Module - Result codes

The aim of this tutorial is show the way, how to define result codes in custom module and how to expose list of codes on frontend.

What do you need before you start

- You need to install CzechIdM 8.1.0 (and higher). We have CzechIdM installed for this tutorial on server `http://localhost:8080/idm-backend`.
- Create an identity, which has permission to read installed backend modules (`MODULE_READ` permission). We are using the default `admin:admin` identity.

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

01 Create module result codes

The first of all, we need to define result codes, which can be used, when some information has to be sent to client - as message or result of some operation. We will create new class in our module (example module is used here), which implements `ResultCode` interface:

```
package eu.bcvsolutions.idm.example.domain;

import org.springframework.http.HttpStatus;
import eu.bcvsolutions.idm.core.api.domain.ResultCode;

/**
 * Enum class for formatting response messages (mainly errors).
 * Every enum contains a string message and corresponding https HttpStatus
code.
 *
 * Used http codes:
 * - 2xx - success
 * - 4xx - client errors (validations, conflicts ...)
 * - 5xx - server errors
 */
public enum ExampleResultCode implements ResultCode {

    EXAMPLE_CLIENT_ERROR(HttpStatus.BAD_REQUEST, "Example client error, bad
value given [%s]"),
    EXAMPLE_SERVER_ERROR(HttpStatus.INTERNAL_SERVER_ERROR, "Example server
error with parameter [%s]");

    private final HttpStatus status;
    private final String message;

    private ExampleResultCode(HttpStatus status, String message) {
```

```
        this.message = message;
        this.status = status;
    }

    public String getCode() {
        return this.name();
    }

    public String getModule() {
        return "example";
    }

    public HttpStatus getStatus() {
        return status;
    }

    public String getMessage() {
        return message;
    }
}
```

Message ca be sent to clien as exception (example only):

```
@ResponseBody
    @RequestMapping(method = RequestMethod.GET, path = "/client-error")
    @ApiOperation(
            value = "Example client error",
            notes= "Example client error with given parameter.",
            nickname = "exampleClientError",
            tags={ ExampleController.TAG },
            authorizations = {
                @Authorization(SwaggerConfig.AUTHENTICATION_BASIC),
                @Authorization(SwaggerConfig.AUTHENTICATION_CIDMST)
            })
    public void clientError(
            @ApiParam(value = "Error parameter", example = "parameter",
defaultValue = "value")
            @RequestParam(required = false, defaultValue = "parameter")
String parameter) {
        // lookout - ImmutableMap parameter values cannot be {@code null}
        throw new
ResultCodeException(ExampleResultCode.EXAMPLE_CLIENT_ERROR,
ImmutableMap.of("parameter", String.valueOf(parameter)));
    }
```

The better usage of result codes is as `OperationResult` - it's used for long running tasks, reports, provisioning operations etc.. Or simply `DefaultResultModel` with code in constructor can be used and returned to client. Component `Advanced.OperationResult` can be used for rendering `OperationResult` on frontend. Or the simpler component `Basic.FlashMessage` with `FlashMessageManager#convertFromResultModel` method, if only `ResultModel` has to be rendered - but `OperationResult` is prefered).

02 Create localization for result codes

All defined result codes should be localized on frontend, add localization into locales:

```
...
  "error": {
    "EXAMPLE_SERVER_ERROR": {
      "title": "Example server error",
      "message": "Example server with parameter [{{parameter}}]."
    },
    "EXAMPLE_CLIENT_ERROR": {
      "title": "Example client error",
      "message": "Example client error, bad value given [{{parameter}}]."
    }
  },
...
```

Only en localization is shown here, complete localization files can be found in example module.

If localization on frontend is not found, then default message defined directly in `ResultCode` implemetation will be used.

03 Expose result codes

Module can provide defined result codes to frontend, update module descriptor in your module, add:

```
...
public class ExampleModuleDescriptor extends PropertyModuleDescriptor {
...
        @Override
    public List<ResultCode> getResultCodes() {
        return Arrays.asList(ExampleResultCode.values());
    }
...
```

04 Test

Go to menu **Setting → Modules → Modules (backend)**, you will see new button in documentation column:

core	CzechIdM Core Impl	BCV solutions s.r.o.	8.0.1-SNAPSHOT	Core services implementations, entities, rest endpoint implementations	Api Html Result codes
example	CzechIdM Module Example	BCV solutions s.r.o.	8.0.1-SNAPSHOT	CzechIdM Module Example / Skeleton. Contains module descriptor, flyway, swagger configuration etc. Can be used as new module template.	Api Html Result codes Deactivate
ic	CzechIdM IC	BCV solutions s.r.o.	8.0.1-SNAPSHOT	CzechIdM Module IC - Identity Connector Framework. Ensure load, configration, call all available connectors. Supproted more IC (first is ConnectId).	Result codes

and after click on the button, modal will be shown: