hypeJunction Documentation

Release 3.0

Ismayil Khayredinov

Nov 29, 2018

Contents

Build modern social media applications with an ever-growing collection of high-quality plugins for Elgg 3+

CHAPTER 1

Installation

1.1 Using ZIP files

Distribution (.zip) plugin files can be downloaded from hypeJunction website. To add a plugin, unzip it into your `/mod` directory, making sure the directories are not nested, and that `manifest.xml` is in the root of the plugin folder.

1.2 Using Composer

To use composer, you need to have an active subscription that gives you access to individual packages.

```
## Add your satis credentials to global config
## You can find your username and token by logging into https://hypejunction.com/ and␣
↪navigating to one of the downloads.

composer config --global --auth http-basic.hypejunction.com <username> <token>
```

Update your `composer.json` to include:

```
"repositories": [
    {
        "type": "composer",
        "url": "https://hypejunction.com/satis"
    }
]
```

1.2.1 New Project

The easiest way to create an Elgg project with hypeJunction plugins installed is by running the following commands.

```
## Create a new Elgg project with hypeJunction Pro plugins pre-installed

composer create-project hypejunction/hypejunction:dev-master ./project-name

cd ./project-name

composer install
composer install # 2nd call is currently required

./elgg-cli install
./elgg-cli plugins:activate hypeCli
./elgg-cli hypejunction:install
```

1.2.2 Existing Project

You can add plugins to an existing project, using

```
composer require hypejunction/<plugin_name>
```

1.3 Additional Info

Learn about installation in Elgg docs.

CHAPTER 2

hypeActivity

Replaces profile pages with an activity stream

- Activity streams as main attribute of the profile page
- Extendable with custom modules/widgets

CHAPTER 3

hypeAjax

Utilities for AJAX requests

- Deferred view rendering

3.1 Developer Notes

3.1.1 Deferred Rendering

To defer view rendering, simply add `'deferred' => true` to view vars.

```
echo elgg_view('my_view', [
   // tells the view system to defer view render
   'deferred' => true,

   // if set to false, placeholder will not be rendered
   // if set to a value, that value will be used as the placeholder
   // if not set, default ajax loader will be used
   'placeholder' => false,

   // you can pass other view vars, as you would with normal views
   // various Elgg data will be serialized and available to the deferred view
   // some of the values may need to be wrapped into a Serializable instance
   'entity' => get_entity(123),
   'user' => get_user(234),
]);
```

CHAPTER 4

hypeAttachments

- API and UI for attaching files and other entities
- Form input for uploading file attachments
- Views for displaying attachments

4.1 Register allowed subtypes

If you add your entity subtype to a list of entities supporting attachments, the plugin will attempt to create all of the UI, necessary to upload and display attachments:

```
elgg_register_plugin_hook_handler('allow_attachments', 'object:my_subtype',
↪'\Elgg\Values::getTrue');
```

Note that this generic approach might not work with all plugins, and may require additional customizations on your side.

4.2 Display an attachment input

```
echo elgg_view('input/attachments');
```

To add an attachments input to your comment and discussion replies forms, use the following code. You will not need to add any code to your save action.

```
echo elgg_view('input/attachments', [
    'name' => 'comment_attachments',
]);
```

To add an attachments input to your personal messages and replies forms, use the following code. You will not need to add any code to your save action.

```
echo elgg_view('input/attachments', [
   'name' => 'message_attachments',
]);
```

Note that if you are not using *hypeDropzone*, your form must have it's encoding set to `multipart/form-data`.

4.3 Attach uploaded files in an action

```
hypeapps_attach_uploaded_files($entity, 'upload', [
   'access_id' => $entity->access_id, // change the access level of uploaded files
]);
```

4.4 Attach an object

```
hypeapps_attach($entity, $attachment);
```

4.5 Display attachments

```
echo elgg_view('output/attachments', [
   'entity' => $entity,
]);
```

4.6 Acknowledgements

- Early version of the plugin development has been partially sponsored by [Social Business World](https://socialbusinessworld.org/)

CHAPTER 5

hypeAutocomplete

Autocomplete input

- Replaces selects with autocomplete inputs that support images and icons
- Adds typeahead support for tags input
- Replaces autocomplete and userpicker inputs
- Adds versatile *input/guids*

CHAPTER 6

hypeBlog

Customized blog

- Uses extendable form interface
- Rich-media oriented

CHAPTER 7

hypeBraintreePayments

A wrapper for Braintree's PHP SDK

7.1 Webhooks

Configure your Braintree application to send webhooks to `` `https://<your-elgg-site>/payments/braintree/webhooks` ``

To digest a webhook, register a plugin hook handler:

```
elgg_register_plugin_hook_handler('subscription_went_past_due', 'braintree',␣
↪HandleExpiredSubscription::class);

class HandleExpiredSubscription {
   public function __invoke(\Elgg\Hook $hook) {
      $webhook = $hook->getParam('webhook');
      /* @var $webhook \Briantree\WebhookNotification */

      // ... do stuff

      return $result; // data to send back to braintree
   }
}
```

7.2 Card Input

To display a card input:

```
// Card number, expiry and CVC
echo elgg_view_field([
   '#type' => 'braintree/card',
```

(continues on next page)

(continued from previous page)

```
    '#label' => 'Credit or Debit Card',
    'required' => true,
]);
```

You can then retrieve the value of the Braintree token in your action:

```
$token = get_input('braintree_token'); // Corresponds to payment_method_nonce

elgg()->{'payments.gateways.braintree'}->pay($transaction, [
    'braintree_token' => $token,
]);
```

CHAPTER 8

hypeBraintreeSubscriptions

Braintree integration for hypeSubscriptions

8.1 Currencies

If you are using multiple currencies, make sure to set up a Braintree merchant account for each currency.

8.2 Subscriptions

Braintree API does not allow plans to be created via API calls, so you will need to setup your plans via Braintree control panel and then import them. Plans will be matched by their plan_id, so you can duplicate your existing plans on Braintree.

CHAPTER 9

hypeCapabilities

Capabilities and roles API

9.1 Registering a role

```
elgg()->roles->register('role_name');
```

9.2 Assigning a role to a user

```
// Site wide role
elgg()->roles->assign('role_name', $user);

// Group specific role
elgg()->roles->assign('role_name', $user, $group);
```

9.3 Remove a role from a user

```
// Site wide role
elgg()->roles->unassign('role_name', $user);

// Group specific role
elgg()->roles->unassign('role_name', $user, $group);
```

9.4 Configuring role permissions

9.4.1 Creating entities of a specific type

```
// Prevent users with given role from creating entities of a given type
elgg()->roles->role_name->onCreate('object', 'blog', Role::DENY);

// Allow users to create entities of a given type regardless of context
elgg()->roles->role_name->onCreate('object', 'blog', Role::ALLOW, Role::OVERRIDE);

// Allow users to create entities of a given type if all other container permissins␣
↪are met
elgg()->roles->role_name->onCreate('object', 'blog', Role::ALLOW, Role::STACK);

// Allow users to create entities when specific conditions are met
// Only allow group blogs
elgg()->roles->role_name->onCreate('object', 'blog', Role::DENY,␣
↪function(\hypeJunction\Capabilities\Context $context) {
   $container = $context->getTarget();
   if (!$container instanceof ElggGroup) {
      return Role::DENY;
   }
});
```

9.4.2 Update and delete permissions

Similar to above, you can use `onUpdate` and `onDelete` methods;

9.4.3 Granting administrative permissions

Administrative permissions imply high-level administrative action on entities, e.g. approving a certain post after moderation. By default, core does not use this privilege level, but you can check if the user has admin permissions over an entity like so:

```
$params = [
   'entity' => $entity,
   'user' => $user,
];
if (!elgg_trigger_plugin_hook('permissions_check:administer', "$entity->type:$entity->
↪subtype", $params, false)) {
   // No permissions to approve
   throw new EntityPermissionsException();
}

// Do something that requires high level permissions, e.g.
$entity->published_status = 'published';
```

Granting/denying admin permissions

```
// Prevent users with given role from creating entities of a given type
// Allow moderator role to administer all blogs regardless of owner/container
elgg()->roles->moderator->onAdminister('object', 'blog', Role::ALLOW, Role::OVERRIDE);

```

(continues on next page)

(continued from previous page)

```
// Allow users to create entities when specific conditions are met
// Allow teacher to administer all group blogs
elgg()->roles->teacher->canAdminister('object', 'blog', Role::ALLOW,␣
↪function(\hypeJunction\Capabilities\Context $context) {
   $entity = $context->getTarget();
   $actor = $context->getActor();

   $container = $entity->getContainerEntity();
   return $container->canEdit($actor->guid);
});
```

9.4.4 Routes

You can allow/deny access to certain routes by route name

```
// Context parameter contain matched route elements
// e.g. prevent access to user profile if users are not friends
elgg()->roles->user->onRouteAccess('view:user', Role::DENY,␣
↪function(\hypeJunction\Capabilities\Context $context) {
   $actor = $context->getActor();

   $username = $context->getParam('username');
   $user = get_user_by_username($username);

   if (!$actor || !$user instanceof ElggUser || !$actor->isFriendOf($user->guid)) {
      register_error('You must be friends to access user profiles');
      return Role::DENY;
   }
});

// Here is an example of how to prevent access to member pages to non-logged in users:
elgg()->roles->guest->onRouteAccess('collection:user:user', Role::DENY);
elgg()->roles->guest->onRouteAccess('collection:user:user:alpha', Role::DENY);
elgg()->roles->guest->onRouteAccess('collection:user:user:newest', Role::DENY);
elgg()->roles->guest->onRouteAccess('collection:user:user:online', Role::DENY);
elgg()->roles->guest->onRouteAccess('collection:user:user:popular', Role::DENY);
elgg()->roles->guest->onRouteAccess('search:user:user', Role::DENY);
elgg()->roles->guest->onRouteAccess('view:user', Role::DENY);
```

9.4.5 Custom (component) capabilities

You can check and alter custom capabilities:

```
// Check a custom role
elgg()->roles->can('read', 'discussions');

// Define how role responds to capability check
elgg()->roles->guest->on('read', 'discussions', Role::DENY);

// Override role response
elgg_register_plugin_hook_handler('capability', 'read:discussions', function(Hook
↪$hook) {

});
```

CHAPTER 10

hypeCaptcha

Protects the site from bots using reCaptcha

10.1 Usage

```
echo elgg_view_field(['#type' => 'captcha']);
```

Users are only requested to solve a captcha one. If they proved they are human, they won't be bothered again.

CHAPTER 11

hypeCli

CLI tools for working with hypeJunction plugins

11.1 Commands

11.1.1 `elgg-cli hypejunction:install [--pack|-p]`

Install all plugins within a pack.

Packs include:

`core` - plugins that most other plugins depend on `content` - plugins for authoring content `tools` - various site tools `theming` - theming tools `commerce` - payments and subscriptions

CHAPTER 12

hypeCountries

Country utilities

- Country info, including name, ISO, ISO-3, ISO numeric, FIPS, TLD, currency code, postal code format and other
- Country input view

12.1 Country select

```
echo elgg_view('input/country', array(
   'name' => 'country',
   'value' => 'CZ',
));
```

12.2 List countries

```
$countries = elgg()->countries->getCountries();
foreach ($countries as $country) {
   /* @var $country \hypeJunction\Country */
   echo "$country->name ($country->iso)";
}
```

12.3 Country info

Get a list of countries with extended details

```
$fields = array(
   'name',
   'iso',
   'iso3',
   //'iso_numeric',
   //'fips',
   'capital',
   //'area',
   //'population',
   //'continent',
   'tld',
   'currency_code',
   'currency_name',
   'phone_code',
   'postal_code_format',
   'postal_code_regex',
   'languages',
   //'geoname_id',
   'neighbours'
);

// Get a list of countries ordered by currency_code
$countries = elgg_get_country_info($fields, 'currency_code');
```

CHAPTER 13

hypeDiscussions

- Implements threaded discussions (using comment system)
- Integrates with hypeInteractions for real time updates
- Enables replies from river
- Adds attachments to discussions (requires hypeAttachments)
- Adds discussions widget to the group and user profile
- Restricted discussion (only group admins can start new discussions)
- Enables discussions outside of groups

CHAPTER 14

hypeDownloads

Digital product downloads

CHAPTER 15

hypeDraft

Implements a publishing workflow for content items.

- Allow users to save draft posts hidden from other users
- Autosave posts
- Maintain revision history

CHAPTER 16

hypeDropzone

Drag&Drop File Uploads for Elgg

- Cross-browser support for drag&drop file uploads
- Easy to integrate into existing forms
- Supports chunked uploads of large files

16.1 Adding a drag&drop file input and processing uploads

To add a drag&drop input to your form, add the following:

```
echo elgg_view('input/dropzone', array(
      'name' => 'upload_guids',
      'accept' => "image/*",
      'max' => 25,
      'multiple' => true,
      'container_guid' => $container_guid, // optional file container
      'subtype' => $subtype, // subtype of the file entities to be created
      // see the view for more options
));
```

In your action, you can retrieve uploaded files with `` `get_input('upload_guids');` ``

You also need to implement a fallback solution for when the browser does not support drag and drop. Check `hypeJunction\DropzoneService` for an example.

16.2 Initializing and resetting dropzone

You can instantiate and clear dropzone by triggering jQuery events on the containing form:

```
$('.elgg-form').trigger('initialize'); // will instantiate dropzone inputs contained␣
↪within the form
$('.elgg-form').trigger('reset'); // will clear previews and hidden guid inputs
```

16.3 Acknowledgements

- Dropzone.js is a really cool library by Matias Meno

http://www.dropzonejs.com/

CHAPTER 17

hypeEmbed

- Search, upload and embed files on the spot
- Search and embed all other registered object types on the spot
- Embed URL previews and rich-media players
- [admin] Embed buttons that match the site styles
- [admin] Embed “insecure” HTML embeds (forms, calendars etc)

17.1 Shortcodes

The plugin supports the following shortcodes:

`embed` shortcode:

- `guid` - GUID of an entity to embed

`button` shortcode:

- `text` - call to action
- `type` - One of the following types *action*, *submit*, *delete*, *cancel* (these values only affect styling and do not carry any functional value)
- `url` - URL to link to
- `target` - Default *self*, *blank* or *lightbox*

Examples:

```
[embed guid="555"]
[button type="action" text="Read Terms" url="/terms" target="lightbox"]
```

Unlisted shortcode attributes will be parsed and passed to the view after sanitization, so extending plugins can add additional options.

By default, only shortcodes passed to *output/longtext* view will be expanded automatically.

17.2 Static assets

If you are using the same images across multiple posts, you may way to use static assets, as they allow you to take advantage of simplecache, thus offering better performance than file entities.

Create a folder in your dataroot */embed/* and place your image files in there, flush the caches, and you will see your images in the Assets tab of the embed lightbox window.

17.3 Acknowledgements

• Upgrade for Elgg 2.3 has been sponsored by ApostleTree, LLC

CHAPTER 18

hypeGroups

- Extended search and sort functionality
- API to add new group subtypes
- API to manage group hierarchies
- API for managing group fields
- API for restricting group tools, as well as using preset tools

18.1 Group Subtypes

Registering new subtypes and configuring them is made easy.

Here is an example of how to remove groups from the top level of the site, and making them subgroups of a new subtype called classroom.

```
$svc = elgg()->groups;
/* @var $svc \hypeJunction\Groups\GroupsService */

$svc->registerSubtype('classroom', [
        'labels' => [
                'en' => [
                        'item' => 'Classroom',
                        'collection' => 'Classrooms',
                ],
        ],
        'root' => true,
        'identifier' => 'classrooms',
        'class' => \CustomPlugin\Classroom::class,
        'collections' => [
                'all' => \CustomPlugin\DefaultClassroomCollection::class,
                'owner' => \CustomPlugin\OwnedClassroomCollection::class,
                'member' => \CustomPlugin\JoinedClassroomCollection::class,
```

(continues on next page)

(continued from previous page)

```
            ],
    ]);

    $svc->registerSubtype('group', [
  'site_menu' => false,
            'labels' => [
                    'en' => [
                            'item' => 'Group',
                            'collection' => 'Groups',
                    ],
            ],
            'root' => false,
            'parents' => ['classroom'],
            'identifier' => 'groups',
]);
```

You can put multiple subtypes into a collection by assigning them to the same *identifier*, e.g. you could create *usa_state* and *canada_province* subtypes and register them for *regions* identifier.

18.2 Fields

Fields are managed by hypePost. Please see the documentation there for more information.

CHAPTER 19

hypeHero

Replaces owner block with a hero

CHAPTER 20

hypeIllustration

Unsplash integration

CHAPTER 21

hypeInteractions

Enhanced commenting and liking UX for Elgg

- Extendable interactions module
- Real-time comments and likes (requires hypeLists)
- Comment attachments (requires hypeAttachments)
- Multi-level comment threads
- Inline comment search (requires hypeLists)
- Inline comment sorting (requires hypeLists)

CHAPTER 22

hypeInvite

- Allows users to invite new users by email
- An option to create an invite-only network
- Keeps track of all invitations to the same email address
- Creates friend requests when invitations are accepted
- Group owners can allow members to invite other members
- Site admins can allow group invitations of non-friends
- Site admins can allow group invitations by email

22.1 Gotchas

- Registration must be enabled on the site for this plugin to work
- In an invite-only network, uservalidationbyemail will be bypassed, as it is assumed that users would have received their invitation code by email
- When invited by email to group, non-existing users will first have to create an account. Upon registration, invitations will be created for every group the email has been invited to before registration.

22.2 Creating Invites

Other plugins may centralize off-site invitations and attach custom behavior to the invites. For example, to invite non-registered users to an event by their email:

```
$invite = users_invite_create_user_invite($email);
add_entity_relationship($invite->guid, 'invited_to', $event->guid);
add_entity_relationship($invite->guid, 'invited_by', $inviter->guid);
```

(continues on next page)

(continued from previous page)

```
// generate a registration link to include in the notification
$registration_link = users_invite_get_registration_link($email, $inviter->guid);


// implement a custom handler
elgg_register_plugin_hook_handler('accept', 'invite', function($hook, $type, $return,
↪$params) {

   $invite = $params['invite'];
   $user = $params['user'];

   $events = elgg_get_entities_from_relationship([
        'types' => 'object',
        'subtypes' => 'event',
      'relationship' => 'invited_to',
      'relationship_guid' => $invite->guid,
      'limit' => 0,
   ]);

   if (!$events) {
      return;
   }

   foreach ($events as $event) {
      add_entity_relationship($user->guid, 'attending', $event->guid);
   }
});
```

CHAPTER 23

hypeLists

A set of tools that improve UX and simplify common list patterns for developers.

- Seamless integration with core and existing plugins
- AJAXed list pagination and infinite scroll
- Lazy loading of preceeding and succeeding list pages
- Auto refreshing of lists
- An interface for creating sortable/searchable lists

23.1 Server-Side

The following options are accepted by `` `elgg_list_entities()` ``, `` `elgg_view_entity_list()` ``, and by `` `page/components/list` `` and `` `page/components/gallery` `` views. These parameters will only take effect, if you have `` `'pagination' => true` `` in your options. Additional options, that need to be passed to the jQuery plugin, can be prefixed with `` `data-` ``

- `` `'list_id'` `` STRING is an optional parameter, but it is strongly recommended to pass it to your list. List id must be unique to the page.
- `` `'pagination_type'` `` STRING `` `default` `` (pagination bar with page number navigation) or `` `infinite` `` (before and after navigation)
- `` `'position'` `` STRING can be used to specify the position of pagination items. `` `before` ``, `` `after` ``, `` `both` ``
- `` `'num_pages'` `` INT can be used to specify how many page number navigation items to show, use 0 to only show Next and Prev links
- `` `'lazy_load'` `` INT can be used to initialize lazy loading of pages
- `` `'auto_refresh'` `` INT can be used to specify at which interval in seconds new items should be fetched
- `` `'reversed'` `` BOOL can be used to specify reversed lists. If list is reversed, it is assumed that the new items will be located at the end of the list

23.2 Client-Side

Lists that have received the necessary parameters server-side will be instantiated automatically. If you need to instantiate a list programmatically, use `` `$.hypeList(options)` ``.

```
// Instantiate a new list
$('.elgg-list.my-list').hypeList({
      baseUrl: false, // Data source
      count: 0, // Number of items in the list
      offset: 0, // Current offset from the beginning of the list
      offsetKey: 'offset', // Offset key
      limit: 10, // Number of items per page
      listId: '', // List identifier unique to the page
      pagination: 'default', // Pagination type: 'default', 'infinite'
      paginationPosition: 'after', // Pagination position: 'before', 'after', 'both'
      paginationNumPages: 10, // Number of page links to display in the pager
      classActive: 'elgg-state-selected', // CSS class pertinent to active elements
      classDisabled: 'elgg-state-disabled', // CSS class pertinent to disabled␣
↪elements
      classLoading: 'elgg-state-loading', // CSS class pertinent to pending elements
      textNoResults: '', // Text displayed when no items were found in the list
      textNext: elgg.echo('next'), // Text for next link
      textPrev: elgg.echo('previous'), // Text for previous link
      keyTextBefore: 'lists:add:before', // Language key for before link (will␣
↪receive limit as parameter)
      keyTextAfter: 'lists:add:after', // Language key for before link (will receive␣
↪limit as parameter)
      lazyLoad: 10, // Number of pages to lazy load
      autoRefresh: 60, // Fetch new items at this interval (in seconds)
      reversed: false, // List is reversed that is new items are appended to the end␣
↪of the list
      scrollTopOffset: -100, // Additional offset in pixels for when the page is␣
↪scrolled to the top of the list
      listTime: 0, // Timestamp at which the list was generated, sent with AJAX␣
↪requests
      showEffect: 'highlight', // jQuery UI effect used for toggling item visibility
      selectorDelete: '.elgg-menu-item-delete > a', // CSS selector of an anchor that␣
↪will trigger a delete action
});

// Public methods

// Navigate to a page with a certain index
// For default pagination type, page with pageIndex is loaded and displayed
// For infinite pagination type, all pages in range from currently visible pages to␣
↪the page with pageIndex are loaded and displayed
$('.elgg-list').trigger('goToPage', [pageIndex]);

// Trigger refresh
// Reloads the page and appends new items if any
// If no pageIndex is provided, it's determined by pagination type
// goToPage parameter can be used to navigate to the page once new items have been␣
↪fetched
// goToPage flag is useful when a new post was made and you want to display the post␣
↪to the user
$('.elgg-list').trigger('fetchNewItems', [pageIndex, goToPage]);
```

(continues on next page)

(continued from previous page)

```
// Remove items from the list and reindex
$('.elgg-list').trigger('removeItems', [$items]);

// Add new items to the list
$('.elgg-list').trigger('addFetchedItems', [ajaxData]);


// Events

// Event triggered whenever the list is first rendered
// Callback will receive list options as a second parameter
$('.elgg-list').on('ready', callback);

// Event triggered whenever an item is added, removed or hidden from a list
// Callback will receive list options as a second parameter
$('.elgg-list').on('change', callback);
```

CHAPTER 24

hypeMapsOpen

API and UI for maps built with open technology

- Geocoding and reverse geocoding via Nominatim
- Maps built with Leaflet.js
- Default map tiles provided by Open Street Maps (customizagle in views)
- User map
- Groups map
- Group members map

24.1 A map of arbitrary locations

```
echo elgg_view('page/components/map', [
   'markers' => [
      'Berlin, Germany',
      'London, UK',
      'Paris, France',
   ]
]);
```

24.2 A map with custom icons

```
$berlin = hypeJunction\MapsOpen\Marker::fromLocation('Berlin, Germany');
$berlin->icon = 'smile-o';
$berlin->color = 'green';
$berlin->tooltip = '<b>Berlin is a happy place</b>';
```

(continues on next page)

(continued from previous page)

```
$paris = hypeJunction\MapsOpen\Marker::fromLocation('Paris, France');
$paris->icon = 'coffee';
$paris->color = 'black';
$paris->tooltip = '<img src="https://s-media-cache-ak0.pinimg.com/736x/ca/ea/57/
↪caea57268e1dee696f3c20a5a0f895f2.jpg" alt="Paris" />';

echo elgg_view('page/components/map', [
   'markers' => [
      $berlin,
      $paris,
   ],
]);
```

24.3 A map of entities

```
echo elgg_view('page/components/map', [
   'markers' => elgg_get_entities_from_metadata([
      'types' => 'object',
      'subtypes' => 'place',
      'metadata_name_value_pairs' => [
         'venue_type' => 'cafe',
      ],
      'limit' => 0,
   ]),
   'center' => hypeJunction\MapsOpen\Marker::fromLocation('London, UK');
]);
```

24.4 A map with data source and search

```
echo elgg_view('page/components/map', [
   // Set src to json data source
   // Data set should be an export of Marker instances
   'src' => '/path/to/data/source/json',
   'show_search' => true,
]);
```

24.5 Change marker icon and color

Use *'marker','<entity_type>'* hook. Supported colors: 'red', 'darkred', 'orange', 'green', 'darkgreen', 'blue', 'purple', 'darkpuple', 'cadetblue'

```
elgg_register_plugin_hook_handler('marker', 'object', function($hook, $type, $return,
↪$params) {

   $entity = elgg_extract('entity', $params);

   if ($entity instanceof Event) {
      $return->icon = 'calendar';
```

(continues on next page)

(continued from previous page)

```
        $return->color = 'darkpurple'
    }

    return $return;
})
```

24.6 Change popup content

Add a view for *maps/tooltip/<entity_type>/<entity_subtype>* or *maps/tooltip/<entity_type>/default*;

24.7 Acknowledgements

- Early version of the plugin has been partially sponsored by [Social Business World] (https://socialbusinessworld.org "Social Business World")

CHAPTER 25

hypeMedia

Media albums

- Support for image and video files
- Support for image and video embed via URLs

CHAPTER 26

hypeMentions

Mentions

- Allows mentioning users and groups in various text fields `@username`
- Hashtag autocomplete `#hashtag`
- Emoji support `:heart:`

26.1 Acknowledgements

- Plugin uses amazing At.js library http://ichord.github.io/At.js/
- A list of emojis was sources from https://github.com/amio/emoji.json

CHAPTER 27

hypeMenus

Menu editor

- Add/remove items from menus using admin interface
- Easily customize menu icons, text, tooltip, URL
- Drag&Drop reordering of menu items and children
- Easily add new menu sections
- Build dynamic URLs using extrapolation or linking an item to an entity

CHAPTER 28

hypeModerator

Implements moderator role

28.1 Approval Workflow

Approval workflow only works with entities that use `publish` event.

To queue entities for approval use a hook handler:

```
elgg_register_plugin_hook_handler('uses:moderation', 'object:<entity_subtype>`,
↪'\Elgg\Values::getTrue`);
```

CHAPTER 29

hypeNotifications

- Facebook-style site notifications
- Email digest: users can specify at which interval they receive notifications for each type
- A tool to update preferred notification methods for all site users
- Leverages *Zend_Mail* (email library used in core) to send out HTML emails
- Allows to configure email transports (Sendmail, SMTP, File Transport, SendGrid, Mailgun, SparkPost)
- Allows to send file attachments
- Inlines CSS styles for improved email client experience
- Simpler testing experience: catch all email address, email/domain whitelist

29.1 Notification preferences

Go to Admin > Administer > Utilities > Notification Methods to update personal and subscription notification preferences globally.

29.2 Notification event types

Notification event types can be filtered using `'notification_events','notifications'` hook. Users will be given an option to unsubscribe from notifications about these events or batch them into a digest. Note that some instant notification events should not be added this list, e.g. password reset and other account related notifications should remain instant.

29.3 Notification testing

You can disable outgoing email by switching to File Transport in plugin settings, this will instead write email as txt files to the filestore under */notifications_log/zend/*

29.4 Sample SMTP config for GMail

To use GMail as your SMTP relay, you will likely need to Allow less secure apps: https://support.google.com/accounts/answer/6010255?hl=en

- Host: smtp.gmail.com
- Port: 587
- Secure Connection: TLS
- Auth: SMTP with AUTH LOGIN
- Username: <your gmail email>
- Password: <your gmail password>

29.5 Sample SMTP config for SendGrid

- Host: smtp.sendgrid.com
- Port: 587
- Secure Connection: TLS
- Auth: SMTP with AUTH LOGIN
- Username: apikey
- Password: <your api key>

29.6 File Attachments

To add attachments to your email, add an array of *ElggFile* objects to notification parameters:

```
notify_user($to, $from, $subject, $body, array(
   'attachments' => array(
      $file1, $file2,
   )
));
```

CHAPTER 30

hypePayments

- Standardized API for handling payments and product sales
- Interface for logging and refunding payments

30.1 New payment

```
namespace hypeJunction\Payments;

// First, we create an itemized order/invoice
$order = new Order();
$order->setCurrency('EUR');

// Add a new product
$order->add($product, 2);

// Add additional fees and charges
$shipping = Amount::fromString('25.25', 'EUR');
$charges[] = new ShippingFee('shipping', 0, $shipping);

$charges[] = new ProcessingFee('paypal_fee', 3.9);

$order->setCharges($charges);

$address = new Address();
$address->street_address = 'Some street 25';
// add other address parts
$address->country_code = 'CZ';

$order->setShippingAddress($address);

// Now create a transaction
$transaction = new Transaction();
```

(continues on next page)

(continued from previous page)

```
$transaction->setOrder($order);
$transaction->setPaymentMethod('paypal');

// Be sure to correctly set the owner and container and access id
// to ensure that both the merchant and the customer have access
// to the transaction entity
$transaction->owner_guid = $payer->guid;
$transaction->container_guid = $payee->guid;

// You can use access_grant to give access to the merchant,
// or create a new acccess collection that contains both the payer and the payee
$transaction->access_id = ACCESS_PRIVATE;

$transaction->save();

// Instantiate a gateway of choice
$gateway = new \hypeJunction\PayPal\API\Adapter();

// What you do with response may depend on where you are executing
// this code. From an action file, you can just return the $response.
$response = $adapter->pay($transaction);
```

CHAPTER 31

hypePaypalPayments

A wrapper for Paypal's PHP SDK

31.1 Webhooks

Configure your Paypal application to send webhooks to `https://<your-elgg-site>/payments/paypal/webhooks`

To digest a webhook, register a plugin hook handler:

```
elgg_register_plugin_hook_handler('BILLING.SUBSCRIPTION.EXPIRED', 'paypal',␣
↪HandleExpiredSubscription::class);

class HandleExpiredSubscription {
   public function __invoke(\Elgg\Hook $hook) {
      $webhook_data = $hook->getParam('data');

      // ... do stuff

      return $result; // Result will be reported back to paypal
   }
}
```

31.2 Paypal Button

To display a pay button:

```
echo elgg_view_field([
   '#type' => 'paypal/paypal',
   'required' => true,
]);
```

You can then retrieve the value of the Paypal's payment and payer ID your action:

```
$payment_id = get_input('paypal_payment_id');
$payer_id = get_input('payer_id');

elgg()->{'payments.gateways.paypal'}->pay($transaction, [
   'paypal_payment_id' => $payment_id,
   'paypal_payer_id' => $payer_id,
]);
```

CHAPTER 32

hypePaypalSubscriptions

Paypal integration for hypeSubscriptions

32.1 Disclaimer

PayPal sucks and so does its API. I have written this plugin to test the flexibility of the subscriptions API. Personally, I would discourage anyone from using PayPal, until they get their act together, hire an army of developers and get up to speed with the developments in the financial e-services.

During testing, I have noticed that PayPal doesn't properly synchronize payment information with billing agreements, so it's really impossible to keep track of changes in real-time. You may end up having to manually consolidate transactions and refunds.

CHAPTER 33

hypePost

Provides utilities for creating and editing posts.

Extendable form fields Post forms are built using an extended field API layer, which makes it easier to define all fields in one place, and not have to worry about maintaining form, action and profile logic separate from each other.

Reusable views Plugins can recycle existing resource views, saving lots of time needed to bootstrap new entity types.

Client site validation Client side form validation

AJAX form save Forms are saved using AJAX with saves user a few steps should the action fail.

33.1 Form Fields

To extend post form, use `fields, <entity_type>:<entity_subtype>` (or less granular `fields, <entity_type>`) plugin hook.

The hook receives an instance of `\hypeJunction\Fields\Collection`, which allows you to easily manipulate fields:

```
elgg_register_plugin_hook_handler('fields', 'object:blog', function(\Elgg\Hook $hook)
↪{

   $fields = $hook->getValue();
   /* @var $fields \hypeJunction\Fields\Collection */

   $fields->add('published_date', new MetaField([
      'type' => 'date',
      'required' => true,
   ]);

   $fields->get('description')->required = false;

   return $fields;
});
```

33.2 Common Properties

Icons

To enable or disable icons, use `uses:icon, <entity_type>:<entity_subtype>` hook. The handler should return *true* or *false*

Cover Images

To enable or disable cover images, use `uses:cover, <entity_type>:<entity_subtype>` hook. The handler should return *true* or *false*

Comments

To enable or disable comments, use `uses:comments, <entity_type>:<entity_subtype>` hook. The handler should return *true* or *false*

CHAPTER 34

hypePaywall

- Restrict access to content items to paying site members
- Restrict downloads to paying site members
- Allow access with to content and downloads with one-off payments

CHAPTER 35

hypeProfile

- Extended member search and sort functionality
- API for managing profile and registration fields
- API for adding fields to search
- Registration form validation
- Admin options to simplify registration form
- An option to validate email before account is created

CHAPTER 36

hypeSatis

Allows hosting composer repository with private repositories as part of the Elgg site, extending hypeDownloads plugin.

36.1 Satis Build

To build Satis composer repository:

```
vendor/bin/elgg-cli satis:build
```

You may want to configure a cron job to run this command to pull in new versions of packages at a given interval.

36.2 Client Setup

First, add username and token of the Elgg account to the client server:

```
composer global config http-basic.{{host}} {{username}} {{token}}
```

In `composer.json` of the client project add:

```
{
    "repositories": [
        {
            "type": "composer",
            "url": "https://{{host}}/satis"
        }
    ]
}
```

Where:

- `{{host}}` is the hostname of your Elgg installation

- `{{username}}` is the username of the Elgg user
- `{{token}}` is the password of the Elgg user

CHAPTER 37

hypeScraper

A tool for scraping, caching and embedding remote resources.

- Scrapes URLs and turns them in responsive preview cards
- Aggressive caching of scraped resources for enhanced performance
- Linkifies #hashtags, @usernames, links and emails

37.1 Card

To display a URL card with an image preview, title and brief description, use `output/card` view:

```
echo elgg_view('output/card', array(
   'href' => 'https://www.youtube.com/watch?v=Dlf1_vuIR4I',
));
```

37.2 Player

To dipslay a rich media player use `output/player` view:

```
echo elgg_view('output/player', array(
   'href' => 'https://www.youtube.com/watch?v=Dlf1_vuIR4I',
));
```

37.3 Linkify

To linkify all URLs, usernames, emails and hashtags that are not wrapped in html tags, use `` `output/linkify` `` view. Pass your text in a `` `value` `` parameter. You can use `` `parse_` `` flags to skip certain qualifiers.

```
$text = '@someone needs to #linkify this article http://example.com and email it to␣
↪someone@example.com';
if (elgg_view_exists('output/linkify')) {
   $text = elgg_view('output/linkify', array(
      'value' => $text,
      //'parse_urls' => false,
      //'parse_hashtags' => false,
      //'parse_usernames' => false,
      //'parse_emails' => false,
   ));
}
```

To generate a preview for multiple URLs extracted from text, use `` `output/url_preview` `` view. Pass your text as a `` `value` `` parameter. The view will parse all URLs and generate previews.

```
$text = 'This video is really cool https://vimeo.com/channels/staffpicks/116498390';
if (elgg_view_exists('output/url_preview')) {
   $text = elgg_view('output/url_preview', array(
      'value' => $text,
   ));
}
```

CHAPTER 38

hypeShortcode

Add support for custom BB-style shortcodes

38.1 Register shortcode

```
elgg()->shortcodes->register('mycode');

// then add a view in shortcodes/mycode
// view vars will contain attributes of the shortcode
```

38.2 Generate a shortcode tag

```
elgg()->shortcodes->generate('mycode', [
   'foo' => 'bar',
]);
```

38.3 Expand shortcodes

```
elgg()->shortcodes->expand($text);
```

38.4 Strip shortcodes

```
elgg()->shortcodes->strip($text);
```

CHAPTER 39

hypeShutdown

API to offset code execution until system shutdown. Allows plugins to execute expensive code until after the page has been rendered in the browser.

39.1 Shutdown event

All shutdown event handlers will be executed after the request has been sent to the browser and the buffer has been flushed.

You can either register a shutdown event handler:

```
elgg_register_event_handler('shutdown', 'system', function() {
   // your long running script
});
```

You can also use a runtime queue:

```
\hypeJunction\Shutdown\Queue::instance()->queue(function() use ($video) {
   $video->convert();
});
```

CHAPTER 40

hypeSlug

Provides slug support

40.1 Adding slugs

If you have hypePost enabled, slug input will be automatically available. Otherwise, extend your forms and actions to store *slug* metadata on an entity.

CHAPTER 41

hypeStash

API for caching common entity data to reduce DB queries

- Caches entity likes count
- Caches entity comments count
- Caches last comment
- Caches user friends count
- Caches group members count

41.1 Stashing Logic

The plugin uses preloader classes to load values from the database on first request. The value is cached and returned on consequent calls. `Preloader::up()` can be used to define when the cached value should be reset. For example, the value of likes is constant until a new like annotation is created, or an old is deleted, so we register our reset function for those events.

41.2 Helpers

You can use helper functions to retrieve counts using caching framework. All available shortcut functions can be found in `/lib/functions.php`

```
elgg_get_total_likes($entity);
elgg_get_total_comments($entity);
```

41.3 Custom Properties

```
$stash = \hypeJunction\Stash\Stash::instance();

// Register a new cacheable property
$stash->register(new CustomProperty()); // Custom property must implement Preloader␣
↪interface

// Get property value
$prop = $stash->get(CustomProperty::PROPERTY, $entity);
```

CHAPTER 42

hypeStaticPages

Admin tool for creating static pages

CHAPTER 43

hypeStripePayments

A wrapper for Stripe's PHP SDK

43.1 Webhooks

Configure your Stripe application to send webhooks to ``https://<your-elgg-site>/payments/stripe/webhooks``

To digest a webhook, register a plugin hook handler:

```
elgg_register_plugin_hook_handler('customer.subscription.deleted', 'stripe',␣
↪HandleExpiredSubscription::class);

class HandleExpiredSubscription {
   public function __invoke(\Elgg\Hook $hook) {
      $stripe_event = $hook->getParam('event');
      /* @var $stripe_event \Stripe\Event */

      $subscription = $stripe_event->data->object;

      // ... do stuff

      return $result; // Result will be reported back to stripe
   }
}
```

43.2 Card Input

To display a card input:

```
// Card number, expiry and CVC
echo elgg_view_field([
   '#type' => 'stripe/card',
   '#label' => 'Credit or Debit Card',
   'required' => true,
]);

// Cardholder name
echo elgg_view_field([
   '#type' => 'stripe/cardholder',
   '#label' => 'Cardholder',
   'required' => true,
]);

// Billing address
// Requires hypeCountries plugin
echo elgg_view_field([
   '#type' => 'stripe/address',
   '#label' => 'Billing address',
   'required' => true,
]);
```

You can then retrieve the value of the Stripe token in your action:

```
$token = get_input('stripe_token');
$address = get_input('address');
$name = get_input('cardholder');

// Use stripe API to create a new card object
// or use the token as the source of the payment
```

CHAPTER 44

hypeStripeSubscriptions

Stripe integration for hypeSubscriptions

CHAPTER 45

hypeSubscriptions

API for implementing paid subscriptions

- Agnostic API that can be extended with any payment provider
- Implement site subscriptions and optionally restrict access to paying subscribers only
- API to implement entity/group specific subscriptions
- API to restrict access to posts and downloads

45.1 Events

To implement custom logic when the subscription is created, listen to `create, subscription` event.

To implement custom logic when the subscription cancelled, listing to `cancel, subscription`. Note that the sbuscription can be cancelled at period end, so check `current_period_end` metadata, before terminating access to features.

CHAPTER 46

hypeTime

Utilities for working with dates and time

- Adds form fields for date and time input
- Allows users to configure their time output preferences

CHAPTER 47

hypeTrees

- Simple API for managing hierarchies of entities

CHAPTER 48

hypeVue

Vue.js bootstrap for Elgg

48.1 Developer Tools

To enable Vue dev tools, set `environment` config value to `development`

```
elgg_set_config('environment', 'development');
```

CHAPTER 49

hypeWall

Rich user interface for sharing status updates, links and content via user walls.

- URL-parsing with embeddable content view
- Geotagging (based on browser location)
- Inline multi-file upload
- Friend tagging
- Content attachments

49.1 Gotchas

- Reverse geocoding is performed via Nominatim http://wiki.openstreetmap.org/wiki/Nominatim.

Reverse geocoding (i.e. browser position coordinates to human readable address) will not work in https. Implement a custom solution using a paid/free/proprietary service that does the same

- Icons are not included with the plugin. You will need to load FontAwesome CSS,

either by registering it in your theme, or using one of the available Elgg plugins.

- You can add wall tabs and forms by extending the `` `'framework/wall/container/extend'` `` view