mge-graphql Documentation

Release latest

Jan 26, 2022

CONTENTS

CHAPTER

ONE

MGE-GRAPHQL

1.1 Introduction

MGE-GraphQL is a Python library for building GraphQL mutations fast and easily.

- **Data Validations:** A similar data validation workflow as Django.
- **Errors:** Support for throwing errors
- **Permissions:** Support for user permissions

1.2 Installation

For instaling MGE-GraphQL, just run this command in your shell

```
pip install "mge-graphql"
```

1.3 Examples

Here is one example for you to get started: Create `error_codes.py` and define some errors

```
from enum import Enum
from mge_graphql.utils.error_codes import (
    MGE_ERROR_CODE_ENUMS,
    generate_error_codes
)


class AccountErrorCode(Enum):
    # Here you define your Error Codes
    INVALID_PASSWORD = "invalid_password"

# Register error codes
MGE_ERROR_CODE_ENUMS.append(AccountErrorCode)
generate_error_codes()
```

Create `enums.py` to define your Graphene Error Enums

```
import graphene
import error_codes as account_error_codes

# Create Graphene Enum
AccountErrorCode = graphene.Enum.from_enum(account_error_codes.AccountErrorCode)
```

Create `types.py` to create your custom error graphql object type

```
from mge_graphql.types.common import Error
from enums import AccountErrorCode

class AccountError(Error):
    # Custom fields
    # Support for error_code
    code = AccountErrorCode(description="The error code.", required=True)
```

Create `mutations.py` to create your first mutation

```
from mge_graphql.mutations.base import BaseMutation
from mge_graphql.exceptions import ValidationError
from enums import AccountErrorCode
from types import AccountError
import graphene

class AccountRegister(BaseMutation):
    # YOUR GRAPHENE FIELDS
    username = graphene.String(required=True)
    password = graphene.String(required=True)

    class Arguments:
        username = graphene.String(required=True)
        password = graphene.String(required=True)

    class Meta:
        description = "Register a new account."
        # Set our custom AccountError class
        error_type_class = AccountError

    @classmethod
    def clean_password(cls, password, errors):
        if len(password) < 6:
            errors["password"].append(
                ValidationError(
                    {
                        "password": ValidationError(
                            "Password cannot be less than 6 characters.",
                            code=AccountErrorCode.INVALID_PASSWORD
                        )
                    }
                )
            )

        return password
```

(continues on next page)

(continued from previous page)

```
    @classmethod
    def clean(cls, **data):
        errors = defaultdict(list)
        cls.clean_password(data["password"], errors)

        if errors:
            raise ValidationError(errors)

        return data

    @classmethod
    def check_permissions(cls, context):
        # Permission Checks.
        # If False, then it will raise an Permission Denied Error
        return True

    @classmethod
    def perform_mutation(cls, _root, info, **data):
        cleaned_data = cls.clean(**data)

        cleaned_username = cleaned_data.get("username")
        cleaned_password = cleaned_data.get("password")

        # User Save // Any Mutation Logic

        return AccountRegister(
            username=cleaned_username,
            password=cleaned_password
        )
```

Create `schema.py` and register your mutation:

```
from mutations import AccountRegister
import graphene

class Mutation(graphene.ObjectType):
    account_register = AccountRegister.Field()


schema = graphene.Schema(mutation=Mutation)
```

1.3.1 And.. we are done! Let's try our mutation

invalid input:

```
mutation {
  accountRegister(username: "test", password: "234") {
    username
    password

    errors {
      field
      message
      code
    }
  }
}
```

```
{
  "data": {
    "accountRegister": {
      "username": null,
      "password": null,
      "errors": [
        {
          "field": "password",
          "message": "Password cannot be less than 6 characters.",
          "code": "INVALID_PASSWORD"
        }
      ]
    }
  }
}
```

valid input:

```
mutation {
  accountRegister(username: "test", password: "123456") {
    username
    password

    errors {
      field
      message
      code
    }
  }
}
```

```
{
  "data": {
    "accountRegister": {
      "username": "test",
      "password": "123456",
```

(continues on next page)

(continued from previous page)

```
      "errors": []
    }
  }
}
```

If method `check_permissions` returns False:

```
mutation {
  accountRegister(username: "test", password: "123456") {
    username
    password

    errors {
      field
      message
      code
    }
  }
}
```

```
{
  "data": {
    "accountRegister": {
      "username": null,
      "password": null,
      "errors": [
        {
          "field": null,
          "message": "You do not have permission to perform this action",
          "code": "PERMISSION_DENIED"
        }
      ]
    }
  }
}
```

1.4 Documentation

Documentation and links to additional resources are available at https://mge-graphql.readthedocs.io/