Cinnamon Documentation

Release 1.0

Alexander Ray

Nov 28, 2018

CONTENTS

Contents:

CHAPTER

ONE

PROJECT.CONTROLLERS PACKAGE

1.1 project.controllers.AddAccountController module

class project.controllers.AddAccountController.**AddAccountController**
Bases: flask.views.View

decorators = [<function login_required>]

dispatch_request()
Handler for add account controller. Takes data from account add form, creates new account with decorator classes

Returns Template

methods = ['GET', 'POST']

1.2 project.controllers.CreateReportController module

class project.controllers.CreateReportController.**CreateReportController**
Bases: flask.views.View

decorators = [<function login_required>]

dispatch_request()
Handler for create report. Takes filename data from form and generates report with user's current report generator

Returns Template

methods = ['GET', 'POST']

1.3 project.controllers.HomeController module

class project.controllers.HomeController.**HomeController**
Bases: flask.views.View

decorators = [<function login_required>]

dispatch_request()
Handler for requests to homepage. Retrieves data from model and passes to view

Returns Template

1.4 project.controllers.LoginController module

class project.controllers.LoginController.**LoginController**
Bases: flask.views.View

dispatch_request()
Login handler. Compares username and password to database, sends to homepage if approved

Returns Template

methods = ['GET', 'POST']

1.5 project.controllers.LogoutController module

class project.controllers.LogoutController.**LogoutController**
Bases: flask.views.View

dispatch_request()
Handler for logout

Returns Redirect to login page

methods = ['GET', 'POST']

1.6 project.controllers.SettingsController module

class project.controllers.SettingsController.**SettingsController**
Bases: flask.views.View

decorators = [<function login_required>]

dispatch_request()
Settings handler. Populates default fields with existing data. Recreates report generator based on user choices, following strategy design pattern.

Returns Template

methods = ['GET', 'POST']

1.7 project.controllers.SignupController module

class project.controllers.SignupController.**SignupController**
Bases: flask.views.View

dispatch_request()
Signup handler. Takes form data and creates a new user

Returns Template

methods = ['GET', 'POST']

1.8 project.controllers.SpendingController module

class project.controllers.SpendingController.**SpendingController**
Bases: flask.views.View

decorators = [<function login_required>]

dispatch_request()
Handler for adding spending instances. Populates default fields and drop down menus. Withdraws amount from specified account, creates a new spending instance.

Returns Template

methods = ['GET', 'POST']

1.9 project.controllers.forms module

class project.controllers.forms.**AddAccountForm**(**args*, ***kwargs*)
Bases: flask_wtf.form.Form

account = <UnboundField(StringField, ('Account',), {'validators': [<wtforms.validator

include_email = <UnboundField(BooleanField, ('Email Updates',), {})>

include_savings = <UnboundField(BooleanField, ('Save by Rounding',), {})>

class project.controllers.forms.**BaseAddressForm**(**args*, ***kwargs*)
Bases: flask_wtf.form.Form

city = <UnboundField(StringField, ('City',), {'validators': [<wtforms.validators.Inpu

state = <UnboundField(SelectField, ('State',), {'coerce': <class 'str'>, 'validators'

street_address = <UnboundField(StringField, ('Street Address',), {'validators': [<wtf

zip = <UnboundField(StringField, ('Zip Code',), {'validators': [<wtforms.validators.I

class project.controllers.forms.**BaseUserForm**(**args*, ***kwargs*)
Bases: flask_wtf.form.Form

email = <UnboundField(EmailField, ('Email',), {'validators': []})>

password = <UnboundField(PasswordField, ('Password',), {'validators': [<wtforms.valid

class project.controllers.forms.**CreateReportForm**(**args*, ***kwargs*)
Bases: flask_wtf.form.Form

filename = <UnboundField(StringField, ('Filename',), {'validators': [<wtforms.validat

include_description = <UnboundField(BooleanField, ('Include Description',), {})>

class project.controllers.forms.**LogSpendingForm**(**args*, ***kwargs*)
Bases: flask_wtf.form.Form

account = <UnboundField(SelectField, ('Account',), {'coerce': <class 'str'>, 'validat

amount = <UnboundField(DecimalField, ('Amount',), {'validators': [<wtforms.validators

date = <UnboundField(DateField, ('Date of Spending',), {'format': '%Y-%m-%d', 'defaul

description = <UnboundField(StringField, ('Description',), {'validators': [<wtforms.v

spending_type = <UnboundField(SelectField, ('Spending Type',), {'coerce': <class 'str

class project.controllers.forms.**SettingsForm**(**args*, ***kwargs*)
Bases: *project.controllers.forms.BaseAddressForm*

income = <UnboundField(IntegerField, ('Income',), {'validators': [<wtforms.validators

report_type = <UnboundField(SelectField, ('Report Type',), {'coerce': <class 'str'>,

summary_type = <UnboundField(SelectField, ('Summary Type',), {'coerce': <class 'str'>

class project.controllers.forms.**SignupForm**(**args*, ***kwargs*)
Bases: *project.controllers.forms.BaseUserForm*, *project.controllers.forms.BaseAddressForm*

income = <UnboundField(IntegerField, ('Income',), {'validators': [<wtforms.validators

CHAPTER

TWO

PROJECT.MODELS PACKAGE

2.1 project.models.Account module

class project.models.Account.**AccountEmailDecorator**(*dec*)
Bases: *project.models.Account.BaseAccountDecorator*

withdraw(*amount*)
Withdraw method for email decorator class. Calls other withdraw methods, then sends email with details

Parameters **amount** – Amount of money to withdraw

Returns Current total expenditure. Needed for decorator design pattern due to SQLAlchemy constraints

class project.models.Account.**AccountSavingsDecorator**(*dec*)
Bases: *project.models.Account.BaseAccountDecorator*

withdraw(*amount*)
Withdraw method for savings decorator. Calls other withdraw methods, adds to savings (rounded up to nearest dollar), and returns current sum of withdrawals.

Parameters **amount** – Amount of money to withdraw

Returns Current total expenditure. Needed for decorator design pattern due to SQLAlchemy constraints

class project.models.Account.**BaseAccount**(***kwargs*)
Bases: sqlalchemy.ext.declarative.api.Model

expenditure
Getter for total expenditure

Returns Sum of all spending in this account

name
Getter for account name

Returns Name of account

withdraw(*amount*)
Abstract withdraw method, to be implemented by subclasses

Parameters **amount** – Amount to withdraw

class project.models.Account.**BaseAccountDecorator**(*dec*)
Bases: *project.models.Account.BaseAccount*

withdraw(*amount*)
Abstract withdraw method, to be implemented by subclasses

Parameters **amount** – Amount to withdraw

class project.models.Account.**ConcreteAccount**(*name*)
Bases: *project.models.Account.BaseAccount*

withdraw(*amount*)
Method for withdrawing money from an account

Parameters **amount** – Amount of money to withdraw

Returns Current total expenditure. Needed for decorator design pattern due to SQLAlchemy constraints

2.2 project.models.Address module

class project.models.Address.**Address**(*street_address*, *city*, *state*, *zip*)
Bases: sqlalchemy.ext.declarative.api.Model

city
Getter for city

Returns City

state
Getter for state

Returns State

street_address
Getter for street address

Returns Street address

zip
Getter for zip code

Returns Zip code

2.3 project.models.ReportGenerator module

class project.models.ReportGenerator.**CSVReportGenerator**(*filename*)
Bases: *project.models.ReportGenerator.ReportGenerator*

generate_report(*instances*, *include_description*)
CSV report generator, using Python's CSV module

Parameters

- **instances** – List of spending instance objects
- **include_description** – Toggle for including description

Returns Flask response object

class project.models.ReportGenerator.**JSONReportGenerator**(*filename*)
Bases: *project.models.ReportGenerator.ReportGenerator*

generate_report(*instances*, *include_description*)
JSON report generator, using Python's JSON module

Parameters

- **instances** – List of spending instance objects
- **include_description** – Toggle for including description

Returns Flask reponse object

class project.models.ReportGenerator.**ReportGenerator**(*filename*)
Bases: sqlalchemy.ext.declarative.api.Model

default_filename
Getter for default filename of report generator

Returns Default filename

generate_report(*instances*, *include_description*)
Abstract method to generate report, must be implemented by subclasses

Parameters

- **instances** – List of spending instance objects
- **include_description** – Boolean value to determine whether or not to include a description

2.4 project.models.ReportType module

class project.models.ReportType.**ReportType**
Bases: enum.Enum

An enumeration.

CSV = 'CSV'

JSON = 'JSON'

2.5 project.models.SpendingHistory module

class project.models.SpendingHistory.**SpendingHistory**(***kwargs*)
Bases: sqlalchemy.ext.declarative.api.Model

add_spending_instance(*instance*)
Method to add a spending instance to your spending history

Parameters **instance** – Instance to add

spending_instances
Getter for spending instances

Returns List of spending instances, sorted in reverse chronological order by date

2.6 project.models.SpendingInstance module

class project.models.SpendingInstance.**DiningSpendingInstance**(*amount*, *account*, *date*, *description*)
Bases: *project.models.SpendingInstance.SpendingInstance*

class project.models.SpendingInstance.**RetailSpendingInstance**(*amount*, *account*, *date*, *description*)

Bases: *project.models.SpendingInstance.SpendingInstance*

class project.models.SpendingInstance.**SpendingInstance**(*amount*, *account*, *date*, *description*)

Bases: sqlalchemy.ext.declarative.api.Model

account
Getter for account associated with spending instance

Returns Account

amount
Getter for spending instance amount

Returns Amount

date
Getter for spending instance date

Returns Date

description
Getter for description associated with spending instance

Returns Description

id
Getter for id

Returns Id

2.7 project.models.SpendingInstanceFactory module

class project.models.SpendingInstanceFactory.**SpendingInstanceFactory**

Bases: object

static factory_method(*amount*, *account*, *date*, *description*, *instance_type*)
Static factory method for simple factory design pattern, to create spending instances. Types defined in SpendingType

Parameters

- **amount** – Amount of instance
- **account** – Account object
- **date** – Date object
- **description** – Optional description
- **instance_type** – Type from SpendingType

Returns Spending instance object

2.8 project.models.SpendingType module

class project.models.SpendingType.**SpendingType**

Bases: enum.Enum

An enumeration.

DINING = 'Dining'

RETAIL = 'Retail'

2.9 project.models.SummaryGenerator module

class project.models.SummaryGenerator.**DateOrientedSummaryGenerator**(***kwargs*)
Bases: *project.models.SummaryGenerator.SummaryGenerator*

class project.models.SummaryGenerator.**SummaryGenerator**(***kwargs*)
Bases: sqlalchemy.ext.declarative.api.Model

summary_template_method()
Template method for summary generator. Enforces ordering of summary generation using the template method design pattern

Returns HTML-formatted string

class project.models.SummaryGenerator.**TotalExpenditureSummaryGenerator**(***kwargs*)
Bases: *project.models.SummaryGenerator.SummaryGenerator*

2.10 project.models.SummaryType module

class project.models.SummaryType.**SummaryType**
Bases: enum.Enum

An enumeration.

DATE_ORIENTED = 'Date-oriented'

TOTAL_EXPENDITURE = 'Total Expenditure'

2.11 project.models.User module

class project.models.User.**User**(*email*, *password*, *info*)
Bases: sqlalchemy.ext.declarative.api.Model

account_names
Account names property to retrieve list of account names

Returns List of account names associated with user

accounts
Getter for list of user accounts

Returns List of user accounts

email
Getter for user email

Returns User email

get_account(*name*)
Get user account by name

Parameters **name** – Name of user account

Returns Account object or null if not found

get_id()
flask-login method for id retrieval

Returns User id

information
Getter for user information object

Returns User information object

is_active()
flask-login method

Returns True

is_anonymous()
flask-login method

Returns False

is_authenticated()
flask-login method

Returns True

password
Getter for user password

Returns User password

report_generator
Getter for report generator object

Returns Report generator object

spending_history
Getter for user spending history

Returns Spending history object

summary_generator
Getter for summary generator object

Returns Summary generator object

2.12 project.models.UserInformation module

class project.models.UserInformation.**UserInformation**(*income*, *address*)
Bases: sqlalchemy.ext.declarative.api.Model

address
Getter for user address

Returns User address object

income
Getter for user income

Returns User income

PYTHON MODULE INDEX

p

INDEX

R

S

T

U

W

Z