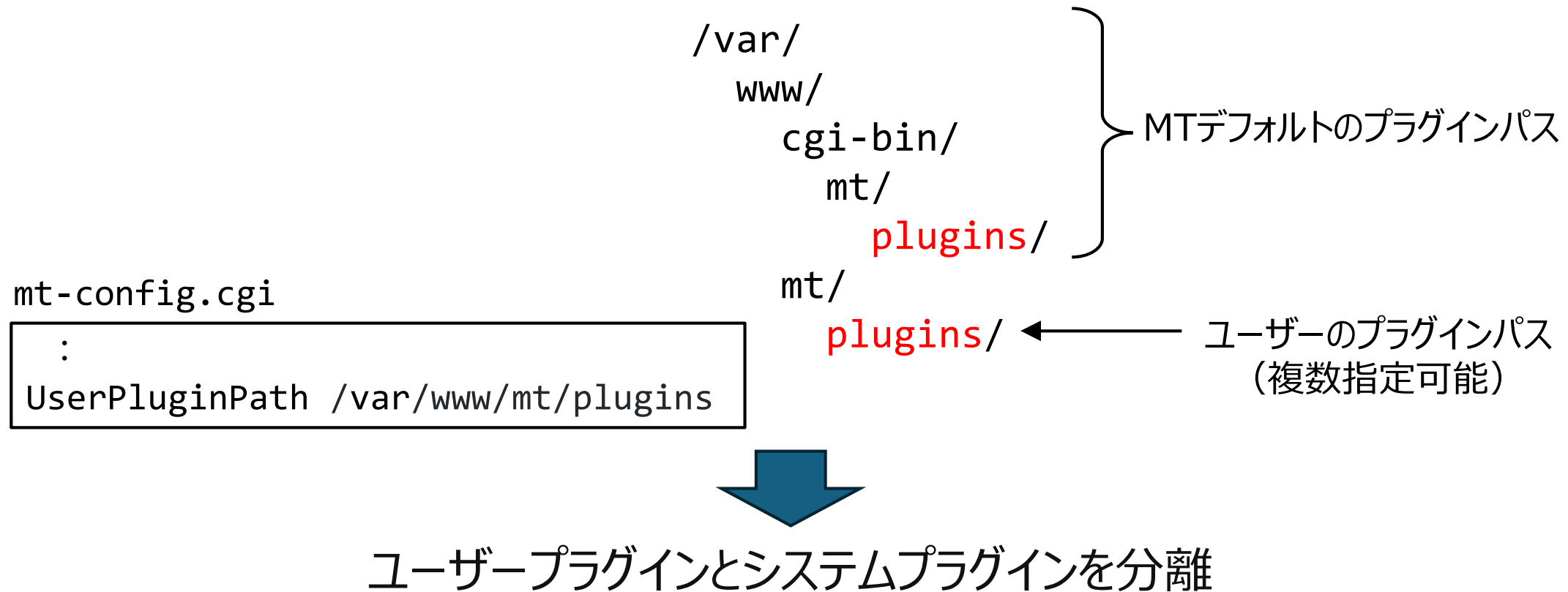
```
plugins/  
  HelloWorld/  
    config.yaml  
    lib/  
      HelloWorld/  
        Tags.pm
```

extlib/ ← Perlモジュールの追加が可能

PluginPath環境変数



UserPluginPath環境変数 (MT8)



mt.cgiの仕組み

CGIサンプル

sample.cgi

```
#!/usr/bin/perl  
  
print "Content-type: text/html ¥n¥n";  
  
print "Hello";
```

CGIサンプル2

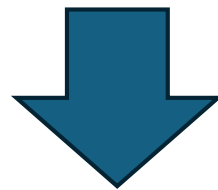
sample.cgi

```
#!/usr/bin/perl  
  
use CGI;  
my $query = CGI->new;  
print $query->header();  
print "Hello";
```

mt.cgiの仕組み (1/13)

mt.cgi

```
#!/usr/bin/env perl
:
use strict;
use lib $ENV{MT_HOME} ? "$ENV{MT_HOME}/lib" : 'lib';
use MT::Bootstrap App => 'MT::App::CMS';
```



赤字を書き換えると…

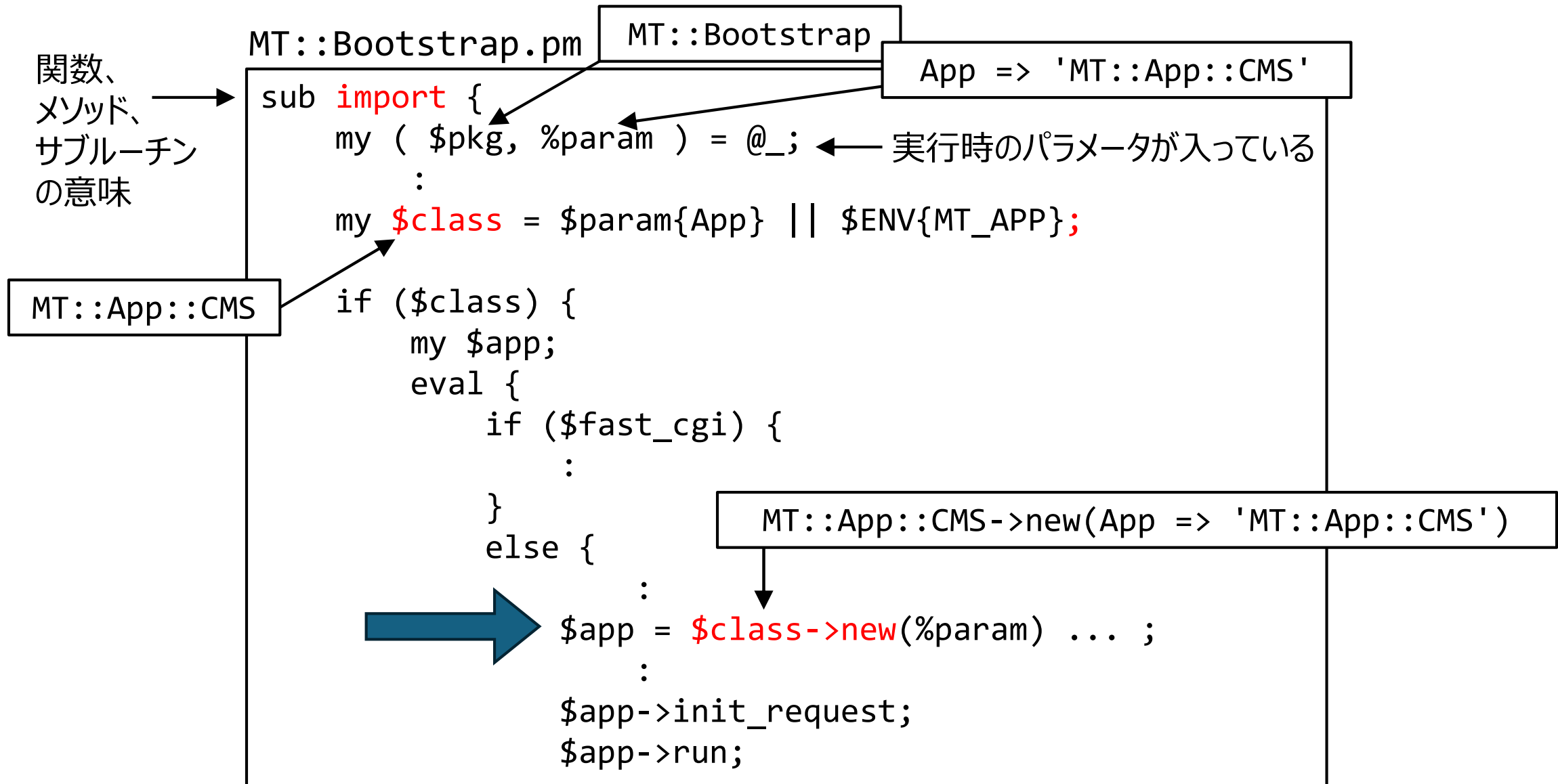
mt.cgiの仕組み (2/13)

mt.cgi

```
#!/usr/bin/env perl
:
use strict;
use lib $ENV{MT_HOME} ? "$ENV{MT_HOME}/lib" : 'lib';
BEGIN {
    require MT::Bootstrap;
    MT::Bootstrap->import( App => 'MT::App::CMS' );
}
```

MTの管理画面アプリケーション

mt.cgiの仕組み (3/13)



補足：アプリケーションごとに実行プログラムが変わる

mt-data-api.cgi

```
#!/usr/bin/env perl
:
use strict;
use lib $ENV{MT_HOME} ? "$ENV{MT_HOME}/lib" : 'lib';
use MT::Bootstrap App => 'MT::App::DataAPI';
```

補足 : mt-data-api.cgiの場合

MT::Bootstrap.pm

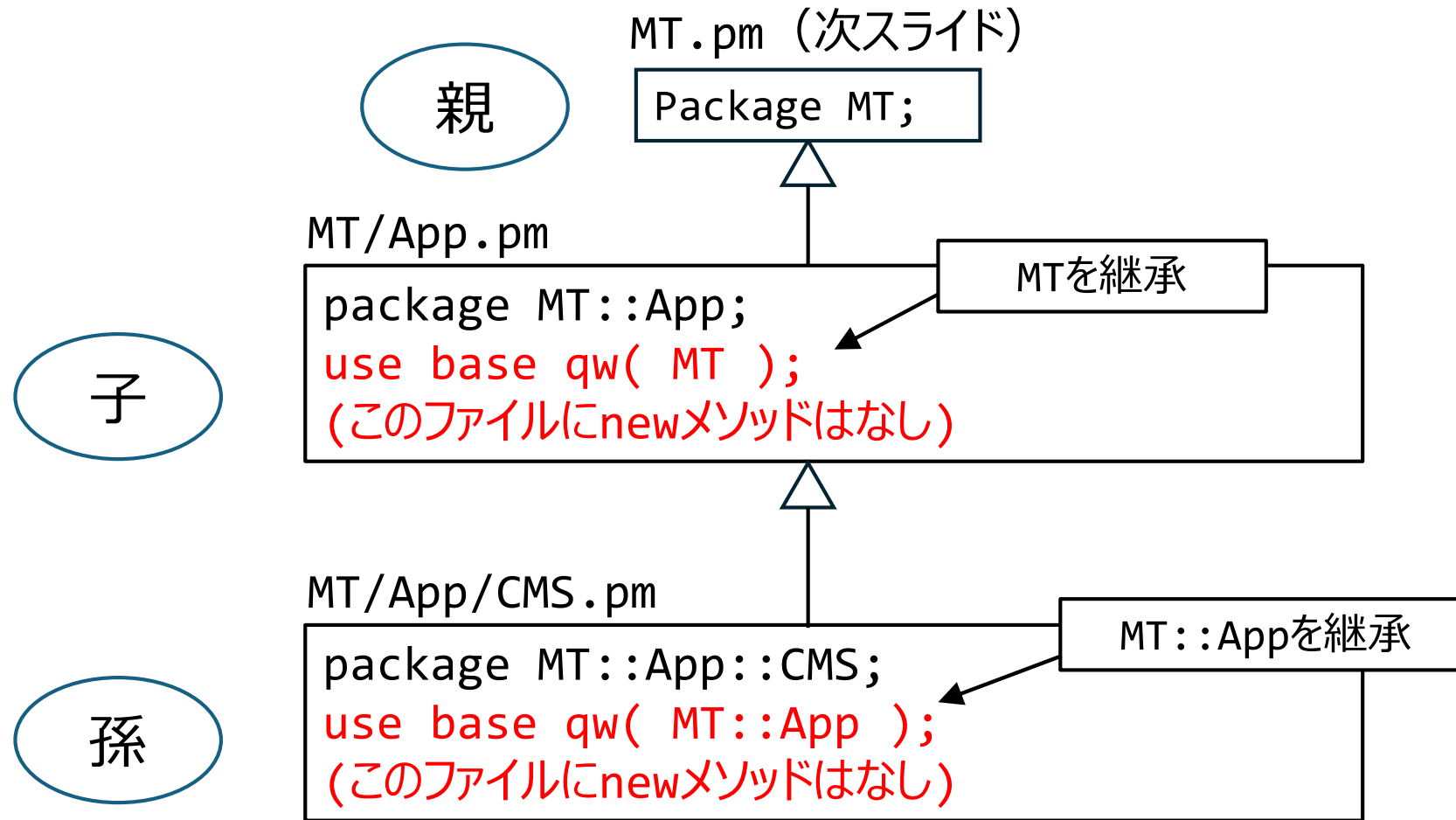
```
sub import {
  my ( $pkg, %param ) = @_ ;
  :
  my $class = $param{App} || $ENV{MT_APP};
  if ($class) {
    my $app;
    eval {
      if ($fast_cgi) {
        :
      }
      else {
        :
        $app = $class->new(%param) ... ;
        :
        $app->init_request;
        $app->run;
      }
    }
  }
}
```

App => 'MT::App::DataAPI'

MT::App::DataAPI

MT::App::DataAPI->new(App => 'MT::App::DataAPI')

mt.cgiの仕組み (4/13)



MT.pmはさらに継承しているが省略

mt.cgiの仕組み (5/13)

MT.pm

```
package MT;
:
sub new {
  my $mt = &instance_of;
  $mt_inst ||= $mt;
  $mt;
}
sub instance_of {
  my $class = shift;
  $mt_inst{$class} ||= $class->construct(@_);
}
sub construct {
  my $class = shift;
  my $mt = bless {}, $class;
  local $mt_inst = $mt;
  $mt->init(@_) or die $mt->errstr;
  $mt;
  @_ = ['App', 'MT::App::CMS'];
}
```

MT::App::CMS

MT::App::CMS
のオブジェクト

bless:リファレンスをオブジェクトとして機能するよう承認

@_ = ['App', 'MT::App::CMS'];

mt.cgiの仕組み (6/13)

MT/App/CMS.pm

```
sub init {  
    my $app = shift;  
    $app->SUPER::init(@_) or return;  
    $app->{state_params} = [  
        '_type',           'id',  
        'tab',             'offset',  
        'filter',         'filter_val',  
        'blog_id',        'is_power_edit',  
        'filter_key',     'type',  
        'content_type_id', 'content_data_id',  
    ];  
    $app->{template_dir}           = 'cms';  
    $app->{plugin_template_path} = '';  
    $app->{is_admin}               = 1;  
    $app->{default_mode}           = 'dashboard';  
    $app;  
}
```

子のメソッドを起動

mt.cgiの仕組み (7/13)

MT/App.pm

```
sub init {  
    my $app    = shift;  
    my %param  = @_;  
    :  
    $app->SUPER::init(%param) or return;  
    :  
    $app->{plugin_template_path} = 'tmpl';  
    $app->run_callbacks( 'init_app', $app, @_ );  
  
    if ( $MT::DebugMode & 128 ) {  
        MT->add_callback( 'pre_run', 1, $app, sub { $app->pre_run_debug } );  
        MT->add_callback( 'take_down', 1, $app,  
            sub { $app->post_run_debug } );  
    }  
    MT->add_callback( 'restore', 10, $app,  
        sub { MT->model('rebuild_trigger')->runner( 'post_restore', @_ ) } );  
  
    $app->{vtbl} = $app->registry("methods");  
    return $app;  
}
```

親のメソッドを起動

mt.cgiの仕組み (8/13)

MT.pm

```
sub init {
    my $mt    = shift;
    my %param = @_;

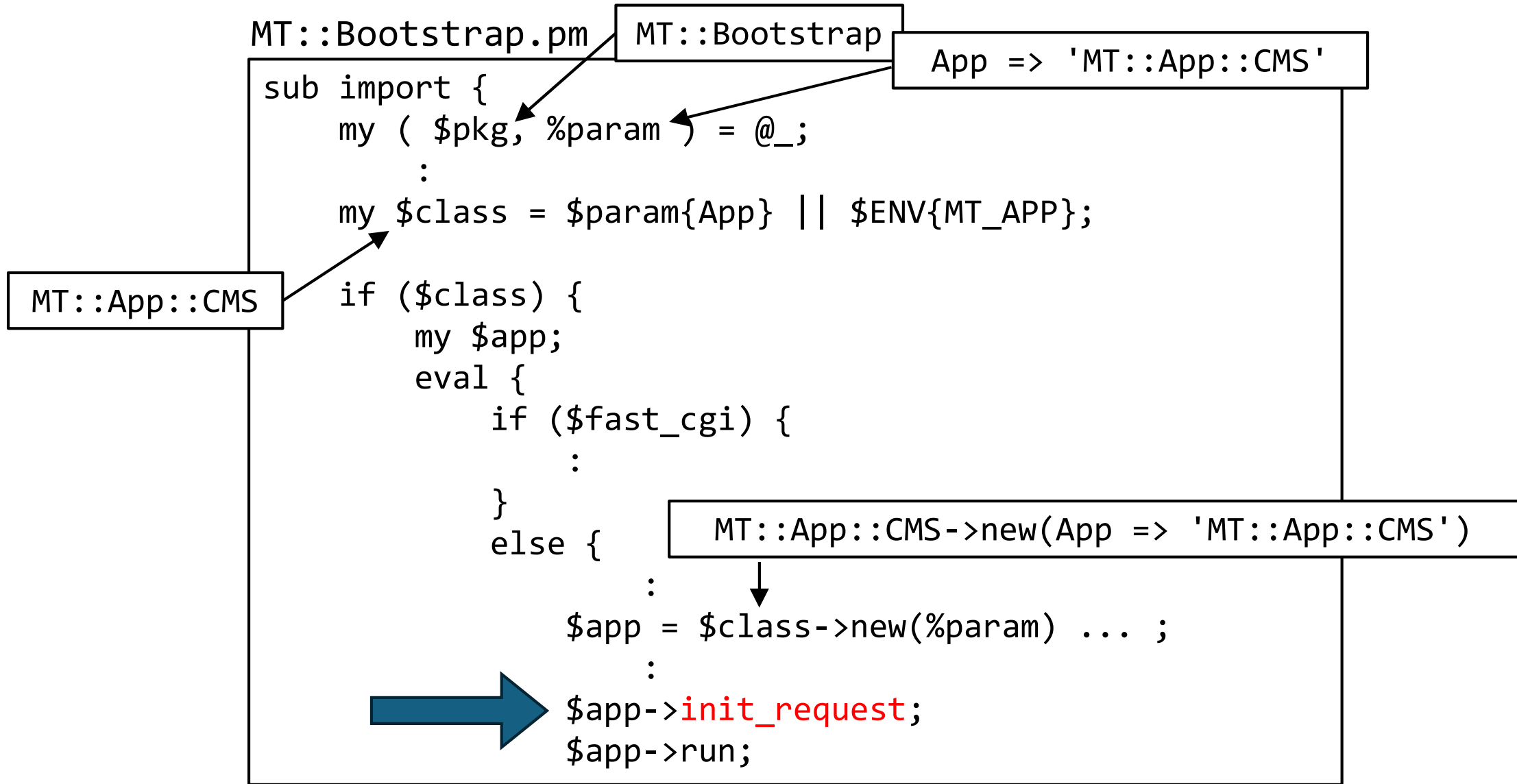
    $mt->bootstrap() unless $MT_DIR;
    $mt->{mt_dir}    = $MT_DIR;
    $mt->{config_dir} = $CFG_DIR;
    $mt->{app_dir}   = $APP_DIR;

    $mt->init_callbacks();

    ## Initialize the language to the default in case any errors occur in
    ## the rest of the initialization process.
    $mt->init_config( %param ) or return;
    $mt->init_lang_defaults(@_) or return;
    require MT::Plugin;
    $mt->init_addons(@_) or return;
    $mt->init_config_from_db( %param ) or return;
    $mt->init_debug_mode;
    $mt->init_plugins(@_) or return;
    $plugins_installed = 1;
    $mt->init_schema();
    $mt->init_permissions();
    :
    $mt->run_callbacks( 'post_init', $mt, %param );
    :
    return $mt;
}
```

もろもろ初期化

mt.cgiの仕組み (9/13)



mt.cgiの仕組み (10/13)

スライド26

MT/App.pm

```
sub init_request {
  my $app = shift;
  my %param = @_;
  :
  require CGI;
  $CGI::POST_MAX = $app->config->CGIMaxUpload;
  $app->{query} = CGI->new( $app->{no_read_body} ? {} : ( ) );
  :
```

```
#!/usr/bin/perl

use CGI;
my $query = CGI->new;
print $query->header();
print "Hello";
```

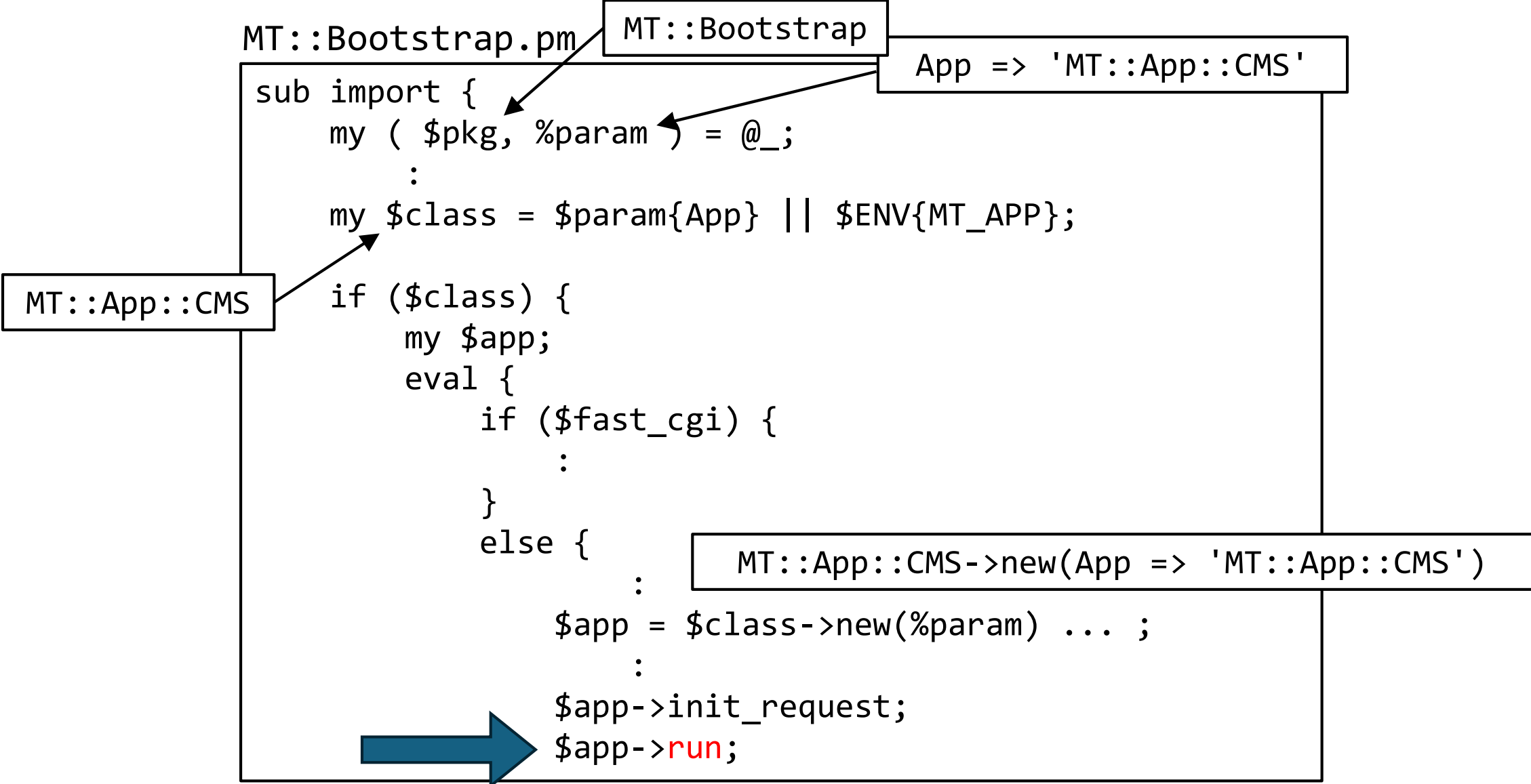
MT/App/CMS.pm

```
sub init_request {
  my $app = shift;
  $app->SUPER::init_request(@_);
  :
```

MT::Appのinit_requestを起動



mt.cgiの仕組み (11/13)



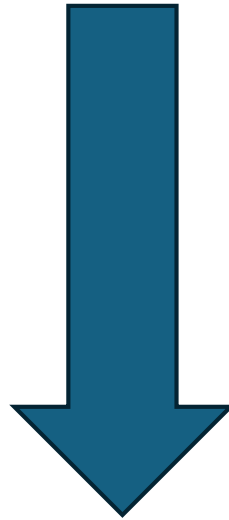
mt.cgiの仕組み (12/13)

MT/App/CMS.pm

```
package MT::App::CMS;
```

```
use base qw( MT::App );
```

(このファイルにrunメソッドはなし)



mt.cgiの仕組み (13/13)

MT/App.pm

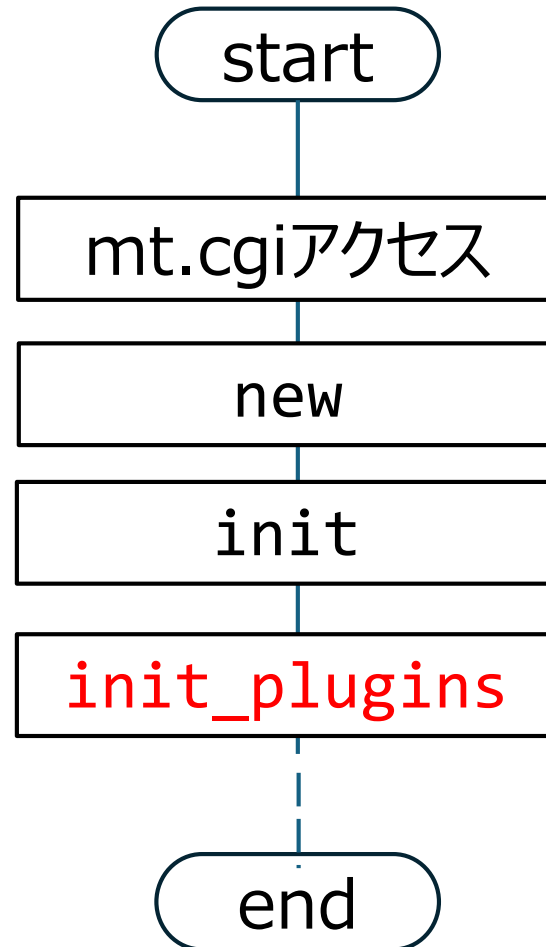
```
sub run {
    my $app = shift;
    my $q    = $app->param;
    :
    my ($body);
    :
    $app->pre_run;
    foreach my $code (@handlers) {
        my $local_component;
        if ( ref $code eq 'HASH' ) {
            my $meth_info = $code;
            $code = $meth_info->{code} || $meth_info->{handler};
        }
        if ( ref $code ne 'CODE' ) {
            $code = $app->handler_to_coderef($code);
        }
        if ($code) {
            my $content = $code->( $app, @forward_params );
            $app->response_content($content) if defined $content;
        }
    }
    $app->post_run;
    :
    $body = $app->response_content();
    :
    $app->send_http_header;
    :
    $app->print_encode($body);
}
```

MTプログラム起動

HTMLレスポンス出力

プラグインが登録される仕組み

プラグインが登録される仕組み (1/14)



プラグインが登録される仕組み (2/14)

MT.pm

```
sub init {
  my $mt      = shift;
  my %param = @_;

  $mt->bootstrap() unless $MT_DIR;
  $mt->{mt_dir}      = $MT_DIR;
  $mt->{config_dir} = $CFG_DIR;
  $mt->{app_dir}    = $APP_DIR;
  $mt->init_callbacks();
  $mt->init_config( ¥%param ) or return;
  $mt->init_lang_defaults(@_) or return;
  require MT::Plugin;
  $mt->init_addons(@_) or return;
  $mt->init_config_from_db( ¥%param ) or return;
  $mt->init_debug_mode;
  $mt->init_plugins(@_) or return;
  $plugins_installed = 1;
  :
  $mt->run_callbacks( 'post_init', $mt, ¥%param );
  :
  return $mt;
}
```

スライド36

アドオンの初期化

プラグインの初期化 (MT::App::CMS、MT::App::DataAPIなど)

プラグインが登録される仕組み (3/14)

MT/App/CMS.pm

```
sub init_plugins {
    my $app = shift;

    MT::App::CMS::Common::init_core_callbacks($app);
    →スライド46

    my $pkg = $app->id . '_';
    my $pfx = '$Core::MT::CMS::';
    $app->_register_core_callbacks(
        {
            $pkg . 'pre_load_filtered_list.entry' =>
                "${pfx}Entry::cms_pre_load_filtered_list",
            $pkg . 'view_permission_filter.entry' => "${pfx}Entry::can_view",
            :
        }
    );
    $app->SUPER::init_plugins(@_);
    →スライド47
}
```

MT/App.pmにinit_pluginsは実装されていないので、
MT.pmのinit_pluginsへ

プラグインが登録される仕組み (4/14)

MT/App/CMS/Common.pm

```
sub init_core_callbacks {  
    my $app = shift;  
    my $pkg = $app->id . '_';  
    my $pfx = '$Core::MT::CMS::';  
    $app->_register_core_callbacks(  
        {  
            # notification callbacks  
            $pkg  
                . 'save_permission_filter.notification' =>  
                "${pfx}AddressBook::can_save",  
            $pkg  
                . 'delete_permission_filter.notification' =>  
                "${pfx}AddressBook::can_delete",  
            :  
        }  
    );  
}
```

コア機能のコールバックを登録

プラグインが登録される仕組み (5/14)

MT.pm

```
sub init_plugins {  
  my $mt = shift;  
  :  
  my $cfg = $mt->config;  
  my $use_plugins = $cfg->UsePlugins; スライド23  
  my @PluginPaths = ($cfg->UserPluginPath, $cfg->PluginPath);  
  my $PluginSwitch = $cfg->PluginSwitch || {}; スライド22  
  my $plugin_sigs = join ', ', sort keys %$PluginSwitch;  
  $mt->_init_plugins_core( $PluginSwitch, $use_plugins, ¥@PluginPaths );  
  :
```

プラグイン関連のコンフィグを
mt-config.cgiから取得

プラグインの初期化

プラグインが登録される仕組み (6/14)

MT.pm

```
sub _init_plugins_core {
    my $mt = shift;
    my ( $PluginSwitch, $use_plugins, $PluginPaths ) = @_;
    :
    foreach my $PluginPath (@$PluginPaths) {
        if ( opendir my $DH, $PluginPath ) {
            my @p = readdir $DH;
            closedir $DH;
            for my $plugin (@p) {
                :
                my $yaml = File::Spec->catdir( $plugin_full_path, 'config.yaml' );
                if ( -f $yaml ) {
                    my $obj = __load_plugin_with_yaml( $use_plugins,
                                                        $PluginSwitch,
                                                        $plugin_dir );
                    push @loaded_plugins, $obj if $obj;
                }
            }
        }
    }
    :
```

プラグインディレクトリを検索

config.yamlを検索

YAML定義のプラグインをロード

プラグインが登録される仕組み (7/14)

MT.pm

```
sub __load_plugin_with_yaml {
  my ( $use_plugins, $PluginSwitch, $plugin_dir ) = @_;
  my $pclass
    = $plugin_dir =~ m/¥.pack$/
    ? 'MT::Component'
    : 'MT::Plugin';
    :
  my $id = lc $plugin_dir;
  $id =~ s/¥.¥w+$/;
  my $p = $pclass->new(
    {
      id      => $id,
      path    => $plugin_full_path,
      envelope => $plugin_envelope
    }
  );
  local $plugin_sig = $plugin_dir;
  $PluginSwitch->{$plugin_sig} = 1;
  MT->add_plugin($p);
  return $p;
}
```

プラグインディレクトリを
IDとして取得・設定

ディレクトリに”.pack”が付与されていれば
アドオンとして登録

プラグインオブジェクトの生成

プラグインオブジェクトをMTに登録

プラグインが登録される仕組み (8/14)

MT.pm

```

sub add_plugin {
  my $class = shift;
  my ($plugin) = @_;
  :
  $plugin->{name} ||= $plug
  $plugin->{plugin_sig} = $

  my $id = $plugin->id;
  :
  $Components{ lc $id } = $plugin if $id;
  $Plugins{$plugin_sig}{object} = $plugin;
  $plugin->{full_path} = $plugin_full_path;
  $plugin->path($plugin_full_path);
  unless ( $plugin->{registry} && ( %{ $plugin->{registry} } ) ) {
    $plugin->{registry} = $plugin_registry;
  }
  push @Components, $plugin;
  1;
}

```

```

$plugin = bless( {
  'plugin_sig' => 'HelloWorld',
  'registry' => undef,
  'name' => 'HelloWorld',
  'full_path' => '/var/www/cgi-bin/mt/plugins/HelloWorld',
  'envelope' => 'plugins/HelloWorld',
  'path' => '/var/www/cgi-bin/mt/plugins/HelloWorld',
  'id' => 'helloworld'
}, 'MT::Plugin' );

```

コンポーネントにプラグインを登録

プラグインが登録される仕組み (9/14) レジストリ

MT.pm スライド48のつづき

```
(sub _init_plugins_core {
    :
    # Drop conflicting plugins
    :
    for my $plugin (values %deduped_plugins) {
        if ($plugin->isa('MT::Plugin')) {
            $plugin->init;
        }
        if ($plugin->{registry}) {
            if (my $settings = $plugin->{registry}{config_settings}) {
                $settings = $plugin->{registry}{config_settings} = $settings->()
                if ref($settings) eq 'CODE';
                $mt->config->define($settings) if $settings;
            }
        }
        $plugin->init_callbacks;
    }
}
```

重複プラグインは削除

プラグインのレジストリ登録等

プラグインのコールバック登録 (後述)

プラグインのコンフィグ設定

プラグインが登録される仕組み (10/14) レジストリ

MT/Plugin.pm

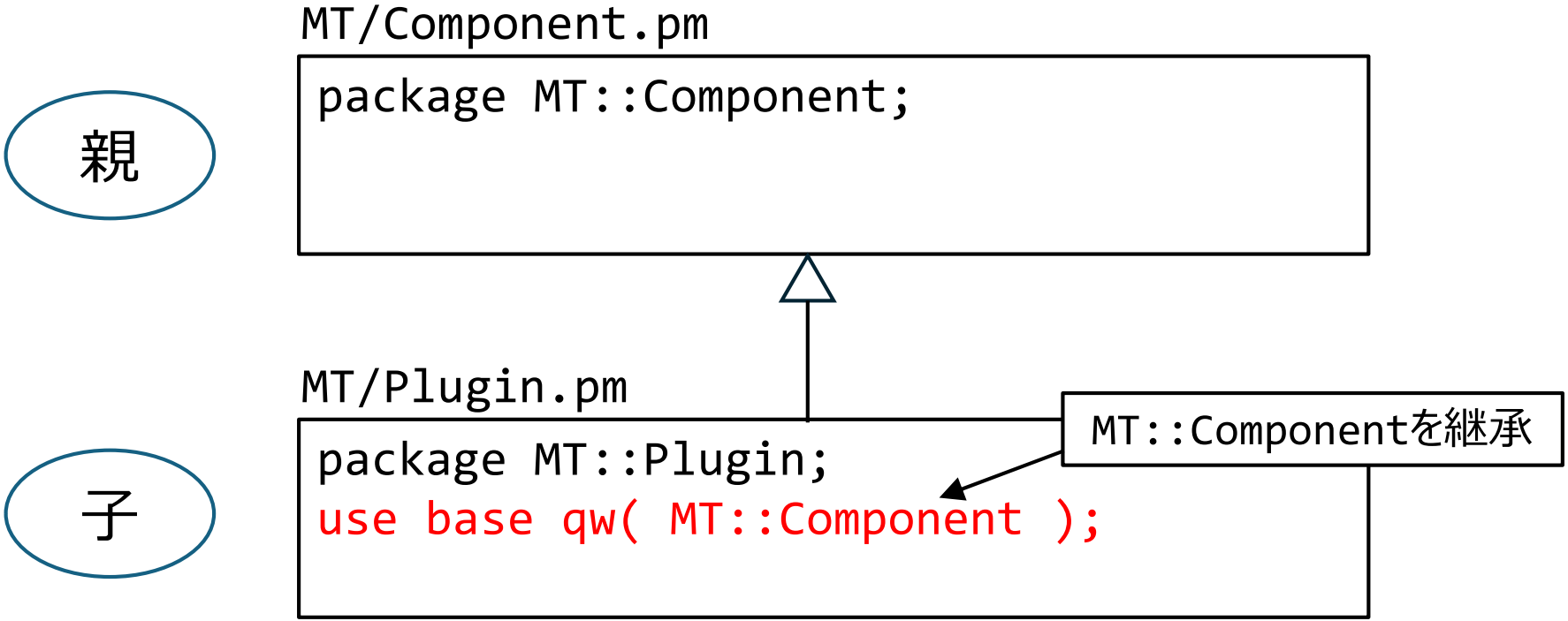
```
use base qw( MT::Component );

sub init {
    my $plugin = shift;
    $plugin->{__settings} = {};
    :
    $plugin->SUPER::init(@_) or return;
    return $plugin;
}
```

親のメソッドを起動



プラグインが登録される仕組み (11/14) レジストリ



MT/Component.pmはさらに継承しているが省略

プラグインが登録される仕組み (12/14) レジストリ

MT/Component.pm

```
sub init {
    my $c = shift;
    $c->init_registry() or return;
    $c;
}

sub init_registry {
    my $c = shift;
    my $r = $c->load_registry("config.yaml");
    if ( !$r ) {
        return 1;
    }
    $c->registry($r);

    foreach my $prop (qw(version schema_version)) {
        $c->$prop( $r->{$prop} ) if exists $r->{$prop};
    }
    $c->name( $r->{label} ) if exists $r->{label};
    return 1;
}
```

config.yamlの読み込み

コンポーネントにレジストリ設定

プラグインバージョンと
スキーマバージョン設定

プラグインが登録される仕組み (13/14) レジストリ

MT/Component.pm

```
sub load_registry {  
    my $c      = shift;  
    my ($file) = @_;  
    my $path   = $c->path or return;  
    $path = File::Spec->catfile( $c->path, $file );  
    return unless -f $path;  
    require MT::Util::YAML;  
    my $y = eval { MT::Util::YAML::LoadFile($path) }  
        or die "Error reading $path: "  
            . ( MT::Util::YAML->errstr || $@ || $! );  
    return $y;  
}
```

YAML形式のデータ読み込み

プラグインが登録される仕組み (14/14) レジストリ

赤色部分 (レジストリ) が追加される

スライド50のデータ

```
$plugin = bless( {  
  'plugin_sig' => 'HelloWorld',  
  'registry' => undef,  
  'name' => 'HelloWorld',  
  'full_path' => '/var/www/cgi-bin/mt/plugins/HelloWorld',  
  'envelope' => 'plugins/HelloWorld',  
  'path' => '/var/www/cgi-bin/mt/plugins/HelloWorld',  
  'id' => 'helloworld'  
}, 'MT::Plugin' );
```

```
$plugin = bless( {  
  'plugin_sig' => 'HelloWorld',  
  'registry' => {  
    'name' => 'HelloWorld',  
    'id' => 'HelloWorld',  
    'tags' => {  
      'function' => {  
        'HelloWorld' => 'sub { "Hello, world!" }'  
      }  
    }  
  },  
  'name' => 'HelloWorld',  
  'full_path' => '/var/www/cgi-bin/mt/plugins/HelloWorld',  
  'envelope' => 'plugins/HelloWorld',  
  'path' => '/var/www/cgi-bin/mt/plugins/HelloWorld',  
  'id' => 'helloworld'  
}, 'MT::Plugin' );
```


レジストリとは

- Movable Typeの情報を階層的に保持するための仕組み
- プラグインの定義情報はレジストリに登録
- コア機能もレジストリを持っている ↓

MT/Core.pm

```
:
BEGIN {
    $core_registry = {
        version          => MT->VERSION,
        schema_version => MT->schema_version,
        object_drivers => {
            'mysql' => {
                label          => 'MySQL Database (Recommended)',
                dbd_package    => 'DBD::mysql',
                config_package => 'DBI::mysql',
                :
            }
        }
    }
}
```

各種プラグインの定義と実装

プラグインの主な種類

MTで用意されているもの

MTで用意されていないもの

プラグイン定義 (config.yaml)
で実現できるもの

- メニュー
- パーミッション
- リスティング
- フィールド追加
- :

プログラムが必要なもの

- コールバック
- テンプレートタグ
- グローバルモディファイア
- テンプレート
- トランスフォーマー
- スケジュールタスク
- リストアクション
- ウィジェット
- DataAPI
- アプリケーション追加
- テーブル追加
- :

共通的なもの

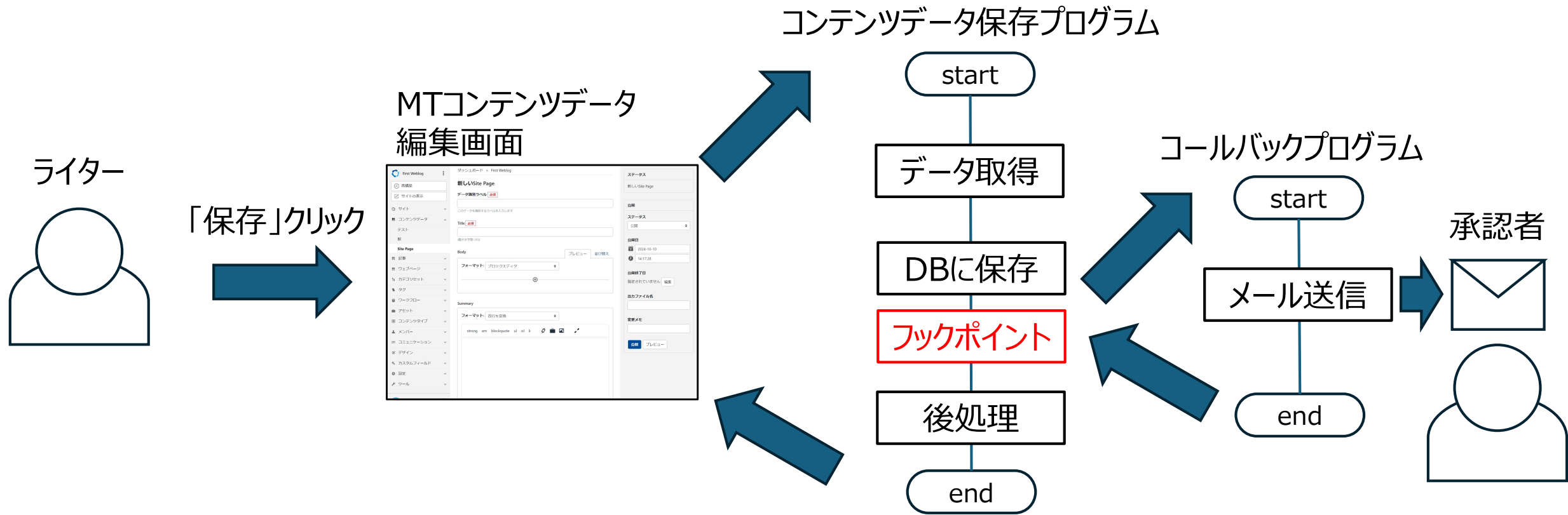
- ローカライズ
- プラグイン設定画面
- 資材 (画像・CSS・JavaScript等)

- コンテンツアクション
- その他もろもろ

コールバック

コールバック概要 (1/4)

Workflowプラグインで、コンテンツデータ保存後、承認者にメールを送信



コールバック概要 (2/4)

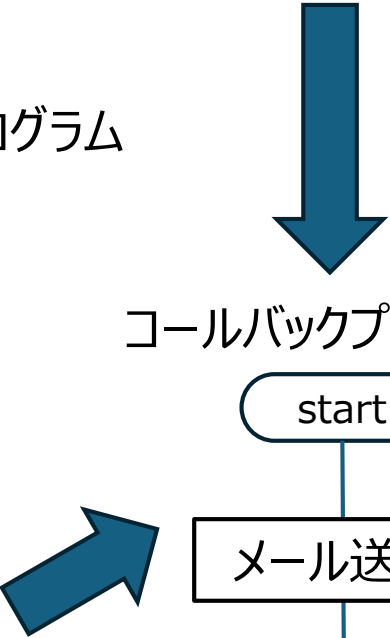
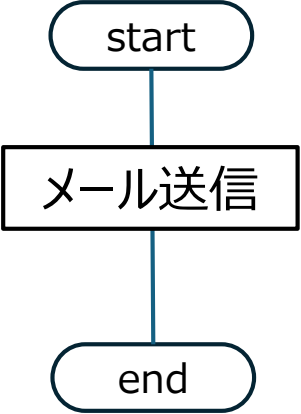
- ①コールバック定義
- ②コールバックプログラム実装
- ③コールバック登録

コンテンツデータ保存プログラム

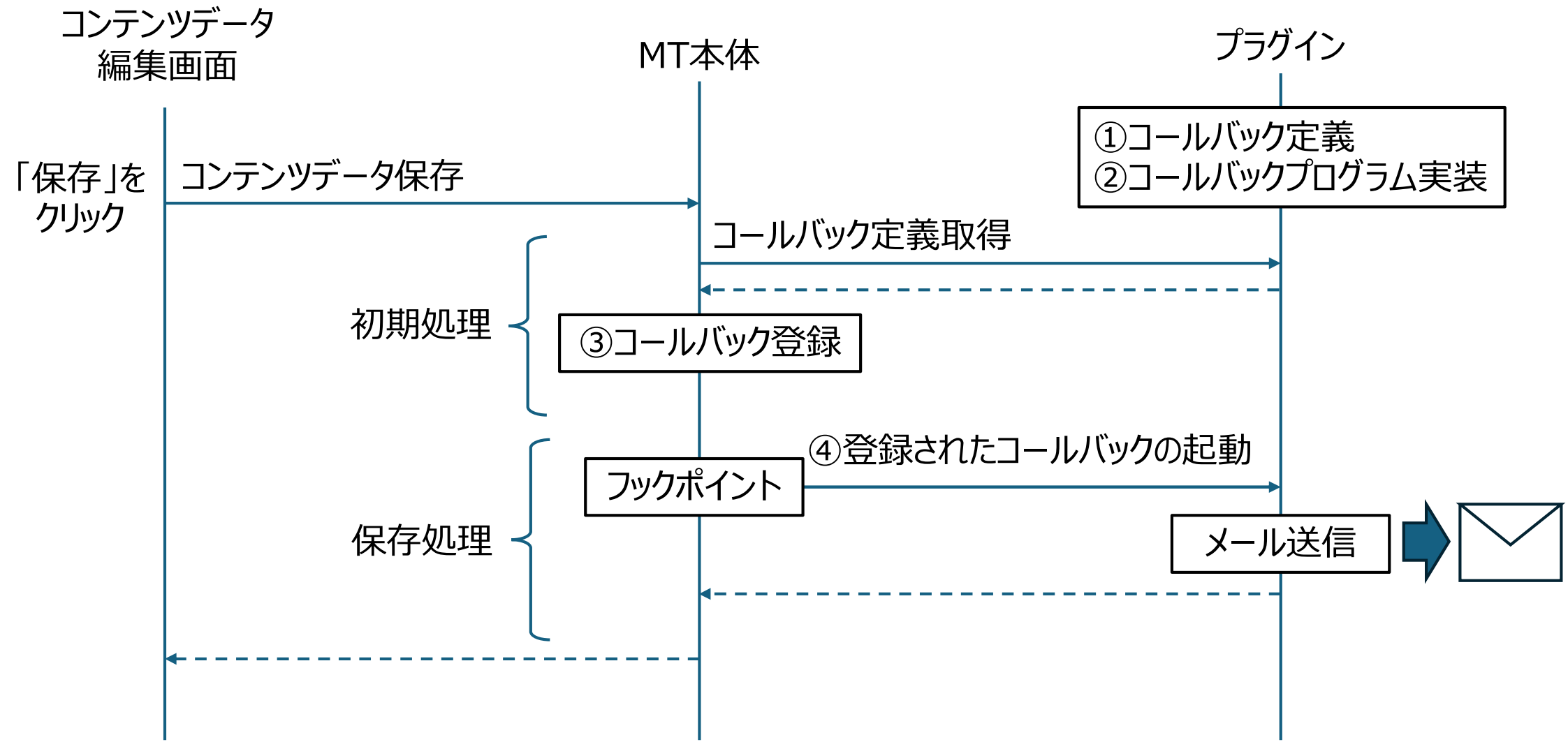


④登録されたコールバックの起動

コールバックプログラム



コールバック概要 (3/4)



コールバック概要 (4/4)

- ひとつのコールバックを複数のプラグイン（コア機能内部からの呼び出し含む）が利用できる
- 同一のコールバックを利用するときはプライオリティに注意

① プラグインにコールバック定義

config.yaml

```
  :  
  callbacks:  
    cms_post_save.content_data: $Workflow::Workflow::App::CMS::post_save
```

フックポイント名

起動させたいコールバックプログラム

②コールバックプログラムの実装

Workflow/lib/Workflow/App/CMS.pm

```
sub post_save {  
    my ($eh, $app, $obj, $orig_obj) = @_;  
    :  
    my $q      = $app->param;  
    my $type   = $q->param('_type');  
    :  
    if ( $type eq 'content_data' ) {  
        _send_mail_cd($app, $obj, $old_status);  
    } else {  
        _send_mail($app, $obj, $old_status);  
    }  
}
```

メール送信

③コールバックプログラム追加

MT/Component.pm

```
sub init_callbacks {
    my $c = shift;
    :
    if ( my $callbacks = $c->callbacks ) {
        if ( ref $callbacks eq 'ARRAY' ) {
            :
            elsif ( ref $callbacks eq 'HASH' ) {
                foreach my $cbname ( keys %$callbacks ) {
                    if (ref $callbacks->{$cbname} eq 'CODE'
                        || ( ref $callbacks->{$cbname} eq ''
                            && $callbacks->{$cbname} )
                    )
                    {
                        MT->add_callback( $cbname, 5, $c, $callbacks->{$cbname} );
                    }
                }
            }
        }
    }
}

```

`$c->callbacks`

```
{
    'MT::App::CMS::template_param.edit_content_data' => '$HelloWorld::HelloWorld::CMS::hdlr_template_param_edit_content_data',
    'MT::App::CMS::template_source.edit_content_data' => '$HelloWorld::HelloWorld::CMS::hdlr_template_source_edit_content_data',
    'frobmitzes' => '$HelloWorld::HelloWorld::CMS::frobmitzes',
    'init_requet' => '$HelloWorld::HelloWorld::CMS::init_request'
}

```

③コールバックプログラム追加 (MTコア)

MT/Object.pm

```
sub add_callback {
  my $class = shift;
  my $meth  = shift;
  MT->add_callback(
    $class . '::' . $meth, @_ );
}
```

MT::ContentData::post_remove

MT.pm

```
sub add_callback {}
```

cms_post_save.content_data

MT/ContentData.pm

post_remove

```
use base qw( MT::Object );
:
MT->add_callback( 'cms_post_save.content_data', 10, MT->component('core'),
  sub { MT->model('rebuild_trigger')->runner( 'post_content_save', @_ ); }
);
MT::ContentData
__PACKAGE__->add_callback(
  'post_remove',
  5, MT->component('core'), ... );
```

App.pm

```
MT->add_callback( 'post_save', 0, $app,
  ¥&_cb_mark_blog );
```

③コールバックプログラム追加

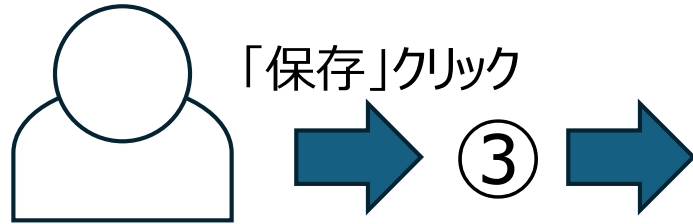
MT.pm

```
sub add_callback {
  my $class = shift;
  my ( $meth, $priority, $plugin, $code ) = @_;
  :
  if ( $priority == 0 || $priority == 11 ) {
    if ( $Callbacks[$priority]->{$meth} ) {
      return $class->trans_error("Two plugins are in conflict");
    }
  }
  :
  require MT::Callback;
  $CallbacksEnabled{$meth} = 1;
  my $cb = MT::Callback->new(
    plugin    => $plugin,
    code      => $code,
    priority  => $priority,
    internal  => $internal,
    method    => $meth
  );
  push @{$Callbacks[$priority]->{$meth}}, $cb;
  $cb;
}
```

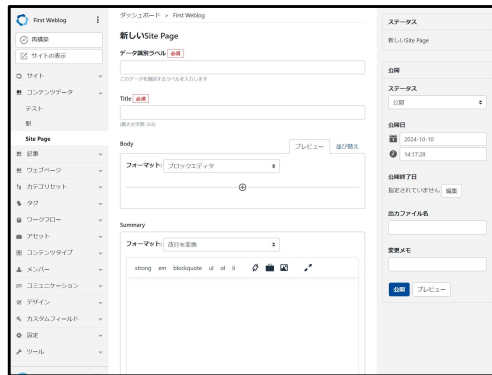
優先度「0」と「11」は重複不可

コールバックを優先度別に登録

④コールバックの起動



MTコンテンツデータ
編集画面



MT/CMS/ContentData.pm

```
sub save {
  my ($app) = @_ ;
  :
  $app->run_callbacks( 'cms_pre_save.content_data',
    $app, $content_data, $orig );

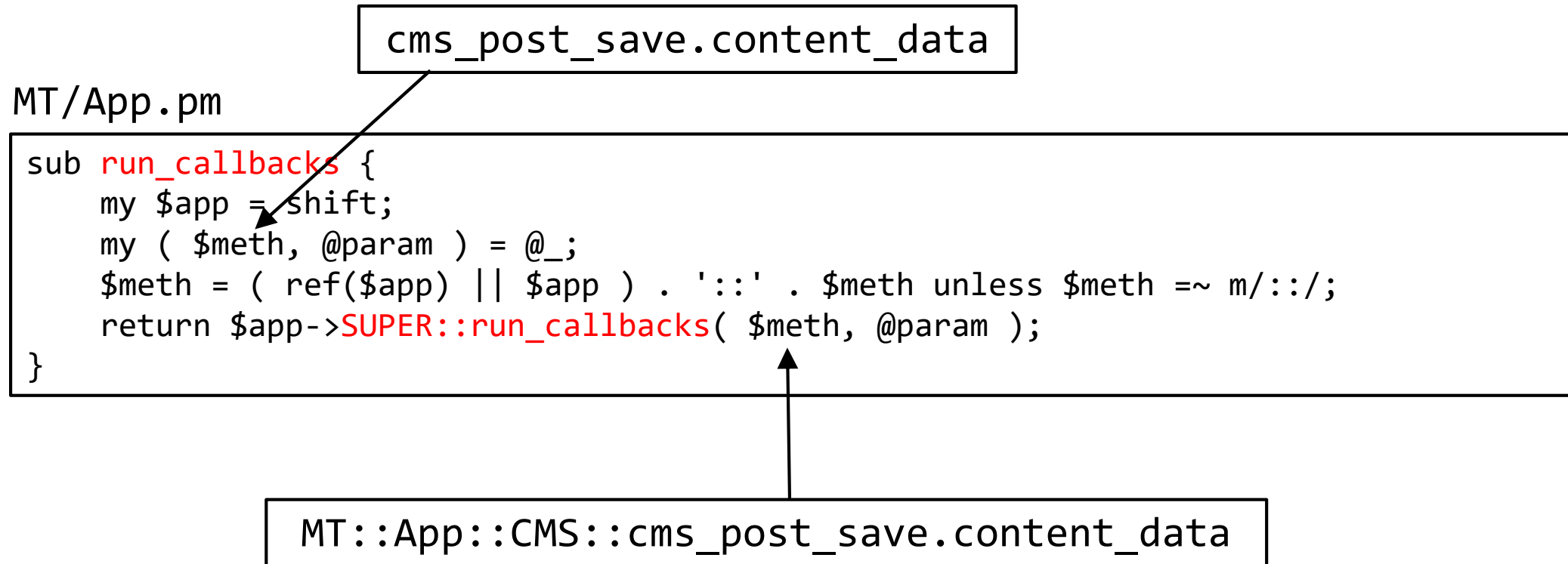
  $content_data->save
  or return $app->error(
    $app->translate(
      "Saving [_1] failed: [_2]", $content_type->name,
      $content_data->errstr
    )
  );

  $app->run_callbacks( 'cms_post_save.content_data',
    $app, $content_data, $orig );
  :
```

データ保存前の
コールバック

データ保存後の
コールバック

④コールバックの起動



④コールバックの起動

MT.pm

```

sub run_callbacks {
  my $class = shift;
  my ( $meth, @args ) = @_;
  $meth = $CallbackAlias{$meth} if exists $CallbackAlias{$meth};
  my @methods;

  if ( $meth =~ /::/ ) {
    my $name = $meth;
    $name =~ s/^\.*::(:[^\:]*)/$1/;
    push @methods, '*::' . $name if $CallbacksEnabled{ '*::' . $name };
    push @methods, $name if $CallbacksEnabled{$name};
  }

  :
  foreach my $callback_sheaf (@Callbacks) {
    for my $meth (@methods) {
      if ( my $set = $callback_sheaf->{$meth} ) {
        for my $cb (@$set) {
          my $result = $class->run_callback( $cb, @args );
          $filter_value &&= $result;
          :
        }
      }
    }
  }
}

```

MT::App::CMS::cms_post_save.content_data

cms_post_save.content_data

['cms_post_save.content_data'];

“cms_post_save.content_data”に登録されている
優先度xのすべてのコールバック

優先度0から
1,2,3...11
の順に実行

cms_post_save.content_data

④コールバックの起動

MT.pm

```
sub run_callback {  
    my $class = shift;  
    my ( $cb, @args ) = @_;  
  
    $cb->error();    # reset the error string  
    my $result = eval { $cb->invoke(@args); };  
    if ( my $err = $@ ) {  
        $cb->error($err);  
        :  
        return 0;  
    }  
    if ( $cb->errstr() ) {  
        return 0;  
    }  
    return $result;  
}
```

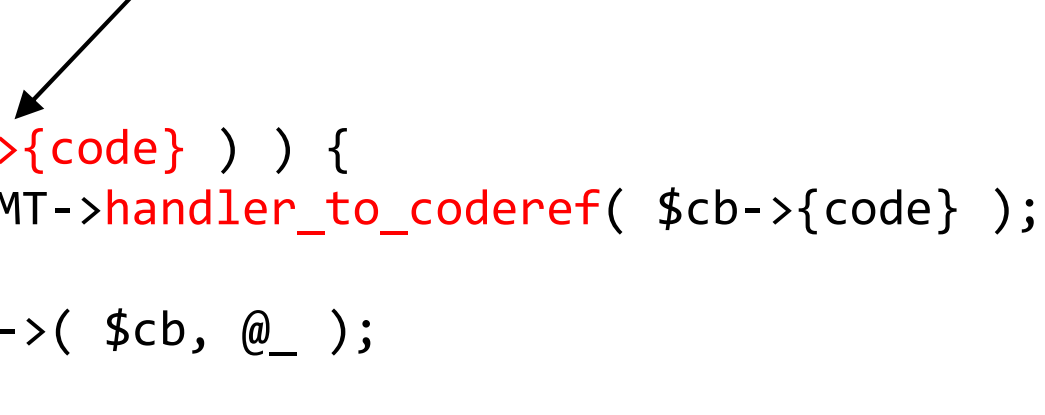
```
$app->run_callbacks( 'cms_post_save.content_data',  
                    $app, $content_data, $orig );
```

④コールバックの起動

MT/Callback.pm

`$Workflow::Workflow::App::CMS::post_save`

```
sub invoke {  
  my $cb = shift;  
  unless ( ref( $cb->{code} ) ) {  
    $cb->{code} = MT->handler_to_coderef( $cb->{code} );  
  }  
  return $cb->{code}->( $cb, @_ );  
}
```

A diagram consisting of a rectangular box at the top containing the text `$Workflow::Workflow::App::CMS::post_save`. An arrow points from the bottom-left corner of this box to the `handler_to_coderef` function call in the Perl code block below.

④コールバックの起動

MT/Callback.pm

```
sub handler_to_coderef {
    my $pkg = shift;
    my ( $name, $delayed ) = @_ ;
    :
    if ( $method ) {
        $code = sub {
            my $mt_inst = MT->instance;
            local $mt_inst->{component} = $component
                if $component;
            return $hdlr_pkg->$method(@_);
        };
    } else {
        if ( $component ) {
            $code = sub {
                no strict 'refs';
                my $hdlr = (
                    defined &$name ? ¥&$name
                    : ( $pkg->can('AUTOLOAD') ? ¥&$name
                    : undef
                )
            );
            use strict 'refs';
            if ( $hdlr ) {
                my $mt_inst = MT->instance;
                local $mt_inst->{component} = $component
                    if $component;
                return $hdlr->(@_);
            }
        }
    }
}
```

コールバックの優先度設定 (Workflow)

MT/Revisable.pm

```
sub install_properties {
  my $pkg      = shift;
  my ($class)  = @_;
  my $props    = $class->properties;
  my $datasource = $class->long_datasource;
  :
  # Callbacks: object-level callbacks could not be
  # prioritized and thus caused problems with plugins
  # registering a post_save and saving
  MT->add_callback( 'data_api_post_save.' . $datasource,
    9, undef, ¥&mt_postsave_obj );
  MT->add_callback( 'api_post_save.' . $datasource,
    9, undef, ¥&mt_postsave_obj );
  MT->add_callback( 'cms_post_save.' . $datasource,
    9, undef, ¥&mt_postsave_obj );
}
```

更新履歴は
優先度9で登録

更新履歴が
登録されたあとに実行

plugins/Workflow/config.yaml

```
:
callbacks:
  cms_post_save.content_data:
    code: $Workflow::Workflow::App::CMS::post_save
    priority: 10
```

- 更新履歴から変更前のステータスを取得
- 変更後のステータスと比較し、承認依頼の場合のみメール送信する制御を実施



① プラグインにコールバック定義（優先度設定）

config.yaml

```
:  
callbacks:  
  cms_post_save.content_data: $Workflow::Workflow::App::CMS::post_save
```



```
:  
callbacks:  
  cms_post_save.content_data:  
    code: $Workflow::Workflow::App::CMS::post_save  
    priority: 10
```

テンプレートタグ

テンプレートタグ (Object)

記事の管理

公開 削除 アクション...

フィルタ: [すべての記事](#)

タイトル

HelloWorld3 [✕](#)

HelloWorld2 [✕](#)

HelloWorld1 [✕](#)

テンプレート名

test

テンプレートの内容

```
<mt:Objects name="entry">
  <mt:Object name="entry" property="title" /><br />
</mt:Objects>
```

```
1 <mt:Objects name="content_type">
2   <mt:Object name="content_type" property="unique_id" />
3 </mt:Objects>
```

localhost/mt/test.html

HelloWorld1
HelloWorld2
HelloWorld3

<https://www.koikikukan.com/archives/2010/11/09-005555.php>

テンプレートタグ

config.yaml

```
  :  
tags:  
  block:   
    Objects: $Object::Object::Tags::_hdlr_objects  
  function:  
    Object: $Object::Object::Tags::_hdlr_object_data
```

ブロックタグ

起動するプログラム

ファンクションタグ

タグ名



ブロックタグ

lib/

Object/

Tags.pm

```

sub _hdlr_objects {
  my ($ctx, $args, $cond) = @_;

  my $name = $args->{name};
  my $class = MT->model($name);
  my $sort = $args->{sort};

  my $res = '';
  my $builder = $ctx->stash('builder');
  my $tokens = $ctx->stash('tokens');

  my ($term, $arg);
  $arg->{ sort } = $sort if $sort;

  my $blog = $ctx->stash('blog');
  $term->{blog_id} = $blog->id;

  $res;
}

```

コンテキスト

パラメータ

stash:隠し場所

オブジェクト (Entry) のロード

```

my @objects = $class->load($term, $arg);
my $count = 0;
my $max = scalar @objects;
my $vars = $ctx->{__stash}{vars} ||= {};
for my $object (@objects) {
  $count++;
  local $ctx->{__stash}{$name} = $object;
  local $vars->{__first__} = $count == 1;
  local $vars->{__last__} = ($count == ($max));
  local $vars->{__counter__} = $count;
  my $out = $builder->build($ctx, $tokens, { %$cond });
  $res .= $out;
}

```

オブジェクトを"__stash"に保持

コンテキストのビルド

HelloWorld1
 HelloWorld2
 HelloWorld3

コンテキストからビルダーとトークン取得

ファンクションタグ

lib/
Object/
Tags.pm

コンテキスト

```
sub _hdlr_object_data {  
  my ($ctx, $args) = @_;  
  my $name = $args->{name};  
  my $prop = $args->{property};  
  
  my $object = $ctx->stash($name);  
  return $object->$prop;  
}
```

"MT::Entry->title"の値を返却

グローバルモディファイア

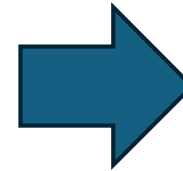
グローバルモディファイア (Split)

テンプレート名

test

テンプレートの内容

```
1 <mt:setvar name="foo" value="a AND b AND c" />
2 <mt:getVar name="foo" split=" AND " setvar="bar">
3 <mt:loop name="bar">
4   <mt:getVar name="__value__" /><br />
5 </mt:loop>
```



localhost/mt/test.html


a
b
c

<https://www.koikikukan.com/archives/2009/01/20-015555.php>

グローバルモディファイア

config.yaml

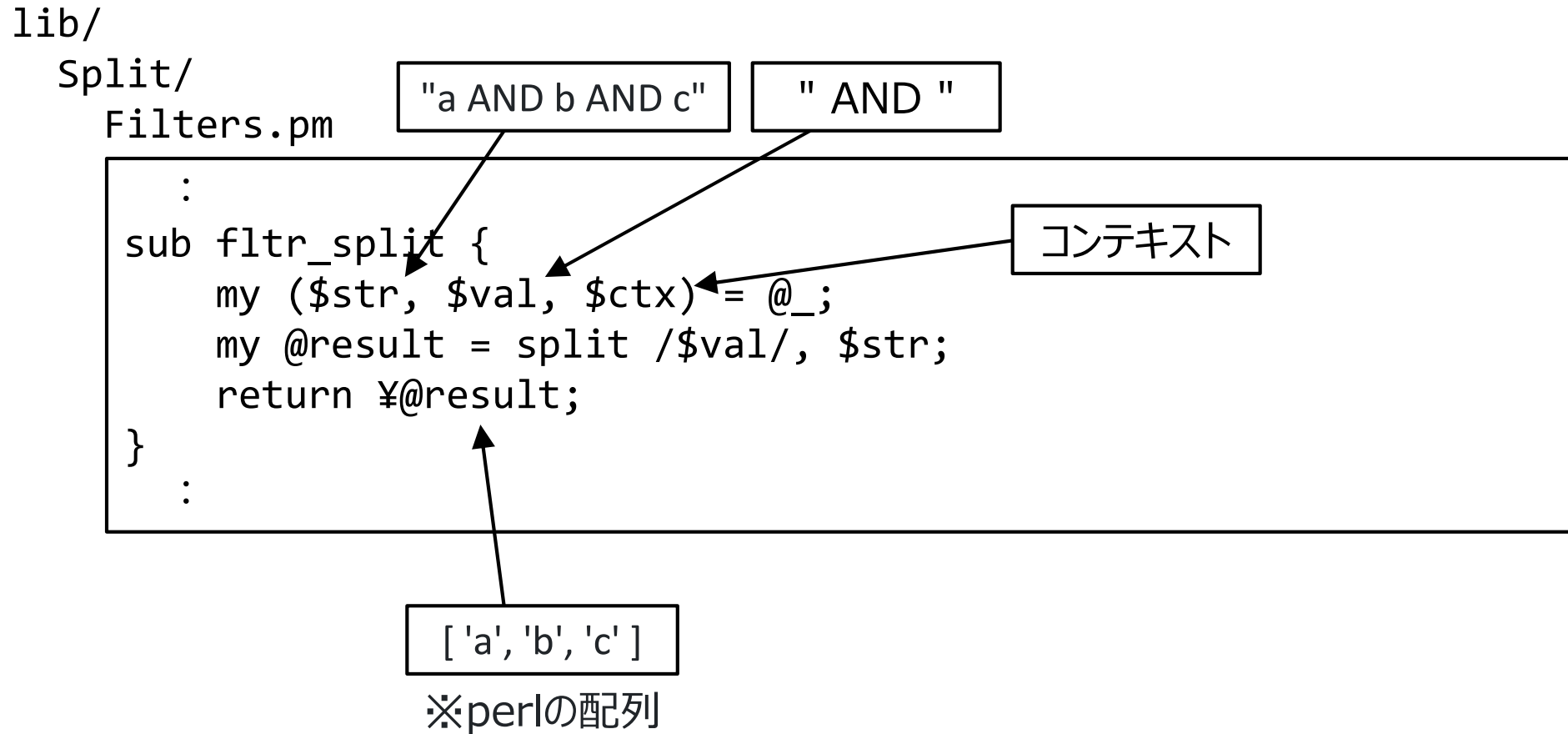
```
  :  
tags:  
  modifier:  
    split: $Split::Split::Filters::fltr_split
```



モディファイア名

起動するプログラム

グローバルモディファイア



テンプレート

テンプレート (Workflow)

テンプレートの管理

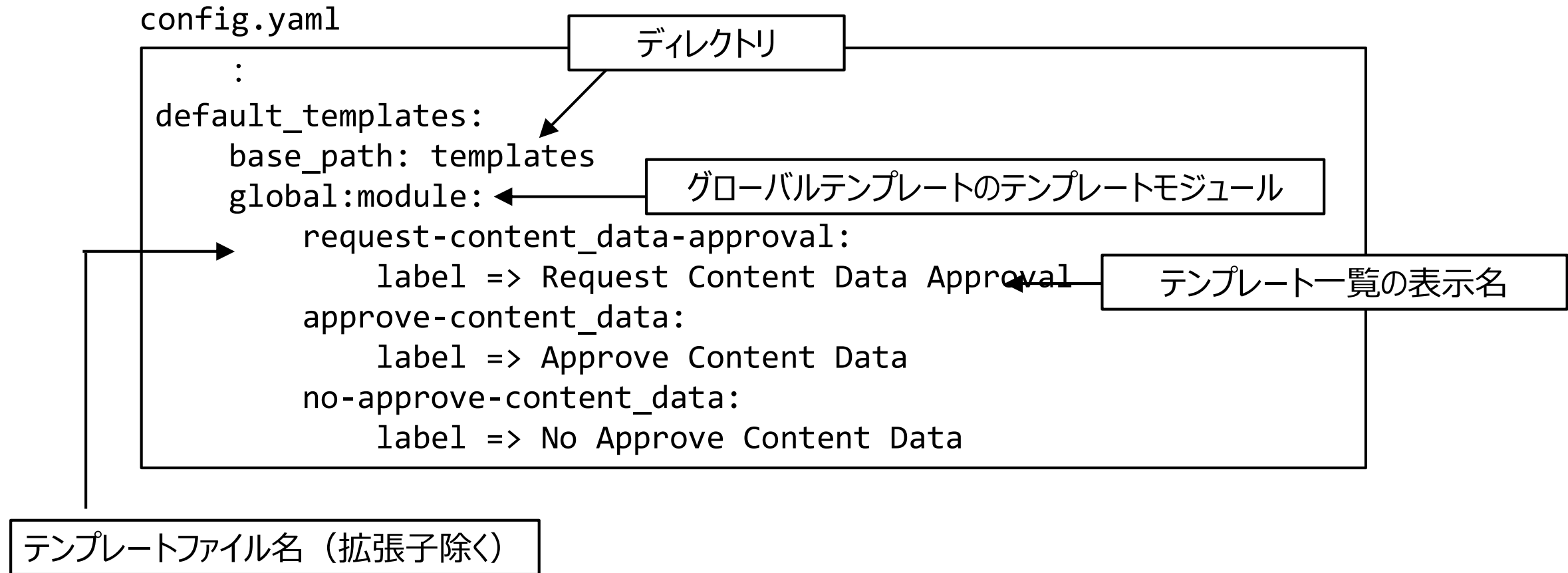
⊕ テンプレートモジュール ⇅ [新規作成](#)

テンプレートモジュール

削除 アクション... ⇅ Go

<input type="checkbox"/>	名前
<input type="checkbox"/>	コンテンツデータのプレビュー依頼
<input type="checkbox"/>	コンテンツデータの差し戻し
<input type="checkbox"/>	コンテンツデータの承認
<input type="checkbox"/>	コンテンツデータの承認依頼
<input type="checkbox"/>	プレビュー依頼
<input type="checkbox"/>	メールフッター
<input type="checkbox"/>	記事の差し戻し

テンプレート (定義)



テンプレート (テンプレート)

```
plugins/  
  Workflow/  
    config.yaml  
    lib/  
      :  
      templates/  
        no-approve-content_data.mtml  
        request-content_data-approval.mtml  
        request-preview-content_data.mtml
```

テンプレート（テンプレートファイルの中身）

templates/request-content_data-approval.mtml

```
このメールは、<mt:if name="review" eq="reviewee">ライター<mt:else>承認者<mt:if name="approver_name">（<$mt:var name="approver_name"$>）</mt:if></mt:if>「<$mt:AuthorDisplayName$>」様からの、
<$mt:BlogName$>の<mt:unless name="published">新しいコンテンツデータ「<$mt:ContentLabel$>」の
公開承認依頼です。<mt:else>コンテンツデータ「<$mt:ContentLabel$>」の更新承認依頼です。</mt:unless>
```

の承認／差し戻しを行うには、以下のURLをクリックして
「承認」または「差し戻し」をクリックしてください。

編集画面

```
<$mt:var name="entry_editurl"$>
```

プレビュー画面

```
<$mt:var name="entry_previewurl"$>
```

```
<mt:if name="message"><$mt:AuthorDisplayName$>様からのメッセージ:
```

```
-----
<$mt:var name="message"$>
```

```
-----</mt:if>
```

注：複数の承認者にこのメールが送信されている場合、承認状態が変更されている可能性があります。

```
<mt:if name="review" eq="reviewee">ライター<mt:else>承認者</mt:if>:
<$mt:AuthorDisplayName$> <<$mt:AuthorEmail$>>
```

トランスフォーマー

トランスフォーマー

`build_page()`

`load_tmpl` (テンプレートのロード)

- `template_source` : テンプレートの書き換え
 - ↓ パラメータ設定等
- `template_param` : テンプレートに独自パラメータ設定
 - ↓ テンプレートのビルド
- `template_output` : ビルドされたファイルの書き換え (説明省略)

template_source (1/10)

[MT_DIR]/tmpl/cms/edit_content_data.tmpl

```

:
<mtapp:setting
  id="basename"
  label="$basename_label"
  label_class="top-label">
<br/>
<span class="basename">
  <input type="text" name="identifier" id="basename" style="color:#ccc"
class="form-control text full" value="<$mt:var name="identifier"
escape="html"$>" /><mt:if name="object_type" eq="page"><span class="file-
extension"><mt:var name="file_extension" escape="html"></span></mt:if>
  <input type="hidden" name="basename_manual" id="basename_manual"
value="0" />
</span>
:

```

出力ファイル名



出力ファイル名

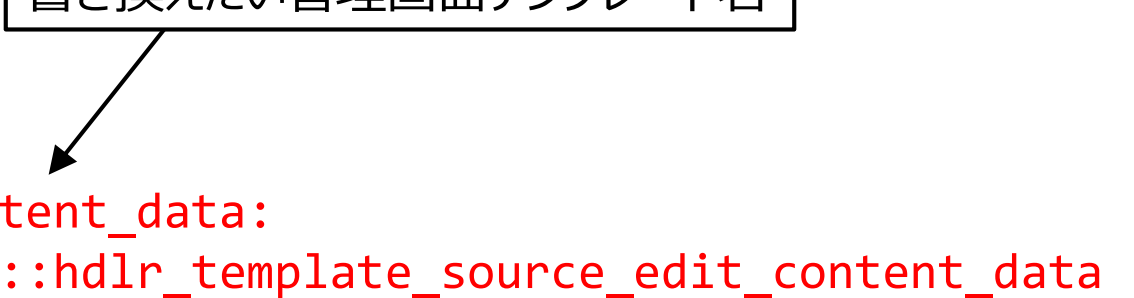
テンプレートに追加したい属性



template_source (2/10)

```
plugins/  
HelloWorld/  
config.yaml
```

```
id: HelloWorld  
name: HelloWorld  
callbacks:  
  MT::App::CMS::template_source.edit_content_data:  
    code: $HelloWorld::HelloWorld::CMS::hdr_template_source_edit_content_data
```



書き換えたい管理画面テンプレート名

template_source (3/10)

```
plugins/  
  HelloWorld/  
    lib/  
      HelloWorld/  
        CMS.pm
```

書き換えたい管理画面テンプレート

```
:  
sub hdlr_template_source_edit_content_data {  
  my ($cb, $app, $tmpl) = @_;  
  
  $$tmpl =~ s/id="basename"/id="basename" style="color:#ccc"/;  
}  
:
```

Perlで文字列置換

template_source (4/10) エレガントな実装

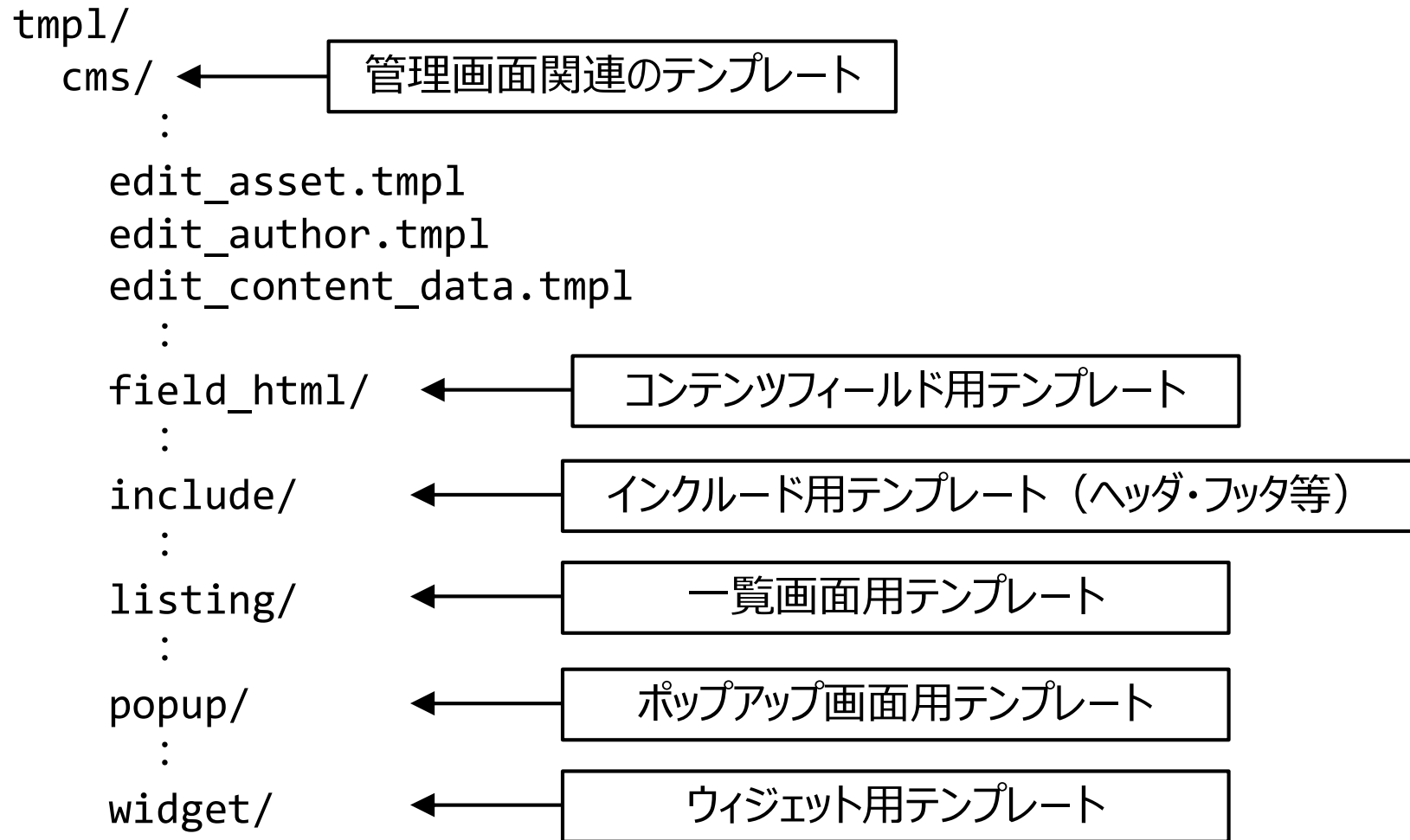
```
sub sample_hdlr_template_source {
  my ($cb, $app, $template) = @_;
  my $old = <<HTML;
  <mt:if name="id">
  <div class="text-right delete-action mt-4 d-none d-md-block">
HTML
    $old = quotemeta($old);
    my $new = <<HTML;
  <mtapp:setting id="plugin-text"
    label="<__trans phrase="Plugin Text">" label_class="top-label">
    <input type="text" name="plugin-text" id="plugin-text"
      class="form-control text full" value="<mt:var name="plugin-text"
        escape="html">" />
  </mtapp:setting>
  <mt:if name="id">
  <div class="text-right delete-action mt-4 d-none d-md-block">
HTML
    $$template =~ s/$old/$new/;
}
```

検索文字列 {

置換文字列
(検索文字列も含める) {

※コアのテンプレートの内容が変わると対応が必要

template_source (5/10) テンプレートの探し方



分からなければ、ファイルの中身を消して、画面が真っ白になればそれ

template_source (6/10) 起動箇所

MT.pm

```
sub load_tmpl {
  my $mt = shift;
  :
  my ( $file, @p ) = @_;
  my $param;
  if ( @p && ( ref( $p[$#p] ) eq 'HASH' ) ) {
    $param = pop @p;
  }
  my $cfg = $mt->config;
  require MT::Template;
  my $tmpl;
  my @paths = $mt->template_paths;
  my $type = { 'SCALAR' => 'scalarref', 'ARRAY' => 'arrayref' }->{ ref $file } || 'filename';
  $tmpl = MT::Template->new(
    type    => $type,
    source  => $file,
    path    => \@paths,
    filter  => sub {
      my ( $str, $fname ) = @_;
      if ( $fname ) {
        $fname = File::Basename::basename($fname);
        $fname =~ s/\.tmpl$//;
        $mt->run_callbacks( "template_source.$fname", $mt, @_ );
      }
    }
  );
}
```

スライド103で起動

ここ

template_param (1/4)

tmpl/cms/edit_content_data.tmpl

```

:
<mtapp:setting
  id="basename"
  label="$basename_label"
  label_class="top-label">
<br/>
<span class="basename">
  <input type="text" name="identifier" id="basename" class="form-control text full" value="<$mt:var
name="identifier" escape="html"$>" /><mt:if name="object_type" eq="page"><span class="file-extension"><mt:var
name="file_extension" escape="html"></span></mt:if>
  <input type="hidden" name="basename_manual" id="basename_manual" value="0" />
</span>
<input type="text" name="url" id="url" class="full-width" value="<mt:var name="url" escape="html">" />
:

```

テンプレートに追加したい要素

template_param (2/4)

```
plugins/  
HelloWorld/  
config.yaml
```

```
id: HelloWorld  
name: HelloWorld  
callbacks:  
  MT::App::CMS::template_param.edit_content_data:  
    code: $HelloWorld::HelloWorld::CMS::hdlr_template_param_edit_content_data
```

A callout box with a black border and white background contains the Japanese text "パラメータを追加したい管理画面テンプレート名" (Name of the management screen template to which parameters are to be added). A black arrow points from the bottom-left corner of this box to the 'code' field in the 'callbacks' section of the YAML configuration.

パラメータを追加したい管理画面テンプレート名

template_param (3/4)

CMS.pm

DOMアクセス

<https://github.com/movabletype/Documentation/wiki/Japanese-plugin-dev-4-3>

```

:
sub hdlr_template_param_edit_content_data {
  my ($cb, $app, $param, $tmpl) = @_;

  my $host_node = $tmpl->getElementById('basename');
  my $innerHTML =
    '<input type="text" name="url" id="url" class="full-width" value="<mt:var name="url" escape="html">" />';
  my $block_node = $tmpl->createElement(
    'app:setting',
    {
      id => 'url',
      label => 'URL',
      label_class => 'top-label',
    }
  );

  $block_node->innerHTML( $innerHTML );
  $tmpl->insertAfter( $block_node, $host_node );

  $param->{url} = "http://www.example.com/" if !$param->{url};
}

```

追加する要素

mt:appsettingタグの生成

パラメータとして渡す

テンプレートに渡すパラメータ

template_param (4/4) 起動箇所

MT.pm

```

sub build_page {
  my $mt = shift;
  my ( $file, $param ) = @_;
  my $tmpl;
  my $mode = $mt->mode;
  $param->{"mode_$mode"} ||= 1;
  :
  pop @{$param->{page_titles}};
  $param->{magic_token} = $mt->current_magic if $mt->user;
  :
  my $tmpl_file = '';
  if ( UNIVERSAL::isa( $file, 'MT::Template' ) ) {
    $tmpl = $file;
    $tmpl_file = ( exists $file->{__file} ) ? $file->{__file} : '';
  }
  else {
    $tmpl = $mt->load_tmpl($file) or return;
    $tmpl_file = $file unless ref($file);
  }
  $tmpl->param($param) if $param;
  if ($tmpl_file) {
    $tmpl_file = File::Basename::basename($tmpl_file);
    $tmpl_file =~ s/¥.tmpl$//;
    $tmpl_file = '.' . $tmpl_file;
  }
  $mt->run_callbacks( 'template_param' . $tmpl_file,
    $mt, $tmpl->param, $tmpl );
  my $output = $mt->build_page_in_mem($tmpl);
  return unless defined $output;
  $mt->run_callbacks( 'template_output' . $tmpl_file,
    $mt, ¥$output, $tmpl->param, $tmpl );
  return $output;
}

```

ページのビルドで起動

スライド99のload_tmpl

ここ

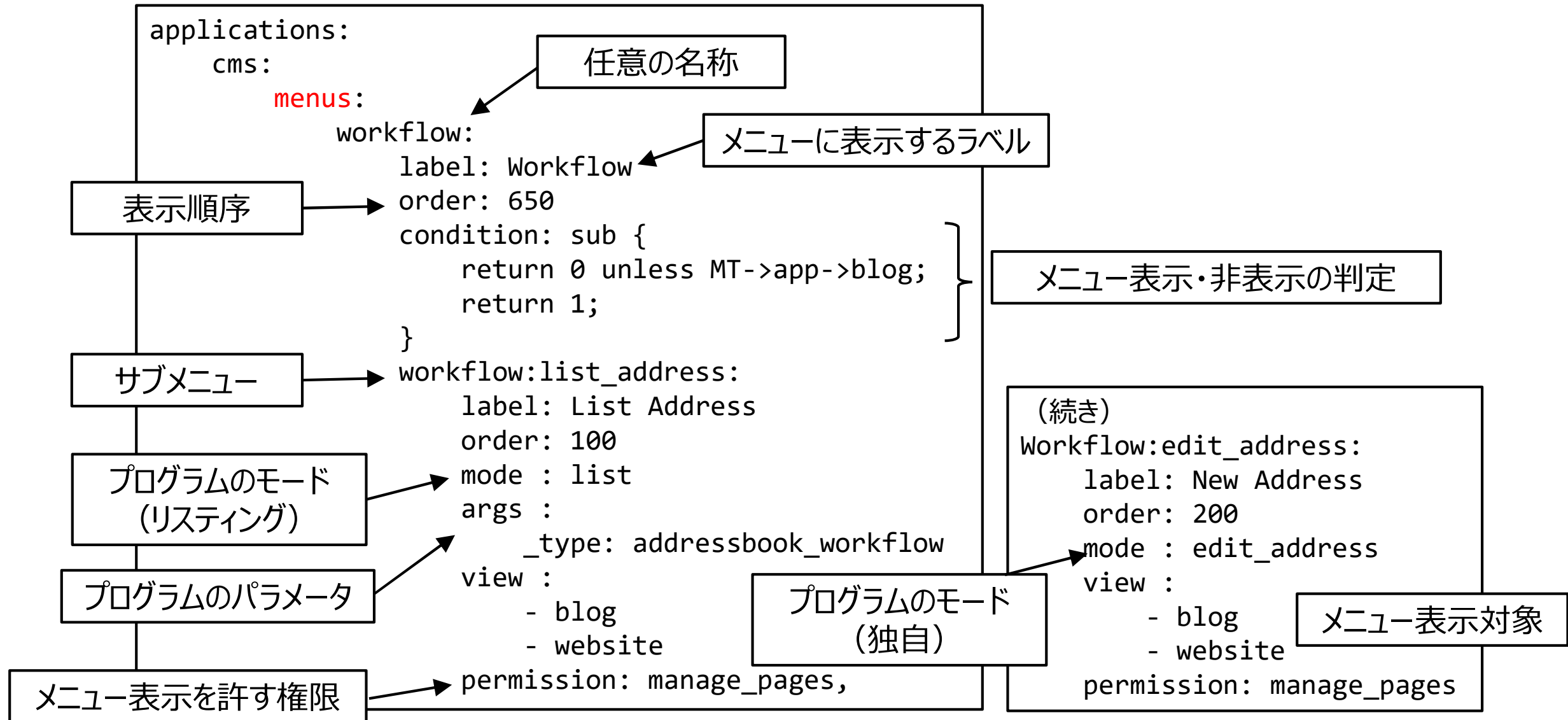
メニユー

メニュー (Workflow)



メニュー

config.yaml



メニュー「新規作成」をクリックしたあとの画面



メールアドレス新規作成

名前 必須

メール 必須

説明

保存

管理画面テンプレート

edit_address.tmp1 (抜粋)

```

<mt:setvarblock name="form_header">
<form method="post" action="<mt:GetVar name="script_url">" id="new_addressbook_form">
<mt:unless name="new_object">
  <input type="hidden" name="id" value="<mt:GetVar name="id" escape="html">" />
</mt:unless>
<input type="hidden" name="__mode" value="save" />
<input type="hidden" name="_type" value="addressbook_workflow" />
<input type="hidden" name="blog_id" value="<mt:GetVar name="blog_id">" />
<input type="hidden" name="return_args" value="<mt:GetVar name="return_args" escape="html">" />
<input type="hidden" name="magic_token" value="<mt:GetVar name="magic_token">" />
</mt:setvarblock>
<mt:setvarblock name="page_content">
  <fieldset>
    <input type="text" name="name" id="name" class="form-control text required" value="<mt:GetVar name="name" />" />
    <input type="text" name="email" id="email" class="form-control text required" value="<mt:GetVar name="email" />" />
    <textarea name="description" id="description" class="form-control textareaz low" cols="" rows=""><mt:var name="description" escape="html"></textarea>
  <mt:if name="new_object">
    <mt:SetVarBlock name="action_buttons">
      <button type="submit" title="<__trans phrase="Create">" class="create action button btn btn-primary"><__trans phrase="Create"></button>
    </mt:SetVarBlock>
  <mt:else>
    <mt:SetVarBlock name="action_buttons">
      <button type="submit" title="<__trans phrase="Save Changes">" class="create action button btn btn-primary"><__trans phrase="Save Changes"></button>
    </mt:SetVarBlock>
  </mt:if>
  <mt:Include name="include/actions_bar.tmp1" bar_position="bottom" hide_pager="1" settings_bar="1">
  </fieldset>
</mt:setvarblock>
<mt:setvarblock name="form_footer">
</form>
</mt:setvarblock>
<mt:include name="layout/default.tmp1">
<mt:var name="layout">

```

ボタンをクリックしたときのモード

オブジェクトのタイプ

その他、必要なパラメータ

入力フィールド（名前・メール・説明）

新規作成と更新で表示するボタンを変更

メニューから起動するプログラムの定義

config.yaml

```
applications:
  cms:
    menus:
      workflow:
        :
        workflow:edit_address:
          label: New Address
          order: 200
          mode : edit_address
          view :
            - blog
            - manage_pages
            :
methods:
  edit_address: $Workflow::Workflow::App::AddressBook::edit_address
```

名前を合わせる

スライド106

メニューから起動するプログラム

plugins/Workflow/lib/Workflow/App/AddressBook.pm

```

package Workflow::App::AddressBook;
:
use MT::AddressBook;
sub edit_address {
  my $app = shift;
  my $q = $app->param;
  my $id = $q->param('id');
  my $plugin = MT->component('Workflow');
  my %param;
  if (!$id) {
    $param{ new_object } = 1;
    $param{ page_title } = $plugin->translate('New Address');
  } else {
    my $address = MT::AddressBook->load({ id => $id, blog_id => $q->param('blog_id') || 0 });
    $param{ id } = $q->param('id');
    $param{ blog_id } = $q->param('blog_id') || 0;
    $param{ name } = $address->name;
    $param{ email } = $address->email;
    $param{ page_title } = $plugin->translate('Update Address');
  }
  my $tmpl = 'edit_address.tmpl';
  return $app->build_page($tmpl, %param);
}

```

クエリーの取得

ローカライズのためコンポーネント取得

管理画面テンプレートに
設定するパラメータ

管理画面のビルド
(スライド103)

管理画面テンプレート

管理画面テンプレート

パラメータ

リスティング・リストアクション

<https://www.slideshare.net/slideshow/deep-mt-r2-8422851/8422851>

リスティング



メールアドレス一覧

表示オプション ▾

削除

1 - 2 / 2

フィルタ: [すべてのメールアドレス](#) ▾

<input type="checkbox"/>	名前 ↑	説明	サイト名
<input type="checkbox"/>	鈴木一郎	課長	First Weblog
<input type="checkbox"/>	山田太郎	部長	First Weblog

◀ 前 1 次 ▶

プラグイン設定

config.yaml

要素別の定義

```
list_properties:
  addressbook_workflow:
    id:
      base: __virtual.id
      order: 100
    name:
      auto: 1
      label: Name
      order: 200
      primary: 1
      display: force
    description:
      auto: 1
      label: Description
      order: 250
      display: optional
      :
```

名前

説明

リスト全体の定義

```
listing_screens:
  addressbook_workflow:
    primary: name
    screen_label: addressbook
    object_label: addressbook
    object_label_plural: all_addressbook
    default_sort_key: name
    view:
      - blog
      - website
      - system
    scope_mode: this
```

リストアクション (CSVDataImExporter)

The screenshot shows the 'Site Pageの管理' (Site Page Management) interface. At the top, there are three links: 'Site Pageを作成' (Create Site Page), 'Site Pageをインポート' (Import Site Page), and 'Site Pageをエクスポート' (Export Site Page). Below these are three buttons: '公開' (Publish), '削除' (Delete), and 'アクション...' (Action...). The 'アクション...' dropdown menu is open, showing three options: 'コンテンツデータの公開を取り消し' (Cancel content data publication), 'プラグインアクション' (Plugin action), and 'コンテンツデータのエクスポート (CSV)' (Export content data (CSV)). The 'コンテンツデータのエクスポート (CSV)' option is highlighted with a red box. Below the dropdown, there is a table with columns for 'データ' (Data), 'og:image', and 'Tag'. The table contains two rows, each with a checkbox, a green checkmark, the text 'aaa', and a small icon.

Site Pageの管理

⊕ Site Pageを作成 ⊕ Site Pageをインポート ⊕ Site Pageをエクスポート

公開 削除 アクション...

フィルタ: [すべて](#)

コンテンツデータの公開を取り消し
プラグインアクション
コンテンツデータのエクスポート (CSV)

データ og:image Tag

aaa aaa

aaa aaa

リストアクション

config.yaml



リストアクション

lib/CSVDataImExporter/Export.pm

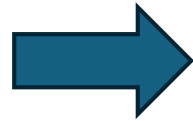
```
    :  
sub export_entries_with_csv {  
    my $app = shift;  
  
    my $q = $app->param;  
    my @ids = $q->param('id');  
    :  
}
```

一覧でチェックしたIDのリスト



ウィジェット

ウィジェット



ウィジェット

config.yaml

```

:
widgets:
  HelloWorldDashboard:
    label: HelloWorld Dashboard
    template: helloworld.tmp1
    singular: 1
    set: main
    handler: $HelloWorld::HelloWorld::CMS::dashboard
    order: 100
    default: 1
    view:
      - user
      - website
      - blog
  
```

任意の名称

ラベル名

ダッシュボード用テンプレート

表示位置 (main/sidebar)

表示順序

起動するプログラム

表示対象

1回のみ表示

(これを設定しないと表示後、プルダウンから消えない)

ウィジェット

tmpl/helloworld.tmpl

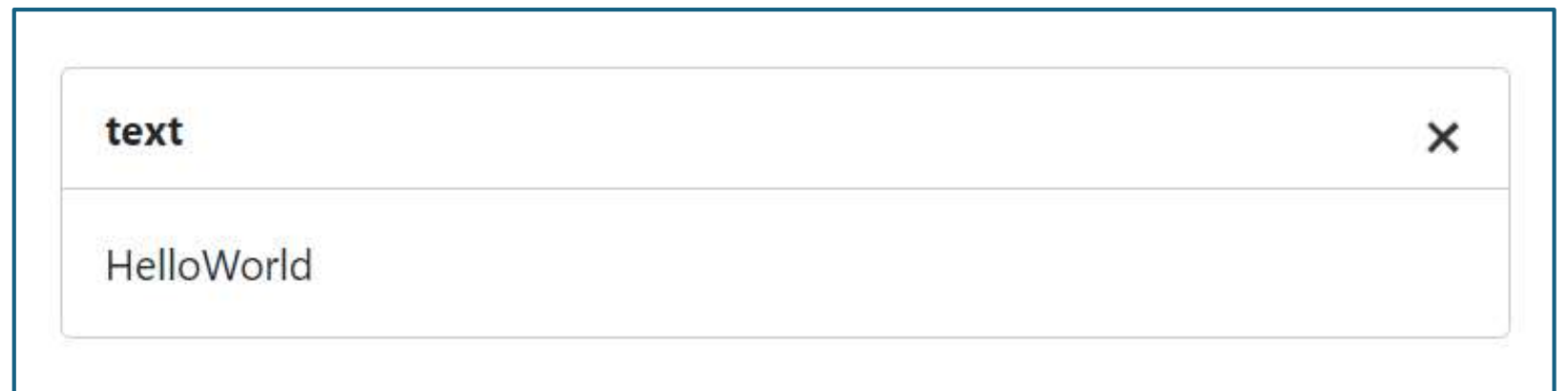
```
<mtapp:widget  
  id="HelloWorldDashboard"  
  class="widget"  
  label="text"  
  can_close="1">  
<mt:var name="data">  
</mtapp:widget>
```

mtapp:widgetタグ

ラベル

×マークの表示

表示したいデータ

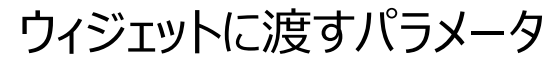


ウィジェット

CMS.pm

```
    :  
sub dashboard {  
    my $app = shift;  
    my ( $tmpl, $param ) = @_;  
  
    $param->{data} = 'HelloWorld';  
}  
    :
```

ウィジェットに渡すパラメータ



パーミッション

パーミッション (Workflow)

ロール作成画面 (抜粋)

権限

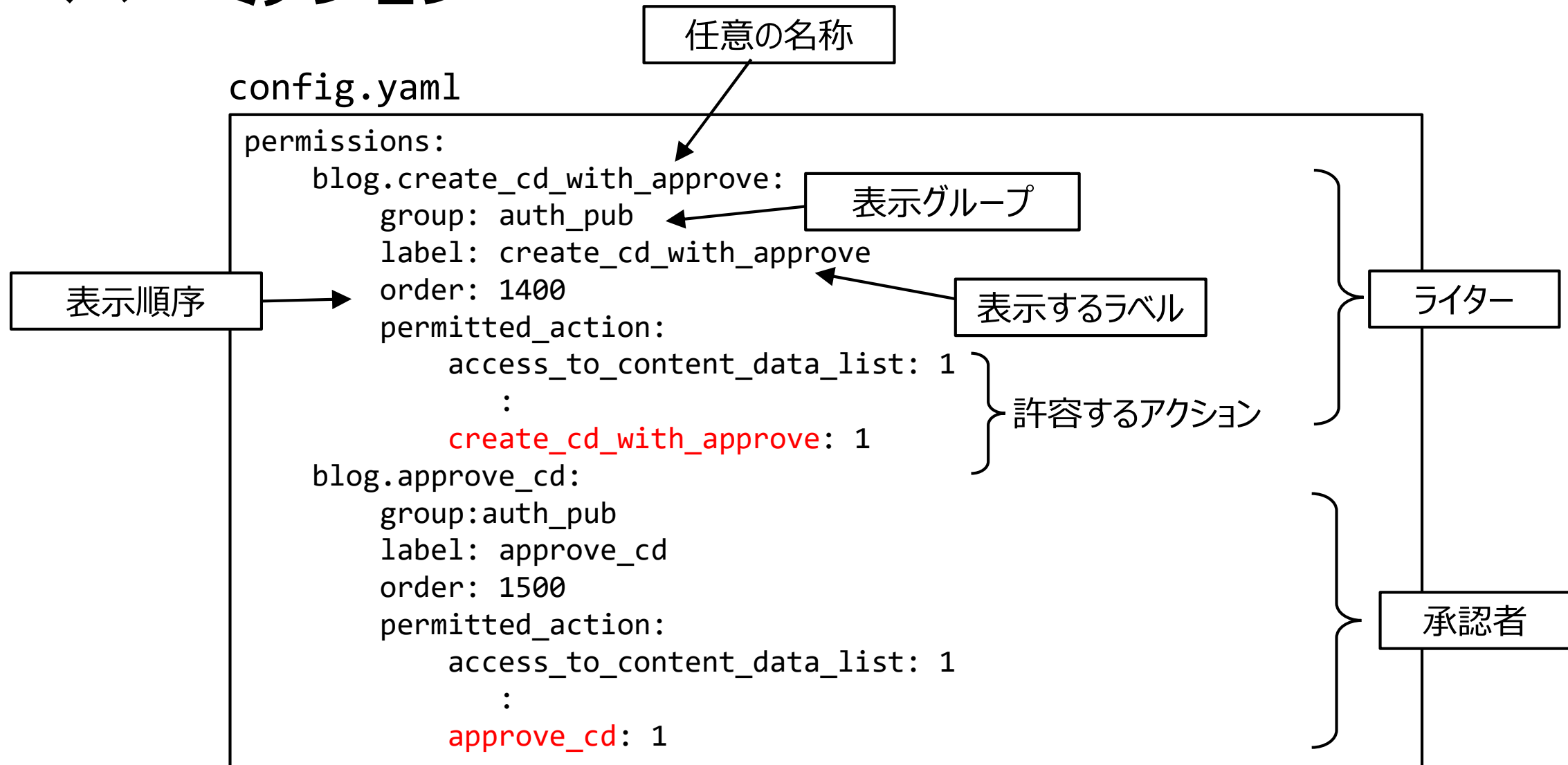
管理

- サイトの管理
- サイトの作成
- 設定の変更
- カテゴリの管理
- アドレス帳の管理
- タグの管理
- ユーザーの管理
- 公開パスの設定
- ログの閲覧
- カテゴリセットの管理

作成と公開

- 記事の作成
- 記事の作成 (承認つき)
- 記事の公開
- 記事の承認
- 通知の送信
- すべての記事の編集
- ウェブページの管理
- サイトを再構築
- すべてのコンテンツデータの管理
- コンテンツデータの作成 (承認つき)
- コンテンツデータの承認

パーミッション



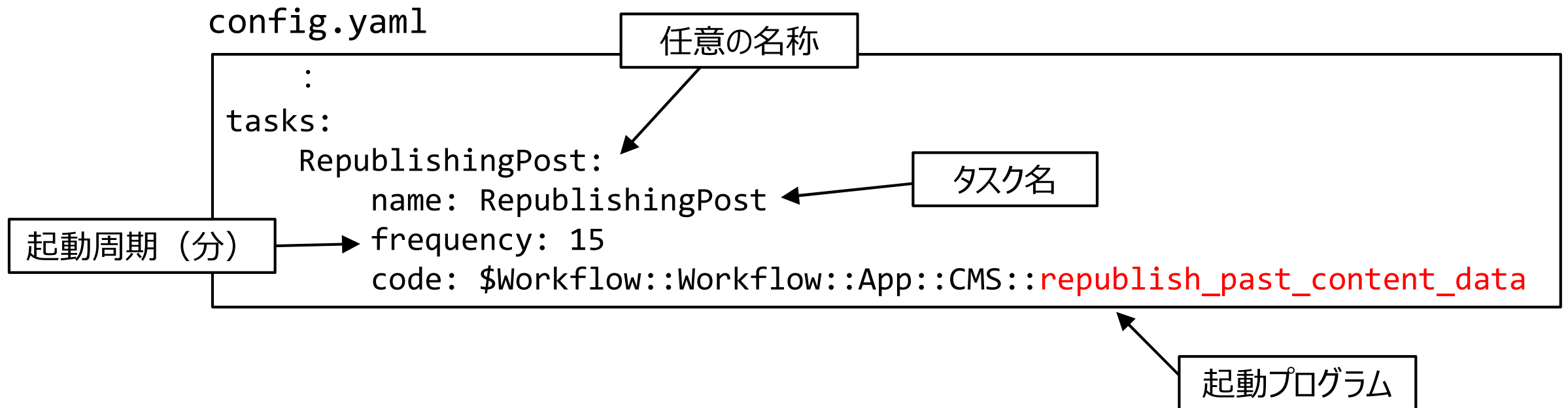
パーミッション

lib/Workflow/CMS/ContentData.pm

```
    :  
    if ( $app->can_do('create_cd_with_approve') ) {  
        //ライター処理  
    }  
  
    if ( $app->can_do('approve_cd') ) {  
        //承認者処理  
    }
```

スケジュールタスク

スケジュールタスク (Workflow)



スケジュールタスク

tools/run-periodic-tasks



lib/Workflow/App/CMS.pm

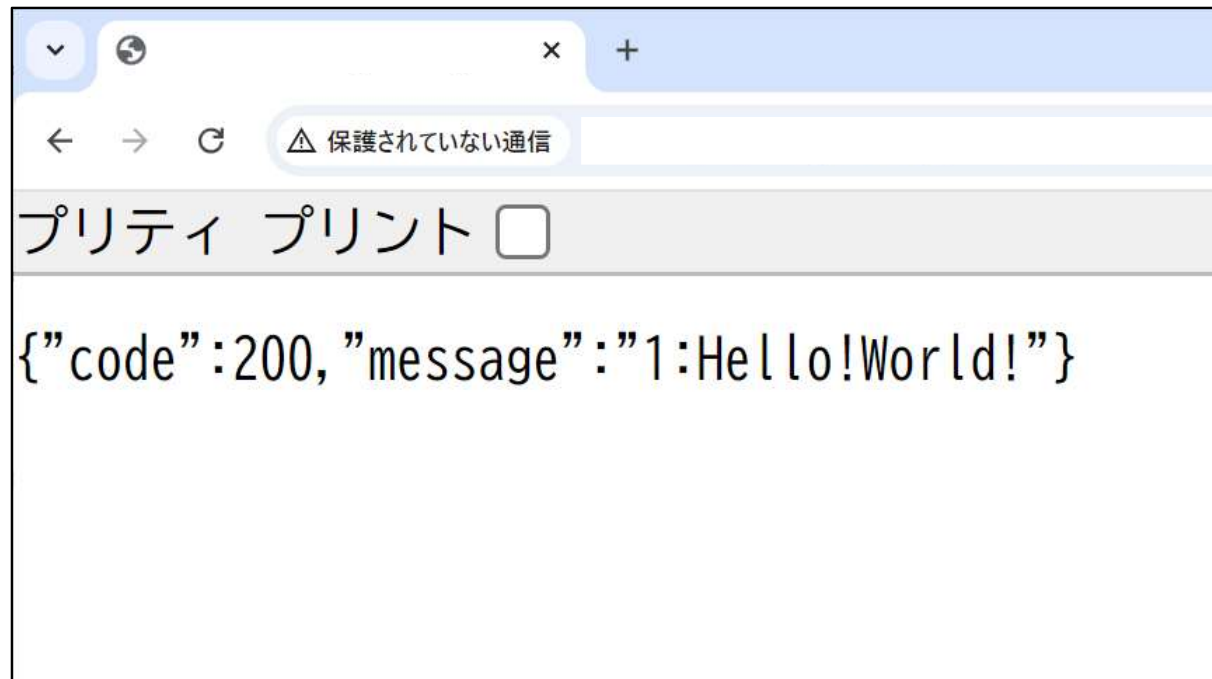
```
    :  
    sub republish_past_content_data {  
        // タスク処理  
    }
```


DataAPI

DataAPI

起動するURL

<http://user-domain/cgi-bin/mt/mt-data-api.cgi/v6/sites/1/helloworld>



DataAPI

Verb:動詞

config.yaml

```
  :
  applications:
    data_api:
      endpoints:
        - id: helloworld
          route: /sites/:site_id/helloworld
          verb: GET
          version: 6
          handler: $HelloWorld::HelloWorld::EndPoint::HelloWorld::get
          requires_login: 0
```

エンドポイントID

エンドポイントの
動作

バージョン

ログイン要否

エンドポイントURI
(:は可変データ)

起動プログラム

DataAPI

lib/HelloWorld/EndPoint/HelloWorld.pm

```
package HelloWorld::EndPoint::HelloWorld;

use MT::DataAPI::Endpoint::Common;
use MT::DataAPI::Resource;

sub get {
    my ( $app, $endpoint ) = @_;
    my ( $blog ) = context_objects(@_) or return;

    return { code => 200, message => $blog->id . ':Hello!World!' };
}

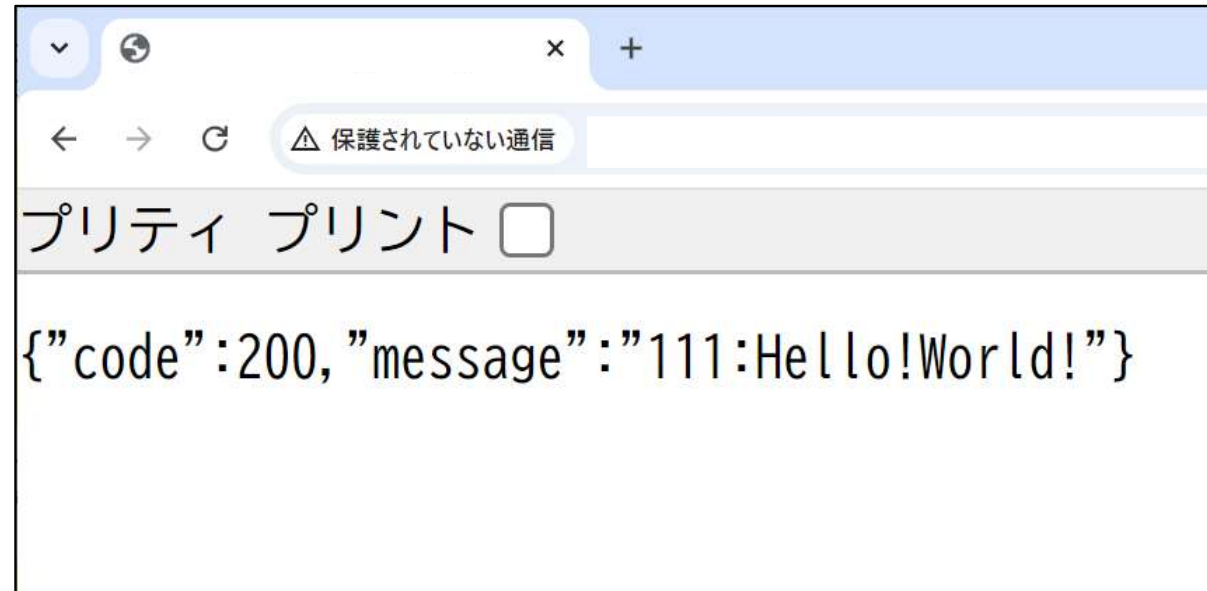
1;
```

routeで指定したリソースを取得できる

返却値をJSONで記述

DataAPI（異なるオブジェクトを複数取得）

<http://user-domain/cgi-bin/mt/mt-data-api.cgi/v6/sites/1/contentTypes/1/contentData/1/helloworld>



DataAPI (異なるオブジェクトを複数取得)

config.yaml

```
:
applications:
  data_api:
    endpoints:
      - id: helloworld
        route: /sites/:site_id/contentTypes/:content_type_id/contentData/:content_data_id/helloworld
    :
```

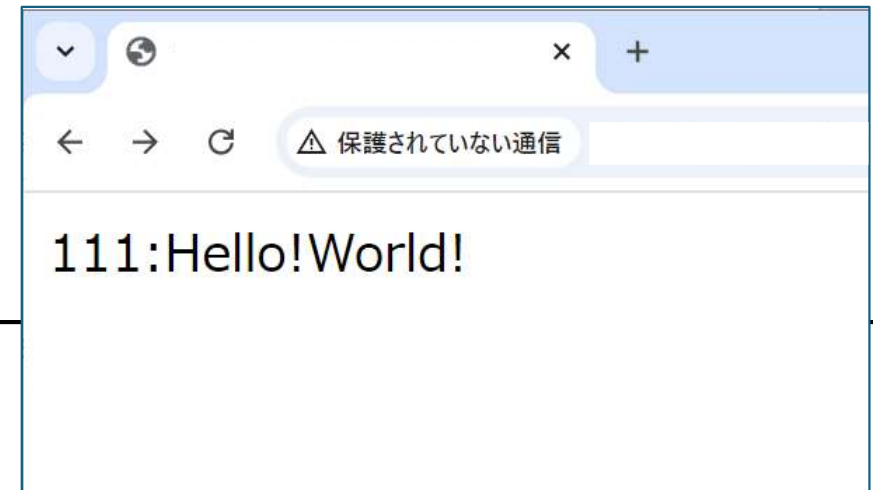
lib/HelloWorld/EndPoint/HelloWorld.pm

```
sub get {
  my ( $app, $endpoint ) = @_;
  my ( $blog, $ct, $cd ) = context_objects(@_) or return;
  return { code => 200,
          message => $blog->id . $ct->id . $cd->id . ':Hello!World!' };
}
```

DataAPI (フロントエンド)

helloworld.html

```
<script src="https://code.jquery.com/jquery-3.7.1.min.js"></script>
<script>
$.ajax({
  url: "http://user-domain/cgi-bin/mt/mt-data-
      api.cgi/v3/sites/1/contentTypes/1/contentData/1/helloworld",
  type: "GET",
  dataType: "json",
}).done(function(data){
  $('#test').text(data.message);
});
</script>
<div id="test"></div>
```



DataAPIの利用設定

サイトの「設定」→「Webサービス」



```
{"error":{"code":403,"message":"Forbidden"}}
```

システムの「設定」→「Webサービス」



```
{"error":{"code":400,"message":"DataAPI is disabled"}}
```


オーバーライド

オーバーライド

- プラグインフレームワークが提供されていない処理
- Perlのオーバーライドを利用して既存処理を書き換える

config.yaml

```
init_app:  
  MT::App::CMS: $HelloWorld::HelloWorld::CMS::init cms_app_routine
```

CMS.pm

```
sub init cms_app_routine {  
  my ( $cb, $app ) = @_;  
  no warnings 'once';  
  no warnings 'redefine';  
  if (ref($app) eq 'MT::App::CMS') {  
    require MT::CMS::ContentData;  
    *MT::CMS::ContentData::make_menus = ¥&plugin_make_menus;  
  }  
}
```

警告対処

オーバーライド

スケジュールタスク

```
# ./run-periodic-tasks
Subroutine permission redefined at /var/www/cgi-bin/mt/lib/MT/ContentType.pm line 358.
2024-11-15T14:58:23 [INFO] --- Start rebuild_content_data. at /var/www/cgi-
bin/mt/lib/MT/ContentPublisher.pm line 559.
2024-11-15T14:58:23 [INFO] Rebuilt /var/www/html/mt/first_weblog/author/writer/index.html
at /var/www/cgi-bin/mt/lib/MT/ContentPublisher.pm line 1450.
2024-11-15T14:58:23 [INFO] --- Start rebuild. at /var/www/cgi-
bin/mt/lib/MT/ContentPublisher.pm line 125.
2024-11-15T14:58:23 [INFO] --- Start rebuild_indexes. at /var/www/cgi-
bin/mt/lib/MT/WeblogPublisher.pm line 1791.
2024-11-15T14:58:24 [INFO] --- End rebuild_indexes. at /var/www/cgi-
bin/mt/lib/MT/WeblogPublisher.pm line 2069.
2024-11-15T14:58:24 [INFO] --- End rebuild. at /var/www/cgi-
bin/mt/lib/MT/ContentPublisher.pm line 400.
Subroutine make_menus redefined at /var/www/cgi-bin/mt/lib/MT/CMS/ContentData.pm line 1341.
```

ローカライズ

ローカライズ

管理画面やログメッセージの
表示言語をユーザー別に最適化

The screenshot shows a user management interface. On the left is a sidebar menu with categories like 'システム' (System), 'サイト' (Site), '記事' (Articles), 'アセット' (Assets), 'ユーザー' (Users), 'グループ' (Groups), 'ロール' (Roles), 'デザイン' (Design), 'カスタムフィールド' (Custom Fields), 'フィルタ' (Filters), '設定' (Settings), and 'ツール' (Tools). The 'ユーザー' section is expanded, showing a list of users with 'mtbook' selected. The main content area is titled 'ユーザー情報の編集' (Edit User Information) and shows the user's details. The '設定' (Settings) section is highlighted with a red box, showing the '使用言語' (Language) dropdown menu set to '日本語' (Japanese). Below the dropdown, a note states: '管理画面で使用する言語です。' (Language to be used in the management screen).

ダッシュボード > システム > ユーザー > mtbook

ユーザー情報の編集

ユーザー情報

ユーザー名 **必須**

表示名 **必須**

コンテンツの公開時に、この名前が表示されます。

電子メール **必須**

ウェブサイトURL

ユーザーの個人ホームページのURL。表示する名前とウェブサイトURLは、コンテンツやコメントの

パスワード

パスワードの変更

設定

使用言語

管理画面で使用する言語です。

カスタム形式

ローカライズ (config.yamlで定義)

config.yaml

```
l10n_lexicon:
```

```
  ja:
```

```
    Import CSV Data: CSVデータのインポート
```

```
    Export Content Data with CSV: コンテンツデータのエクスポート (CSV)
```

ローカライズ（YAMLを別ファイルで定義）

- 管理画面やログメッセージの表示言語をユーザー別に最適化

config.yaml

```
l10n_lexicon:  
  ja: l10n_ja.yaml  
  en_us: l10n_en_us.yaml
```

en_usの指定なければl10n_ja.yamlのキーが使われる

l10n_ja.yaml

```
Import CSV Data: CSVデータのインポート  
Export Content Data with CSV: コンテンツデータのエクスポート (CSV)
```

ローカライズ (Perlで定義)

```
plugins/
  HelloWorld/
    config.yaml
```

```
l10n_class: HelloWorld::L10N
```

```
lib/
  HelloWorld/
    L10N/
      L10N.pm →
      ja.pm
```

```
package HelloWorld::L10N;
:
use base 'MT::Plugin::L10N';
:
```

```
package HelloWorld::L10N::ja;
:
use base 'HelloWorld::L10N';
our %Lexicon = (
  'HelloWorld' => 'こんにちは！世界！',
:
);
1;
```


ローカライズ（__transタグ）

config.yaml

```
id: ContentInfoWidget
name: ContentType Information Widget
version: 0.01
author_link: https://www.movabletype.org/
author_name: Six Apart Ltd.
description: <__trans phrase="Show ContentType information on the right sidebar of Edit Template.">
l10n_class: ContentInfoWidget::L10N
```

※"MT_TRANS"や"__TRANS"もOK

ローカライズ（テンプレート）

import_excel.tpl

```
<mt:setvarblock name="page_title">  
  <__trans phrase="Import CSV Data">  
</mt:setvarblock>  
:
```

ダッシュボード > First Weblog > CSVデータのインポート

CSVデータのインポート

インポートするサイト： First Weblog (1)

インポートファイル 選択されていません

コンテンツデータID指定時の処理

指定記事IDがある場合

同一記事IDの記事を上書き

インポートしない

指定記事IDがない場合

ローカライズ（プラグインの設定を利用）

```
：  
<__trans_section component="Workflow"> ← コンポーネント名  
  <span class="">  
    <mtapp:svgicon id="ic_checkbox" title="<__trans phrase="Draft">" size="sm">  
  </span>  
</__trans_section>  
：
```

プラグインの定義を利用

lib/MT/L10N/ja.pm

```
：  
'Draft' => '下書き'  
：
```

plugins/Workflow/lib/Workflow/L10N/ja.pm

```
：  
'Draft' => '原稿'  
：
```

ローカライズ (config.yamlのlabel)

config.yaml

```
list_actions:  
  content_data:  
    export_content_data_with_csv:  
      label: Export Content Data with CSV  
      order: 100  
      handler: $CSVDataImE  
      permission: administ  
      :
```

ダッシュボード > First Weblog > Site Page

Site Pageの管理

⊕ Site Pageを作成 ⊕ Site Pageをインポート ⊕ Site Pageをエクスポート

公開 削除 アクション...

フィルタ: [すべて](#)

- コンテンツデータの公開を取り消し
- コンテンツデータのエクスポート (CSV)

	データ	summary
<input checked="" type="checkbox"/>	test2	test2
<input checked="" type="checkbox"/>	test1	test1

ローカライズ (プログラム)

AddressBook.pm

```
    :  
my $plugin = MT->component('Workflow');  
$param{ page_title } = $plugin->translate('New Address');  
    :
```

コア機能のローカライズを利用

```
    :  
$hoge = MT->translate('xxxxx');  
    :
```

プラグインデータ・プラグイン設定

プラグインデータ

- 既存のテーブルに関連のないデータ、テーブルにフィールドを追加できない場合等

作成と保存

```
use MT::PluginData;

my $data = MT::PluginData->new;
$data->plugin('my-plugin');
$data->key('unique-key');
$data->data($big_data_structure);
$data->save or die $data->errstr;
```

取得

```
use MT::PluginData;

$data = MT::PluginData->load(
    { plugin => 'my-plugin',
      key=> 'unique-key' }
);
my $value = $data->data;
```

プラグイン設定（サンプル）



The screenshot shows a settings window for a plugin named 'HelloWorld'. At the top left is a puzzle piece icon followed by the text 'HelloWorld'. Below this are three tabs: '詳細' (Details), 'リソース' (Resources), and '設定' (Settings). The '設定' tab is selected, indicated by a blue underline. Under the '設定' tab, there is a label 'メッセージ' (Message) followed by an empty text input field. At the bottom, there are two buttons: a blue button labeled '変更を保存' (Save Changes) and a white button labeled '初期化' (Reset).

プラグイン設定 (定義)

```
plugins/  
  HelloWorld/  
    config.yaml
```

```
id: HelloWorld  
name: HelloWorld  
config_template: config.tpl  
blog_config_template: blog_config.tpl
```

システム管理画面用のテンプレート名

サイト管理画面用のテンプレート名

プラグイン設定 (テンプレートの配置)

```
plugins/  
  HelloWorld/  
    config.yaml
```

```
id: HelloWorld  
name: HelloWorld  
config_template: config.tpl  
blog_config_template: blog_config.tpl
```

```
tpl/
```

```
  config.tpl
```

```
  blog_config.tpl
```

システム管理画面用テンプレート

サイト管理画面用テンプレート

プラグイン設定 (テンプレートの中身)

```
plugins/  
  HelloWorld/  
  :  
  tmpl/  
  config.tpl
```

フォームを整形するための
管理画面用テンプレートタグ

```
<mtapp:setting  
  id="message"  
  label="<__trans phrase="message">"  
>  
  <input type="text"  
    id="message"  
    name="message"  
    value="<mt:var name="message">"  
  />  
</mtapp:setting>
```

フォーム本体

プラグイン設定画面

mtapp:settingタグあり

The screenshot shows the settings page for the 'HelloWorld' plugin. At the top, there is a puzzle piece icon followed by the text 'HelloWorld'. Below this, there are three tabs: '詳細', 'リソース', and '設定'. The '設定' tab is selected, indicated by a blue underline. Underneath the tabs, there is a text input field with the label 'message' and a blue button labeled '変更を保存'.

mtapp:settingタグなし

The screenshot shows the settings page for the 'HelloWorld' plugin. At the top, there is a puzzle piece icon followed by the text 'HelloWorld'. Below this, there are three tabs: '詳細', 'リソース', and '設定'. The '設定' tab is selected, indicated by a blue underline. Underneath the tabs, there is an empty text input field and a blue button labeled '変更を保存'.

プラグイン設定 (ローカライズ)

```
plugins/  
  HelloWorld/  
    config.yaml
```

```
id: HelloWorld  
name: HelloWorld  
l10n_lexicon:  
  ja: l10n_ja.yaml  
config_template: config.tmpl  
blog_config_template: blog_config.tmpl
```

```
l10n_ja.yaml
```

```
message: メッセージ
```

プラグイン設定画面（ローカライズ後）



プラグイン設定（初期値設定）

```
plugins/  
  HelloWorld/  
    config.yaml
```

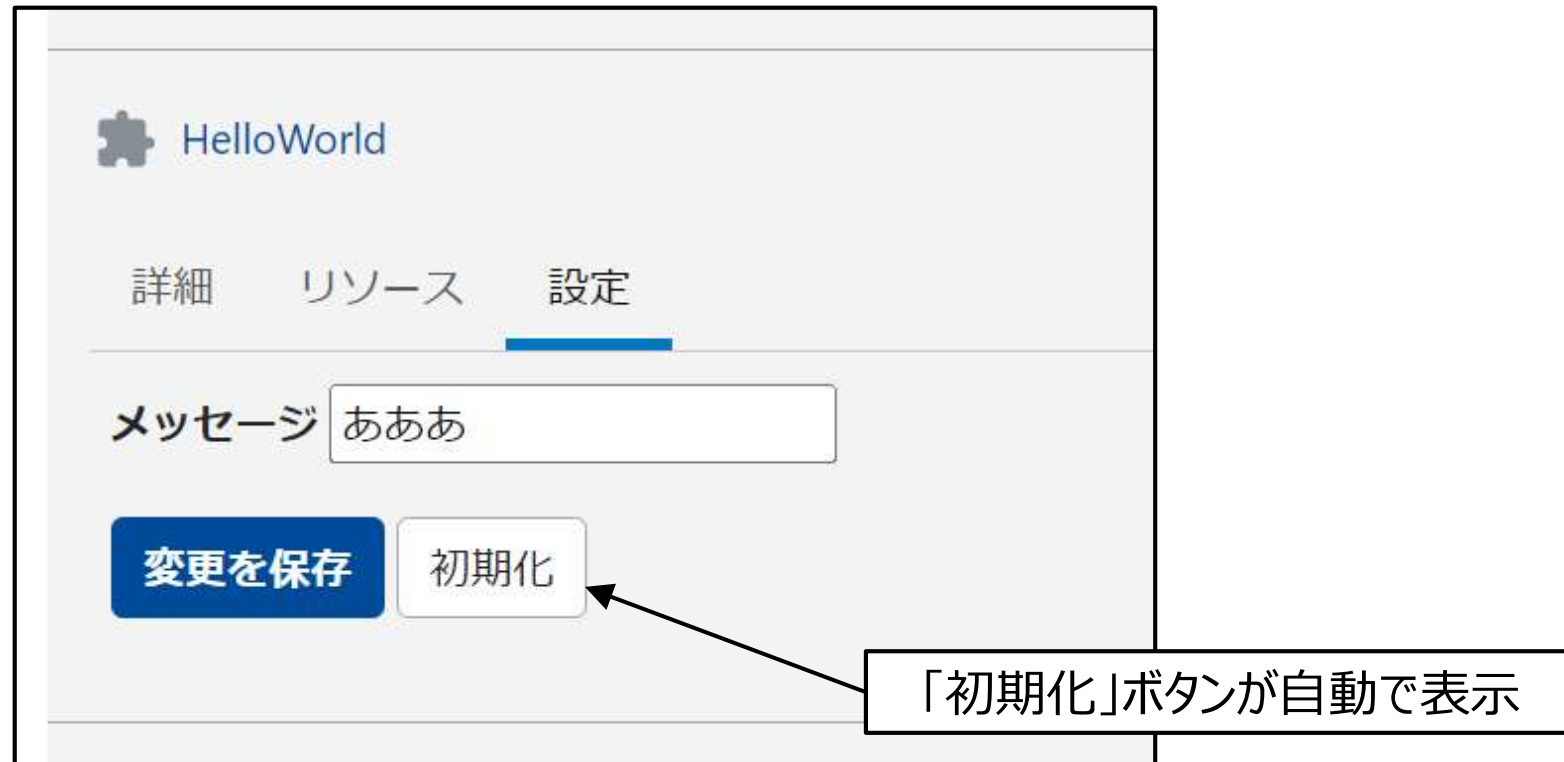
```
id: HelloWorld  
name: HelloWorld  
l10n_lexicon:  
  ja: l10n_ja.yaml  
config_template: config.tpl  
blog_config_template: blog_config.tpl  
settings:  
  message:  
    scope: system  
    default: hoge
```

※空の値でも何か設定しておかないと「変更を保存」押下後に保存されない

プラグイン設定画面（初期値表示）



プラグイン設定画面（初期値変更後）



プラグイン設定をプログラムから読み出す

サンプル：冒頭のHelloWorldタグから出力する値を、プラグイン設定から読み出す

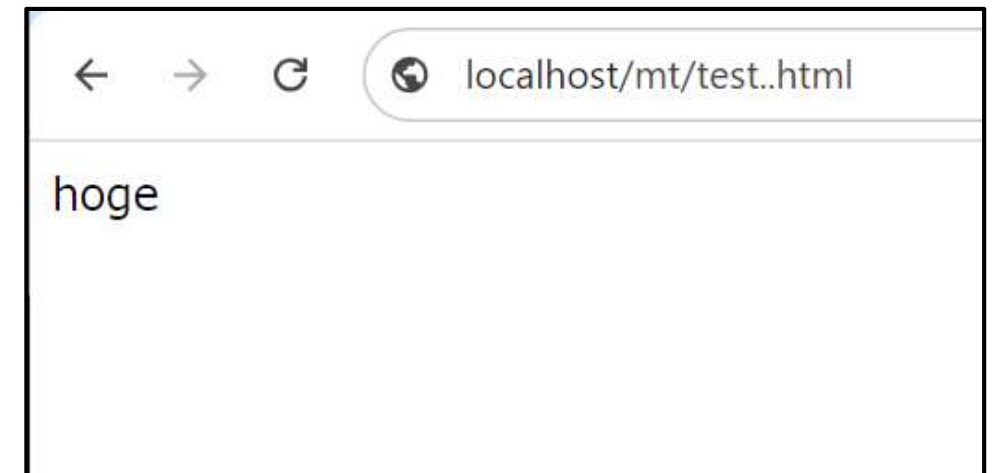
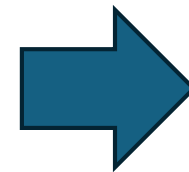
インデックステンプレートの作成

テンプレート名

テンプレートの内容

```
1 <$mt:HelloWorld$>
```

再構築
or
プレビュー



プラグイン設定を読み出す (変更前)

```
plugins/  
  HelloWorld/  
    config.yaml
```

```
id: HelloWorld  
name: HelloWorld  
tags:  
  function:  
    HelloWorld: $HelloWorld::HelloWorld::Tags::hdlr_hello_world
```

```
lib/  
  HelloWorld/  
    Tags.pm
```

```
sub hdlr_hello_world {  
  return "Hello, world!";  
}
```

プラグイン設定を読み出す (変更後)

```
plugins/  
  HelloWorld/  
    config.yaml
```

```
id: HelloWorld  
name: HelloWorld  
tags:  
  function:  
    HelloWorld: $HelloWorld::HelloWorld::Tags::hdlr_hello_world
```

```
lib/  
  HelloWorld/  
    Tags.pm
```

```
sub hdlr_hello_world {  
  my $plugin = MT->component("HelloWorld");  
  my $message = $plugin->get_config_value('message', 'system');  
  return $message;  
}
```

プラグインを指定



コンフィグを指定

プログラムでプラグイン設定に値を設定する

```
plugins/  
  HelloWorld/  
    config.yaml
```

```
callbacks:  
  init_request: $HelloWorld::HelloWorld::CMS::init_request
```

```
lib/  
  HelloWorld/  
    CMS.pm
```

```
:  
sub init_request {  
  my $plugin = MT->component("HelloWorld");  
  $plugin->set_config_value('message', 'aaa');  
}  
:
```

プラグイン設定画面（管理画面アクセス直後）



アプリケーション

<https://www.koikikukan.com/archives/2023/12/24-235555.php>

アプリケーション (Workflow)

上長等がMTログインせずに承認前のページをプレビューする機能

URL例 : <https://user-domain/mt/plugins/Workflow/mt-preview.cgi>

アプリケーション (Workflow)

```
plugins/  
  Workflow/  
    mt-preview.cgi
```

```
#!/usr/bin/perl
```

```
use strict;  
use warnings;  
:
```

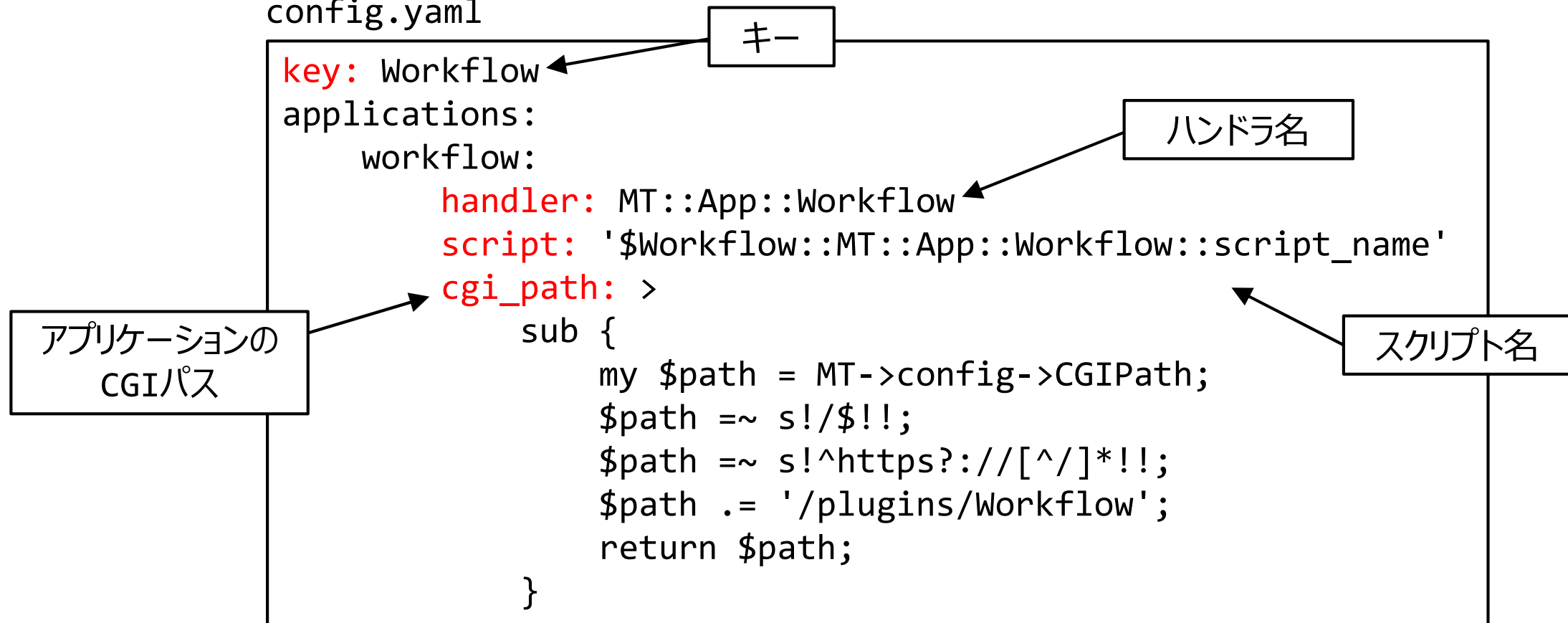
```
use MT::Bootstrap App => 'MT::App::Workflow';
```

キー



アプリケーション (Workflow)

plugins/
Workflow/
config.yaml



アプリケーション (Workflow)

plugins/Workflow/lib/MT/App/Workflow.pm

```
package MT::App::Workflow;
use strict;
use base qw( MT::App );
sub id { 'workflow' }
sub script_name { 'mt-preview.cgi' }
sub init {
    my $app = shift;
    $app->SUPER::init(@_);
    $app->add_methods( preview => \&preview );
    $app->{ default_mode } = 'main';
    $app;
}
sub init_request {
    my $app = shift;
    $app->SUPER::init_request(@_);
    $app->{requires_login} = 0;
}
sub preview {
    my $app = shift;
    :
}
1;
```

MT::Appを継承

idを実装

script_nameを実装

メソッド追加

起動したいメソッドの実装

データベースにテーブルまたはフィールド追加

テーブル追加

config.yaml

```
id: HelloWorld
name: HelloWorld
schema_version: 1.0
object_types:
  addressbook_workflow: MT::AddressBook
```

プラグインのスキーマバージョン
(データ構造の変更時は値を更新すること)

データソース名

対応するクラス名
(ここにテーブル定義を記述)

テーブル追加

lib/MT/AddressBook.pm

```

package MT::AddressBook;
use strict;
use base qw( MT::Object );

__PACKAGE__->install_properties({
  column_defs => {
    'id' => 'integer not null auto_increment',
    'blog_id' => 'integer not null',
    'status' => {
      type => 'smallint',
      label => 'Status',
    },
    'name' => {
      type => 'string',
      size => 255,
    },
  },
  'email' => {
    type => 'string',
    size => 255,
    label => 'Email Address',
  },
  'description' => {
    type => 'text',
  },
  indexes => {
    id => 1,
  },
  datasource => 'addressbook_workflow',
  primary_key => 'id',
});
1;

```

MT::Objectを
継承

フィールド名

型

サイズ

フィールド
定義

インデックステーブルのフィールド

プライマリーキー
の指定

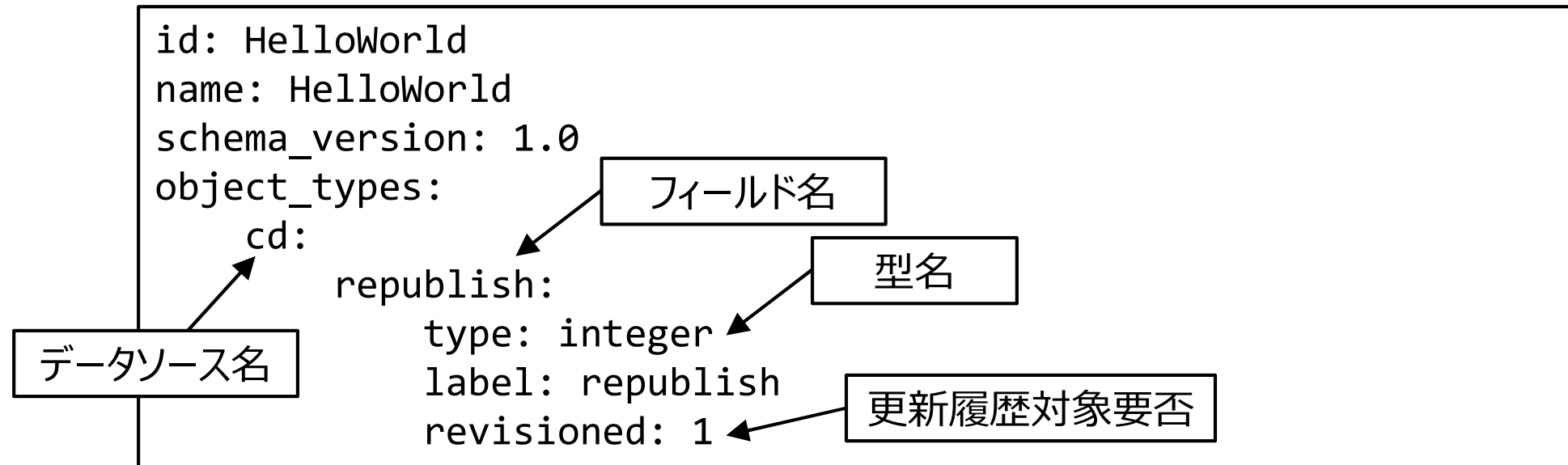
データソース名

テーブルデータへのアクセス

```
use MT::AddressBook;
:
sub sample {
    my $address = MT::AddressBook->load(
        { id => $id, blog_id => $q->param('blog_id') }
    );
    my $name = $address->name;
    my $email = $address->email;
}
```

フィールド追加

config.yaml



アクセス方法は他のフィールドと同じ

プラグイン資材

資材（画像・CSS・JavaScript等）の配置

```
/var/  
  www/  
    cgi-bin/  
      mt/  
        plugins/  
          HelloWorld/  
            config.yaml  
              :  
html/  
  mt-static/  
    plugins/  
      HelloWorld/  
        images/  
          hello.png  
        css/  
          hello.css  
        js/  
          hello.js
```

```
<script type="text/javascript"  
  src="<$mt:var name="static_uri"$>plugins/HelloWorld/js/hello.js">  
</script>
```

デバッグ

デバッグ

プログラムエラーは大体システムログに出力される

ログ

時刻はすべてGMT+09:00です。
⚙️ ログを消去 📄 ログをダウンロード(CSV)

削除

フィルタ: [すべてのログ](#)

<input type="checkbox"/>	作成日	ログ	サイト名
<input type="checkbox"/>	直前	HelloWorldでエラーが発生しました: failed loading package HelloWorld::CMS for routine HelloWorld::CMS::init_app: syntax error at /var/www/cgi-bin/wk_araki/mt/MT7-R5502/plugins/HelloWorld/lib/HelloWorld/CMS.pm line 33, near "aaaa sub frobnitzes " syntax error at /var/www/cgi-bin/wk_araki/mt/MT7-R5502/plugins/HelloWorld/lib/HelloWorld/CMS.pm line 36, near "}" Can't use global @_ in "my" at /var/www/cgi-bin/wk_araki/mt/MT7-R5502/plugins/HelloWorld/lib/HelloWorld/CMS.pm line 39, near "= @ " Can't use global @_ in "my" at /var/www/cgi-bin/wk_araki/	システム

デバッグ

mt-conig.cgiのDebugMode利用

```
mt-config.cgi
```

```
⋮  
DebugMode 1
```

- 1 - デバッグメッセージを表示
- 2 - スタックトレースメッセージ
- 4 - Data::ObjectDriver で発生したクエリ
- 8 - 構築に 1/4 秒以上かかったテンプレートのレポート
- 128 - APLレベルのリクエスト/レスポンス情報を標準エラーへ出力（大量に出力するので常時設定は非推奨）

組み合わせも可能。

例：1と2と4を表示するには7を設定

参考：<https://www.movabletype.jp/documentation/appendices/config-directives/debugmode.html>

デバグ

- システムログに出力

```
MT->instance->log("hoge");
```

ハッシュや配列等

```
use Data::Dumper;
```

```
MT->instance->log(Dumper($data));
```

- システムログを時間順にソートするLogSupplementalsプラグイン
<https://www.koikikukan.com/archives/2013/07/10-015555.php>
- Movable Typeでデバグを捗らせる
<https://tec.toi-planning.net/mt-column/tips/how-to-debug-mt/>

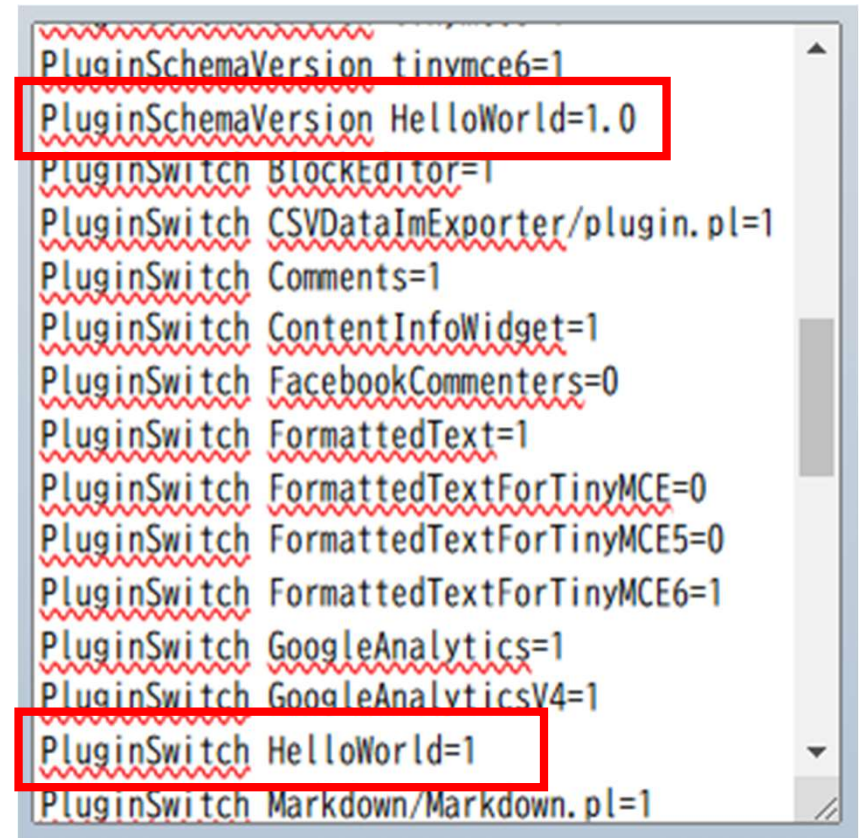
ノウハウ

プラグインの管理情報をクリアしたい

phpMyAdminでプラグイン管理データを削除



削除



削除

元の管理画面に戻したい

```
methods:  
  save_snapshot_wf: $Workflow::Workflow::App::CMS::save_snapshot_wf
```

```
sub save_snapshot_wf {  
  my $app = shift;  
  :  
  $app->call_return;  
}
```

管理画面にメッセージを表示したい (1/3)

例：Workflowで公開後の下書き保存

Site Pageの編集

更新を保存しました (変更内容は公開されません) ×

データ識別ラベル **必須**

test123

このデータを識別するラベルを入力します

Title **必須**

test1

管理画面にメッセージを表示したい (2/3)

config.yaml

```
methods:  
  save_snapshot_wf: $Workflow::Workflow::App::CMS::save_snapshot_wf  
callbacks:  
  MT::App::CMS::template_source.edit_content_data:  
    code: $Workflow::Workflow::CMS::ContentData::add_status  
  MT::App::CMS::template_param.edit_content_data:  
    code: $Workflow::Workflow::CMS::ContentData::set_entry_param
```

「保存」クリック時に起動

```
graph TD; A["「保存」クリック時に起動"] --> B["save_snapshot_wf"]; B --- C["MT::App::CMS::template_source.edit_content_data"]; C --- D["MT::App::CMS::template_param.edit_content_data"]; C --- E["画面表示時に起動"]; D --- F["画面表示時に起動"];
```

画面表示時に起動

画面表示時に起動

管理画面にメッセージを表示したい (3/3)

管理画面上部にメッセージを
表示するテンプレートタグ

CMS.pm

```
sub save_snapshot_wf {  
  my $app = shift;  
  :  
  $app->add_return_arg('saved_snapshot' => 1);  
  $app->call_return;  
}
```

ContentData.pm

```
sub add_status {  
  :  
  $$template =  
  <mt:if name="saved_snapshot">  
  <mtapp:statusmsg>  
  更新を保存しました (変更内容は公開されません)  
  </mtapp:statusmsg>  
  </mt:if>  
  :  
}  
  
sub set_entry_param {  
  my $app = shift;  
  :  
  $param->{'saved_snapshot'} = 1  
  if $q->param('saved_snapshot');  
  :  
}
```

パフォーマンス

不要なプラグインは無効または削除しましょう

システム

サイト

アセット

ユーザー

グループ

ロール

デザイン

カスタムフィールド

フィルタ

設定

全般

ユーザー

Webサービス

プラグイン

再構築トリガー

システム情報

ツール

mtbook

ダッシュボード > システム > プラグイン設定

プラグイン設定

プラグインを探す

プラグイン

BlockEditor 1.11	<input checked="" type="checkbox"/> 有効 <input type="checkbox"/> 無効
Comments 1.13	<input checked="" type="checkbox"/> 有効 <input type="checkbox"/> 無効
ContentType Information Widget 0.01	<input checked="" type="checkbox"/> 有効 <input type="checkbox"/> 無効
FacebookCommenters	<input type="checkbox"/> 無効 <input checked="" type="checkbox"/> 有効
FormattedText 1.4	<input checked="" type="checkbox"/> 有効 <input type="checkbox"/> 無効
FormattedTextForTinyMCE	<input type="checkbox"/> 無効 <input checked="" type="checkbox"/> 有効
FormattedTextForTinyMCE5	<input type="checkbox"/> 無効 <input checked="" type="checkbox"/> 有効
FormattedTextForTinyMCE6 6.0.1	<input checked="" type="checkbox"/> 有効 <input type="checkbox"/> 無効
GoogleAnalytics 1.3	<input checked="" type="checkbox"/> 有効 <input type="checkbox"/> 無効
GoogleAnalyticsV4 1.0	<input checked="" type="checkbox"/> 有効 <input type="checkbox"/> 無効

不要なアドオンは削除しましょう
addons/Commercial.pack

失敗談

HTTPリクエストが2回起動される

原因：オートセーブが実行されたため

```
if ( $app->param('_autosave') ) {  
    return;  
}
```

オブジェクトのメソッドが存在しないエラー

原因：useまたはrequireもれ

```
my $content_type = MT::ContentType->load($content_type_id);
```



```
require MT::ContentType;  
my $content_type = MT::ContentType->load($content_type_id);
```

または

```
use MT::ContentType;  
:  
my $content_type = MT::ContentType->load($content_type_id);
```

use：実行時最初に読み込まれる

require：スクリプト実行時逐次呼ばれる

オブジェクトのメソッドが存在しないエラー

```
my $content_type = MT::ContentType->load($content_type_id);
```



```
my $content_type = MT::model('content_type')->load($content_type_id);
```

MT/Core.pm

```
:
object_types => {
    'entry'           => 'MT::Entry',
    'author'         => 'MT::Author',
    'asset'          => 'MT::Asset',
    :
    'content_type'   => 'MT::ContentType',
    :
```

プラグイン・プラグイン開発関連サイト

公式プラグインディレクトリ

<https://plugins.movabletype.jp/>

The screenshot shows the homepage of the Movable Type Plugins And Themes Directory. The header includes the site name and navigation links for 'サインイン' and 'サインアップ'. Below the header is a search bar with the placeholder text 'Plugin, Theme name or description' and a '検索' button. A '登録する' button is also visible. The main content area features three columns of text: 'GitHubと一緒に使えます', '公認で良いモノを集めました', and 'インスピレーションの宝庫'. Below this is a 'キーワード一覧' section with various tags like 'MT6 Ready (101)', 'MT7 Ready (57)', 'MT8 Ready (35)', 'MTクラウド対応 (70)', 'Data API (13)', 'EC (2)', 'MTタグ・テンプレート (15)', 'アイテム (12)', 'アプリ (3)', 'ウェブサービス (11)', 'カスタムフィールド (7)', 'コミュニティ (2)', 'コメント・トラックバック (2)', 'コンテンツタイプ (18)', 'ソーシャル (3)', 'ダイナミック・パブリッシング (5)', 'デザイン (9)', 'モバイル・スマートフォン (4)', 'ユーティリティ (27)', '再構築 (10)', '検索 (3)', '管理画面 (57)', '編集 (12)', '記事とウェブページ (45)', '開発 (9)', 'テーマ: パーソナルブログ向け (9)', 'テーマ: ビジネス向け (7)', 'テーマ: モバイル・スマートフォン (10)', and 'テーマ: レスポンシブ対応 (12)'. The '最近の注目作品' section highlights two items: 'SetDefaultCategory' (a plugin) and 'Simple Corporate' (a theme).

Movable Type Plugins And Themes Directory [サインイン](#) | [サインアップ](#)

作品を探す ヘルプ

Build Best MT Ever.

MTをもっと楽しく使うための工具箱

GitHubと一緒に使えます

ディレクトリの面倒な管理は不要です。登録時に [GitHub](#) と連携させることで、自動的に情報を更新することができます。手軽に多くのユーザーへ告知することができます。

公認で良いモノを集めました

検索してもなかなか良いツールが見つからず困ったことはありませんか？ディレクトリでは Six Apart が公認する高品質のプラグインとテーマだけ集めました。

インスピレーションの宝庫

Movable Type を拡張する様々なツールをみることで、新しいアイデアが生まれることがあります。「MTってこんなことも出来るんだ」という驚きを見つけてください。

キーワード一覧

MT6 Ready (101) **MT7 Ready (57)** MT8 Ready (35) **MTクラウド対応 (70)** Data API (13) EC (2) MTタグ・テンプレート (15)

アイテム (12) アプリ (3) ウェブサービス (11) カスタムフィールド (7) コミュニティ (2) コメント・トラックバック (2)


コンテンツタイプ (18) ソーシャル (3) ダイナミック・パブリッシング (5) デザイン (9) モバイル・スマートフォン (4) ユーティリティ (27)

再構築 (10) 検索 (3) 管理画面 (57) 編集 (12) 記事とウェブページ (45) 開発 (9) テーマ: パーソナルブログ向け (9)


テーマ: ビジネス向け (7) テーマ: モバイル・スマートフォン (10) テーマ: レスポンシブ対応 (12)

最近の注目作品

プラグイン

 [SetDefaultCategory](#)
SetDefaultCategory は Movable Type の記事にデフォルトカテゴリを設定することができるプラグインです。

テーマ

 [Simple Corporate](#)
「Simple Corporate」は、シンプルな企業サイトを簡単に作成することができる、レスポンシブデザインのテーマです。

プラグイン開発向けドキュメント

<https://github.com/movabletype/Documentation/wiki/Japanese-developer-guide>

The screenshot shows the GitHub repository page for the Movable Type Japanese developer guide. The page title is "Japanese developer guide" and it was edited by yuji on Feb 20, 2013. The main heading is "Movable Type 開発者向けガイド" (Movable Type Developer Guide). Below this, there is a sub-heading "プラグイン開発ガイド" (Plugin Development Guide) and a paragraph explaining that plugins are small software that add unique functionality to Movable Type. A list of 15 links provides a table of contents for the guide, covering topics from first steps to creating custom objects. On the right side, there is a sidebar with a search bar and a list of 315 pages, including "Home", "Applying Filters to Titles", "Asset", "Auto expire entries", "Better website blog integration", "Breadcrumbs", "Canonical Google Sitemap template", "Category Archive Listing", "Category, Folder", "Category:Template Tag Recipes", "Chart api area charts", "Chart api bar charts", "Chart api donut charts", and "Chart api getting started".

Product Solutions Resources Open Source Enterprise Pricing Search or jump to... Sign in Sign up

movabletype / Documentation Public Notifications Fork 9 Star 85

<> Code Issues 1 Pull requests Actions Projects Wiki Security Insights

Japanese developer guide

yuji edited this page on Feb 20, 2013 · 23 revisions

Movable Type 開発者向けガイド

開発者向けのドキュメントです。

プラグイン開発ガイド

プラグインは、Movable Typeに独自の機能を追加する小さなソフトウェアです。Movable Typeのカスタマイズの中核ともいえるプラグインの開発方法を紹介します。

- [プラグイン開発のためのファーストステップ](#)
- [レジストリ、YAMLについて](#)
- [環境変数について](#)
- [プラグインのローカライゼーションについて](#)
- [テストドリブンでのプラグインの開発について](#)
- [グローバル・モディファイアプラグインの開発について](#)
- [ファンクションタグ プラグインの開発について](#)
- [ブロックタグ プラグインの開発について](#)
- [コンディショナルタグ プラグインの開発について](#)
- [プラグインのデバッグ](#)
- [プラグインの設定方法](#)
- [コールバックとフックポイント](#)
- [スケジュールタスクの開発](#)
- [MTオブジェクトの利用方法](#)
- [独自オブジェクトの作成](#)

Pages 315

Find a page...

- Home
- Applying Filters to Titles
- Asset
- Auto expire entries
- Better website blog integration
- Breadcrumbs
- Canonical Google Sitemap template
- Category Archive Listing
- Category, Folder
- Category:Template Tag Recipes
- Chart api area charts
- Chart api bar charts
- Chart api donut charts
- Chart api getting started

DataAPIリファレンス (v6)

<https://movabletype.github.io/mt-docs-data-api-reference/v6.html>

The screenshot displays the documentation for the Movable Type Data API (v6). The main content area is titled "Assets" and includes a section for "Upload a file" with "Permissions" (upload) and "AUTHORIZATIONS" (mtauth). The "QUERY PARAMETERS" section lists the "overwrite_once" parameter, which is an integer with an enum of 0 or 1. The "REQUEST BODY SCHEMA" is defined as "multipart/form-data" and includes the "autoRenameIfExists" parameter, which is an integer with a default of 0 and an enum of 0 or 1. The right-hand panel shows the "POST /assets/upload" endpoint with response samples for status codes 200, 400, 401, 404, 409, and 413. The content type is "application/json", and a sample response is shown with fields for "blog", "class", "createdBy", "id", "userpicUrl", "createdDate", and "customFields".

小粋空間プラグイン一覧

<https://www.koikikukan.com/movabletype/plugin/>

小粋空間でリリース中のMovableType/WordPressプラグイン

MovableTypeプラグイン

[CSVAuthorDataImExporter](#)

主な機能

- CSV形式でMovable Typeのユーザーデータのエクスポート・インポートが可能です。

[CSVDataImExporter](#)

主な機能

- CSV形式でMovable Typeのコンテンツデータ・記事・ウェブページのエクスポート・インポートが可能です。
- インポート時のデータ更新は、コンテンツデータID/記事IDによる指定の他、ラベルや出力ファイル名による指定が可能です。これにより、常にインポートだけを行なう運用が可能です。
- 一覧画面から選択してのエクスポートが可能です。
- 記事・ウェブページは、カテゴリ・フォルダ・関連アイテム・カスタムフィールドに対応します（グレードによって対応機能が異なります）。
- システム管理画面からサイト指定によるコンテンツデータのエクスポート・インポートが可能です。（機能拡張版）
- ロール編集画面に「コンテンツデータのインポート・エクスポート」権限を追加し、ロール別にコンテンツデータのインポート・エクスポートの有効・無効の設定が可能です。（機能拡張版）
- コンテンツデータ別の有効・無効の設定が可能です。（機能拡張版）

[ListingFieldEditor](#)

主な機能

- コンテンツデータ・カスタムフィールドのデータを一覧画面で編集できます。保存はAjaxで行います。

[PowerEditContentData](#)

主な機能

ご清聴ありがとうございました