Dynamic form - creating a new attribute renderer

In this tutorial we will describe how to create a new attribute renderer for a dynamic form attribute.

A new render will be used for a persistent type `INT`. Renderer define a new face `PRIORITY-SELECT` and an input will be represented as five radio buttons.

What do you need before you start

- You need to install CzechIdM 7.4.0 (and higher). We have CzechIdM installed for this tutorial on server `http://localhost:8080/idm-backend`.

Source codes for this tutorial can be found in the example module

01 Create a renderer component

We will use `AbstractFormAttributeRenderer` component as superclass - contains some default method implementations. And define / override methods for getting / setting form value from radio buttons. Radio form component is not implemented in core components yet, so we are using simple radio inputs.

```
import React from 'react';
//
import { Advanced } from 'czechidm-core';

/**
 * Priority form value component. Doesn't support:
 * - multiple
 * - confidential
 * - placeholder
 * - helpBlock
 * - validation
 * - required
 */
export default class PrioritySelectFormAttributeRenderer extends
Advanced.AbstractFormAttributeRenderer {

  constructor(props, context) {
    super(props, context);
    // prepare radio value
    this.state = {
      value: null
    };
  }
```

```
  /**
   * Fill form value field by persistent type from input value
   *
   * @param  {FormValue} formValue - form value
   * @param  {object} formComponent value
   * @return {FormValue}
   */
  fillFormValue(formValue, rawValue) {
    formValue.longValue = rawValue;
    // TODO: validations for numbers
    return formValue;
  }

  /**
   * Returns value to ipnut from given (persisted) form value
   *
   * @param  {FormValue} formValue
   * @return {object} value by persistent type
   */
  getInputValue(formValue) {
    return formValue.longValue;
  }

  /**
   * Filled form value
   *
   * @return {FormValue}
   */
  toFormValue() {
    const { values } = this.props;
    //
    return this.fillFormValue(this.prepareFormValue(values ? values[0] :
null), this.state.value);
  }

  /**
   * Set value on radio is clicked (changed)
   *
   * @param {event} event
   */
  setValue(event) {
    this.setState({
      value: event.target.value
    });
  }

  renderSingleInput() {
    const { attribute, readOnly, values } = this.props;
    const singleValue = this.state.value || this.toInputValue(values);
```

```
    // create radio inputs
    const inputs = [];
    for (let i = 1; i <= PrioritySelectFormAttributeRenderer.RADIO_COUNT;
i++) {
      inputs.push(
        <label className="radio-inline">
          <input
            name={ attribute.name }
            type="radio"
            readOnly={ readOnly || attribute.readonly }
            value={ i }
            defaultChecked={ singleValue === i }/> { i }
        </label>
      );
    }
    // raw bootstrap styles are used in this example (Basic radio component
should be created instead)
    return (
      <div className="form-group">
        <label className="control-label">{ attribute.name }</label>
        <div className="radio" onChange={this.setValue.bind(this)}>
          { inputs }
        </div>
      </div>
    );
  }
}

PrioritySelectFormAttributeRenderer.RADIO_COUNT = 5;
```

02 Register renderer component

Component is created, now we need to register her in component descriptor for start using her in application. New renderer will be available as new face type on form attribute detail page too, when persistent type `INT` will be selected.

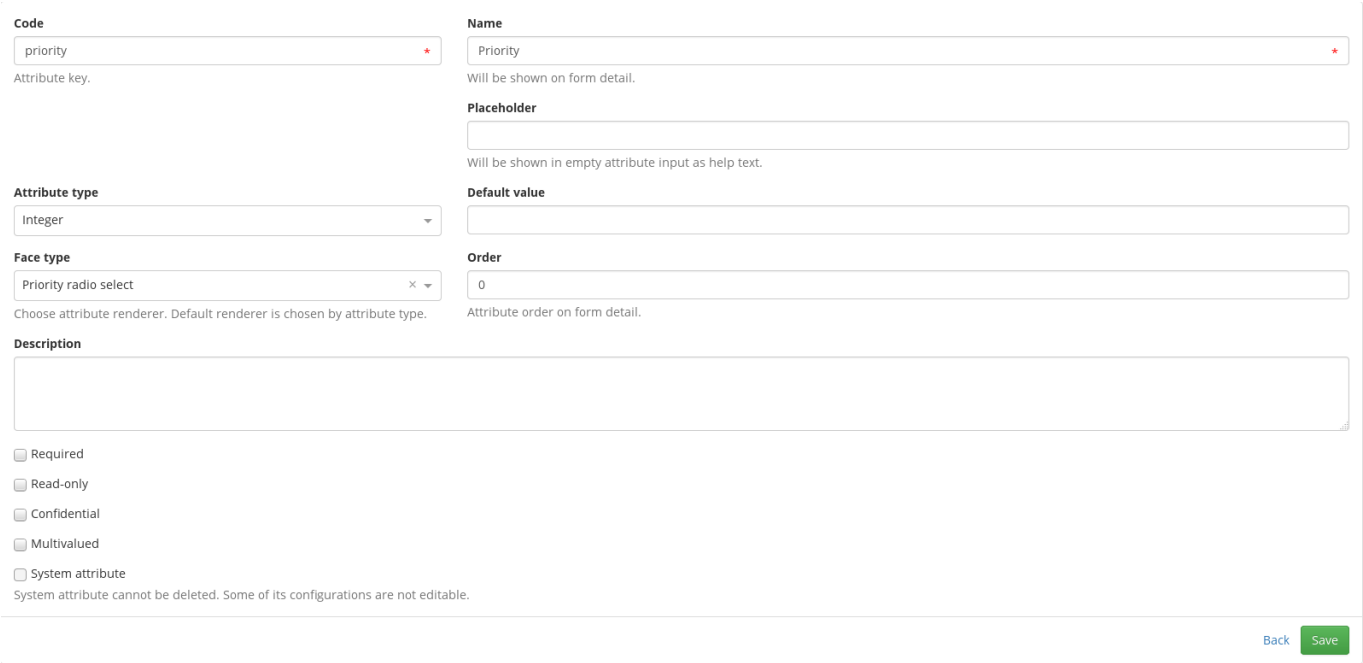
Add component into module's `component-descriptor.js`:

```
...
module.exports = {
  ...
  'components': [
    ...
    {
      'id': 'priority-select-form-value',
      'type': 'form-attribute-renderer',
      'persistentType': 'INT',
      'faceType': 'PRIORITY-SELECT',
      'component':
```

```
require('./src/components/PrioritySelectFormAttributeRenderer'),
      'labelKey': 'Priority radio select'
    }
    ...
  ]
};
...
```

03 Define form attribute

We can select renderer for some dynamic form atribute with peristent type `INT` in form definitions agenda - e.g. we will add new attribute to default identity form definition:



On identity detail in tab more information, we can see, how result's look like:

Advanced

Renderer name can be localized. See `labelKey` attribute in component definition. You can add localization key here instead text.