Create custom Automations

- Vocabulary
  - Plugin API
  - Scriptable API
    - field
    - triggerables and triggerable!
    - placeholders
    - script
  - Localization

:information_source: This is a draft, and may need some extra work.

Vocabulary

- **trigger**: represents the name of the trigger, eg: `user_added_to_group`
- **triggerable**: represents the code logic associated to a trigger, eg: `triggers/user_added_to_group_.rb`
- **script**: represents the name of the script, eg: `send_pms`
- **scriptable**: represents the code logic associated to a script, eg: `scripts/send_pms.rb`

Plugin API

```
add_automation_scriptable(name, &block)
add_automation_triggerable(name, &block)
```

Scriptable API

field

`field :name, component:` lets you add a customisable value in your automation's UI.

List of valid components:

```
# foo must be unique and represents the name of your field.

field :foo, component: :text # generates a text input
field :foo, component: :list # generates a multi select text input where users can fill values
field :foo, component: :choices, extra: { content: [ {id: 1, name: 'your.own.i18n.key.path' }
] } # generates a combo-box with a custom content
field :foo, component: :boolean # generate a checkbox input
field :foo, component: :category # generates a category-chooser
field :foo, component: :group # generates a group-chooser
field :foo, component: :date_time # generates a date time picker
field :foo, component: :tags # generates a tag-chooser
field :foo, component: :user  # generates a user-chooser
field :foo, component: :pms  # allows to create one or more PM templates
field :foo, component: :categories  # allows to select zero or more categories
field :foo, component: :key-value  # allows to create key-value pairs
field :foo, component: :message  # allows to compose a PM with replaceable variables
field :foo, component: :trustlevel  # allows to select one or more trust levels
```

triggerables and triggerable!

```
# Lets you define the list of triggerables allowed for a script
triggerables %i[recurring]

# Lets you force a triggerable for your script and also lets you force some state on fields
field :recurring, component: :boolean
triggerable! :recurring, state: { foo: false }
```

placeholders

```
# Lets you mark a key as replaceable in texts using the placeholder syntax `%%sender%%`
placeholder :sender
```

Note that it's the responsibility of the script to provide values for placeholders and to apply the replacement using `input = utils.apply_placeholders(input, { sender: 'bob' })`

script

This is the heart of an automation and where all the logic happens.

```
# context is sent when the automation is triggered, and can differ a lot between triggers
script do |context, fields, automation|
```

```
end
```

Localization

Each field you will use will depend on i18n keys and will be namespaced to their trigger/script.

For example a scriptable with this content:

```
field :post_created_edited, component: :category
```

Will require the following keys in client.en.yml:

```
en:
  js:
    discourse_automation:
      scriptables:
        post_created_edited:
          fields:
            restricted_category:
              label: Category
               description: Optional, allows to limit trigger execution to this category
```

Note that description is optional here.

□□□ #2
□ □□□ □□□ 2026-10-31 13:31:51 UTC
□ □□□ □□□ 2026-10-31 13:31:51 UTC