Entwurf
MatFlow
Workflowanwendung für Machine Learning Experimente

Florian Küfner, Soeren Raymond, Alessandro Santospirito,
Lukas Wilhelm, Nils Wolters

Dezember 2021

Inhaltsverzeichnis

1 Einleitung **3**

2 Systemüberblick **4**

3 Kommunikation **5**

4 Komponentenbeschreibung **6**
4.1 Client 6
4.2 Server 7

5 Paketbeschreibung **8**
5.1 Client Application 8
5.2 Server Application 8
5.3 Database 17

6 Abläufe **20**

7 Datenbank **21**
7.1 Datenbankaufbau 21

1 Einleitung

2 Systemüberblick

3 Kommunikation

4 Komponentenbeschreibung

4.1 Client

4.2 Server

5 Paketbeschreibung

5.1 Client Application

5.2 Server Application

5.2.1 Workflow Package

This package defines the logic behind the creation and the management of workflow templates, workflow instances and versions of workflow instances. It receives requests in the singleton object of WorkflowManager from the client application through the api package of the server application. The package also communicates with the database interface to safe and recall data aswell as with the API of Apache Airflow.

Class: WorkflowManager This class provides a singleton object that receives all requests addressed to this package. This class is also communicating with the API of Airflow as well as with the Database package.

Methods

getInstance():WorkflowManager Returns the singleton object if already existing otherwise it calls the private constructor

createTemplate(template:Template):void Causes the creation of a new template entry in the database. The new template can be used for creating workflow instances afterwards.

Parameters

- template: A Template object that provides all the necessery components for creating a new workflow template

Exceptions

- DoubleTemplateNameException Gets thrown if the user trys to use an already existing template name
- InvalidDagFileException Gets thrown if the user hands over an invalid dag-definition-file

createWorkflowInstanceFromTemplate(templateName:String, workflowInstanceName:String, configFiles:File[]):void Causes the instantiation of a workflow instance under the use of a workflow template.

Parameters

- templateName: The identifier of the template that is used for instantiation
- workflowInstanceName: The name of the new workflow instance
- configFiles: Cointains all the files needed for the execution of the workflow

Exceptions

- DoubleWorkflowInstanceNameException Gets thrown if the user trys to use an already existing workflow instance name
- EmptyConfigFolderException Gets thrown if the user hands over an empty config-folder

getTemplateNames():String[] Makes database request and returns the names of all templates in the system

getTemplateFromName(templateName:String):Template Forwards the request for the named template to the database and returns result

Parameters

- templateName: The identifier of the desired template

getNamesOfWorkflowsAndConfigFiles():String[][] Forwards the request to the database and returns the names of all workflow instances in combination with the associated config-files

getKeyValuePairsFromConfigFile(workflowInstanceName:String,configFileName:String):String[][] Requests the desired config-file from the current version of the named workflow instance from the database. Afterwards changes the format from a file into an array of key-value-pairs and returns the result

Parameters

- workflowInstanceName: The identifier of the workflow instance
- configFileName: The name of the desired config file

createNewVersionOfWorkflowInstance(workflowInstanceName:String,changedFiles:ReducedConfigF versionNote:String):void Causes the creation of a new version of the workflow instance in the database. The predecessor of the new version is the active version of the instance.

Parameters

- workflowInstanceName: The identifier of the workflow instance of which a new version is created
- changedFiles: The config-files that were changed in comparison to the predecessor version in key-value-pair representation
- versionNote Note about the new version given by the user

getVersionsFromWorkflowInstance(workflowInstanceName:string):FrontendVersion[]

Requests information about all the versions of the given workflow instance from the database. Afterwards calculates the difference to the predecessor for every version and returns that information in combination with the version numbers and version notes

Parameters

- workflowInstanceName: The identifier of the given workflow instance

setActiveVersionThroughNumber(workflowInstanceName:String, versionNumber:String):void

Changes the active version of a workflow instance in the database

Parameters

- workflowInstanceName: The name of the workflow instance of which the active version shall be changed
- versionNumber: The number of the new active version

Exceptions

- WorkflowRunningException Gets thrown if the workflow instance is running. In that case the active version can not be changed

Class: Template This class represents a workflow template. It contains the identifying name of the template as well as a dag-definition-file.

Constructor

Template(name:String, dagDefinitionFile:File):Template

Parameters

- name: The name of the new template
- dagDefinitionFile: The file which defines the behavior of workflows instantiated of this template

Exceptions

- DoubleTemplateNameException Gets thrown if the database already contains a template with the given name
- InvalidDagFileException Gets thrown if the handed dag-definition-file is invalid

Class: WorkflowInstance This class represents a workflow template. It contains the identifying name of the template as well as a dag-definition-file.

Constructor

WorkflowInstance(name:String, dagDefinitionFile:File, configFolder:File[]):WorkflowInstance

Parameters

- name: The identifying name of the workflow instance
- dagDefinitionFile: The file which defines the behavior of the workflow instance
- configFolder: An array of files that is needed for executing the workflow

Exceptions

- DoubleWorkflowInstanceNameException Gets thrown if the workflow instance name is already given away to another workflow instance
- EmptyConfigFolderException Gets thrown if the handed configFolder-array is empty

Class: ParameterChange This class represents the change of a key-value pair. It contains the old aswell as the new version of both the key aswell as the value. Furthermore it holds the name of the file the change was made in.

Constructor

ParameterChange(oldKey:String, newKey:String, oldValue:String, newValue:String, configFileName:String):ParameterChange

Class: ReducedConfigFile This class holds all information to represent a config-file in the frontend.

Constructor

ReducedConfigFile(fileName:String, keyValuePairs:(String,String)[]):ReducedConfigFile

Parameters

- fileName: The name of the config-file through which the config-file can be identified among other config files of the workflow instance
- keyValuePairs: The contents of the file in key-value-pair representation

Class: ConfigFile This is a subclass of ReducedConfigFile and additionally holds the config-file itself aswell as the name of the associated workflow instance.

Constructor

ConfigFile(workflowInstanceName:String, configFileName:String, file:File):ConfigFile It isn't necessary to hand the key-value-pairs because they can be derived from the file itself.

Parameters

- workflowInstanceName: The name of the associated workflow instance
- configFileName: The name of the config-file through which the config-file can be identified among other config files of the workflow instance
- file: The actual config-file

Methods

applyChanges(changes:ReducedConfigFile):void The method compares the key-value-pairs given through the ReducedConfigFile to the pairs of the object itself. Changes detected are applied to the actual config file that is located in the database.

Parameters

- changes: Contains the key-value-pairs of the file after the changes made by the user in the frontend

Class: Version This class represents a version of a workflow instance. It's main task is to switch between the two different representations of versions one in the frontend, one in the database.

Constructor

Version(versionNumber:VersionNumber, note:String):Version

Parameters

- versionNumber: Number that identifies the new Version
- note: Note from user that can be used for documenting the version

Class: FrontendVersion This class inherits from Version and is specialized to satisfy the need for information of the client application

Constructor

FrontendVersion(versionNumber:VersionNumber, note:String, changes:ParameterChange[]):Fronten

Parameters

- versionNumber: Number that identifies the new Version
- note: Note from user that can be used for documenting the version
- changes: Contains the difference to the predecessor version in form of the changed key-value-pairs

Class: DatabaseVersion This class inherits from Version and is specialized to fit the way versions are managed in the database

Constructor

DatabaseVersion(versionNumber:VersionNumber, note:String, changedConfigFiles:File[]):DatabaseV

Parameters

- versionNumber: Number that identifies the new Version
- note: Note from user that can be used for documenting the version
- changedConfigFiles: Contains all the config-files that were changed in this version

Class: VersionNumber This class represents string expressions that are valid version numbers

Methods

static fromString(number:String):VersionNumber This static method converts a string of correct syntax into a VersionNumber by calling the private constructor.

Parameters

- number: String that has to fit the regular expression of a version number

getSuccessor(workflowInstance:String):VersionNumber This method calculates the smallest free successor of the VersionNumber for the given workflow instance

Parameters

- workflowInstance: Identifier of the workflow instance to which the new number corresponds to

5.2.2 API Package

Package: api This package is the interface between the client's application and the server application. All requests are sent to this API(package) in JSON format.

Class: FrontendAPI This class is the main interface of the whole application.The decision to design FrontendAPI as **Singleton** pattern means that the API should be a globally accessible instance that runs on the server on a certain port. The way this happens happens is an implementation detail with which the framework **Flask** helps, but the base idea must be respected in the software design decision. The client application can get the api status code to obtain information regarding the execution of api calls (e.g. status code changes when an exception has been thrown)

static get_FrontendAPI(): FrontendAPI returns the FrontendAPI in singleton design fashion, meaning there is only one instance of FrontendAPI in circulation at all times.

get_status_code(): int return status code

set_status_code(status_code: int) sets the status code, only performed by the **ExceptionHandler**

- **status_code** the status code showing the state of the latest API call

get_server_details(): String gets all server details (container limit, cpu resources, gpu resources, servername, ip address, executing server (yes/no)) and returns them in a json format (json.dumps is interpreted as String in native python)

set_server_details(json_details: String) sets all server details (container limit, cpu resources, gpu resources, servername, ip address, executing server (yes/no)) in a json format (json objects are interpreted as String in native python). Only one bulk update as a whole due to long delay with server communication from client's perspective

- **server_details** all server details in json format

get_all_users_and_details(): String gets all users and their details (user names, privileges, statuses) in a json format (json objects are interpreted as String in native python). Only one bulk update as a whole due to long delay with server communication from client's perspective

set_all_users_and_details(json_details: String) sets all user details (user name, privilege, statuse) for a specific user in a json format (json objects are interpreted as String in native python). Only one bulk update as a whole due to long delay with server communication from client's perspective. If user does not already exist, one will be created.

delete_user(json_details: String) deletes a user by username

- **json_details** json object which only contains the username key and value

get_wf_instance_version_number (json_details: String): String returns a json object containing workflow instance's version number key and value

- **json_details** json object which only contains the workflow instance name and version number key and value

replace_wf_instance_version_number (json_details: String) replaces the (active) workflow instance's version number

- **json_details** json object which only contains the workflow instance name key and value and version number key and value

set_config_file(json_details) changes multiple config files based on input/ key value pair changes in client's applications during workflow instance configuration

- **json_details** json object which contains the changed config files and respectively their changed values, contains workflow instance key and value

get_config_from_wf_instance(json_details): String gets all config file names related to specific workflow instance (version)

- **json_details** json object which contains the wanted workflow instance name and its respective version

get_all_wf_instances_and_config_files (json_details: String) gets all config file **names** (method is used when no config file has been referenced to workflow instance)

verify_lgin(json_details:String) verifies username with associated password

- **json_details** contains username and password

register(json_details:String) registers a new user

- **json_details** contains username, password and repeated password

Class : JSONToPython This class converts all json data into the wanted object by extracting certain keys and values and instantiating a new (temporary) object which executes the wanted methods (e.g. extract server and then execute set new container limit). Instantiated objects will be deleted by the garbage collection after they are finished writing back / getting data from the database.

static extract_user(json_details:String): User extracts json details and builds a new User based off of these json details

- **json_details** contains username, privilege and status

static extract_server(json_details:String): Server extracts json details and builds a new Server based off of these json details

- **json_details** contains servername, container limit, cpu and gpu resources, executing server (yes/no) and ip address

static extract_template(json_details:String): Template extracts json details and builds a new Template based off of these json details

- **json_details** contains dag definition file name and template name

static extract_configs(json_details:String): ConfigFile[] extracts json details and builds a new ConfigFile array based off of these json details

- **json_details** contains config files which in themselves contain config file name, workflow instance name, and key value pairs

Class : PythonToJSON This class converts all python objects into json data by extracting certain keys and values and dumping them into a json object.

static encode_user(user: User): String extracts all user attributes and dumps them into json object

- **user** user whose attributes are to be encoded

static encode_server(server: Server): String extracts all server attributes and dumps them into json object

- **user** server whose attributes are to be encoded

static encode_template(template: Template): String extracts template attributes and dumps them into json object

- **user** template whose attributes are to be encoded

static encode_wf_instance(wf_instance: WorkflowInstance): String extracts workflow instance attributes and dumps them into json object

- **user** workflow instance whose attributes are to be encoded

static encode_config(config: ConfigFile): String extracts config file attributes and dumps them into json object

- **user** config file whose attributes are to be encoded

Class: Exception Handler This class handles all MatFlowExceptions and those who inherit from MatFlowException. It is responsible for changing the API's status code.

handle_exception(exception: MatFlowException) sets the API's status code to the specific exception's status code by calling the private method set_status_code.

- **exception** specific MatFlowException that was thrown during request

success() sets status code to success code

5.2.3 Exception package

5.3 Database

5.3.1 WorkflowData

Diese Klasse ist die Schnittstelle für alle Klassen die auf den Datenbankeinträgen für Workflows arbeiten wollen. Sie erstellt für jeden Zugriff den passenden Datenbankbefehl und konvertiert die Ergebnisse in das gewünschte Format des Aufrufers. Sie wirft bei fehlerhaften Werten eine **MatFlowException**.

createWorkflowInstanceFromTemplate(tName: String, wfName: String, confFold: Folder):void Create a new instance of a workflow by using the dag-file of a Template, with the Version set to 1. Search confDirect for every .conf-file and add them into the Database. Set this Version as active Version.

- **tName** name of Template of which the dag should be taken from
- **wfName** name of the new Workflow
- **confFold** the folder where all data of the Workflow is saved into

getNamesOfWorkflowsAndConfigFiles():String[][]] Return all Workflow names and the names of their corresponding config files. The returning value is a list of lists of Strings where an inner list has the form [<Workflow Name>, <config File1 Name>, <config File2 Name>, ...]

createNewVersionOfWorkflowInstance(wfName:String, newVersion:DatabaseVersion, oldVersionNr:String):void Create a new Version of an existing Workflow with changed config Files. Search which config Files changed and set those in the new Version.

- **wfName** name of the Workflow
- **newVerson** new Version identifier
- **oldVersionNr** identifier of Version the new one is based on

getDatabaseVersionsOfWorkflowInstance(wfName:String):DatabaseVersion[] Return all Versions of a Workflow Instance as DatabaseVersion Objects in a list. Asks the Database for the data and constructs every Object.

- **wfName** name of the Workflow

setActiveVersionThroughNumber(wfName:String, version:String):void Set the previous Version as inactive and a new Version as active

- **wfName** name of the Workflow
- **version** version to be set active

getVersionNumbersOfWorkflowInstance(wfName:String):String[] Return a sortet list of Strings of all Versions of a Workflow.

- **wfName** name of Workflow which versions should be listed

5.3.2 TemplateData

Diese Klasse ist die Schnittstelle für alle Klassen die auf den Datenbankeinträgen für Workflow_Template arbeiten wollen. Sie erstellt für jeden Zugriff den passenden Datenbankbefehl und konvertiert die Ergebnisse in das gewünschte Format des Aufrufers. Sie wirft bei fehlerhaften Werten eine **MatFlowException**.

createTemplate(template:Template):void Read Values from template and convert them into a sql query. Throw error if name already exists.

- **template** Template object to convert

getTemplateNames():String[] Return a list of all existing Template names. List is empty when no Template exists.

getTemplateByNames(name:String):Template Return Template that is asked for. Throw error if Template doesn't exist.

- **name** name/identifier of the wanted Template

5.3.3 DatabaseTable

DatabaseTable ist die einzige Klasse die direkt mit der Datenbank kommuniziert. Ihre Funktion ist hauptsächlich MySQL Befehle entgegennimmt, diese an die Datenbank weiterzugeben und die dadurch erhaltenen Antworten zurückgibt. Im Fall eines Fehlers, wie einem leeren Ergebnis oder eines ungültigen Befehls wird eine **MatFlowException** geworfen.

set(create: String):String Create new entry on Database. Try to execute create on Database and return value.

- **create** a sql query to create an entry

delete(del: String):String Delete an existing entry. Try to execute del on Database and return value.

- **del** a sql query to delete an entry

modify(change: String):String Modify an existing entry. Try to execute change on Database and return value.

- **change** a sql query to change values

6 Abläufe

7 Datenbank

7.1 Datenbankaufbau

Die Datenbank beinhaltet folgende Tabellen:

Workflow_Template		
name	String	$key; notNull$
dag	.py -File	$notNull$

Workflow		
name	String	$key; notNull$
files	String	$notNull$
dag	.py -File	$notNull$

Version		
wfName	String	$key; notNull$; name from Workflow
version	String	$key; notNull$
note	String	

ActiveVersion		
wfName	String	$key; notNull$; name from Workflow
version	String	$notNull$; from Version

VersionFile		
wfName	String	$key; notNull$; name from Workflow
version	String	$key; notNull$; from Version
confKey	String	$key; notNull$; from ConfFiles

ConfFile		
confKey	String	$key; notNull$
file	.conf -File	$notNull$

Notiz Die dag Datei hätte als Schlüssel zwischen Workflow_Template und Workflow fungieren können, ist aber redundant für alle Operationen auf einem Workflow. Deshalb wurde entschieden die Datei bei einer Workflowerstellung direkt zu kopieren.