Command

Plugin Documentation

Table of Contents

Chapter 1. Introduction

The Command Plugin, gives Grails a convention for command objects, and makes them a grails Artefact. It also adds an AST Transform call @ErrorsHandler to eliminate some of the boilerplate of dealing with errors from command objects.

Chapter 2. Version History

- 1.0
  - Initial Release including the ErrorsHandler annotation.

Chapter 3. Getting Started

1.In grails 3 add the following dependency to your gradle.build:

```
compile "org.grails.plugins:command:1.0.0"
```

2.Run grails create-command:

```
grails create-command >package>.<commandName>
```

or for the default package

```
grails create-command <commandName>
```

3.If this is the first command you've created run the refresh dependencies in Intellij, This will add the command folder to the source set. Or you can manually add the folder to the source set, by right clicking on the folder selecting "Mark Directory as" and selecting, "Sources Root". In a future release I'll look for a better way to automate, and eliminate this step.

3.1. Example application

GitHub testCommand

Chapter 4. The ErrorsHandler AST Transforms

The ErrorsHandler AST(@ErrorHandler()) transform Enforce injects a call to the default, injected by the plugin(see bellow). The ErrorsHandler can either be applied at the action level of a controller or at the class level. If the ErrorsHandler is applied at the class level, it will be injected to each action, but, applying it at the action level will override. The ErrorHandler can also be passed an optional String name of an alternative ErrorHandler. The ErrorHandler will not be applied to private methods or Method annotated with @SkipErrorsHandler.

If you use parameter outside of a command object, and those parameters have binding errors, those will be included in the list sent to the error handler, but for each parameter you will have to include an entry in your i18n message bundle for the parameter like:

```
params.<Your parameter name here>.conversion.error = Your error massage for <Your
parameter name> had an error binding.
```

Example Usage:

```
    package test.command

    import com.virtualdogbert.ast.ErrorsHandler
    import grails.converters.JSON
    @ErrorsHandler
    class TestController {

        def index(TestCommand test) {
            //some controller code
        }

        @ErrorsHandler(handler = 'someOtherErrorHandler') //over rides the default.
        def list(TestCommand test) {
            //some controller code
        }

        @SkipErrorsHandler //Skips the error handler injection from the class
annotation.
        def list(TestCommand test) {
            //some controller code
        }

        //Your error handler
        private boolean someOtherErrorHandler(List commandObjects) {
            List errors = commandObjects.inject([]) { result, commandObject ->

                if (commandObject.hasErrors()) {
                    result + (commandObject.errors as JSON)
                } else {
                    result
                }

            } as List

            if (errors) {
                //Do something
                return true
            }
            //possibly do something else
            return false
        }
    }
```

Code injected by the ErrorHandler, into the actions:

```
    if(errorsHandler( [<every commandObject in the actions parameter list>] )){ return
null }
```

Default Error Handler injected into all controllers:

```
    boolean errorsHandler(List commandObjects) {
        List errors = commandObjects.inject([]) { result, commandObject ->

            if (commandObject.hasErrors()) {
                result + (commandObject.errors as JSON)
            } else {
                result
            }

        } as List

        if (errors) {
            render errors
            return true
        }

        return false
    }
```