Overview of EAVs and dynamic forms

configuration, eav, attributes, form, attachment

Dynamic forms are used for:

- extending standard entity attributes with custom, project specific attributes,
- dynamic configurations - e.g. system connector configurations.

Dynamic forms are supported for selected entities:

- `SysSystem` - linked to the systems
- `IdmIdentity` - identities
- `IdmIdentityRole` - assigned roles to identities (respectively assigned to identity contracts)
- `IdmIdentityContract` - Contractual relationships
- `IdmRole` - roles
- `IdmTreeNode` - tree structure items.
- `IdmForm` - common forms
- `VsAccountFormValue` - virtual system attributes

Dynamic form instances (values) are saved in individual tables according to the entity which they are linked to ⇒ which is their owner (e.g. the entity `IdmIdentityFormValue`, `IdmRoleFormValue`). Form values are not saved if `null` value (by persistent type) is given ⇒ only filled values are saved.

Agenda for working with forms

On the FE, there is an agenda of forms - their definition and attributes. Each definition can contain zero or more attributes.

Dynamic form attribute supports the following data types (`persistentType`):

- CHAR - one character
- TEXT - strings (long text). Not indexed - SHORTTEXT usage is preferred.
- SHORTTEXT - strings (2000 chars). Indexed.
- INT - integer
- LONG - long
- DOUBLE - saved as bigdecimal
- BOOLEAN - true / false / null
- DATE - date (without time)
- DATETIME - date with time
- BYTEARRAY - byte[]
- UUID - uuid identifier. Indexed.
- ATTACHMENT - attachment (~binary file). Read more about attachments.

Changing `persistentType` and `confidential` is possible only for attributes without persisted values ⇒ when attribute is not used for some values. Data migration, when attribute's `persistentType` or `confidential` is changed is not supported now.

with these specified settings:

- `readonly`
- `multi values` - Is represented on the front-end by a textarea, where a line is a value (a new line separates the values). This property is supported for persistent types `CHAR`, `TEXT`, `INT`, `LONG`, `DOUBLE` and `UUID`.
- `confidential` - .The values are stored in an confidential storage). Stored values of these attributes - substitute characters only - are loaded on the front-end. The value can only be changed and determined whether it is filled in. This property is supported for persistent types `CHAR`, `TEXT`, `INT`, `LONG`, `DOUBLE`, `UUID`, `BYTEARRAY`.
- `required` - value validation, read more.
- `unique` - value validation, read more.
- `min` - value validation, read more.
- `max` - value validation, read more.
- `regex` - value validation, read more.

* `validationMessage` - custom message, when some validation fails, read more.

It is necessary to be cautious when editing individual form attributes as the logic linked to this form can be rendered non-functional.

Common forms

Common forms is used for saving internal dynamic forms for: - report filters - long running task properties (comming soon) - authorization policiy properties (comming soon)

Forms have to have a form definition and an owner, the latter possibly being an entity that implements the `FormableEntity` interface. One owner can own more than one form. When the owner is deleted, then all forms have to be deleted, too - override owner's service delete method properly. A form can be shared between owners - e.g. report filter can be used as input (properties) for a long running task - this is the main reason why common forms exist, don't use this common form for storing extended attributes mentioned above (e.g. they will not be shown on frontend).

Localization

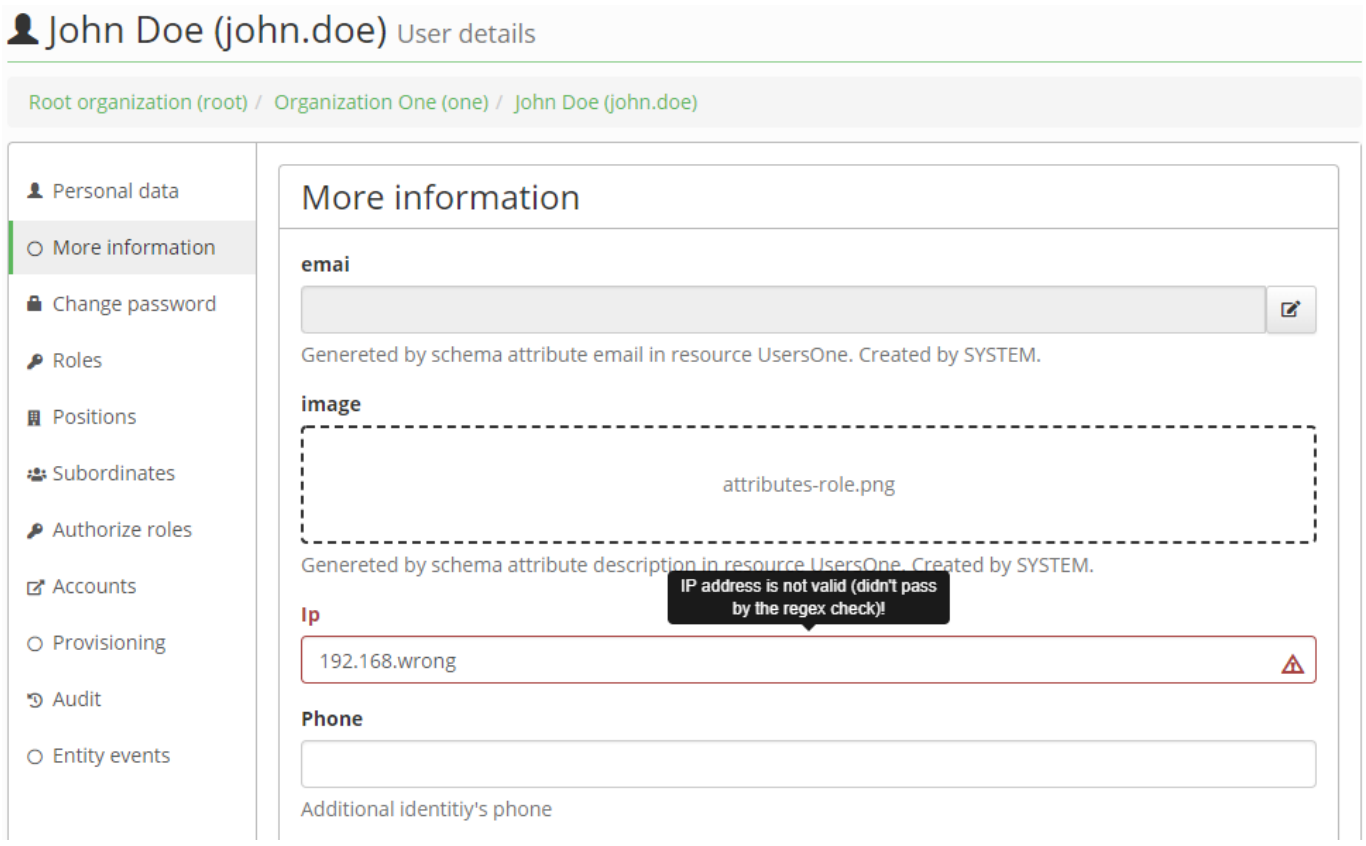
For form attributes, it is possible to add localization into cs.json and en.json for each module.

New tab on the form definition detail can be used for creating localization.

Validation

validation

For form attribute values, it is possible to configure prepared validations. Validation are evaluated (**on the backend**), when form with extended attributes is saved and sent to backend. Simple validations as `required`, `min`, `max` are evaluated on frontend after value is changed.


John Doe (john.doe) User details

Root organization (root) / Organization One (one) / John Doe (john.doe)

More information

emai

Genereted by schema attribute email in resource UsersOne. Created by SYSTEM.

image

attributes-role.png

Genereted by schema attribute description in resource UsersOne. Created by SYSTEM.

IP address is not valid (didn't pass by the regex check)!

Ip

192.168.wrong

Phone

Additional identitiy's phone

Required

Value is required.

Unique

Value has to be unique.

Unique validation is not supported for `BYTEARRAY` and `ATTACHMENT` persistent types.

Min, Max

Value has to be greater than (lesser than) or equal given `min` (`max`) values. Real number (38,4) can be configured.

Min and max validation is supported for numeric `DOUBLE`, `INT`, `LONG` persistent types.

Code lists

A code list can be defined and used on frontend forms → defines options for the select box (e.g. used on role detail for the `environment` attribute). Code lists items could have additional extended attributes. Code list works as decorator only. When the whole code list is deleted, then the only impact is that raw item codes will be rendered on frontend ⇒ when a code list or code list item is not found, then a raw value (item's code) is shown instead.

Frontend localization is supported in the item's name. For example item with name `environment.development.title` can be localized.

EAVs and authorization policies

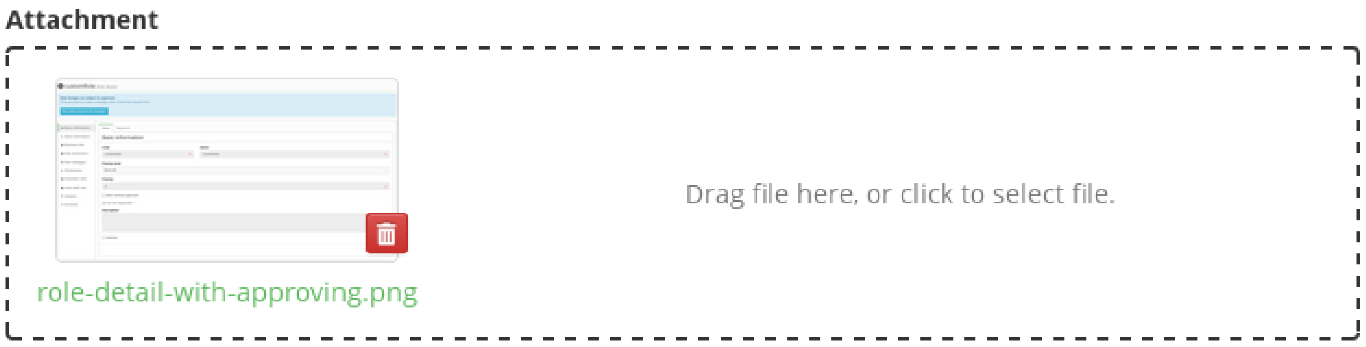
Identity form values can be secured by authorization policies when some identity extended attributes have to be watched carefully - see how in configure authorization policies.

Authorization policies only support identity extended attribute values. Support for other entities can be added in the future.

Attachments

Attachments can be uploaded for the attributes with the persistent type `ATTACHMENT`. Attachment is uploaded immediately after user has selected a file from their file system - attachment is uploaded as temporary to server and it's identifier is used as an extended attribute value. The extended attribute can then be saved (when the entire extended form is submitted).

Attachment can be downloaded from frontend. If an attachment is a picture (~ attachment mime type starts with `image/*`), then a preview of the saved extended attribute's attachment is available directly in the extended attribute form. Downloading and previewing is supported for the following agendas (entity types): identity, role, tree node, contract and contract slice ⇒ role requests (role lifecycle) and identity roles (parameters for the assigned roles) are


Attachment
Drag file here, or click to select file.
role-detail-with-approving.png

Read more about attachments usage in the application (e.g. how to increase maximum file size).