BREAKTHROUGH
TECHNOLOGIES

Drupal 8 Integration with Slack

WHO ARE WE?

Jonathan “Jack” Franks

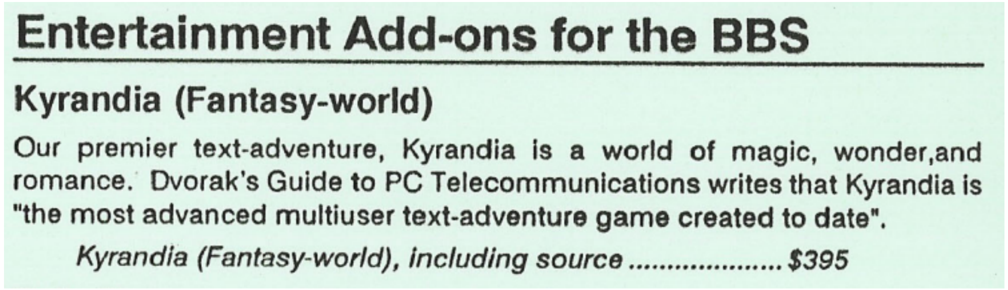
- Software developer since 1990 or so.
- Drupal developer since 2013.
- I work for Breakthrough Technologies.
- Love the Drupal community (that’s you!).
- I enjoy back-end development the most.
- Former BBSer.

What am I doing with this?

MULTI-USER DUNGEON

MUDs are Multi-User Dungeon games. They're usually text-based adventure games that involve killing monsters or solving puzzles to gain levels and get more powerful.

I build my first modem when I was 8 or 9. It was a 300 baud Heathkit and I used it to dial into Compuserve and a bunch of bulletin board systems. The first multiline one I remember calling was a "ddial" BBS called Point Zer0, which was a multi-line chat BBS. After that, I spent a whole lot of time on Galacticom MajorBBS systems like Electropolis and Mulligan's Place. One of my favorite things on these BBSs was the gaming. There were multi-player interactive games - picture Zork, but for 32 players. The one I've ported is called Kyrandia, and it was available for the MajorBBS software starting in 1989.



Entertainment Add-ons for the BBS

Kyrandia (Fantasy-world)

Our premier text-adventure, Kyrandia is a world of magic, wonder,and romance. Dvorak's Guide to PC Telecommunications writes that Kyrandia is "the most advanced multiuser text-adventure game created to date".

Kyrandia (Fantasy-world), including source $395

Demo time!

DEMO

Let’s take a quick look at the app and see:

- What kinds of interactions we have,
- What kinds of data Drupal serves up,
- How Slack presents them,
- And what the app actually looks like.

Architecture

ARCHITECTURE

The app is usually hosted on a free Pantheon site. (But we're using ngrok on a local instance today.)

The Slack app is created through usual Slack app creation, and the URLs are set to the dev Pantheon site (or our ngrok URLs for today).

We use the following new/custom modules for this game:

- `word_grammar`
- `slack_incoming`
- `slack_mud`
- `kyrandia`
- `kyrandia_migrate`

WORD GRAMMAR

Needed mechanism for:

- “a” or “an” for nouns
- Nicely formatted lists with Oxford commas (which is a parameter but, c’mon, it should always be TRUE)

Service for these operations in module `word_grammar`.

Available now at:

https://drupal.org/projects/word_grammar

PORTING THE OLD GAME

Data files:

- Locations
- Items
- Spells
- Descriptions
- Message text
- Levels

C functions

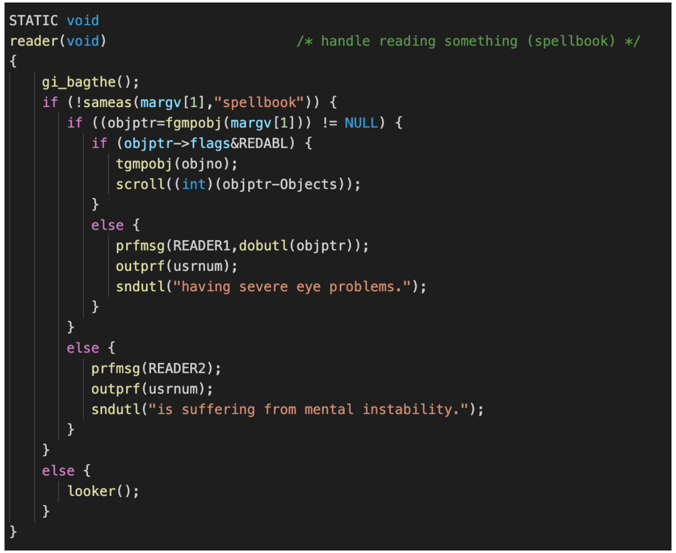
- Command routines
- Location handling routines
- System interaction routines

Importing Ancient Data

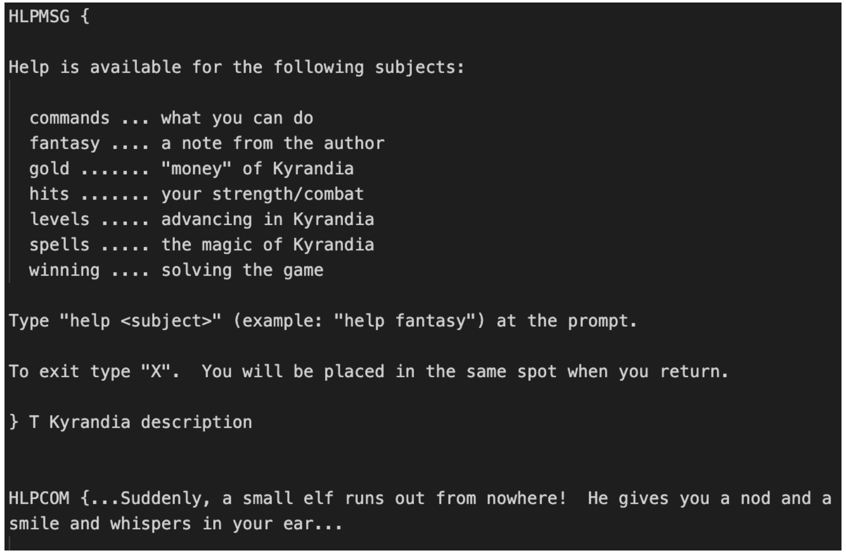
IMPORTING FROM MAJORBBS SOURCE CODE

I have the original source code for Kyrandia, written by Scott Brinker and Richard Skurnick for the Galacticomm MajorBBS software. It’s C code with MSG files that contain the actual messages and descriptions.

First, let’s look at how we can import all of that.



```
STATIC void
reader(void)                        /* handle reading something (spellbook) */
{
    gi_bagthe();
    if (!sameas(margv[1],"spellbook")) {
        if ((objptr=fgmpobj(margv[1])) != NULL) {
            if (objptr->flags&REDABL) {
                tgmpobj(objno);
                scroll((int)(objptr-Objects));
            }
            else {
                prfmsg(READER1,dobutl(objptr));
                outprf(usrnum);
                sndutl("having severe eye problems.");
            }
        }
        else {
            prfmsg(READER2);
            outprf(usrnum);
            sndutl("is suffering from mental instability.");
        }
    }
    else {
        looker();
    }
}
```



```
HLPMSG {

Help is available for the following subjects:

  commands ... what you can do
  fantasy .... a note from the author
  gold ....... "money" of Kyrandia
  hits ....... your strength/combat
  levels ..... advancing in Kyrandia
  spells ..... the magic of Kyrandia
  winning .... solving the game

Type "help <subject>" (example: "help fantasy") at the prompt.

To exit type "X".  You will be placed in the same spot when you return.

} T Kyrandia description


HLPCOM {...Suddenly, a small elf runs out from nowhere!  He gives you a nod and a
smile and whispers in your ear...
```

MSG FILES

GALKYRM.MSG

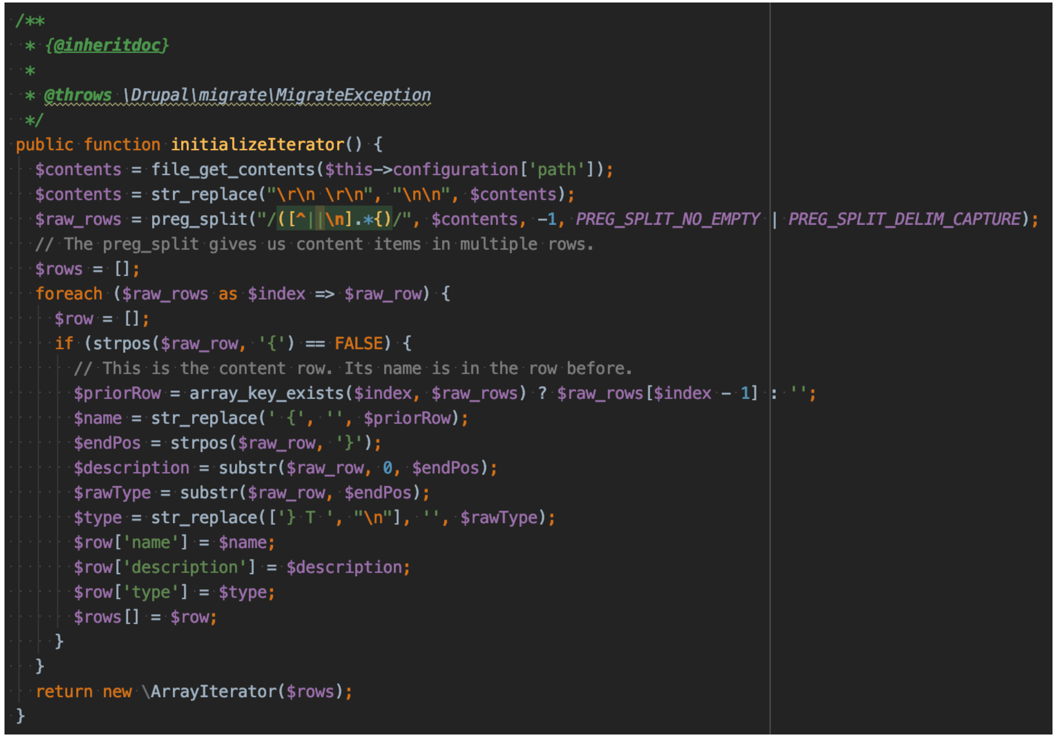
Actual record:

```
FDES04 {...Dressed in a heavy dark cape, %s sticks out in the crowd.  Lightly
sprinkled upon the cape is a substance of silvery dust.  You detect a strong
aura of magical power which makes you very weary of this person.  Sewn by black
threads on the side of the black cape is a Patch of Sorcery inscribed with the
word "Magician".  She seems to be holding } T Kyrandia description
```

We need a Migration Source that can parse this!

CUSTOM MIGRATION SOURCE

Custom iterator to pull the data out.

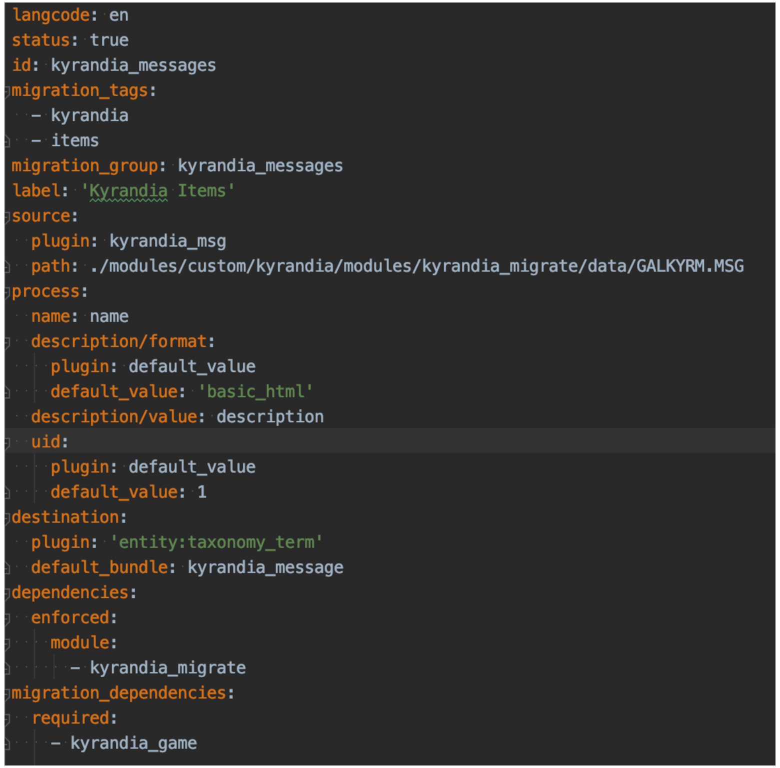


```php
/**
 * {@inheritdoc}
 *
 * @throws \Drupal\migrate\MigrateException
 */
public function initializeIterator() {
  $contents = file_get_contents($this->configuration['path']);
  $contents = str_replace("\r\n \r\n", "\n\n", $contents);
  $raw_rows = preg_split("/([^||\n].*{)/", $contents, -1, PREG_SPLIT_NO_EMPTY | PREG_SPLIT_DELIM_CAPTURE);
  // The preg_split gives us content items in multiple rows.
  $rows = [];
  foreach ($raw_rows as $index => $raw_row) {
    $row = [];
    if (strpos($raw_row, '{') == FALSE) {
      // This is the content row. Its name is in the row before.
      $priorRow = array_key_exists($index, $raw_rows) ? $raw_rows[$index - 1] : '';
      $name = str_replace(' {', '', $priorRow);
      $endPos = strpos($raw_row, '}');
      $description = substr($raw_row, 0, $endPos);
      $rawType = substr($raw_row, $endPos);
      $type = str_replace(['} T ', "\n"], '', $rawType);
      $row['name'] = $name;
      $row['description'] = $description;
      $row['type'] = $type;
      $rows[] = $row;
    }
  }
  return new \ArrayIterator($rows);
}
```

CUSTOM MIGRATION SOURCE

In the migration:

- Source plugin
- Path to MSG file
- Process pulls fields
- Saves to a node or a taxonomy term



```yaml
langcode: en
status: true
id: kyrandia_messages
migration_tags:
  - kyrandia
  - items
migration_group: kyrandia_messages
label: 'Kyrandia Items'
source:
  plugin: kyrandia_msg
  path: ./modules/custom/kyrandia/modules/kyrandia_migrate/data/GALKYRM.MSG
process:
  name: name
  description/format:
    plugin: default_value
    default_value: 'basic_html'
  description/value: description
  uid:
    plugin: default_value
    default_value: 1
destination:
  plugin: 'entity:taxonomy_term'
  default_bundle: kyrandia_message
dependencies:
  enforced:
    module:
      - kyrandia_migrate
migration_dependencies:
  required:
    - kyrandia_game
```

DRUPAL ENTITIES

We bring in the ported data as nodes and taxonomy terms. Let’s have a look at what those look like in Drupal.

- Locations
- Items
- Messages
- Levels
- Spells

CRON

We have a couple of things we need to do with cron.

Every day, the game resets. Default items need to appear where they belong, and extra stuff needs to be cleaned up.

The regular game cycle needs to run. This used to run every 8 seconds on the original BBS. This reduces or removes temporary effects in the game, like spell duration or actions performed in locations with temporary effects.

Drupal Cron doesn’t like running every 8 seconds so I set it up with a free service: https://cron-job.org. This lets me run it every minute, which isn’t as fast as MajorBBS but it works.

Playing the Game

COMMANDS

Text adventure games use textual commands. Players type in commands to do things like move around, look at or talk to people, interact with objects, cast spells, and so on.

```
> inventory

> move north

> get diamond

> drop ruby

> attack dragon
```

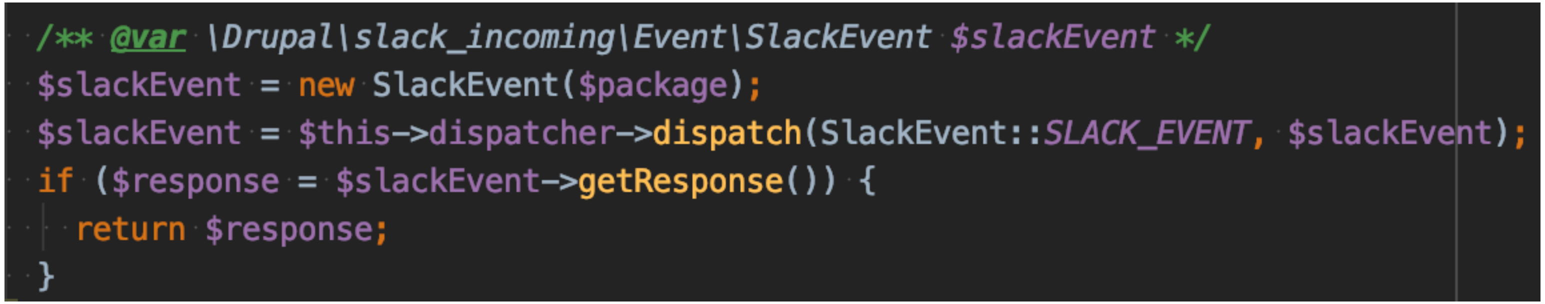
MUD COMMAND ARCHITECTURE

To handle these commands and re-use the same gaming engine across multiple games, we'll use custom plugins to handle the players' commands. The MUD event listener gets the command string that the player typed, determines what plugin to use, then instantiates that plugin and performs the action.

Events

EVENTS

This is how an event is fired.



```php
/** @var \Drupal\slack_incoming\Event\SlackEvent $slackEvent */
$slackEvent = new SlackEvent($package);
$slackEvent = $this->dispatcher->dispatch(SlackEvent::SLACK_EVENT, $slackEvent);
if ($response = $slackEvent->getResponse()) {
  return $response;
}
```

Instantiate your custom event class (`SlackEvent`).

Pass the event name and the instantiated event class to the dispatcher's `dispatch()` method. (Get Dispatcher from the container with `'event_dispatcher'`.)

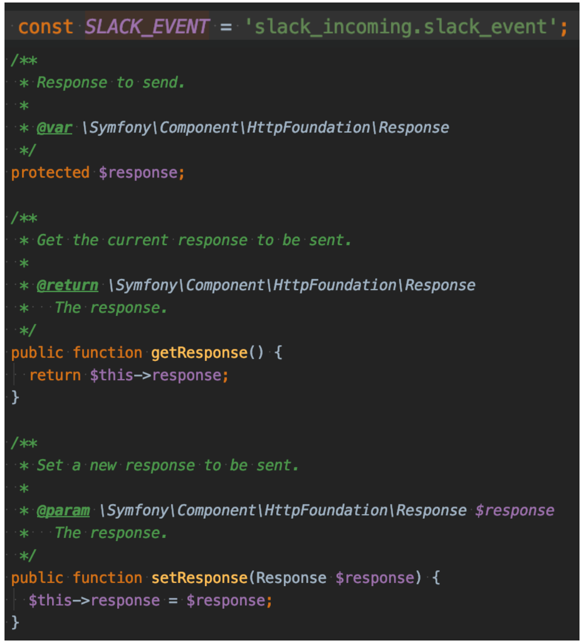
Each event subscribers listening for that event will run and do their thing.

Then we can get the results from `SlackEvent`'s `getResponse()` method, which returns an array of responses created/modified by event subscribers.

CUSTOM EVENT CLASS

The event class has a constant that defines the event name that a listener will use.

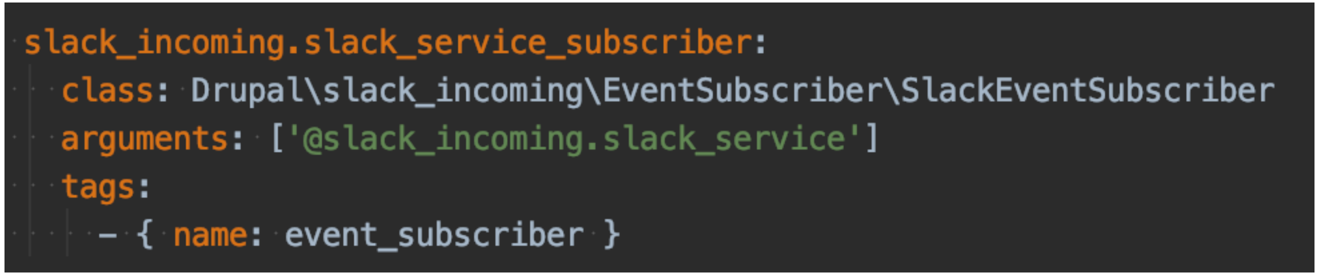
It also has the properties that identify the relevant information to the listeners, and the properties that the listeners will use to pass data back to the event caller.



```
const SLACK_EVENT = 'slack_incoming.slack_event';
/**
 * Response to send.
 *
 * @var \Symfony\Component\HttpFoundation\Response
 */
protected $response;

/**
 * Get the current response to be sent.
 *
 * @return \Symfony\Component\HttpFoundation\Response
 *   The response.
 */
public function getResponse() {
  return $this->response;
}

/**
 * Set a new response to be sent.
 *
 * @param \Symfony\Component\HttpFoundation\Response $response
 *   The response.
 */
public function setResponse(Response $response) {
  $this->response = $response;
}
```

CUSTOM EVENT SUBSCRIBER SERVICE

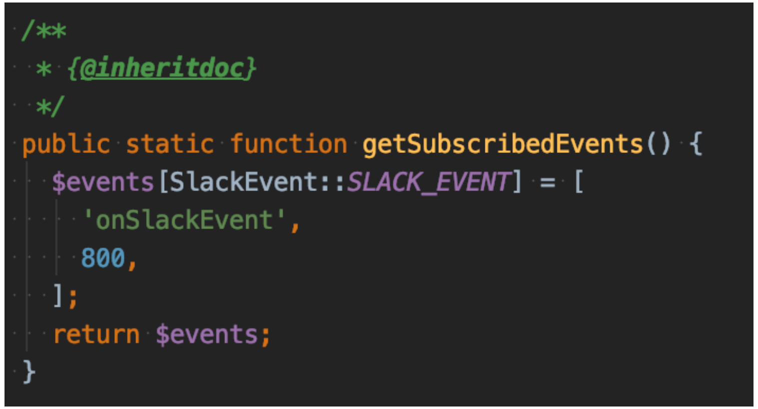
The event subscriber needs to be defined in the module's `services.info.yml` file.
Inherit from `EventSubscriberInterface`.



```yaml
slack_incoming.slack_service_subscriber:
  class: Drupal\slack_incoming\EventSubscriber\SlackEventSubscriber
  arguments: ['@slack_incoming.slack_service']
  tags:
    - { name: event_subscriber }
```

CUSTOM EVENT SUBSCRIBER CLASS

The subscriber class needs to define which events it's listening for in `getSubscribedEvents()`. `SLACK_EVENT` is the same constant we defined in the event class.



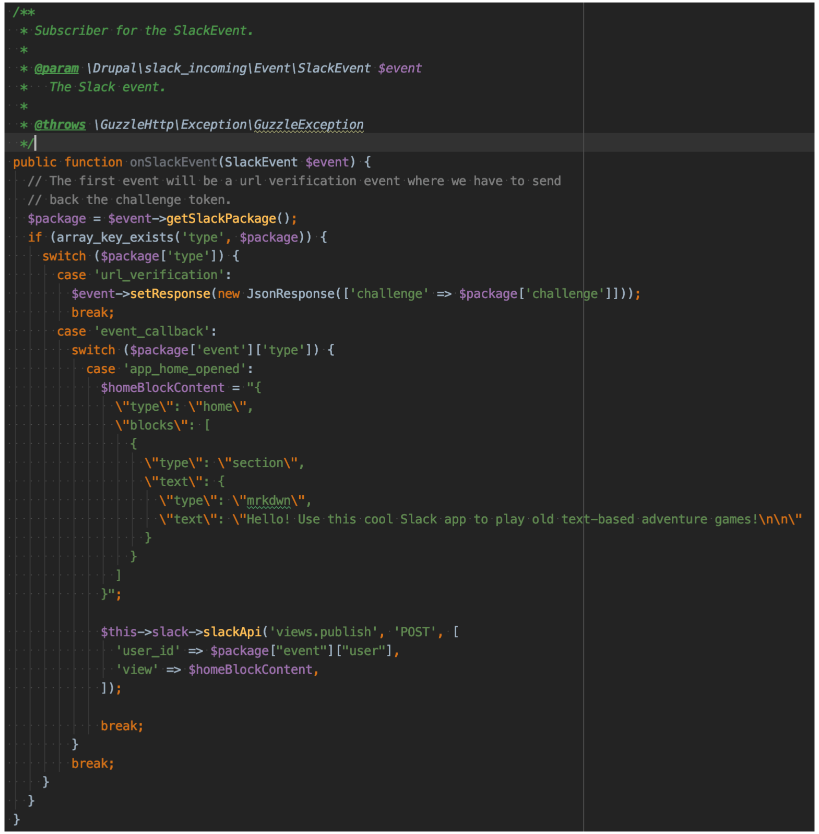
```
/**
 * {@inheritdoc}
 */
public static function getSubscribedEvents() {
  $events[SlackEvent::SLACK_EVENT] = [
    'onSlackEvent',
    800,
  ];
  return $events;
}
```

CUSTOM EVENT LISTENER

This is the function we listed in `getSubscribedEvents()`.

It takes the event as an argument.

Then we do some stuff with it. One condition sets the response on the event to the challenge response. The other posts a Slack message back to the user in the app's Home tab.



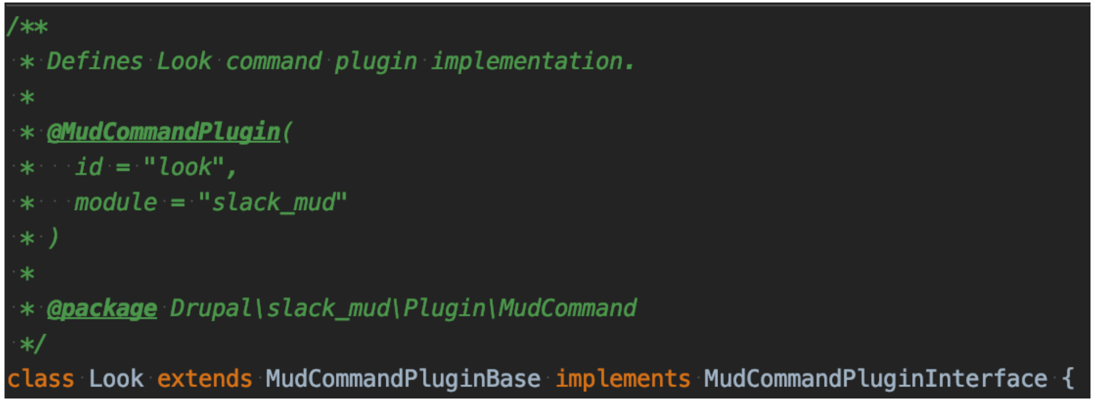
```php
/**
 * Subscriber for the SlackEvent.
 *
 * @param \Drupal\slack_incoming\Event\SlackEvent $event
 *   The Slack event.
 *
 * @throws \GuzzleHttp\Exception\GuzzleException
 */
public function onSlackEvent(SlackEvent $event) {
  // The first event will be a url verification event where we have to send
  // back the challenge token.
  $package = $event->getSlackPackage();
  if (array_key_exists('type', $package)) {
    switch ($package['type']) {
      case 'url_verification':
        $event->setResponse(new JsonResponse(['challenge' => $package['challenge']]));
        break;
      case 'event_callback':
        switch ($package['event']['type']) {
          case 'app_home_opened':
            $homeBlockContent = "{
              \"type\": \"home\",
              \"blocks\": [
                {
                  \"type\": \"section\",
                  \"text\": {
                    \"type\": \"mrkdwn\",
                    \"text\": \"Hello! Use this cool Slack app to play old text-based adventure games!\n\n\"
                  }
                }
              ]
            }";

            $this->slack->slackApi('views.publish', 'POST', [
              'user_id' => $package["event"]["user"],
              'view' => $homeBlockContent,
            ]);

            break;
        }
        break;
    }
  }
}
```

Plugins

CUSTOM PLUGINS

Custom plugins need:

- Annotations
- Plugin manager
- And to be helpful, a base plugin class



```php
/**
 * Defines Look command plugin implementation.
 *
 * @MudCommandPlugin(
 *   id = "look",
 *   module = "slack_mud"
 * )
 *
 * @package Drupal\slack_mud\Plugin\MudCommand
 */
class Look extends MudCommandPluginBase implements MudCommandPluginInterface {
```

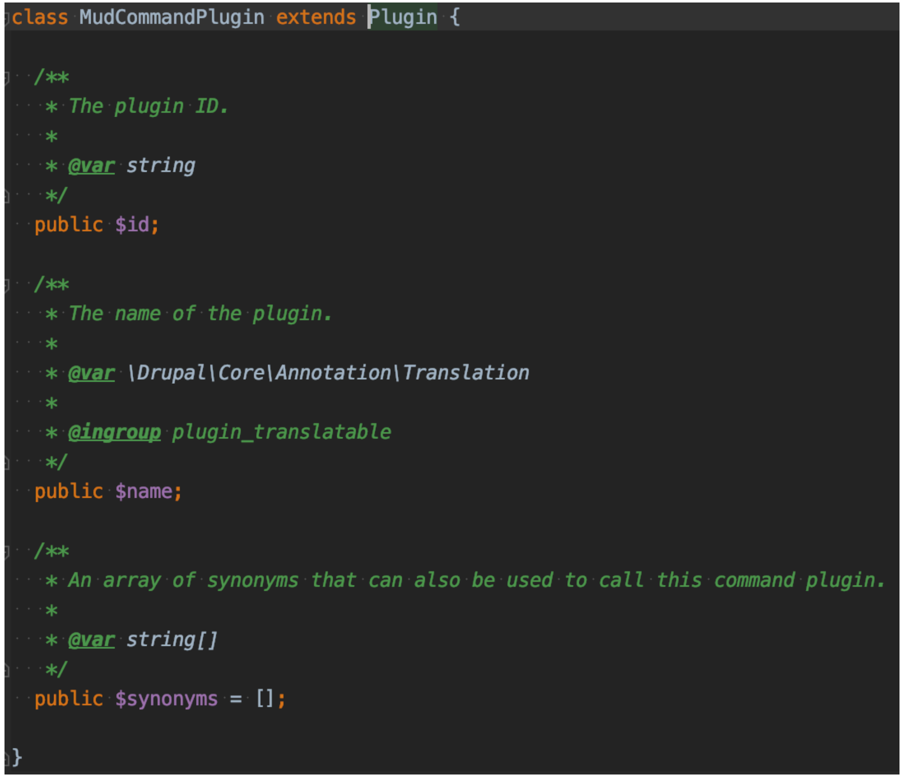
CUSTOM PLUGIN ANNOTATIONS

The annotation class defines what parameters the plugin can use in its annotation.

We have an id string parameter, a name string parameter, and a synonyms parameter.

(Although synonyms aren't working yet!)

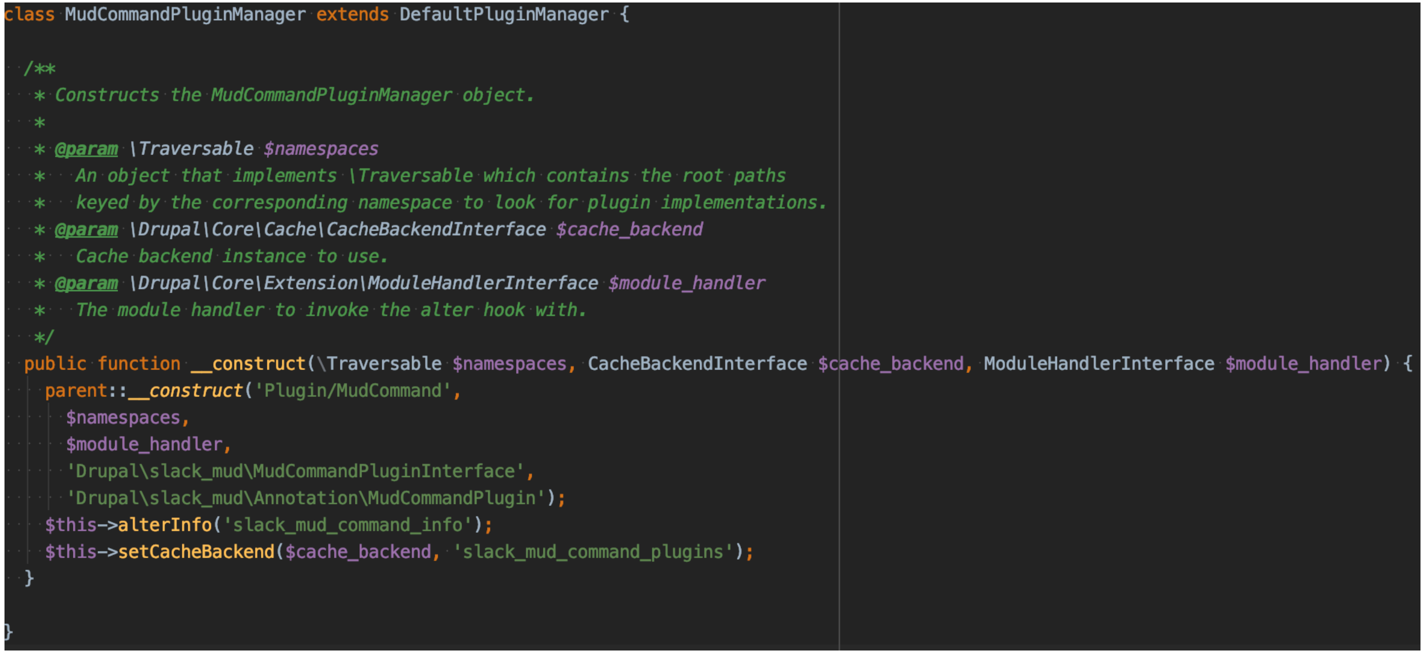
This class lives in `slack_mud/src/Annotation`.



```php
class MudCommandPlugin extends Plugin {

  /**
   * The plugin ID.
   *
   * @var string
   */
  public $id;

  /**
   * The name of the plugin.
   *
   * @var \Drupal\Core\Annotation\Translation
   *
   * @ingroup plugin_translatable
   */
  public $name;

  /**
   * An array of synonyms that can also be used to call this command plugin.
   *
   * @var string[]
   */
  public $synonyms = [];

}
```

CUSTOM PLUGIN MANAGER

Custom plugins need a plugin manager to define the directory structure, namespaces, module handler, plugin interface, and plugin annotation class. It also defines the `alterInfo` method and determines the caching mechanism. It goes in `slack_mud/src`.

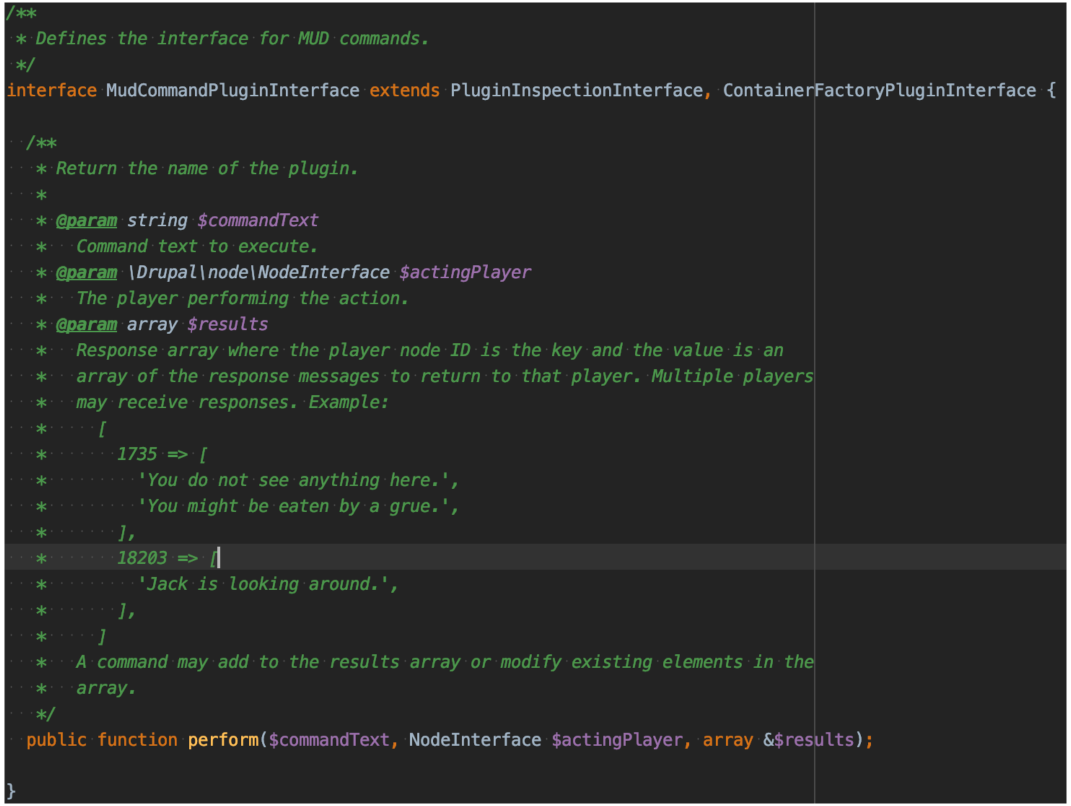


```php
class MudCommandPluginManager extends DefaultPluginManager {

  /**
   * Constructs the MudCommandPluginManager object.
   *
   * @param \Traversable $namespaces
   *   An object that implements \Traversable which contains the root paths
   *   keyed by the corresponding namespace to look for plugin implementations.
   * @param \Drupal\Core\Cache\CacheBackendInterface $cache_backend
   *   Cache backend instance to use.
   * @param \Drupal\Core\Extension\ModuleHandlerInterface $module_handler
   *   The module handler to invoke the alter hook with.
   */
  public function __construct(\Traversable $namespaces, CacheBackendInterface $cache_backend, ModuleHandlerInterface $module_handler) {
    parent::__construct('Plugin/MudCommand',
      $namespaces,
      $module_handler,
      'Drupal\slack_mud\MudCommandPluginInterface',
      'Drupal\slack_mud\Annotation\MudCommandPlugin');
    $this->alterInfo('slack_mud_command_info');
    $this->setCacheBackend($cache_backend, 'slack_mud_command_plugins');
  }

}
```

CUSTOM PLUGIN INTERFACE

Your plugin's interface defines the methods used by all the custom plugins of this type. For our MUD Command plugin, we need a `perform()` method that each plugin will use to process the commands and return block or text results.

The interface is used by the plugin manager to create and return plugin instances.



```php
/**
 * Defines the interface for MUD commands.
 */
interface MudCommandPluginInterface extends PluginInspectionInterface, ContainerFactoryPluginInterface {

  /**
   * Return the name of the plugin.
   *
   * @param string $commandText
   *   Command text to execute.
   * @param \Drupal\node\NodeInterface $actingPlayer
   *   The player performing the action.
   * @param array $results
   *   Response array where the player node ID is the key and the value is an
   *   array of the response messages to return to that player. Multiple players
   *   may receive responses. Example:
   *     [
   *       1735 => [
   *         'You do not see anything here.',
   *         'You might be eaten by a grue.',
   *       ],
   *       18203 => [
   *         'Jack is looking around.',
   *       ],
   *     ]
   *   A command may add to the results array or modify existing elements in the
   *   array.
   */
  public function perform($commandText, NodeInterface $actingPlayer, array &$results);

}
```

CUSTOM PLUGIN BASE CLASS

Your plugin's base class should define your default `create()` and `__construct()` methods. This makes services from the container available to all plugin classes that extend from the base class. It should also implement your interface.

Let's look at `\Drupal\slack_mud\Plugin\MudCommand\MudCommandPluginBase`.

CUSTOM PLUGINS

We’re working with MudCommand plugins, so our classes all go in `src/Plugin/MudCommand`.

We define the annotation parameters with an id and a name, extend from the base class, and implement the interface.

Let’s look at Kyrandia’s `kyrandia_cast` plugin and see how it works.

Slack

INTERACTING WITH THE GAME

Here’s the part we’ve all been waiting for:

How do we integrate the game with Slack?

SLACK_INCOMING MODULE

The `slack_incoming` accepts messages from Slack for all sorts of interactions:

- Messaging
- Slash commands
- Events

It can also make Slack API calls like `chat.postMessage` and `views.publish`.

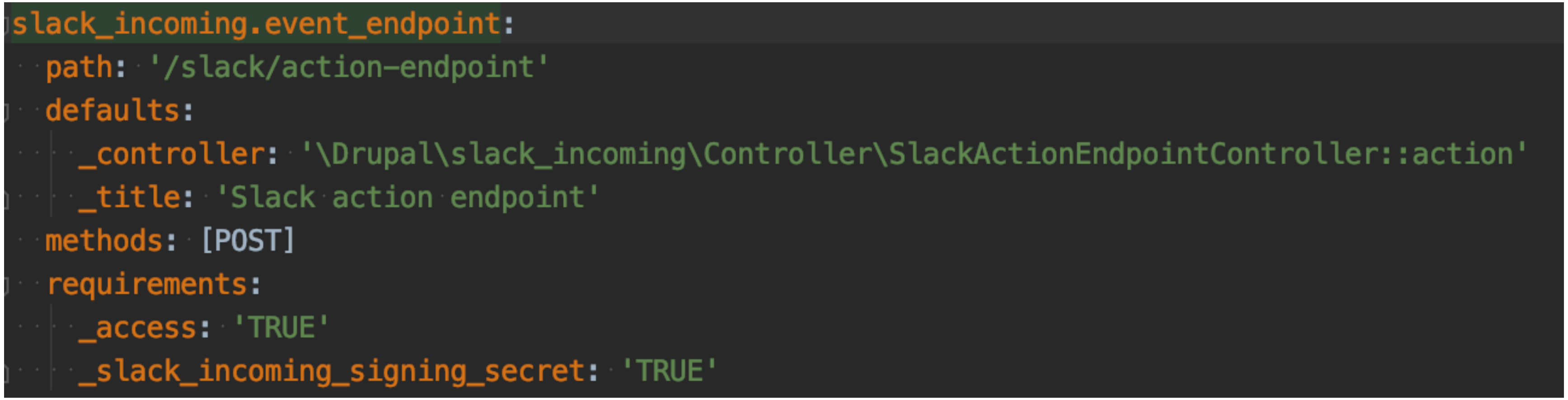
You can find this module at https://drupal.org/project/slack_incoming.

AUTHENTICATION

Slack messages should be authenticated to make sure some rando isn't sending your app weird messages pretending to be Slack.

Slack's scheme for authentication uses a signing secret defined for your application.

The `slack_incoming` module uses an authentication provider to check the incoming messages and is defined as a parameter on any additional routes you might want to add like additional slash commands.



```yaml
slack_incoming.event_endpoint:
  path: '/slack/action-endpoint'
  defaults:
    _controller: '\Drupal\slack_incoming\Controller\SlackActionEndpointController::action'
    _title: 'Slack action endpoint'
  methods: [POST]
  requirements:
    _access: 'TRUE'
    _slack_incoming_signing_secret: 'TRUE'
```

SLACK EVENTS

Slack message behavior:

1. User sends the bot a message in Slack.
2. Slack sends the event request URL a message.
3. The slack_incoming provider authenticates the message.
4. The slack_incoming controller decodes the message and fires a SlackEvent.
5. The SlackEvent's listeners process the message and generate responses.
6. The slack_incoming controller returns a 200 response. Slack expects a 200 within 3 seconds or it will re-send the messages. (Sometimes several times.)
7. The SlackEvent's listeners use slackApi() to send a message back to the user.

SLASH COMMANDS

Slack slash command behavior:

1. Define slash command in Slack application.
2. Define controller method and route, including the `_slack_incoming_signing_secret` requirement.
3. User sends a slash command in Slack.
4. Slack sends a message to the endpoint defined in the Slack application.
5. The slash command controller method reads the message and does some stuff.
6. The controller returns a response.
7. Slack displays the response to the user.

SLACK WORKSPACE

Create a new Slack workspace for your own testing.

Go to https://slack.com/create and enter your email. Confirm your identity.

Describe your team. We'll pick "Something else."

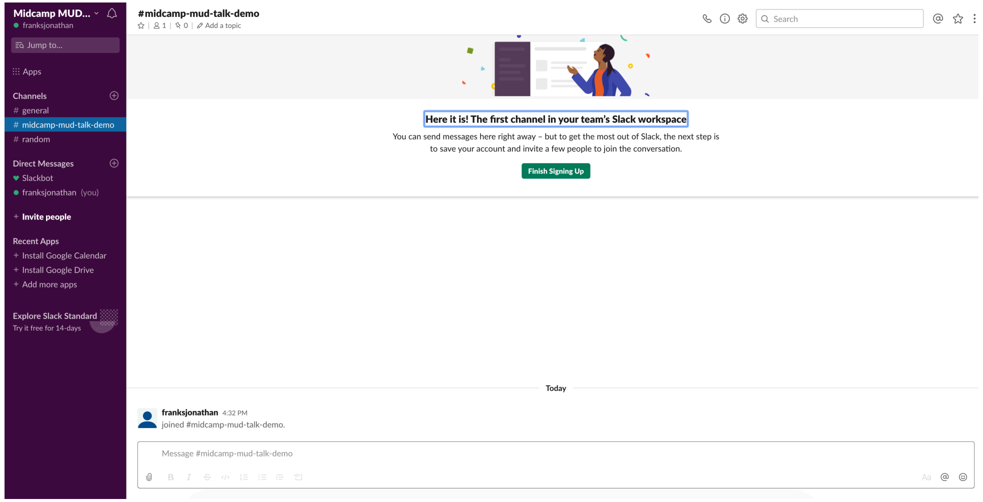
Enter a name for your team.

Enter what you're working on.

Skip adding people.

Click "Launch Workspace".

Your workspace has been created!


Midcamp MUD...
franksjonathan
Jump to...
Apps
Channels
# general
# midcamp-mud-talk-demo
# random
Direct Messages
Slackbot
franksjonathan (you)
+ Invite people
Recent Apps
+ Install Google Calendar
+ Install Google Drive
+ Add more apps
Explore Slack Standard
Try it free for 14-days

#midcamp-mud-talk-demo
1 | 0 | Add a topic
Search

Here it is! The first channel in your team's Slack workspace
You can send messages here right away – but to get the most out of Slack, the next step is to save your account and invite a few people to join the conversation.
Finish Signing Up

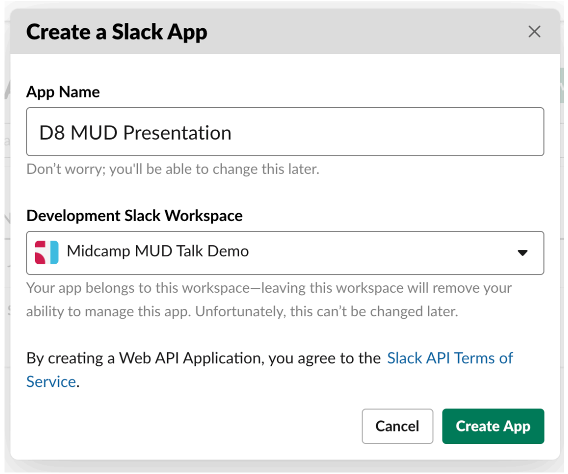
Today
franksjonathan 4:32 PM
joined #midcamp-mud-talk-demo.

Message #midcamp-mud-talk-demo

CREATING A SLACK APPLICATION

How to set up a new Slack application.

- Go to https://api.slack.com and log in. Click “Your Apps”.
- Create new application.
- Enter an App Name.
- For your Development Slack Workspace, drop down and sign into a workspace.
- Enter the URL for your new workspace and press Continue.
- You might need to come back to api.slack.com and go to create app again.
- Enter your App Name, select your new Development Slack Workspace.
- Press Create App.



Create a Slack App

App Name

D8 MUD Presentation

Don't worry; you'll be able to change this later.

Development Slack Workspace

Midcamp MUD Talk Demo

Your app belongs to this workspace—leaving this workspace will remove your ability to manage this app. Unfortunately, this can't be changed later.

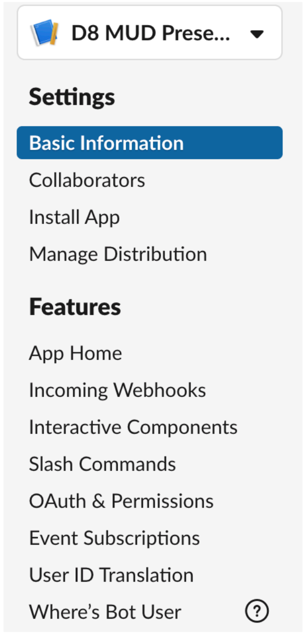
By creating a Web API Application, you agree to the Slack API Terms of Service.

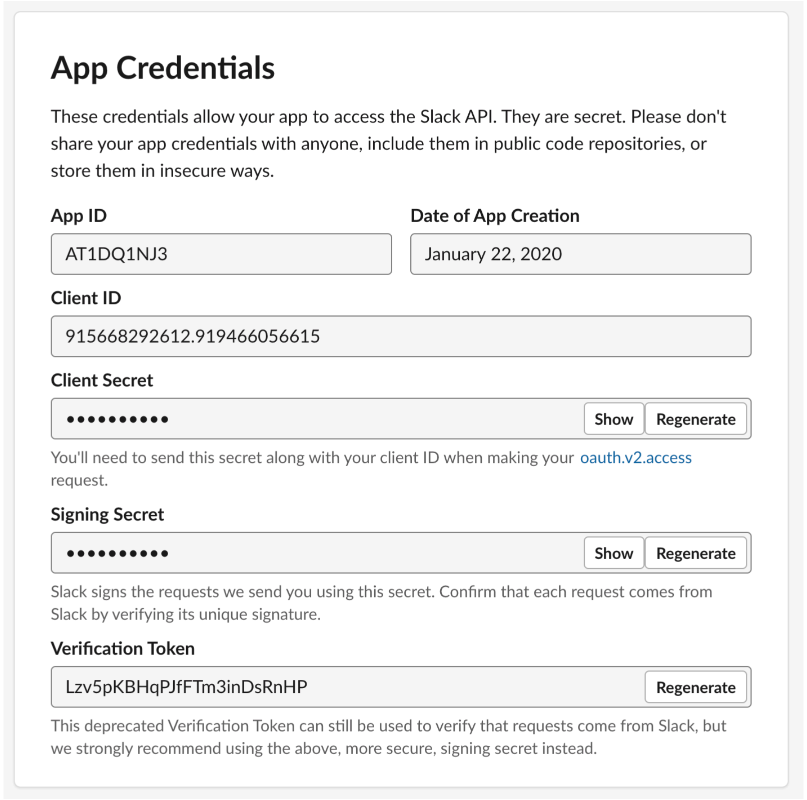
SETTING UP SLACK_INCOMING

In your Drupal site, go to Configuration > Web Services > Slack.

In your Slack application, go to Basic Information.

Enter your Signing Secret in `slack_incoming` configuration and press “Save configuration”.





App Credentials

These credentials allow your app to access the Slack API. They are secret. Please don't share your app credentials with anyone, include them in public code repositories, or store them in insecure ways.

App ID: AT1DQ1NJ3

Date of App Creation: January 22, 2020

Client ID: 915668292612.919466056615

Client Secret: •••••••••• Show Regenerate

You'll need to send this secret along with your client ID when making your oauth.v2.access request.

Signing Secret: •••••••••• Show Regenerate

Slack signs the requests we send you using this secret. Confirm that each request comes from Slack by verifying its unique signature.

Verification Token: Lzv5pKBHqPJfFTm3inDsRnHP Regenerate

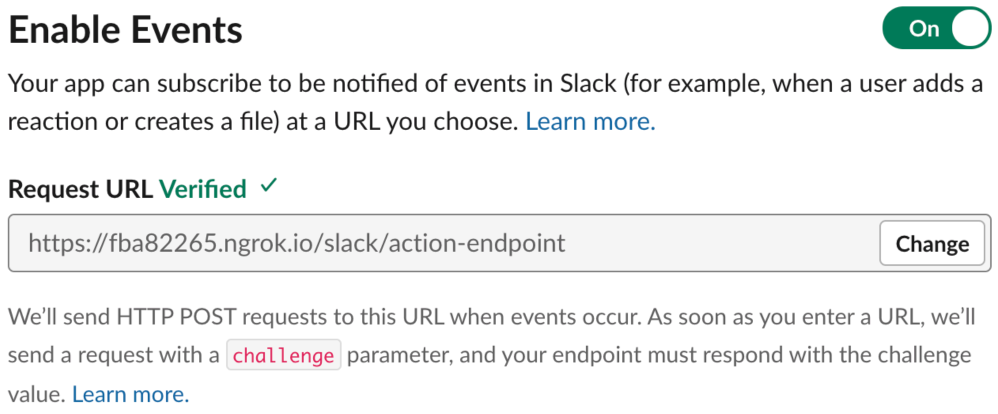
This deprecated Verification Token can still be used to verify that requests come from Slack, but we strongly recommend using the above, more secure, signing secret instead.

SETTING UP YOUR SLACK APPLICATION

Enable events and set your URL. Click the “Event Subscriptions” link in the “Features” section. Click the On/Off control to turn events On.

Enter your URL. Using the slack_incoming module, your controller is at /slack/action-endpoint.

This makes a call to your Drupal site that slack_incoming handles. It returns the challenge parameter back to Slack when the endpoint is hit with the verification event.



Enable Events

Your app can subscribe to be notified of events in Slack (for example, when a user adds a reaction or creates a file) at a URL you choose. Learn more.

Request URL Verified ✓

https://fba82265.ngrok.io/slack/action-endpoint

We’ll send HTTP POST requests to this URL when events occur. As soon as you enter a URL, we’ll send a request with a `challenge` parameter, and your endpoint must respond with the challenge value. Learn more.

SUBSCRIBING TO EVENTS

Slack has a bunch of events to subscribe to. Our MUD only cares about a few. We're going to select:

- `app_home_opened` – This allows us to populate the Home tab of the Slack app's workspace.
- `message.im` – This allows the app to receive messages via DM, which is how users communicate with the game.

Now press Save Changes.

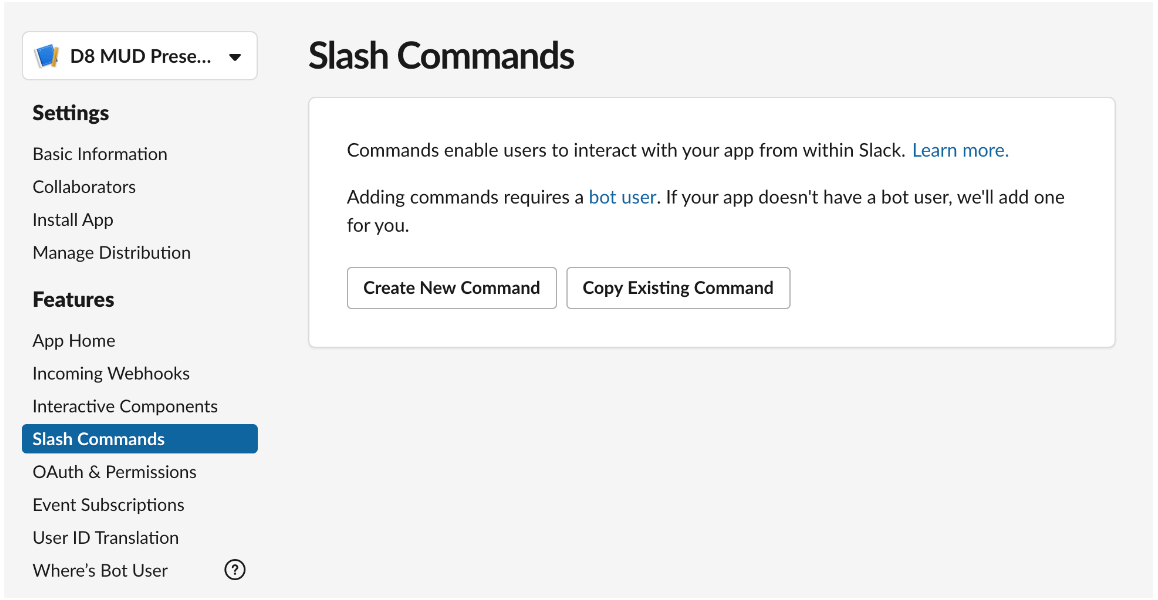
SLASH COMMANDS

We'll have two slash commands:

- /games
- /joingame <game id>

We set these up in our slack_mud controller.

Enter them into the Slack app by clicking Slash Commands in the left menu and clicking Create New Command.



Slash Commands

Commands enable users to interact with your app from within Slack. Learn more.

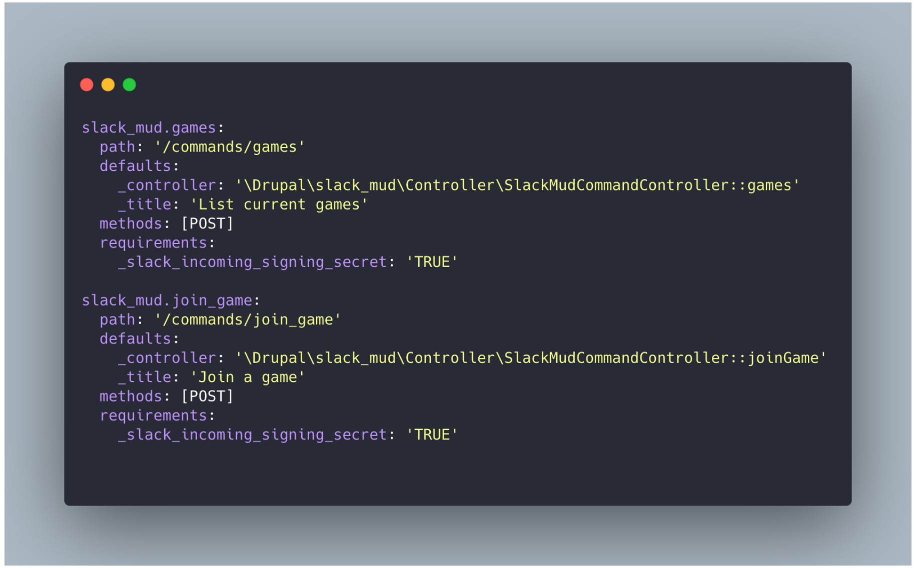
Adding commands requires a bot user. If your app doesn't have a bot user, we'll add one for you.

SLASH COMMANDS

We'll have two slash commands:

- `/games` at `/commands/games`
- `/joingame <game id>` at `/commands/join_name`

We set these up in our `slack_mud` controller.

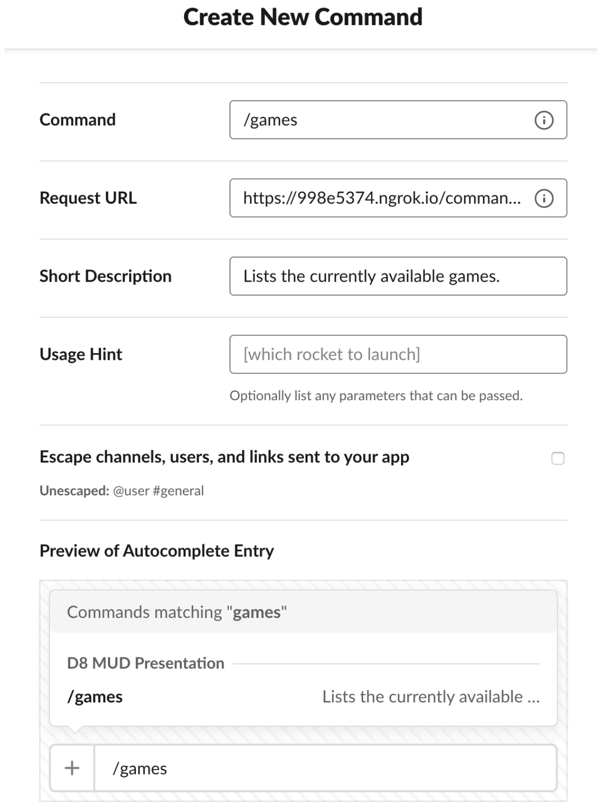


```yaml
slack_mud.games:
  path: '/commands/games'
  defaults:
    _controller: '\Drupal\slack_mud\Controller\SlackMudCommandController::games'
    _title: 'List current games'
  methods: [POST]
  requirements:
    _slack_incoming_signing_secret: 'TRUE'

slack_mud.join_game:
  path: '/commands/join_game'
  defaults:
    _controller: '\Drupal\slack_mud\Controller\SlackMudCommandController::joinGame'
    _title: 'Join a game'
  methods: [POST]
  requirements:
    _slack_incoming_signing_secret: 'TRUE'
```

SLASH COMMANDS - /GAMES

We declared `/games` in our `slack_mud` controller at `/commands/games`.

Press Save when you're done entering your slash command info.



Create New Command

Command	/games
Request URL	https://998e5374.ngrok.io/comman...
Short Description	Lists the currently available games.
Usage Hint	[which rocket to launch]
	Optionally list any parameters that can be passed.

Escape channels, users, and links sent to your app ☐

Unescaped: @user #general

Preview of Autocomplete Entry

Commands matching "**games**"

D8 MUD Presentation

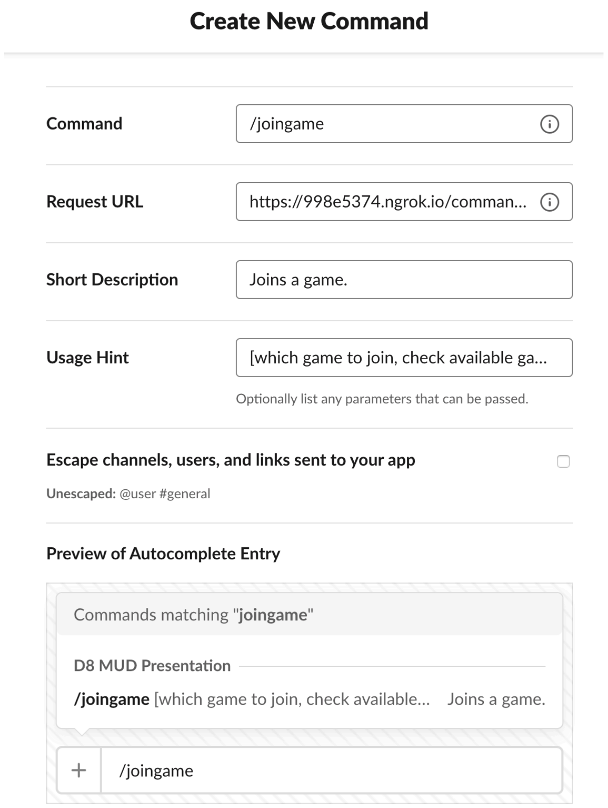
/games Lists the currently available ...

+ /games

SLASH COMMANDS - /JOINGAME

We declared `/joingame` in our `slack_mud` controller at `/commands/join_game`.

Press Save when you're done entering your slash command info.



Create New Command

Command	/joingame
Request URL	https://998e5374.ngrok.io/comman...
Short Description	Joins a game.
Usage Hint	[which game to join, check available ga...

Optionally list any parameters that can be passed.

Escape channels, users, and links sent to your app ☐

Unescaped: @user #general

Preview of Autocomplete Entry

Commands matching "**joingame**"

D8 MUD Presentation

/joingame [which game to join, check available... Joins a game.

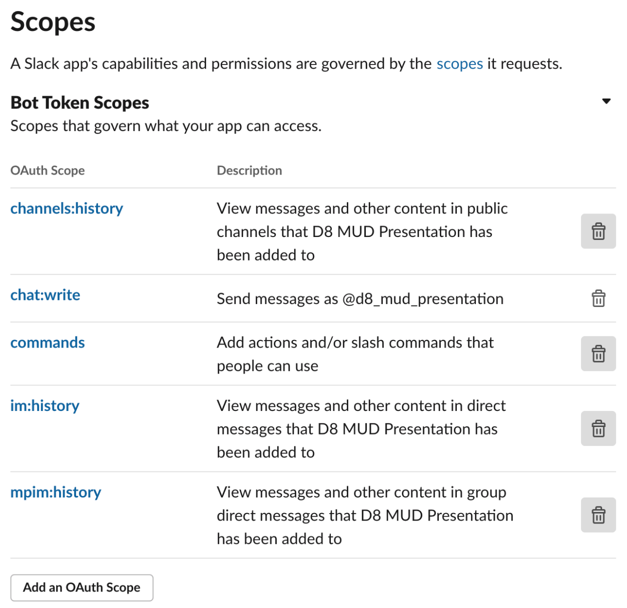
/joingame

SETTINGS

Go back to the App Home page and enable Always Show My Bot as Online and Home Tab.

We'll need to add a scope that allows `slack_incoming` to message users. Click OAuth & Permissions and scroll down to Scopes. Click Add an OAuth Scope and select `chat:write`. This is important! If you don't do this, your app won't be able to send messages to the users.

If your messages aren't appearing in Slack, double check your OAuth bot key first, then this scope setting second.



Scopes

A Slack app's capabilities and permissions are governed by the scopes it requests.

Bot Token Scopes

Scopes that govern what your app can access.

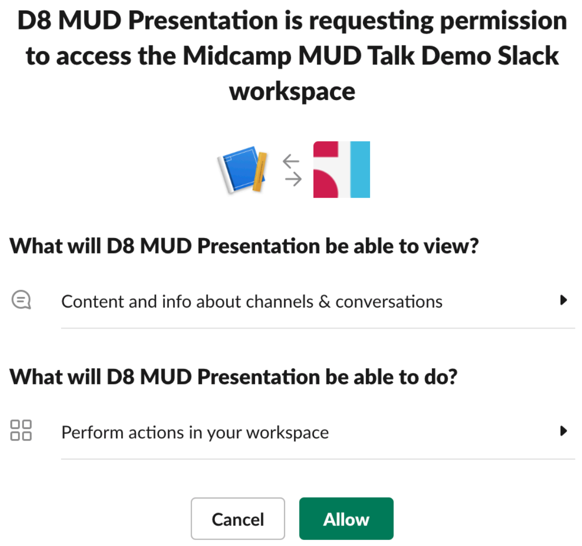
OAuth Scope	Description
channels:history	View messages and other content in public channels that D8 MUD Presentation has been added to
chat:write	Send messages as @d8_mud_presentation
commands	Add actions and/or slash commands that people can use
im:history	View messages and other content in direct messages that D8 MUD Presentation has been added to
mpim:history	View messages and other content in group direct messages that D8 MUD Presentation has been added to

OAUTH

We need one more key to fully integrate the app.

Go to OAuth & Permissions in api.slack.com and press Install App to Workspace. It will verify the permissions it needs. Press Allow to finish installing the app.

Now you'll have a Bot User OAuth Token. Copy that and paste it into your slack_incoming settings (Configuration > Web services > Slack).



D8 MUD Presentation is requesting permission to access the Midcamp MUD Talk Demo Slack workspace

What will D8 MUD Presentation be able to view?

Content and info about channels & conversations

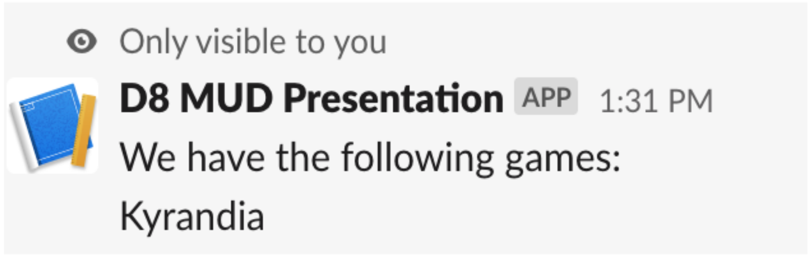
What will D8 MUD Presentation be able to do?

Perform actions in your workspace

USING THE APP

Go to your workspace in Slack. You'll see the new application bot under Recent Apps.

Test a command, like /games.



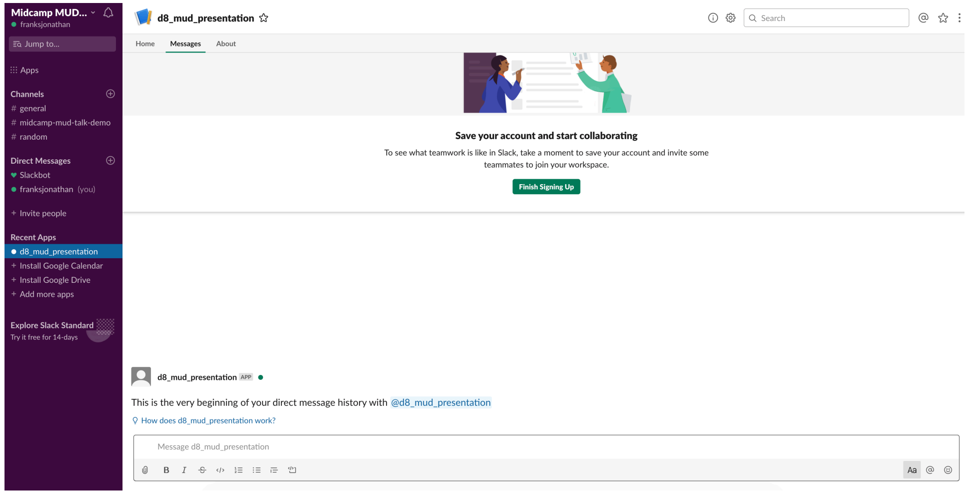
Only visible to you

D8 MUD Presentation APP 1:31 PM

We have the following games:

Kyrandia

Now /joingame kyrandia.



Save your account and start collaborating

To see what teamwork is like in Slack, take a moment to save your account and invite some teammates to join your workspace.

d8_mud_presentation APP

This is the very beginning of your direct message history with @d8_mud_presentation

How does d8_mud_presentation work?

Testing

TESTING

- Testing the plugins with Behat - Kernel tests would have been another option, but for every command we need to test, we need to mock so many things that it was faster and more readable to use Behat.
- Manually testing Slack.

Resources

SOURCE CODE AND SLIDES

Everything we used in this talk is located at github. The original Galacticomm source code is posted without permission in the repository with my ported version.

The `slack_incoming` module can be found on d.o.

The `word_grammar` module can be found on d.o.

(The `slack_mud` module might eventually be found on d.o. Not available yet!)

You can find the source code from this presentation at https://tiny.cc/SlackMud.

Please leave feedback at https://mid.camp/6280.

Q&A

https://www.breaktech.com

Session feedback
http://mid.camp/266

Jonathan "Jack" Franks
Jonathan.franks@breaktech.com