REST-for-Physics Framework

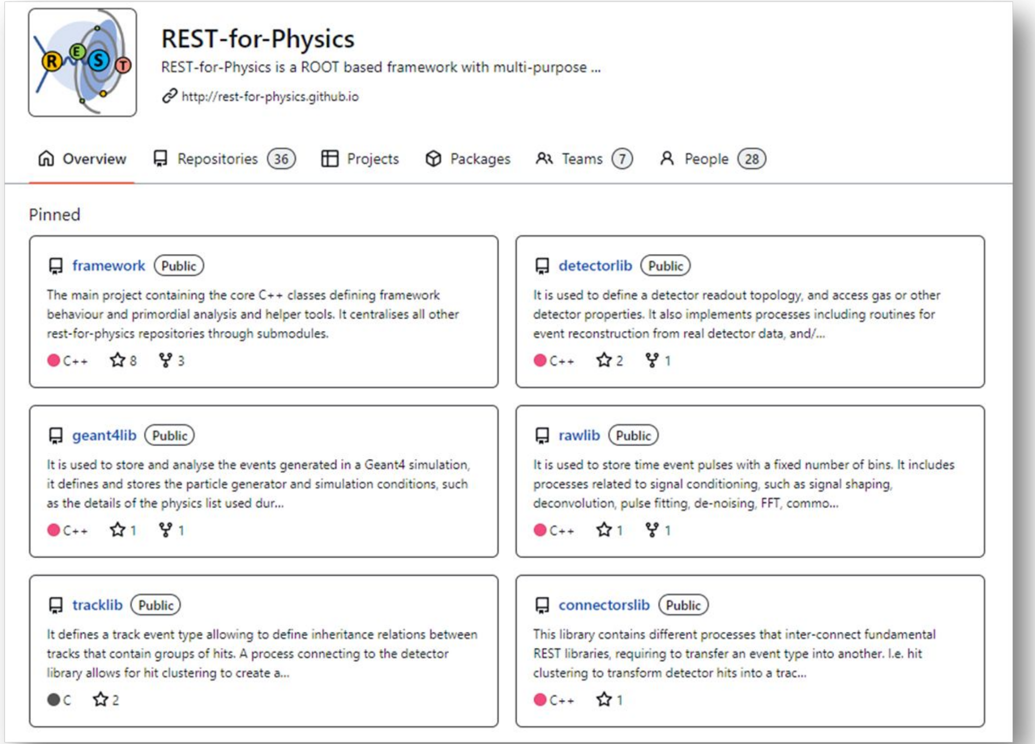
4.1 Contributing to REST-for-Physics

Edit Theme and change
Date - Author name - e-mail

Centro de Astropartículas y Física de Altas Energías
Universidad Zaragoza

REST-for-Physics - https://github.com/rest-for-physics

- Main project
  - Framework
- Libraries for detector/simulated data
  - rawlib / geant4lib
  - detectorlib
  - tracklib
- Library to transform between event types
  - connectorslib



REST-for-Physics

REST-for-Physics is a ROOT based framework with multi-purpose ...

http://rest-for-physics.github.io

Overview | Repositories 36 | Projects | Packages | Teams 7 | People 28

Pinned

framework Public
The main project containing the core C++ classes defining framework behaviour and primordial analysis and helper tools. It centralises all other rest-for-physics repositories through submodules.
● C++ ☆ 8 forks 3

detectorlib Public
It is used to define a detector readout topology, and access gas or other detector properties. It also implements processes including routines for event reconstruction from real detector data, and/...
● C++ ☆ 2 forks 1

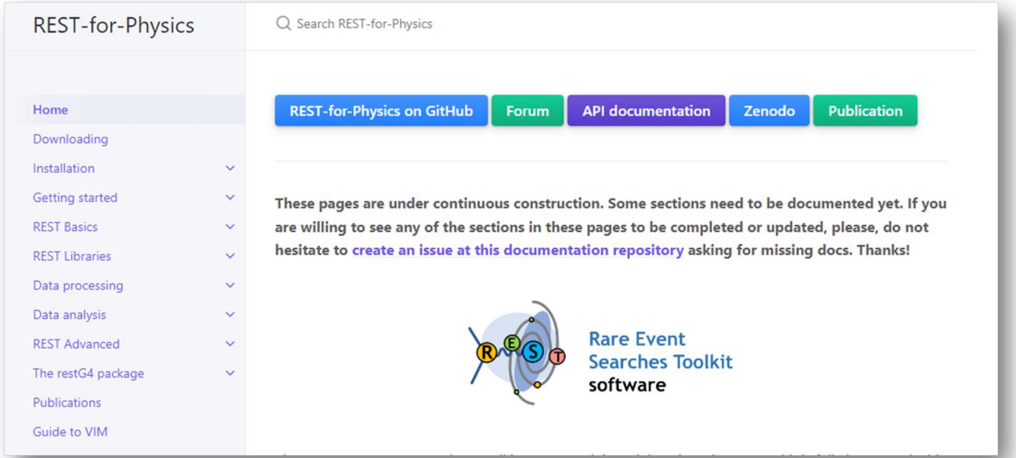
geant4lib Public
It is used to store and analyse the events generated in a Geant4 simulation, it defines and stores the particle generator and simulation conditions, such as the details of the physics list used dur...
● C++ ☆ 1 forks 1

rawlib Public
It is used to store time event pulses with a fixed number of bins. It includes processes related to signal conditioning, such as signal shaping, deconvolution, pulse fitting, de-noising, FFT, commo...
● C++ ☆ 1 forks 1

tracklib Public
It defines a track event type allowing to define inheritance relations between tracks that contain groups of hits. A process connecting to the detector library allows for hit clustering to create a...
● C ☆ 2

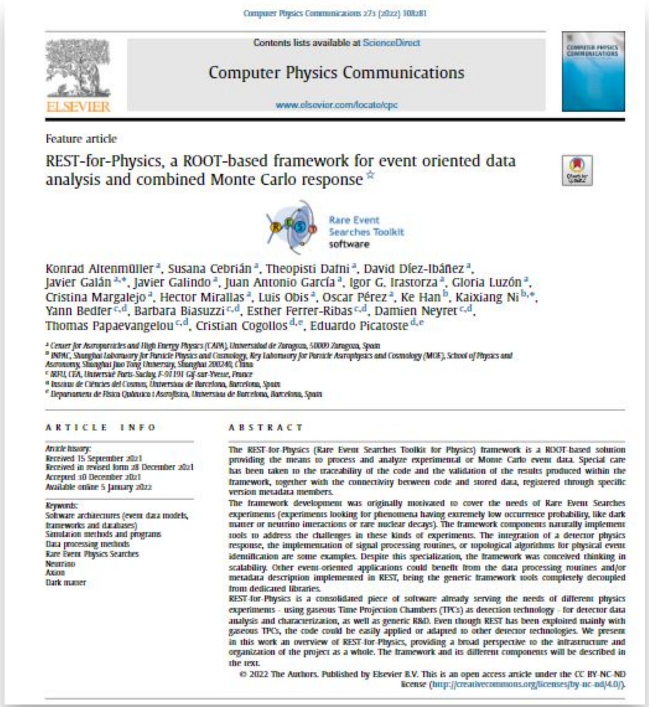
connectorslib Public
This library contains different processes that inter-connect fundamental REST libraries, requiring to transfer an event type into another. I.e. hit clustering to transform detector hits into a trac...
● C++ ☆ 1

Where to find more info: API, user guide, publications



These pages are under continuous construction. Some sections need to be documented yet. If you are willing to see any of the sections in these pages to be completed or updated, please, do not hesitate to create an issue at this documentation repository asking for missing docs. Thanks!

https://rest-for-physics.github.io/



Computer Physics Communications 273 (2022) 108281

Contents lists available at ScienceDirect

Computer Physics Communications

www.elsevier.com/locate/cpc

Feature article

REST-for-Physics, a ROOT-based framework for event oriented data analysis and combined Monte Carlo response ☆

Konrad Altenmüller [a], Susana Cebrián [a], Theopisti Dafni [a], David Díez-Ibáñez [a], Javier Galán [a,*], Javier Galindo [a], Juan Antonio García [a], Igor G. Irastorza [a], Gloria Luzón [a], Cristina Margalejo [a], Hector Mirallas [a], Luis Obis [a], Oscar Pérez [a], Ke Han [b], Kaixiang Ni [b,*], Yann Bedfer [c,d], Barbara Biasuzzi [c,d], Esther Ferrer-Ribas [c,d], Damien Neyret [c,d], Thomas Papaevangelou [c,d], Cristian Cogollos [d,e], Eduardo Picatoste [d,e]

[a] Center for Astroparticles and High Energy Physics (CAPA), Universidad de Zaragoza, 50009 Zaragoza, Spain
[b] INPAC, Shanghai Laboratory for Particle Physics and Cosmology, Key Laboratory for Particle Astrophysics and Cosmology (MOE), School of Physics and Astronomy, Shanghai Jiao Tong University, Shanghai 200240, China
[c] IRFU, CEA, Université Paris-Saclay, F-91191 Gif-sur-Yvette, France
[d] Institut de Ciències del Cosmos, Universitat de Barcelona, Barcelona, Spain
[e] Departament de Física Quàntica i Astrofísica, Universitat de Barcelona, Barcelona, Spain

ARTICLE INFO

Article history:
Received 15 September 2021
Received in revised form 28 December 2021
Accepted 10 December 2021
Available online 5 January 2022

Keywords:
Software architectures (event data models, frameworks and databases)
Simulation methods and programs
Data processing methods
Rare Event Physics Searches
Neutrino
Axion
Dark matter

ABSTRACT

The REST-for-Physics (Rare Event Searches Toolkit for Physics) framework is a ROOT-based solution providing the means to process and analyze experimental or Monte Carlo event data. Special care has been taken to the traceability of the code and the validation of the results produced within the framework, together with the connectivity between code and stored data, registered through specific version metadata members.

The framework development was originally motivated to cover the needs of Rare Event Searches experiments (experiments looking for phenomena having extremely low occurrence probability, like dark matter or neutrino interactions or rare nuclear decays). The framework components naturally implement tools to address the challenges in these kinds of experiments. The integration of a detector physics response, the implementation of signal processing routines, or topological algorithms for physical event identification are some examples. Despite this specialization, the framework was conceived thinking in scalability. Other event-oriented applications could benefit from the data processing routines and/or metadata description implemented in REST, being the generic framework tools completely decoupled from dedicated libraries.

REST-for-Physics is a consolidated piece of software already serving the needs of different physics experiments - using gaseous Time Projection Chambers (TPCs) as detection technology - for detector data analysis and characterization, as well as generic R&D. Even though REST has been exploited mainly with gaseous TPCs, the code could be easily applied or adapted to other detector technologies. We present in this work an overview of REST-for-Physics, providing a broad perspective to the infrastructure and organization of the project as a whole. The framework and its different components will be described in the text.

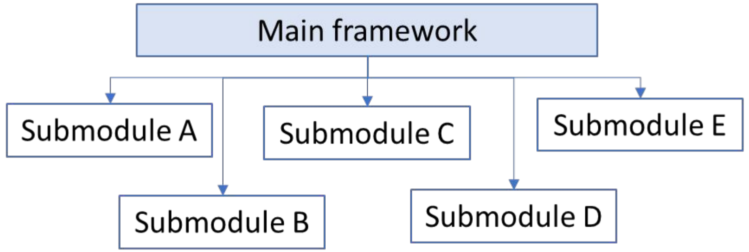
© 2022 The Authors. Published by Elsevier B.V. This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/).

REST-for-Physics publication:

https://doi.org/10.1016/j.cpc.2021.108281

REST-for-Physics - https://github.com/rest-for-physics

- Main framework
  - It defines the basic functions and describes the behavior of the main elements of REST
  - It centralizes all the REST-for-Physics components, such as packages or libraries, that are integrated as git submodules.
- The git submodules strategy
  - This scheme allows to independently monitor the development activity in each of the submodules, to isolate technical issues, and to focus on their functionality.
  - Each submodule evolves independently with its own version or tracking system.
  - A particular state of the code at each of those submodules is fixed in the main framework through a git commit hash, or a unique number. When that happens, the corresponding git commit becomes the **official submodule version of REST**



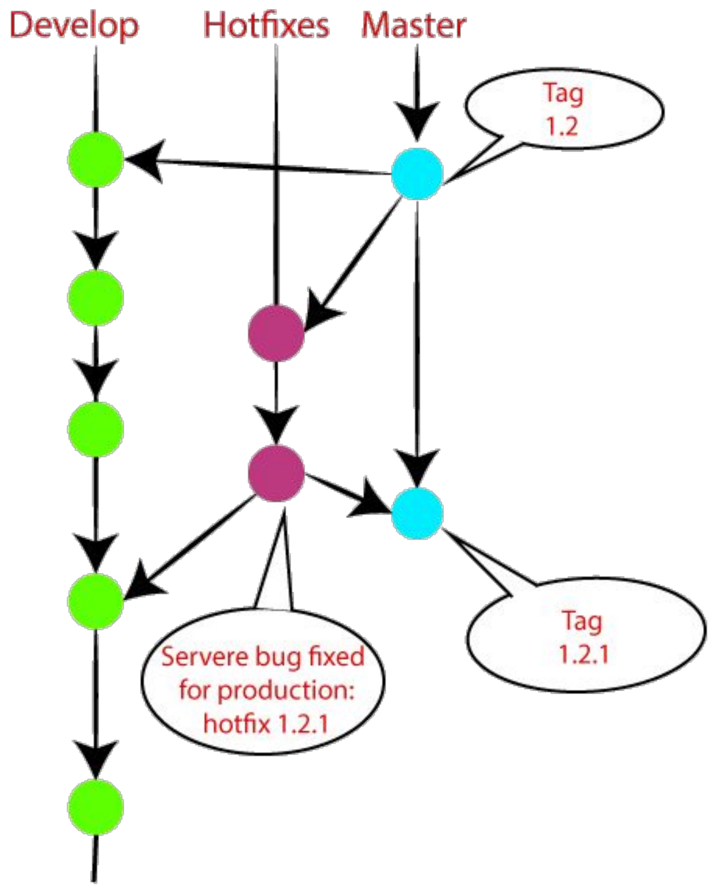
- The framework repository fully centralizes the versioning system of REST, understood as the state of the code at a given time, including the state of the official git submodules attached to it.
- Any REST metadata object written to disk using the ROOT I/O scheme will be stamped with metadata values (e.g., the REST release number, latest commit hash, release date, etc) that ensure that the data written to disk has been processed with a given version, or state of the code

Introduction to Git

Git is an open source distributed version control system originally authored by Linus Torvalds.

Git allow tracking changes in any set of files, usually used for coordinating work among programmers collaboratively developing source code during software development. Its goals include speed, data integrity, and support for distributed, non-linear workflows (thousands of parallel branches running on different systems).

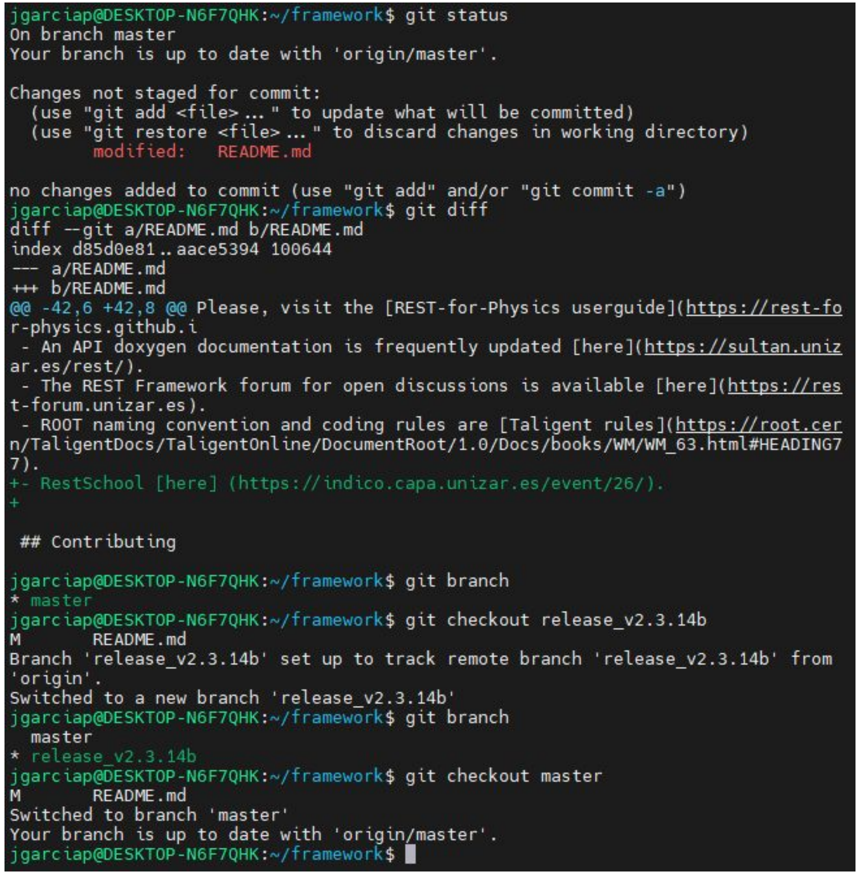
Developers make changes in their local workspace and after commit, these changes become a part of the repository. Git takes it one step further by providing them a private copy of the whole repository. Users can perform many operations with this repository such as add file, remove file, rename file, move file, commit changes, and many more.



Introduction to Git

Basic git commands:

- ***git status*** check status of the project (unstaged files).
- ***git diff*** check unstaged changed in the project.
- ***git branch*** check remote branches staged in your local workspace.
- ***git checkout*** change to a remote repository.

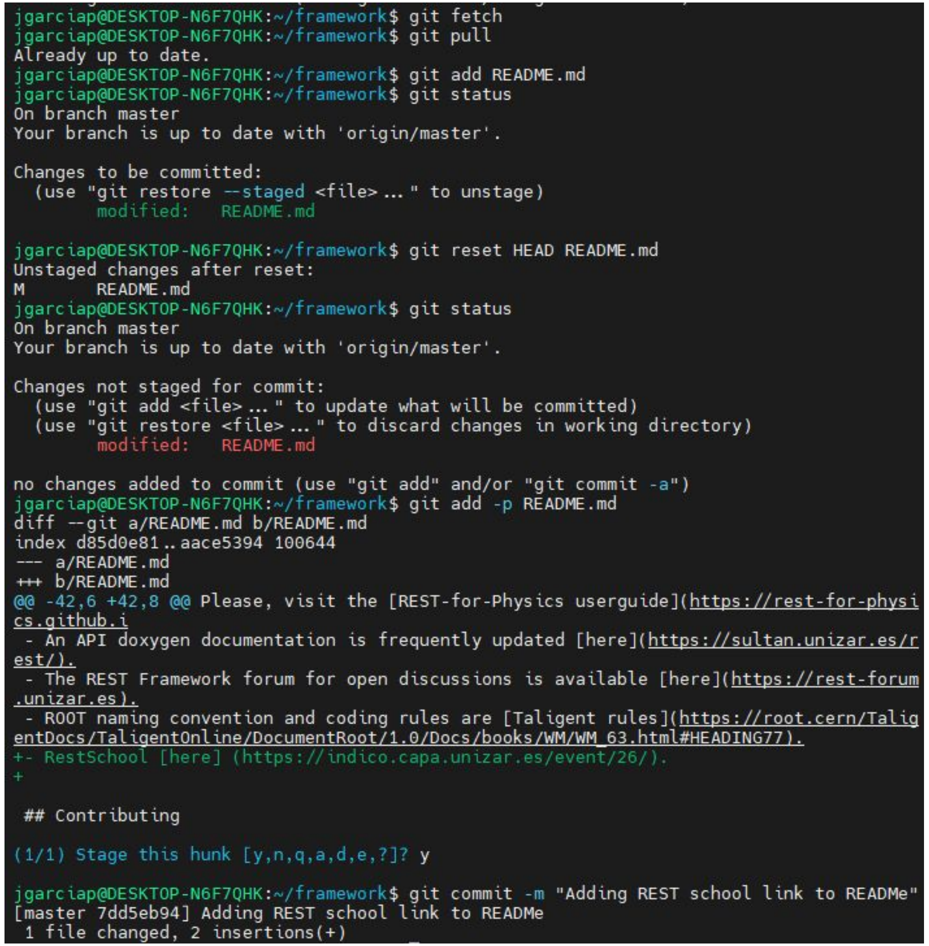


```
jgarciap@DESKTOP-N6F7QHK:~/framework$ git status
On branch master
Your branch is up to date with 'origin/master'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   README.md

no changes added to commit (use "git add" and/or "git commit -a")
jgarciap@DESKTOP-N6F7QHK:~/framework$ git diff
diff --git a/README.md b/README.md
index d85d0e81..aace5394 100644
--- a/README.md
+++ b/README.md
@@ -42,6 +42,8 @@ Please, visit the [REST-for-Physics userguide](https://rest-fo
r-physics.github.i
 - An API doxygen documentation is frequently updated [here](https://sultan.uniz
ar.es/rest/).
 - The REST Framework forum for open discussions is available [here](https://res
t-forum.unizar.es).
 - ROOT naming convention and coding rules are [Taligent rules](https://root.cer
n/TaligentDocs/TaligentOnline/DocumentRoot/1.0/Docs/books/WM/WM_63.html#HEADING7
7).
+- RestSchool [here] (https://indico.capa.unizar.es/event/26/).
+

 ## Contributing

jgarciap@DESKTOP-N6F7QHK:~/framework$ git branch
* master
jgarciap@DESKTOP-N6F7QHK:~/framework$ git checkout release_v2.3.14b
M       README.md
Branch 'release_v2.3.14b' set up to track remote branch 'release_v2.3.14b' from
'origin'.
Switched to a new branch 'release_v2.3.14b'
jgarciap@DESKTOP-N6F7QHK:~/framework$ git branch
  master
* release_v2.3.14b
jgarciap@DESKTOP-N6F7QHK:~/framework$ git checkout master
M       README.md
Switched to branch 'master'
Your branch is up to date with 'origin/master'.
jgarciap@DESKTOP-N6F7QHK:~/framework$ 
```

Introduction to Git

Basic git commands:

- ***git fetch*** locally fetch all the info from remote repository.
- ***git pull*** pull changes from remote repository in local workspace.
- ***git add filename*** stage changes on filename to local repository.
- ***git reset*** unstage changes in local repository
- ***git commit*** save changes to the local repository.

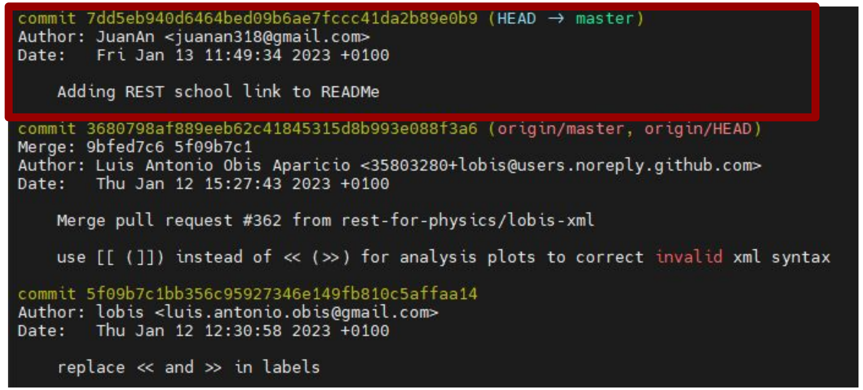


```
jgarciap@DESKTOP-N6F7QHK:~/framework$ git fetch
jgarciap@DESKTOP-N6F7QHK:~/framework$ git pull
Already up to date.
jgarciap@DESKTOP-N6F7QHK:~/framework$ git add README.md
jgarciap@DESKTOP-N6F7QHK:~/framework$ git status
On branch master
Your branch is up to date with 'origin/master'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   README.md

jgarciap@DESKTOP-N6F7QHK:~/framework$ git reset HEAD README.md
Unstaged changes after reset:
M       README.md
jgarciap@DESKTOP-N6F7QHK:~/framework$ git status
On branch master
Your branch is up to date with 'origin/master'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   README.md

no changes added to commit (use "git add" and/or "git commit -a")
jgarciap@DESKTOP-N6F7QHK:~/framework$ git add -p README.md
diff --git a/README.md b/README.md
index d85d0e81..aace5394 100644
--- a/README.md
+++ b/README.md
@@ -42,6 +42,8 @@ Please, visit the [REST-for-Physics userguide](https://rest-for-physi
cs.github.i
 - An API doxygen documentation is frequently updated [here](https://sultan.unizar.es/r
est/).
 - The REST Framework forum for open discussions is available [here](https://rest-forum
.unizar.es).
 - ROOT naming convention and coding rules are [Taligent rules](https://root.cern/Talig
entDocs/TaligentOnline/DocumentRoot/1.0/Docs/books/WM/WM_63.html#HEADING77).
+- RestSchool [here] (https://indico.capa.unizar.es/event/26/).
+

 ## Contributing

(1/1) Stage this hunk [y,n,q,a,d,e,?]? y

jgarciap@DESKTOP-N6F7QHK:~/framework$ git commit -m "Adding REST school link to README"
[master 7dd5eb94] Adding REST school link to README
 1 file changed, 2 insertions(+)
```

Introduction to Git

Basic git commands:

- ***git log*** locally fetch all the info from
- ***git push*** save changes to the remote repository.



```
commit 7dd5eb940d6464bed09b6ae7fccc41da2b89e0b9 (HEAD -> master)
Author: JuanAn <juanan318@gmail.com>
Date:   Fri Jan 13 11:49:34 2023 +0100

    Adding REST school link to READMe

commit 3680798af889eeb62c41845315d8b993e088f3a6 (origin/master, origin/HEAD)
Merge: 9bfed7c6 5f09b7c1
Author: Luis Antonio Obis Aparicio <35803280+lobis@users.noreply.github.com>
Date:   Thu Jan 12 15:27:43 2023 +0100

    Merge pull request #362 from rest-for-physics/lobis-xml

    use [[ (]]) instead of << (>>) for analysis plots to correct invalid xml syntax

commit 5f09b7c1bb356c95927346e149fb810c5affaa14
Author: lobis <luis.antonio.obis@gmail.com>
Date:   Thu Jan 12 12:30:58 2023 +0100

    replace << and >> in labels
```

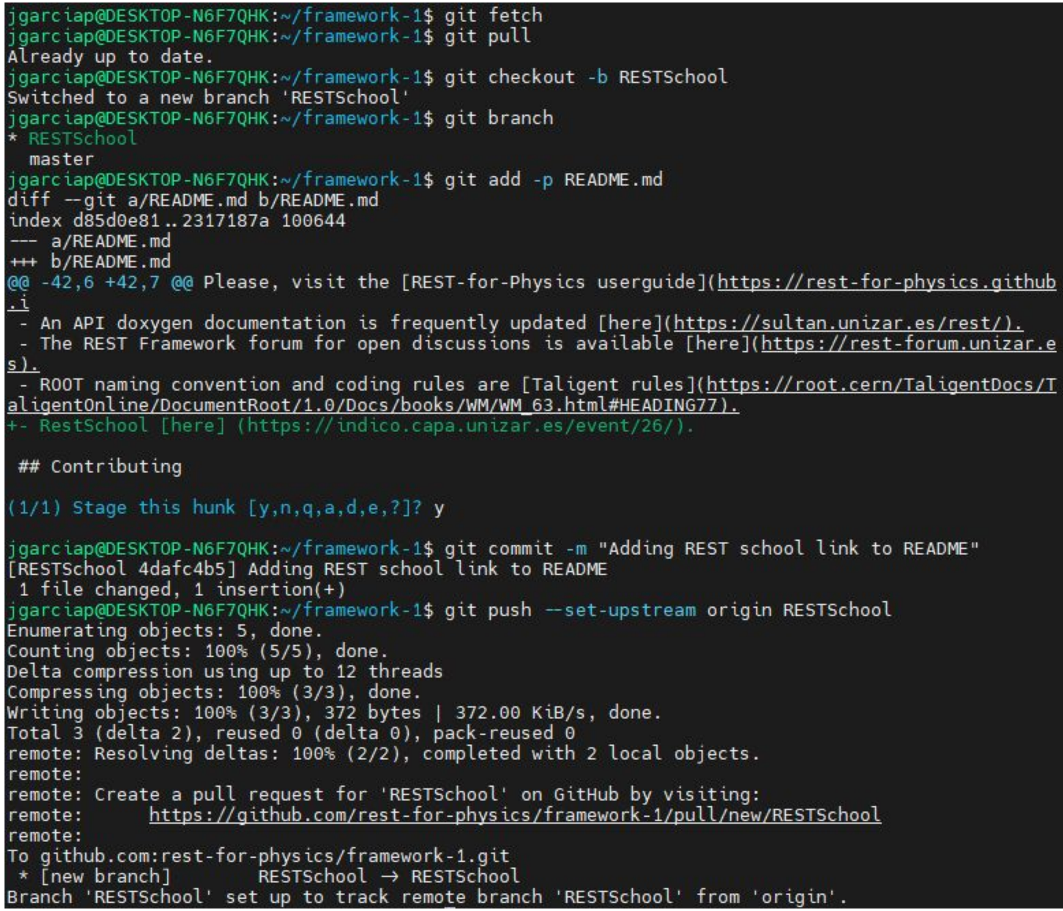
Many other git commands are available, for more info check:

https://git-scm.com/docs

https://dzone.com/articles/top-20-git-commands-with-examples

REST-for-Physics workflow:

- ***git fetch***
- ***git pull***
- ***git checkout -b branchName***
  create a new branch locally
- **git add -p file/folder**
- **git commit -m “Feature description”**
- **git push --set-upstream origin branchName**

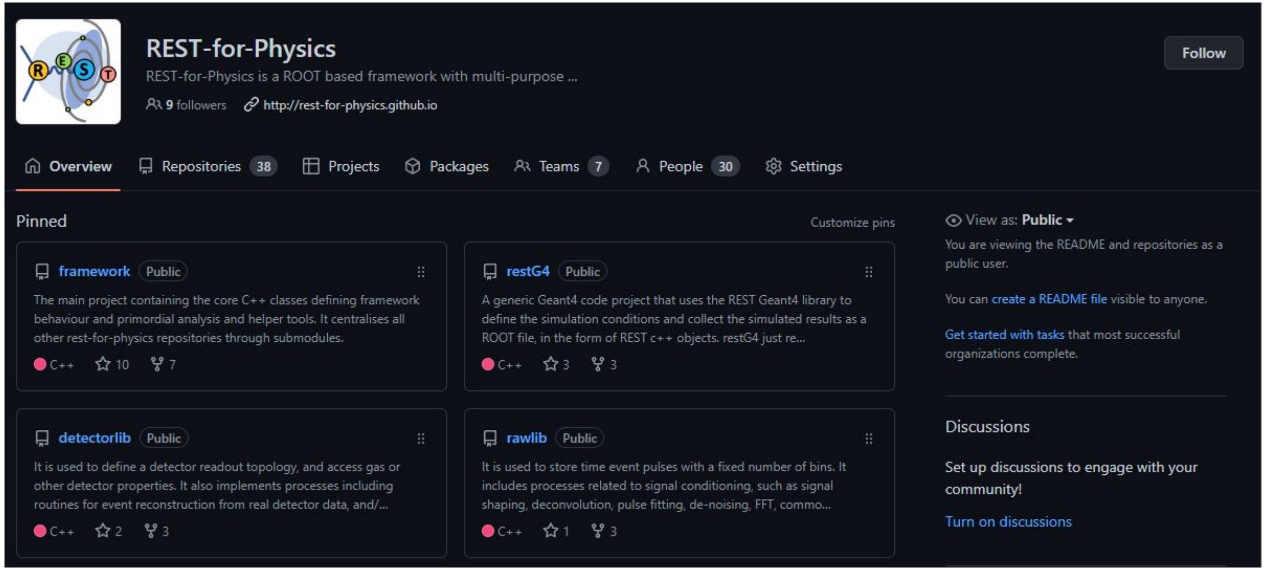


```
jgarciap@DESKTOP-N6F7QHK:~/framework-1$ git fetch
jgarciap@DESKTOP-N6F7QHK:~/framework-1$ git pull
Already up to date.
jgarciap@DESKTOP-N6F7QHK:~/framework-1$ git checkout -b RESTSchool
Switched to a new branch 'RESTSchool'
jgarciap@DESKTOP-N6F7QHK:~/framework-1$ git branch
* RESTSchool
  master
jgarciap@DESKTOP-N6F7QHK:~/framework-1$ git add -p README.md
diff --git a/README.md b/README.md
index d85d0e81..2317187a 100644
--- a/README.md
+++ b/README.md
@@ -42,6 +42,7 @@ Please, visit the [REST-for-Physics userguide](https://rest-for-physics.github
.i
 - An API doxygen documentation is frequently updated [here](https://sultan.unizar.es/rest/).
 - The REST Framework forum for open discussions is available [here](https://rest-forum.unizar.e
s).
 - ROOT naming convention and coding rules are [Taligent rules](https://root.cern/TaligentDocs/T
aligentOnline/DocumentRoot/1.0/Docs/books/WM/WM_63.html#HEADING77).
+- RestSchool [here] (https://indico.capa.unizar.es/event/26/).

 ## Contributing

(1/1) Stage this hunk [y,n,q,a,d,e,?]? y

jgarciap@DESKTOP-N6F7QHK:~/framework-1$ git commit -m "Adding REST school link to README"
[RESTSchool 4dafc4b5] Adding REST school link to README
 1 file changed, 1 insertion(+)
jgarciap@DESKTOP-N6F7QHK:~/framework-1$ git push --set-upstream origin RESTSchool
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 12 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 372 bytes | 372.00 KiB/s, done.
Total 3 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
remote:
remote: Create a pull request for 'RESTSchool' on GitHub by visiting:
remote:      https://github.com/rest-for-physics/framework-1/pull/new/RESTSchool
remote:
To github.com:rest-for-physics/framework-1.git
 * [new branch]        RESTSchool -> RESTSchool
Branch 'RESTSchool' set up to track remote branch 'RESTSchool' from 'origin'.
```

Introduction to GitHub

GitHub, is an Internet hosting service for software development and version control using Git. It provides the distributed version control of Git plus access control, bug tracking, software feature requests, task management, continuous integration, and wikis for every project.

REST-for-Physics is integrated under GitHub https://github.com/rest-for-physics/



REST-for-Physics is a ROOT based framework with multi-purpose ...

9 followers http://rest-for-physics.github.io

Pinned

framework Public
The main project containing the core C++ classes defining framework behaviour and primordial analysis and helper tools. It centralises all other rest-for-physics repositories through submodules.
C++ 10 7

restG4 Public
A generic Geant4 code project that uses the REST Geant4 library to define the simulation conditions and collect the simulated results as a ROOT file, in the form of REST c++ objects. restG4 just re...
C++ 3 3

detectorlib Public
It is used to define a detector readout topology, and access gas or other detector properties. It also implements processes including routines for event reconstruction from real detector data, and/...
C++ 2 3

rawlib Public
It is used to store time event pulses with a fixed number of bins. It includes processes related to signal conditioning, such as signal shaping, deconvolution, pulse fitting, de-noising, FFT, commo...
C++ 1 3

View as: Public

You are viewing the README and repositories as a public user.

You can create a README file visible to anyone.

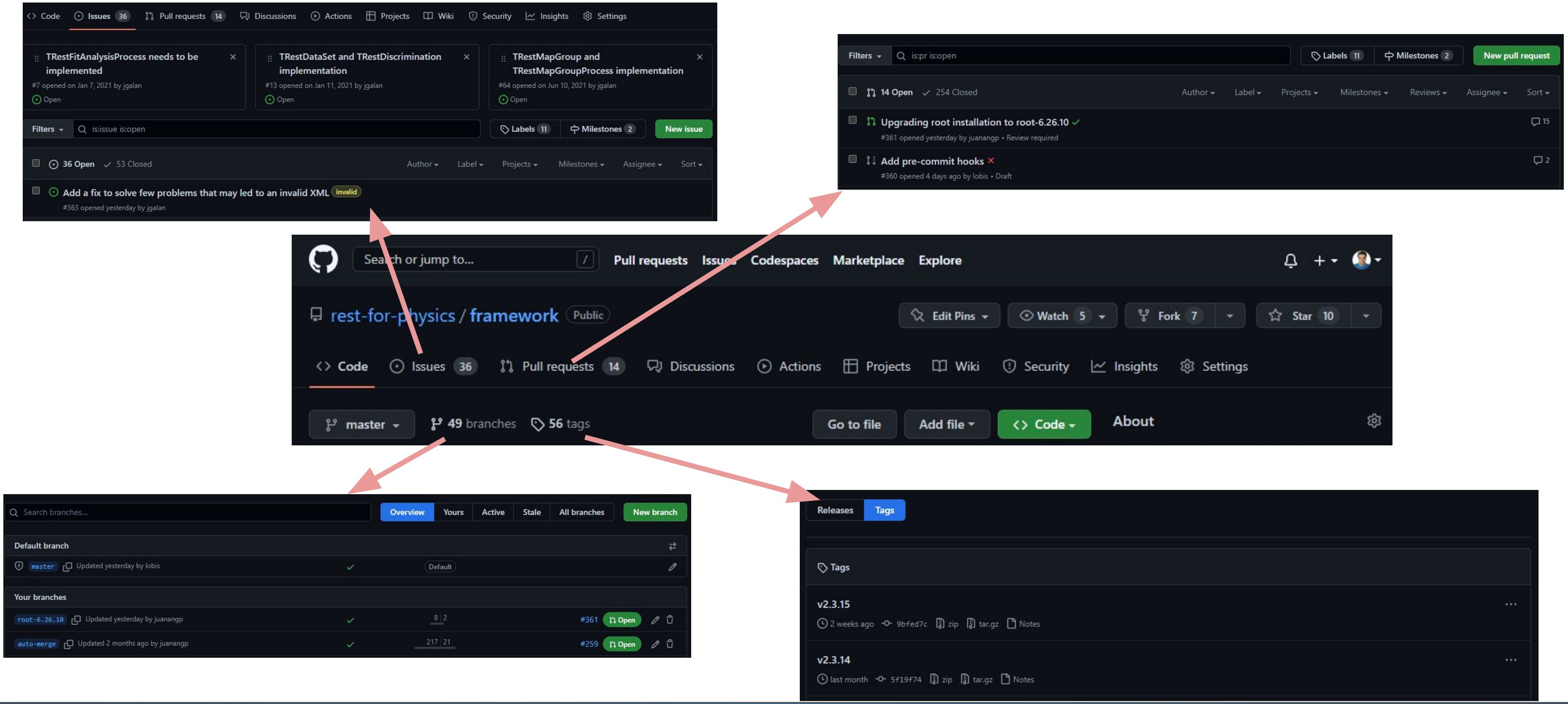
Get started with tasks that most successful organizations complete.

Discussions

Set up discussions to engage with your community!

Turn on discussions

Navigating through REST-for-Physics in GitHub



TRestFitAnalysisProcess needs to be implemented
#7 opened on Jan 7, 2021 by jgalan
Open

TRestDataSet and TRestDiscrimination implementation
#13 opened on Jan 11, 2021 by jgalan
Open

TRestMapGroup and TRestMapGroupProcess implementation
#64 opened on Jun 10, 2021 by jgalan
Open

36 Open ✓ 53 Closed

Add a fix to solve few problems that may led to an invalid XML invalid
#363 opened yesterday by jgalan

14 Open ✓ 254 Closed

Upgrading root installation to root-6.26.10 ✓ 15
#361 opened yesterday by juanangp • Review required

Add pre-commit hooks ✗ 2
#360 opened 4 days ago by lobis • Draft

Default branch
master Updated yesterday by lobis ✓ Default

Your branches
root-6.26.10 Updated yesterday by juanangp ✓ 8 | 2 #361 Open
auto-merge Updated 2 months ago by juanangp ✓ 217 | 21 #259 Open

Tags
v2.3.15
2 weeks ago 9bfed7c zip tar.gz Notes
v2.3.14
last month 5f19f74 zip tar.gz Notes

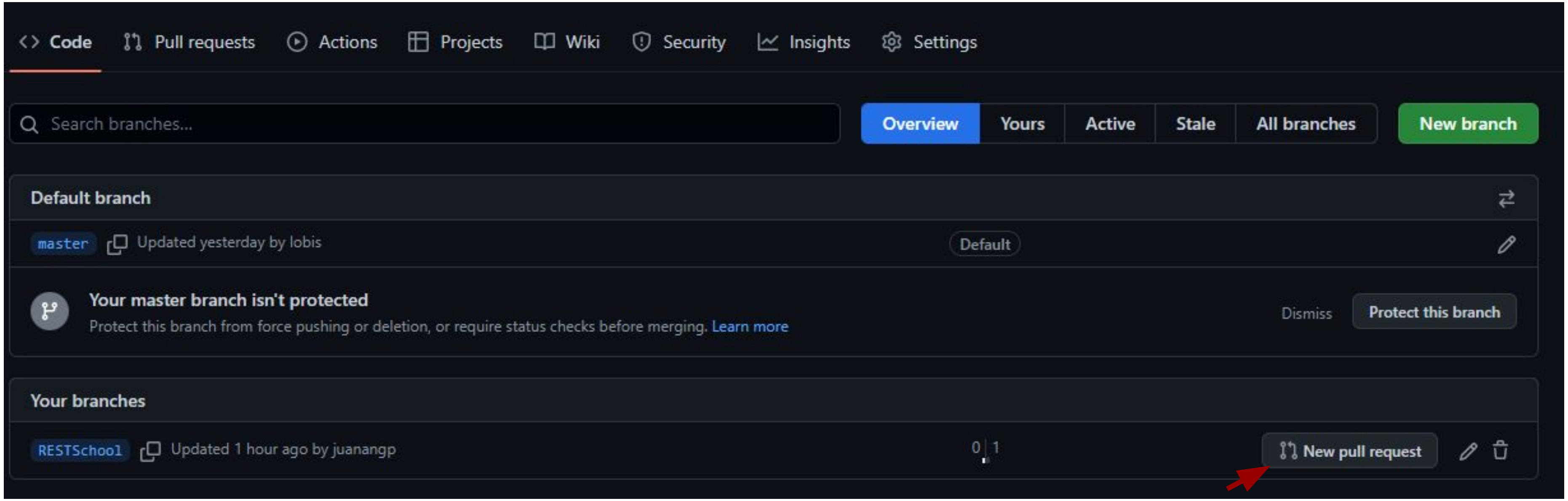
GitHub Pull request

There are some internal rules to deploy REST-for-Physics developments/features:

- Default branch is ***master***
- Master branch is protected, which means that nobody is allowed to push commits directly to master. However, you can push your local changes to a development/feature branch.
- The master branch can only be updated via pull request.
- Only developers can create a pull request to master.
- Before merging a pull request to master you need the approval of a reviewer.
- Some code validation must pass before merging a pull request.
- Branches have to be up-to-date with master before merging the pull request.

GitHub Pull request

After you have finished your development/feature you may want to update master branch with your changes. To do that just create a pull request after you committed all your changes to your development/feature branch:



Default branch

master Updated yesterday by lobis Default

Your master branch isn't protected

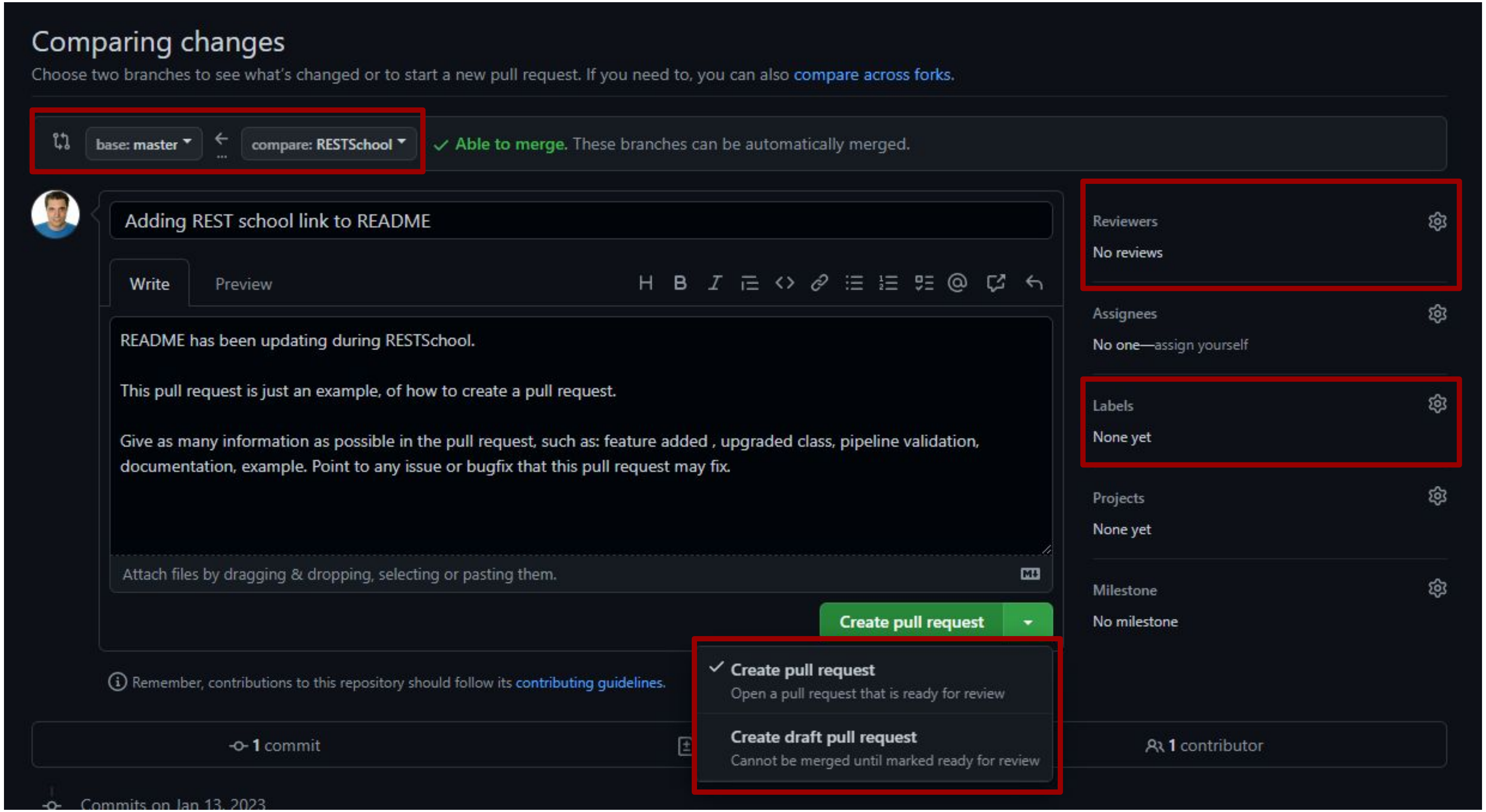
Protect this branch from force pushing or deletion, or require status checks before merging. Learn more

Dismiss Protect this branch

Your branches

RESTSchool Updated 1 hour ago by juanangp 0 | 1 New pull request

GitHub Pull request



Comparing changes

Choose two branches to see what's changed or to start a new pull request. If you need to, you can also compare across forks.

base: master ← compare: RESTSchool ✓ Able to merge. These branches can be automatically merged.

Adding REST school link to README

Write Preview

README has been updating during RESTSchool.

This pull request is just an example, of how to create a pull request.

Give as many information as possible in the pull request, such as: feature added , upgraded class, pipeline validation, documentation, example. Point to any issue or bugfix that this pull request may fix.

Attach files by dragging & dropping, selecting or pasting them.

Create pull request

✓ Create pull request
Open a pull request that is ready for review

Create draft pull request
Cannot be merged until marked ready for review

Remember, contributions to this repository should follow its contributing guidelines.

Reviewers
No reviews

Assignees
No one—assign yourself

Labels
None yet

Projects
None yet

Milestone
No milestone

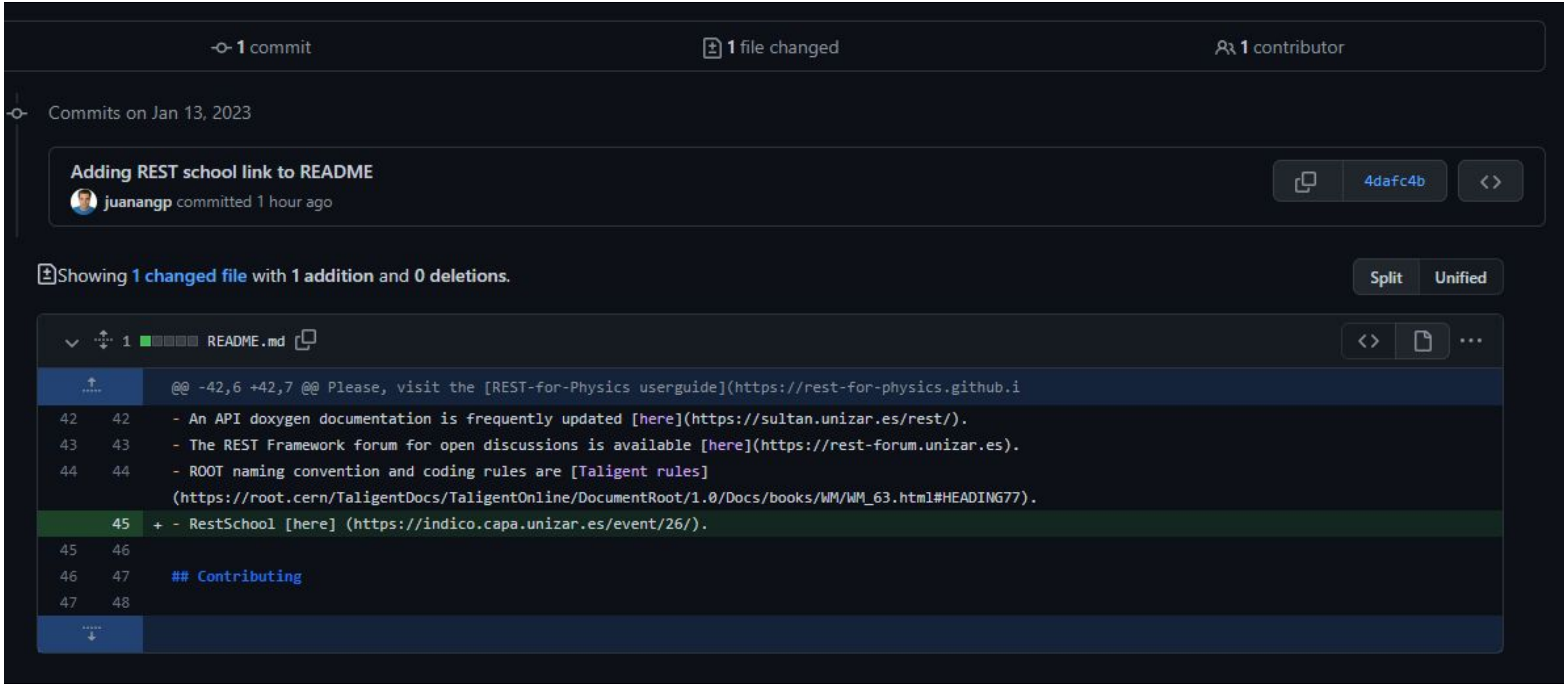
1 commit

1 contributor

Commits on Jan 13, 2023

GitHub Pull request

Changes and commits are displayed at the end of the page:


1 commit 1 file changed 1 contributor

Commits on Jan 13, 2023

Adding REST school link to README
juanangp committed 1 hour ago 4dafc4b

Showing 1 changed file with 1 addition and 0 deletions.

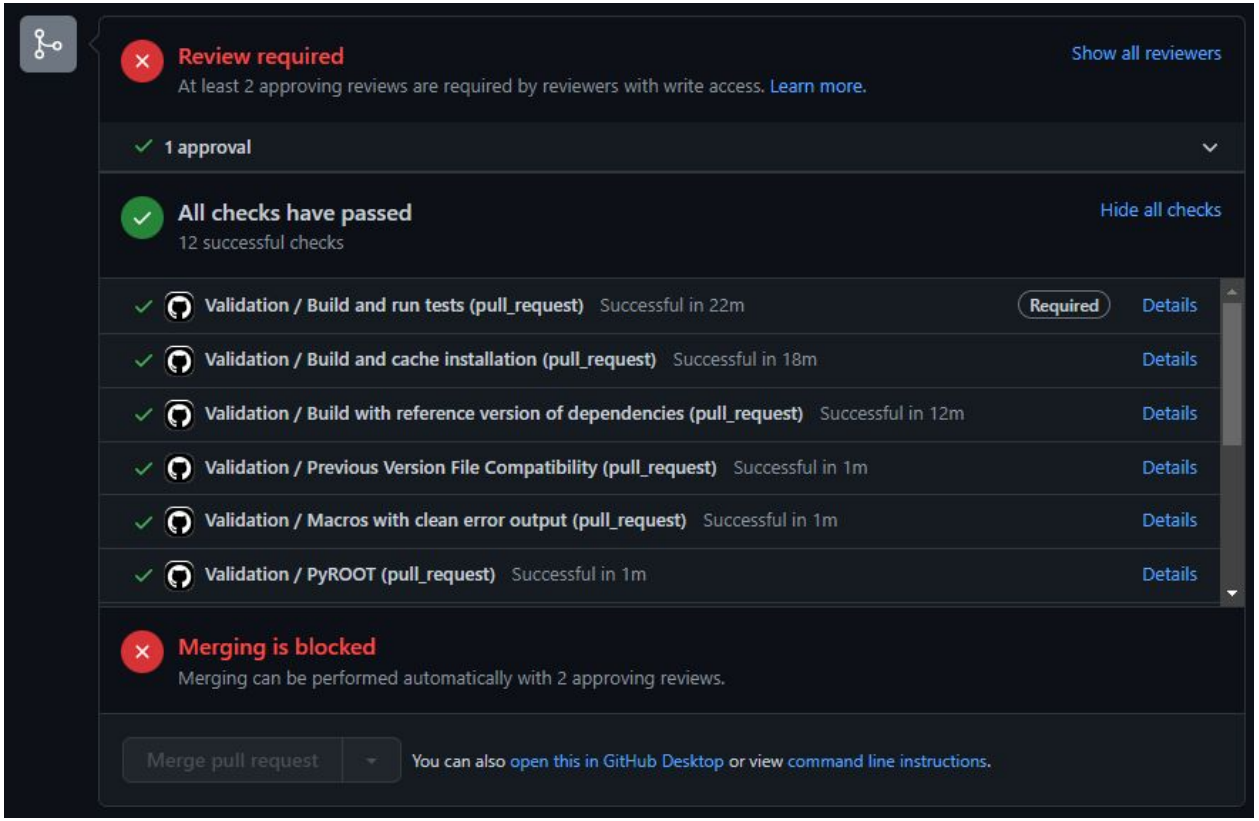
Split Unified

1 README.md

```
@@ -42,6 +42,7 @@ Please, visit the [REST-for-Physics userguide](https://rest-for-physics.github.i
- An API doxygen documentation is frequently updated [here](https://sultan.unizar.es/rest/).
- The REST Framework forum for open discussions is available [here](https://rest-forum.unizar.es).
- ROOT naming convention and coding rules are [Taligent rules]
(https://root.cern/TaligentDocs/TaligentOnline/DocumentRoot/1.0/Docs/books/WM/WM_63.html#HEADING77).
+ - RestSchool [here] (https://indico.capa.unizar.es/event/26/).

## Contributing

```

GitHub Pull request

Once the pull request is created it will trigger the validation pipeline:


Review required

At least 2 approving reviews are required by reviewers with write access. Learn more.

Show all reviewers

✓ 1 approval

All checks have passed

12 successful checks

Hide all checks

✓ Validation / Build and run tests (pull_request) Successful in 22m Required Details

✓ Validation / Build and cache installation (pull_request) Successful in 18m Details

✓ Validation / Build with reference version of dependencies (pull_request) Successful in 12m Details

✓ Validation / Previous Version File Compatibility (pull_request) Successful in 1m Details

✓ Validation / Macros with clean error output (pull_request) Successful in 1m Details

✓ Validation / PyROOT (pull_request) Successful in 1m Details

Merging is blocked

Merging can be performed automatically with 2 approving reviews.

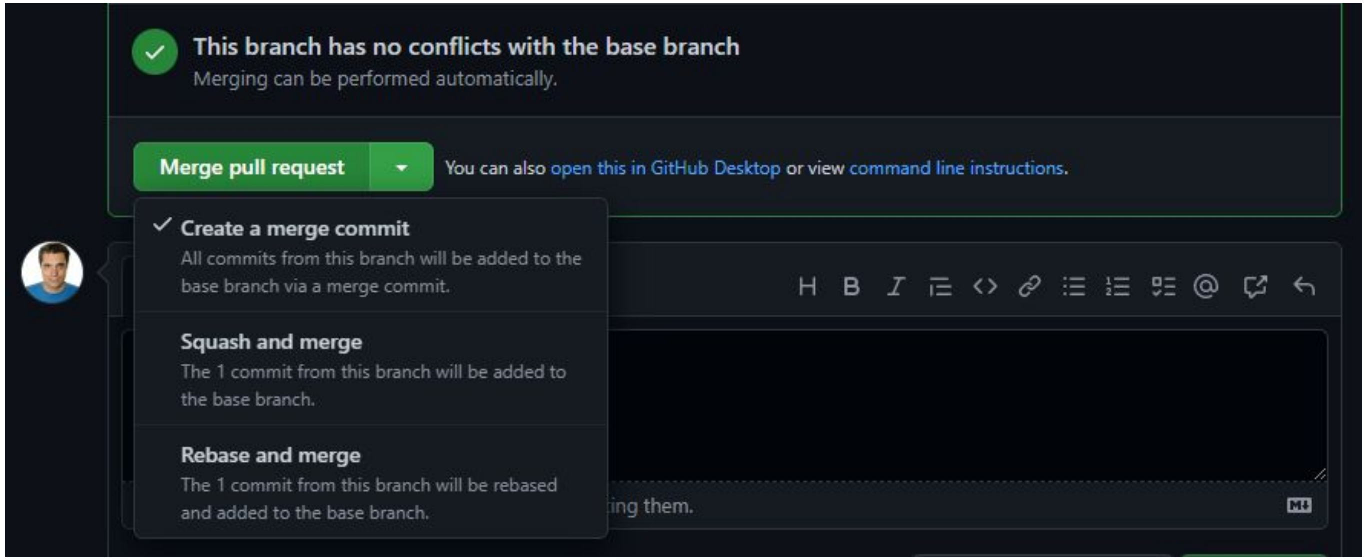
Merge pull request

You can also open this in GitHub Desktop or view command line instructions.

GitHub Pull request

You might receive comments or suggestions from the reviewers, you can keep pushing commits to your development/feature branch and it will be updated accordingly. However, note that it will triggers the pipeline validation again.

Once you have the approval from the reviewers and the pipeline succeed you can go ahead and merge your pull request. It will trigger the pipeline validation in master.


This branch has no conflicts with the base branch
Merging can be performed automatically.

You can also open this in GitHub Desktop or view command line instructions.

✓ Create a merge commit
All commits from this branch will be added to the base branch via a merge commit.

Squash and merge
The 1 commit from this branch will be added to the base branch.

Rebase and merge
The 1 commit from this branch will be rebased and added to the base branch.

GitHub Pull request

Typically developers works on a small part of the code aka submodule (framework, libraries or packages). However, some considerations have to be taken in case a development/feature is distributed across multiple submodules:

- Use the same branch name in the different submodules (e.g. framework and rawlib)
- Create different pull request per submodule
- Make sure that all the different submodule pull request are ready to merge
- Merge all the submodule pull requests at time

Validation pipeline

Continuous integration (CI) is a software practice that requires frequently committing code to a shared repository. Committing code more often detects errors sooner and reduces the amount of code a developer needs to debug when finding the source of an error. Frequent code updates also make it easier to merge changes from different members of a software development team.

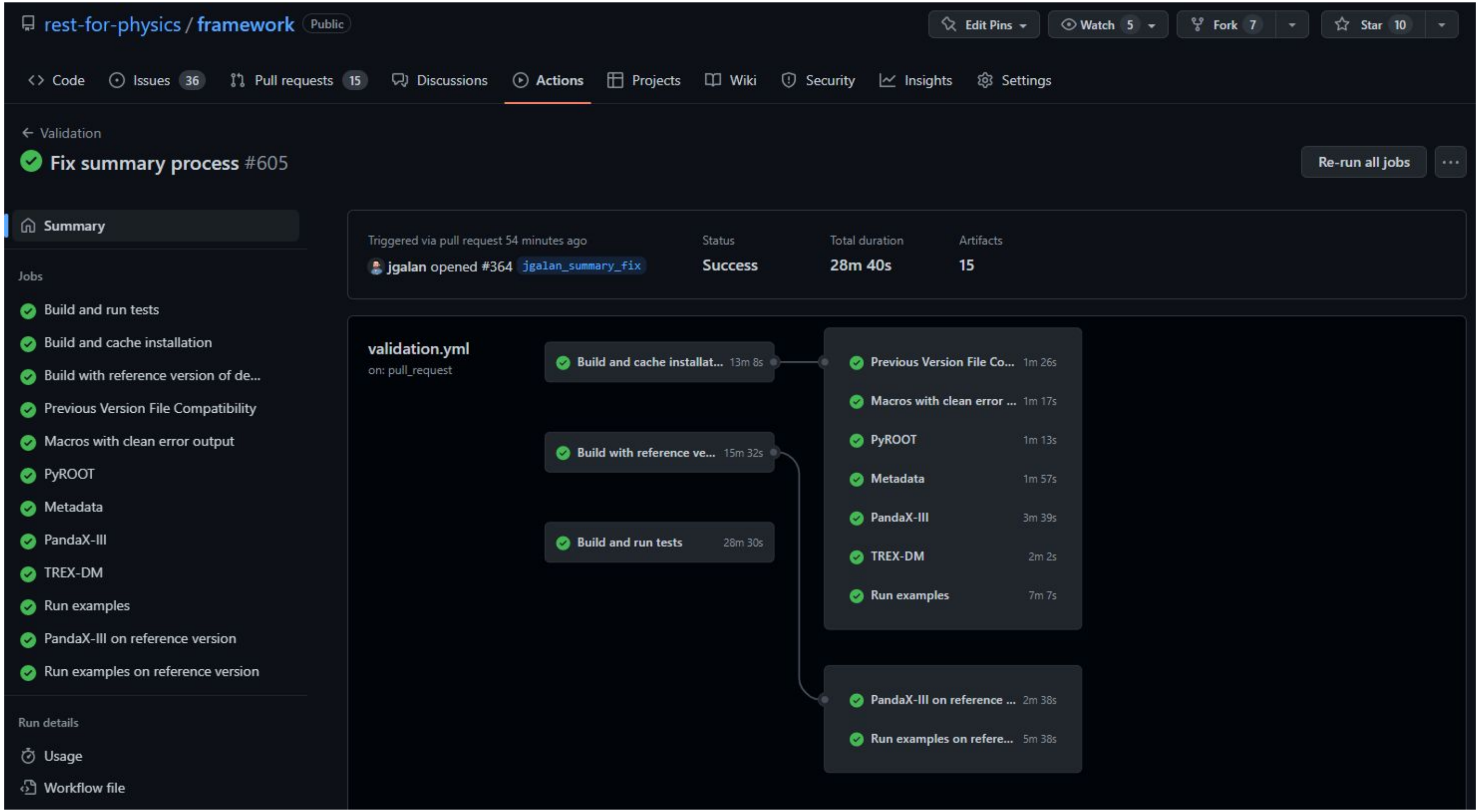
When you commit code to your repository, you can continuously build and test the code to make sure that the commit doesn't introduce errors. Your tests can include code linters (which check style formatting), security checks, code coverage, functional tests, and other custom checks.

Building and testing your code requires a server. You can build and test updates locally before pushing code to a repository, or you can use a CI server that checks for new code commits in a repository.

CI (aka validation pipeline) is performed in REST-for-Physics using GitHub actions

Validation pipeline

REST-for-Physics framework validation pipelines:



Fix summary process #605

Triggered via pull request 54 minutes ago
jgalan opened #364 `jgalan_summary_fix`

Status: Success
Total duration: 28m 40s
Artifacts: 15

Jobs:
- Build and run tests
- Build and cache installation
- Build with reference version of de...
- Previous Version File Compatibility
- Macros with clean error output
- PyROOT
- Metadata
- PandaX-III
- TREX-DM
- Run examples
- PandaX-III on reference version
- Run examples on reference version

validation.yml
on: pull_request

Job	Duration
Build and cache installat...	13m 8s
Build with reference ve...	15m 32s
Build and run tests	28m 30s
Previous Version File Co...	1m 26s
Macros with clean error ...	1m 17s
PyROOT	1m 13s
Metadata	1m 57s
PandaX-III	3m 39s
TREX-DM	2m 2s
Run examples	7m 7s
PandaX-III on reference ...	2m 38s
Run examples on refere...	5m 38s

Validation pipeline

REST-for-Physics validation pipelines includes:

- Build and install the repository
- Google test
- Backward compatibility
- Several examples and reference data processing.

When to create a validation pipeline:

- New library or package added
- New development added
- An issue that has not been spotted in current validation pipeline

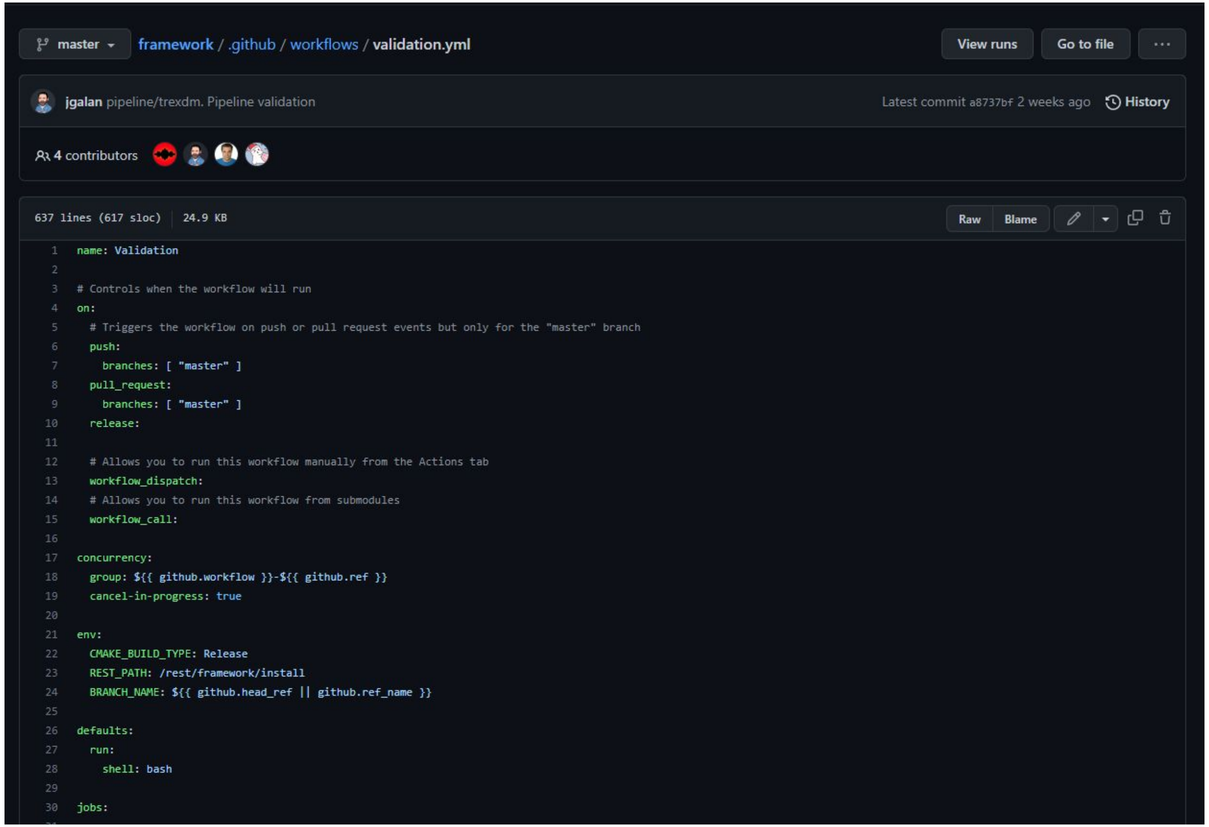
Validation pipeline

GitHub actions are written in yaml and requires:

- Workflow control (when the pipeline is triggered)
- An operative system to run the pipeline (we use ubuntu-latest)
- A shell to run the different commands (e.g bash, sh, python, cmd,...)
- A docker image (here we have a custom image with necessary software e.g. ROOT, Garfield and Geant4)
- Different jobs with dependencies between them
    - Build and cache installation
        - Backward compatibility
        - Macros
        - PyROOT
        - Examples
        - …
    - Build and google test

Validation pipeline

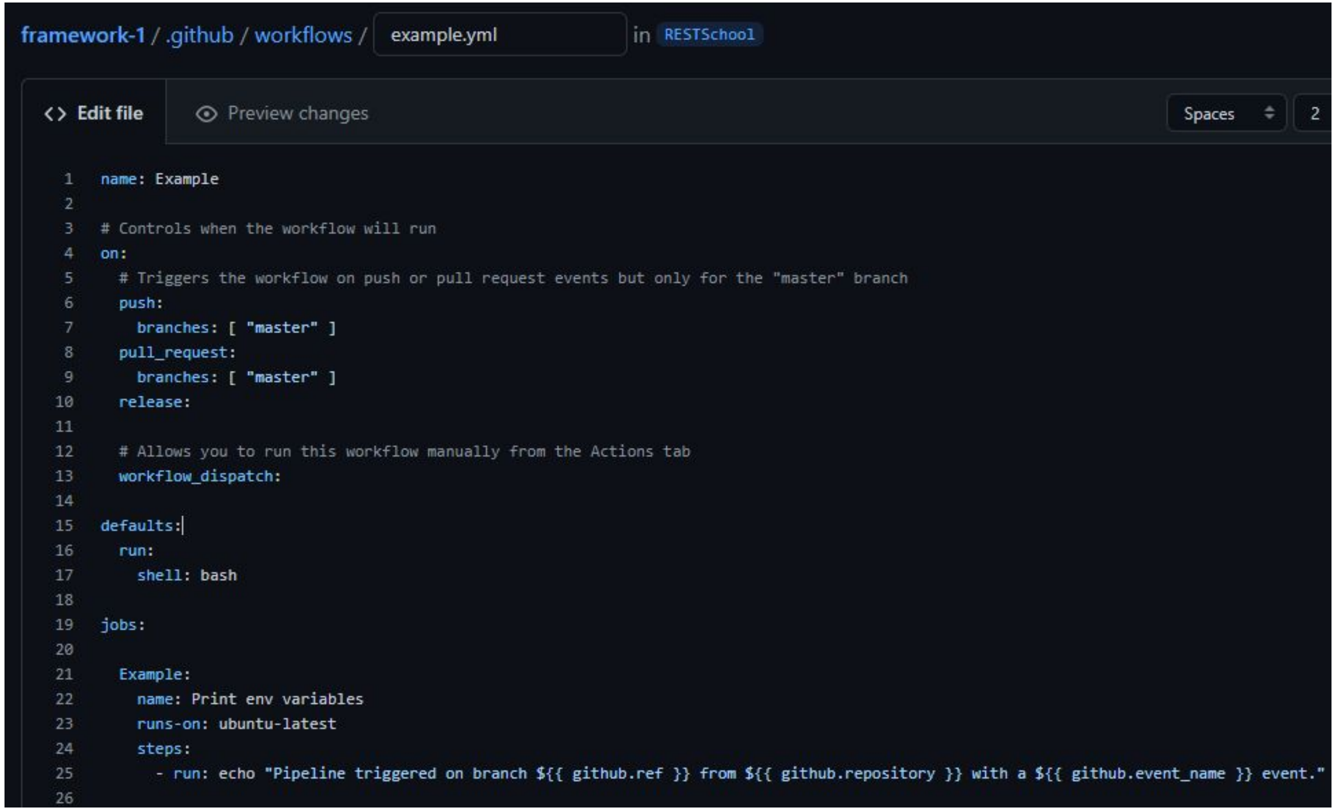
Validation pipelines are under .github/workflows/validation.yml


master
framework / .github / workflows / validation.yml
View runs
Go to file
jgalan pipeline/trexdm. Pipeline validation
Latest commit a8737bf 2 weeks ago
History
4 contributors
637 lines (617 sloc) | 24.9 KB
Raw
Blame

```yaml
name: Validation

# Controls when the workflow will run
on:
  # Triggers the workflow on push or pull request events but only for the "master" branch
  push:
    branches: [ "master" ]
  pull_request:
    branches: [ "master" ]
  release:

  # Allows you to run this workflow manually from the Actions tab
  workflow_dispatch:
  # Allows you to run this workflow from submodules
  workflow_call:

concurrency:
  group: ${{ github.workflow }}-${{ github.ref }}
  cancel-in-progress: true

env:
  CMAKE_BUILD_TYPE: Release
  REST_PATH: /rest/framework/install
  BRANCH_NAME: ${{ github.head_ref || github.ref_name }}

defaults:
  run:
    shell: bash

jobs:
```

Validation pipeline

Example, validation pipeline to print some environmental variables



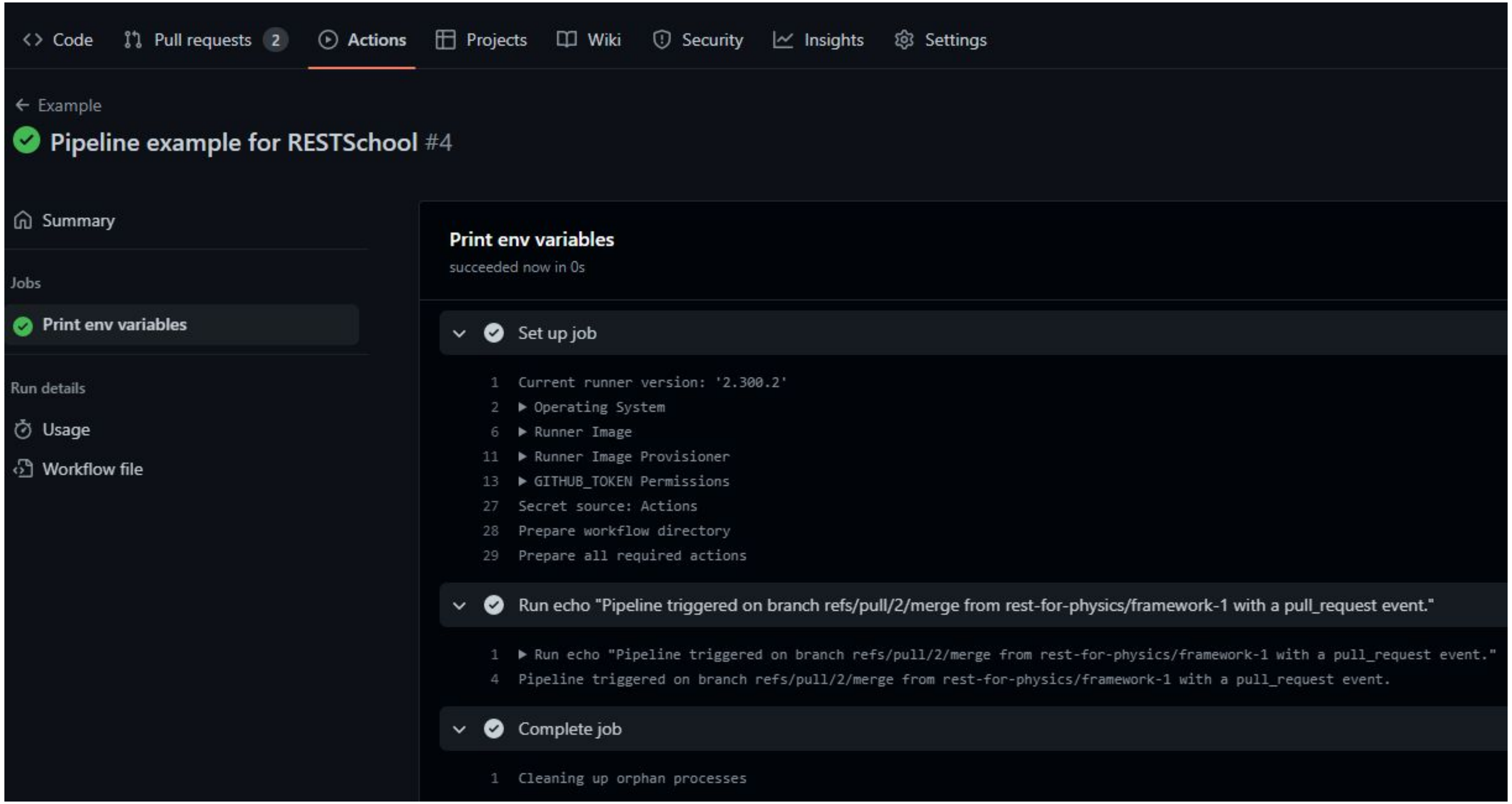
framework-1 / .github / workflows / example.yml in RESTSchool

```yaml
name: Example

# Controls when the workflow will run
on:
  # Triggers the workflow on push or pull request events but only for the "master" branch
  push:
    branches: [ "master" ]
  pull_request:
    branches: [ "master" ]
  release:

  # Allows you to run this workflow manually from the Actions tab
  workflow_dispatch:

defaults:
  run:
    shell: bash

jobs:

  Example:
    name: Print env variables
    runs-on: ubuntu-latest
    steps:
      - run: echo "Pipeline triggered on branch ${{ github.ref }} from ${{ github.repository }} with a ${{ github.event_name }} event."

```

Tip: You can use GitHub editor to edit the pipeline

Validation pipeline

In case of failure you should check the logs under Actions tab.



Pipeline example for RESTSchool #4

Print env variables

succeeded now in 0s

Set up job

```
Current runner version: '2.300.2'
▶ Operating System
▶ Runner Image
▶ Runner Image Provisioner
▶ GITHUB_TOKEN Permissions
Secret source: Actions
Prepare workflow directory
Prepare all required actions
```

Run echo "Pipeline triggered on branch refs/pull/2/merge from rest-for-physics/framework-1 with a pull_request event."

```
▶ Run echo "Pipeline triggered on branch refs/pull/2/merge from rest-for-physics/framework-1 with a pull_request event."
Pipeline triggered on branch refs/pull/2/merge from rest-for-physics/framework-1 with a pull_request event.
```

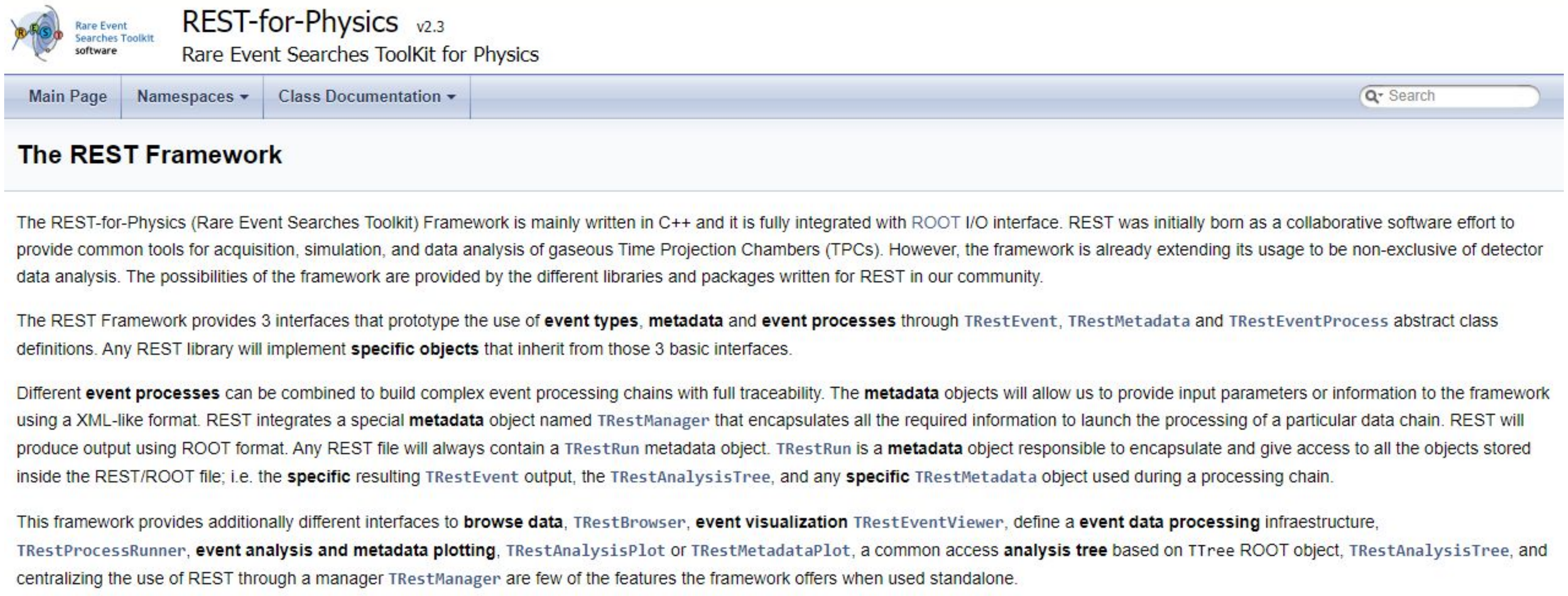
Complete job

```
Cleaning up orphan processes
```

REST-for-Physics documentation

REST-for-Physics documentation can be found in https://sultan.unizar.es/rest/

Documentation is created using Doxygen which relies on commenting out the code with the proper syntax



The REST Framework

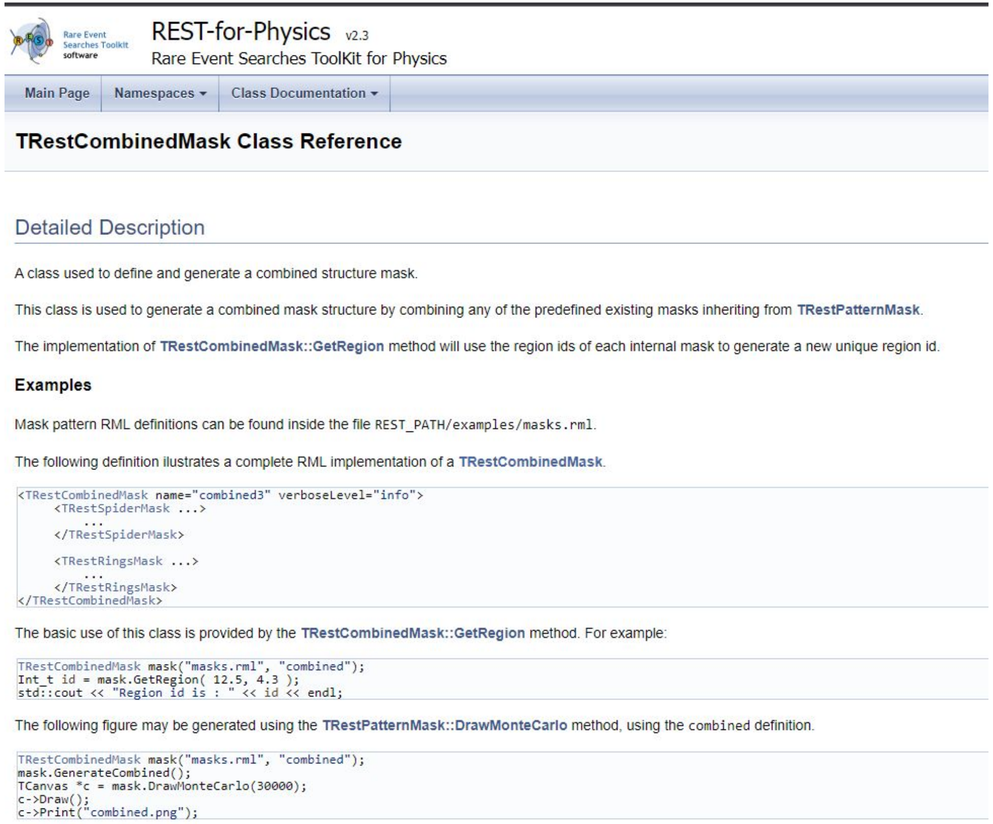
The REST-for-Physics (Rare Event Searches Toolkit) Framework is mainly written in C++ and it is fully integrated with ROOT I/O interface. REST was initially born as a collaborative software effort to provide common tools for acquisition, simulation, and data analysis of gaseous Time Projection Chambers (TPCs). However, the framework is already extending its usage to be non-exclusive of detector data analysis. The possibilities of the framework are provided by the different libraries and packages written for REST in our community.

The REST Framework provides 3 interfaces that prototype the use of **event types**, **metadata** and **event processes** through `TRestEvent`, `TRestMetadata` and `TRestEventProcess` abstract class definitions. Any REST library will implement **specific objects** that inherit from those 3 basic interfaces.

Different **event processes** can be combined to build complex event processing chains with full traceability. The **metadata** objects will allow us to provide input parameters or information to the framework using a XML-like format. REST integrates a special **metadata** object named `TRestManager` that encapsulates all the required information to launch the processing of a particular data chain. REST will produce output using ROOT format. Any REST file will always contain a `TRestRun` metadata object. `TRestRun` is a **metadata** object responsible to encapsulate and give access to all the objects stored inside the REST/ROOT file; i.e. the **specific** resulting `TRestEvent` output, the `TRestAnalysisTree`, and any **specific** `TRestMetadata` object used during a processing chain.

This framework provides additionally different interfaces to **browse data**, `TRestBrowser`, **event visualization** `TRestEventViewer`, define a **event data processing** infraestructure, `TRestProcessRunner`, **event analysis and metadata plotting**, `TRestAnalysisPlot` or `TRestMetadataPlot`, a common access **analysis tree** based on `TTree` ROOT object, `TRestAnalysisTree`, and centralizing the use of REST through a manager `TRestManager` are few of the features the framework offers when used standalone.

Example TRestCombinedMask


Rare Event Searches Toolkit software
REST-for-Physics v2.3
Rare Event Searches ToolKit for Physics
Main Page
Namespaces
Class Documentation

TRestCombinedMask Class Reference

Detailed Description

A class used to define and generate a combined structure mask.

This class is used to generate a combined mask structure by combining any of the predefined existing masks inheriting from TRestPatternMask.

The implementation of TRestCombinedMask::GetRegion method will use the region ids of each internal mask to generate a new unique region id.

Examples

Mask pattern RML definitions can be found inside the file `REST_PATH/examples/masks.rml`.

The following definition ilustrates a complete RML implementation of a TRestCombinedMask.

```
<TRestCombinedMask name="combined3" verboseLevel="info">
     <TRestSpiderMask ...>
          ...
     </TRestSpiderMask>

     <TRestRingsMask ...>
          ...
     </TRestRingsMask>
</TRestCombinedMask>
```

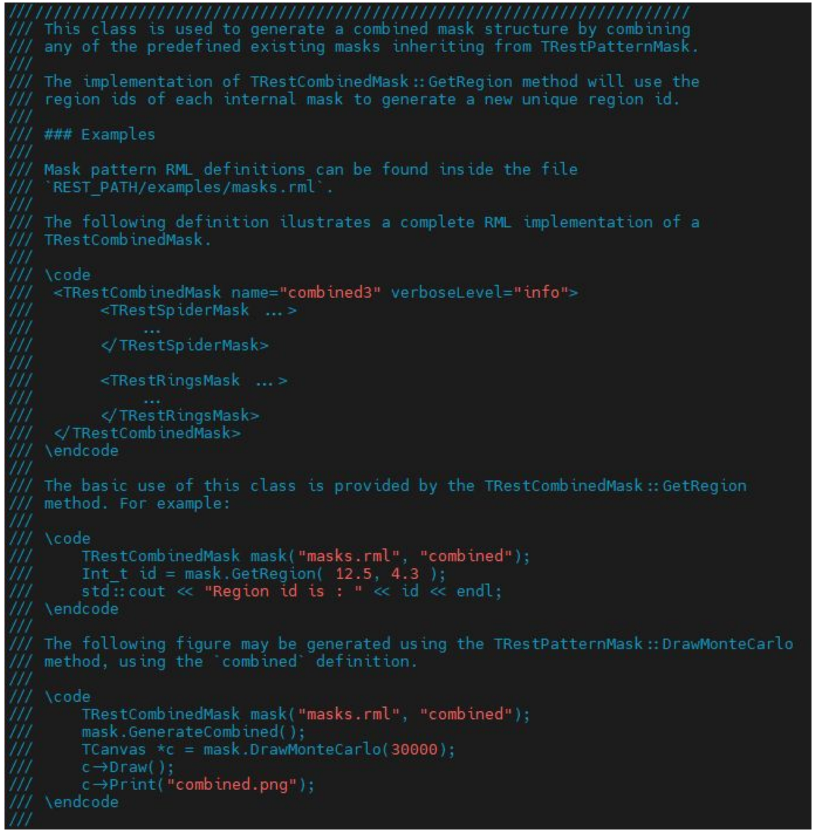
The basic use of this class is provided by the TRestCombinedMask::GetRegion method. For example:

```
TRestCombinedMask mask("masks.rml", "combined");
Int_t id = mask.GetRegion( 12.5, 4.3 );
std::cout << "Region id is : " << id << endl;
```

The following figure may be generated using the TRestPatternMask::DrawMonteCarlo method, using the `combined` definition.

```
TRestCombinedMask mask("masks.rml", "combined");
mask.GenerateCombined();
TCanvas *c = mask.DrawMonteCarlo(30000);
c->Draw();
c->Print("combined.png");
```

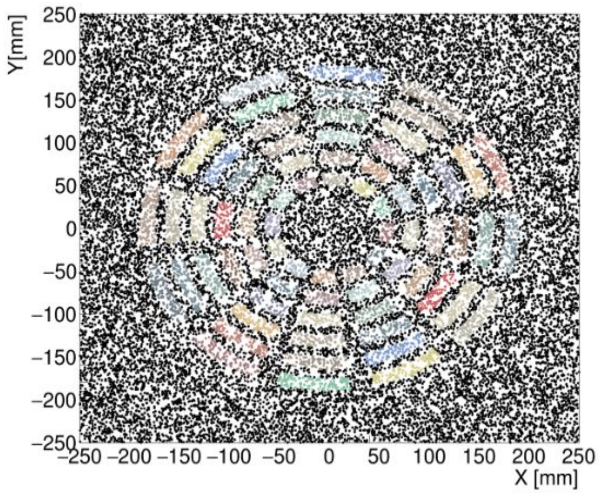
TRestCombinedMask.cxx



```
//////////////////////////////////////////////////////////////////////////
/// This class is used to generate a combined mask structure by combining
/// any of the predefined existing masks inheriting from TRestPatternMask.
///
/// The implementation of TRestCombinedMask::GetRegion method will use the
/// region ids of each internal mask to generate a new unique region id.
///
/// ### Examples
///
/// Mask pattern RML definitions can be found inside the file
/// `REST_PATH/examples/masks.rml`.
///
/// The following definition ilustrates a complete RML implementation of a
/// TRestCombinedMask.
///
/// \code
///  <TRestCombinedMask name="combined3" verboseLevel="info">
///       <TRestSpiderMask  ...>
///            ...
///       </TRestSpiderMask>
///
///       <TRestRingsMask  ...>
///            ...
///       </TRestRingsMask>
///  </TRestCombinedMask>
/// \endcode
///
/// The basic use of this class is provided by the TRestCombinedMask::GetRegion
/// method. For example:
///
/// \code
///     TRestCombinedMask mask("masks.rml", "combined");
///     Int_t id = mask.GetRegion( 12.5, 4.3 );
///     std::cout << "Region id is : " << id << endl;
/// \endcode
///
/// The following figure may be generated using the TRestPatternMask::DrawMonteCarlo
/// method, using the `combined` definition.
///
/// \code
///     TRestCombinedMask mask("masks.rml", "combined");
///     mask.GenerateCombined();
///     TCanvas *c = mask.DrawMonteCarlo(30000);
///     c->Draw();
///     c->Print("combined.png");
/// \endcode
///
```

Example TRestCombinedMask

TRestCombinedMask.cxx



An illustration of the montecarlo mask test using DrawMonteCarlo

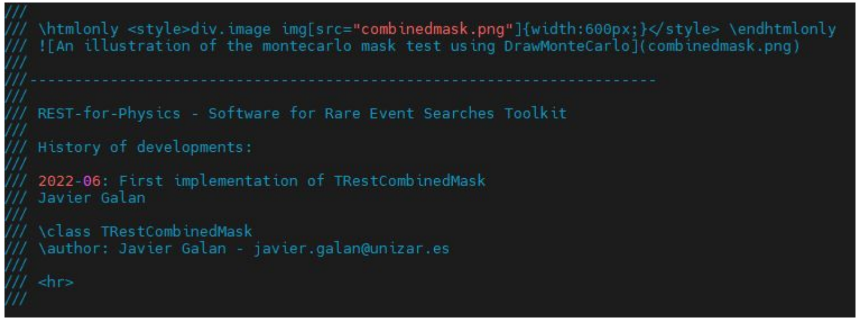
REST-for-Physics - Software for Rare Event Searches Toolkit

History of developments:

2022-06: First implementation of **TRestCombinedMask** Javier Galan

Author

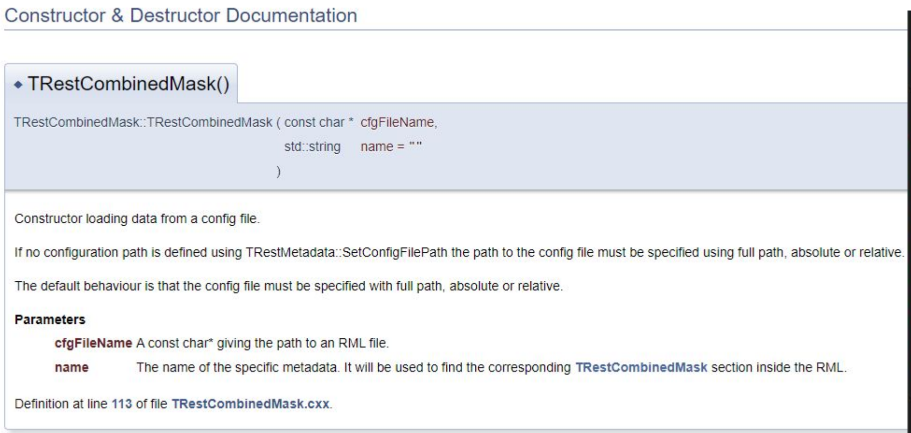
: Javier Galan - javier.galan@unizar.es



```
///
/// \htmlonly <style>div.image img[src="combinedmask.png"]{width:600px;}</style> \endhtmlonly
/// ![An illustration of the montecarlo mask test using DrawMonteCarlo](combinedmask.png)
///
///----------------------------------------------------------------------
///
/// REST-for-Physics - Software for Rare Event Searches Toolkit
///
/// History of developments:
///
/// 2022-06: First implementation of TRestCombinedMask
/// Javier Galan
///
/// \class TRestCombinedMask
/// \author: Javier Galan - javier.galan@unizar.es
///
/// <hr>
///
```

Example TRestCombinedMask

TRestCombinedMask.cxx



Constructor & Destructor Documentation

◆ TRestCombinedMask()

```
TRestCombinedMask::TRestCombinedMask ( const char *  cfgFileName,
                                       std::string   name = ""
                                     )
```

Constructor loading data from a config file.

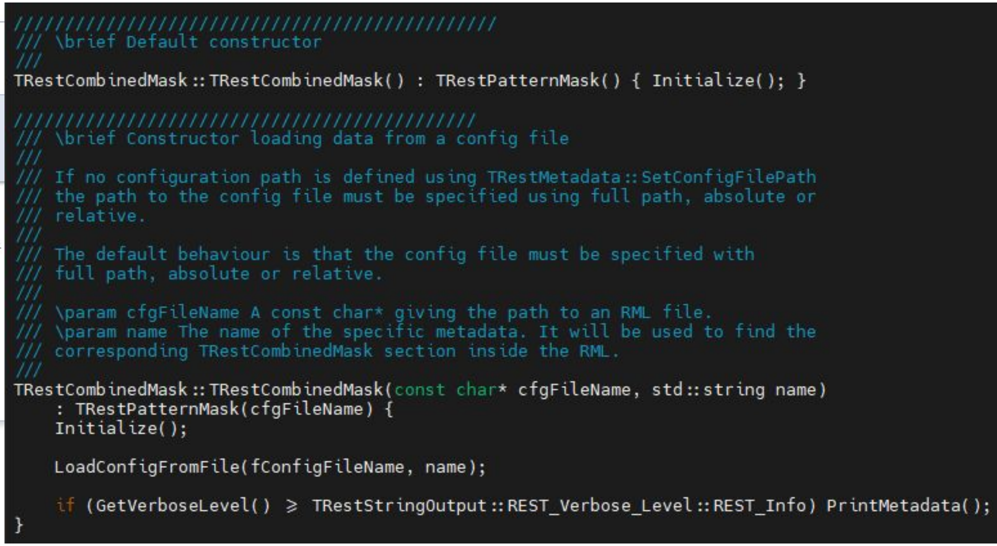
If no configuration path is defined using TRestMetadata::SetConfigFilePath the path to the config file must be specified using full path, absolute or relative.

The default behaviour is that the config file must be specified with full path, absolute or relative.

Parameters

cfgFileName	A const char* giving the path to an RML file.
name	The name of the specific metadata. It will be used to find the corresponding **TRestCombinedMask** section inside the RML.

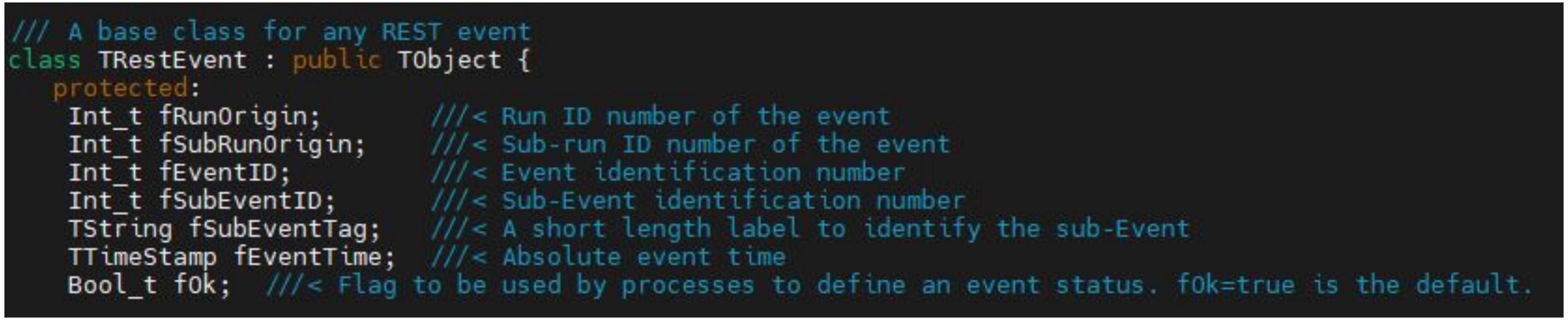
Definition at line 113 of file **TRestCombinedMask.cxx**.



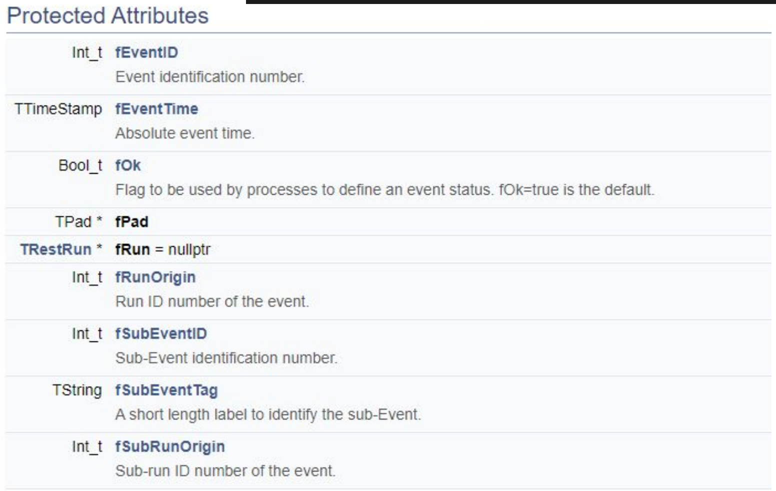
```
///////////////////////////////////////////////
/// \brief Default constructor
///
TRestCombinedMask::TRestCombinedMask() : TRestPatternMask() { Initialize(); }

//////////////////////////////////////////////
/// \brief Constructor loading data from a config file
///
/// If no configuration path is defined using TRestMetadata::SetConfigFilePath
/// the path to the config file must be specified using full path, absolute or
/// relative.
///
/// The default behaviour is that the config file must be specified with
/// full path, absolute or relative.
///
/// \param cfgFileName A const char* giving the path to an RML file.
/// \param name The name of the specific metadata. It will be used to find the
/// corresponding TRestCombinedMask section inside the RML.
///
TRestCombinedMask::TRestCombinedMask(const char* cfgFileName, std::string name)
    : TRestPatternMask(cfgFileName) {
    Initialize();

    LoadConfigFromFile(fConfigFileName, name);

    if (GetVerboseLevel() ≥ TRestStringOutput::REST_Verbose_Level::REST_Info) PrintMetadata();
}
```

Example TRestEvent

TRestEvent.h



```
/// A base class for any REST event
class TRestEvent : public TObject {
  protected:
   Int_t fRunOrigin;         ///< Run ID number of the event
   Int_t fSubRunOrigin;      ///< Sub-run ID number of the event
   Int_t fEventID;           ///< Event identification number
   Int_t fSubEventID;        ///< Sub-Event identification number
   TString fSubEventTag;     ///< A short length label to identify the sub-Event
   TTimeStamp fEventTime;    ///< Absolute event time
   Bool_t fOk;   ///< Flag to be used by processes to define an event status. fOk=true is the default.
```



Protected Attributes

Int_t	**fEventID** Event identification number.
TTimeStamp	**fEventTime** Absolute event time.
Bool_t	**fOk** Flag to be used by processes to define an event status. fOk=true is the default.
TPad *	**fPad**
TRestRun *	**fRun** = nullptr
Int_t	**fRunOrigin** Run ID number of the event.
Int_t	**fSubEventID** Sub-Event identification number.
TString	**fSubEventTag** A short length label to identify the sub-Event.
Int_t	**fSubRunOrigin** Sub-run ID number of the event.

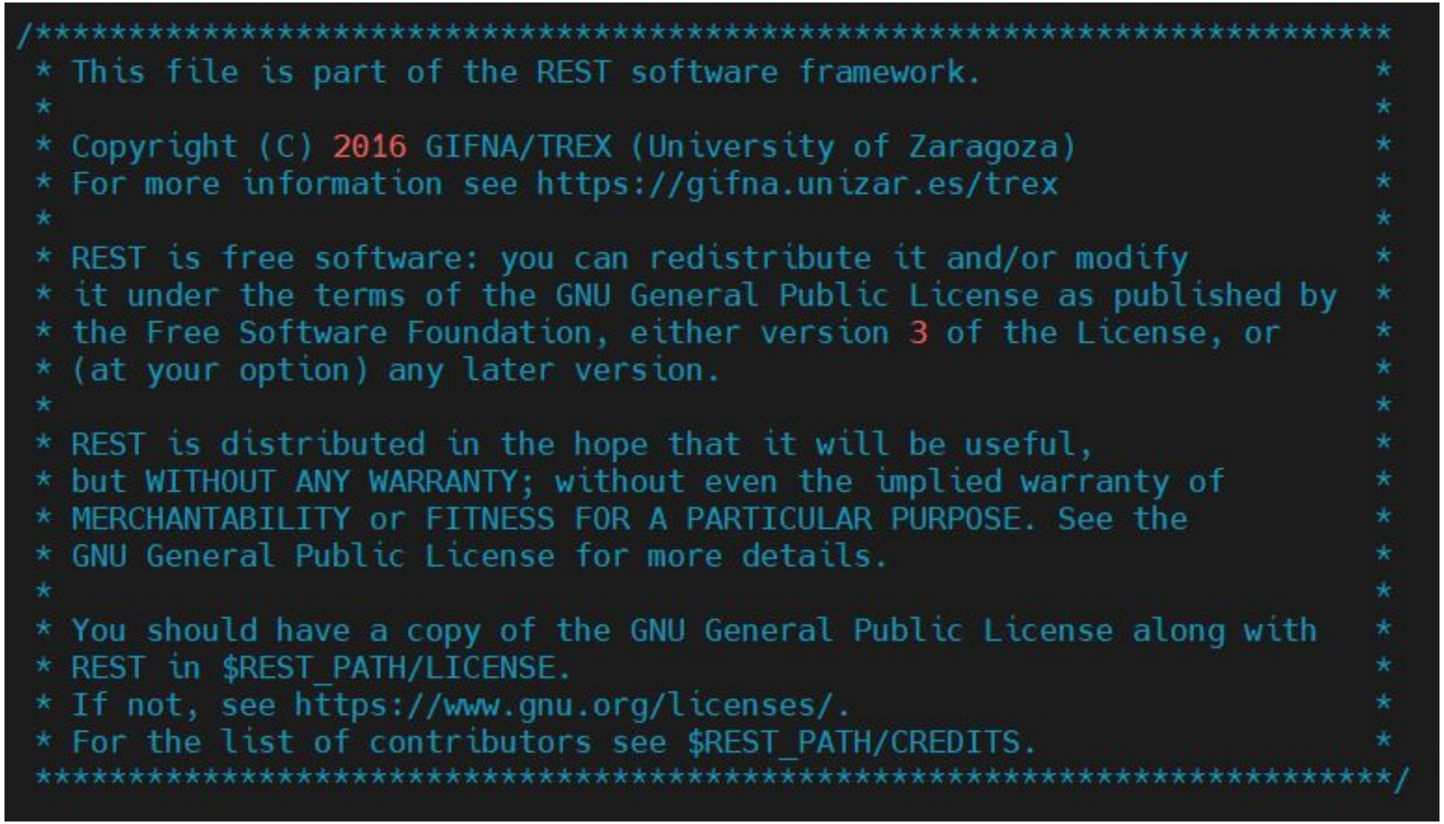
REST-for-Physics documentation

General rules for the documentation in REST-for-Physics:

- Add a detailed description of the class or function
- Give several examples of how to use the class
- For a metadata class explain properly all the metadata members in the description
- Add a brief description of all the members of a class
- Repository is not fully documented, any help is welcome for documenting the code.

REST-for-Physics documentation

Template for documentation:

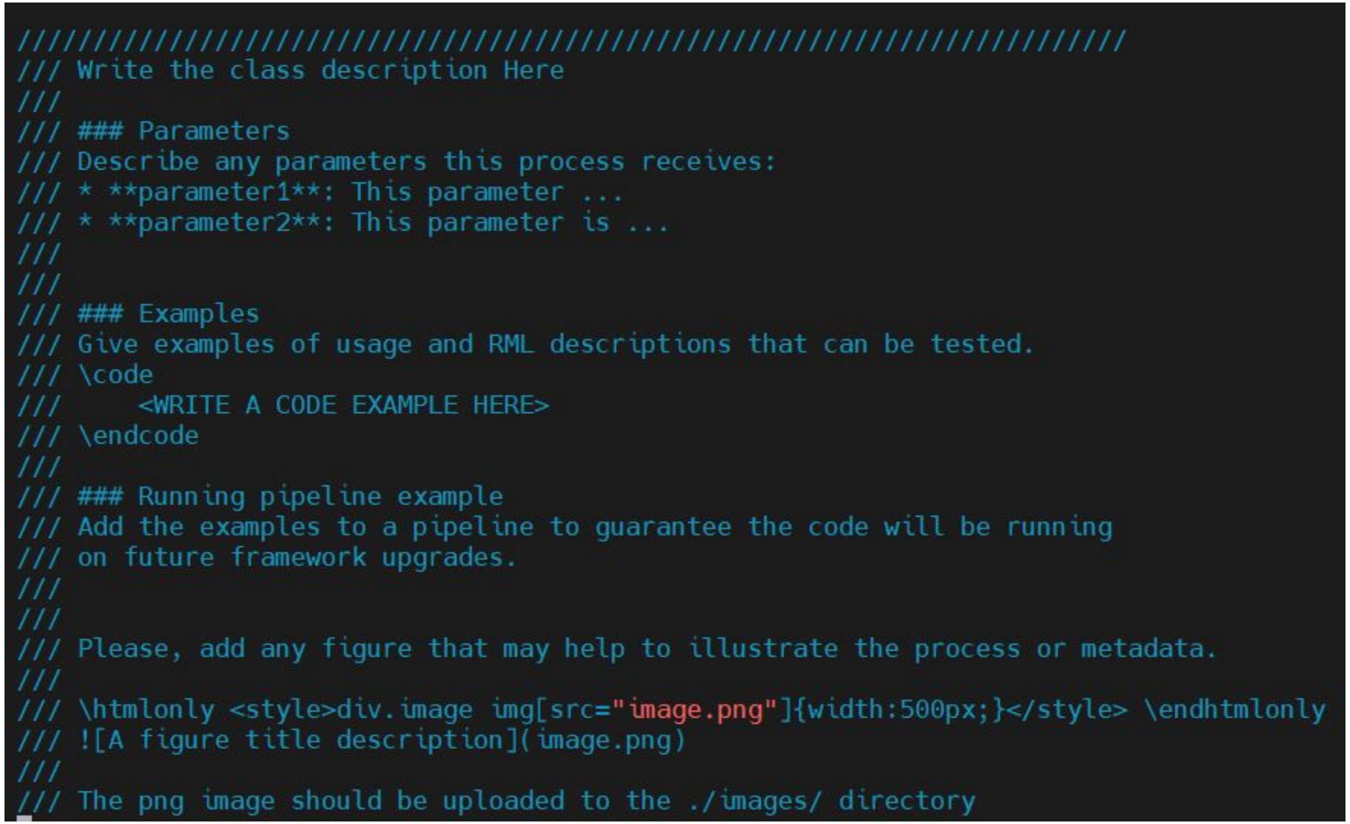


```
/*************************************************************************
 * This file is part of the REST software framework.                     *
 *                                                                       *
 * Copyright (C) 2016 GIFNA/TREX (University of Zaragoza)                *
 * For more information see https://gifna.unizar.es/trex                 *
 *                                                                       *
 * REST is free software: you can redistribute it and/or modify          *
 * it under the terms of the GNU General Public License as published by  *
 * the Free Software Foundation, either version 3 of the License, or     *
 * (at your option) any later version.                                   *
 *                                                                       *
 * REST is distributed in the hope that it will be useful,               *
 * but WITHOUT ANY WARRANTY; without even the implied warranty of        *
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the          *
 * GNU General Public License for more details.                          *
 *                                                                       *
 * You should have a copy of the GNU General Public License along with   *
 * REST in $REST_PATH/LICENSE.                                           *
 * If not, see https://www.gnu.org/licenses/.                            *
 * For the list of contributors see $REST_PATH/CREDITS.                  *
 *************************************************************************/
```

Copyright/License at the beginning of the header or source file

REST-for-Physics documentation

Template for documentation (source file):



```
//////////////////////////////////////////////////////////////////////////
/// Write the class description Here
///
/// ### Parameters
/// Describe any parameters this process receives:
/// * **parameter1**: This parameter ...
/// * **parameter2**: This parameter is ...
///
///
/// ### Examples
/// Give examples of usage and RML descriptions that can be tested.
/// \code
///     <WRITE A CODE EXAMPLE HERE>
/// \endcode
///
/// ### Running pipeline example
/// Add the examples to a pipeline to guarantee the code will be running
/// on future framework upgrades.
///
///
/// Please, add any figure that may help to illustrate the process or metadata.
///
/// \htmlonly <style>div.image img[src="image.png"]{width:500px;}</style> \endhtmlonly
/// ![A figure title description](image.png)
///
/// The png image should be uploaded to the ./images/ directory
```

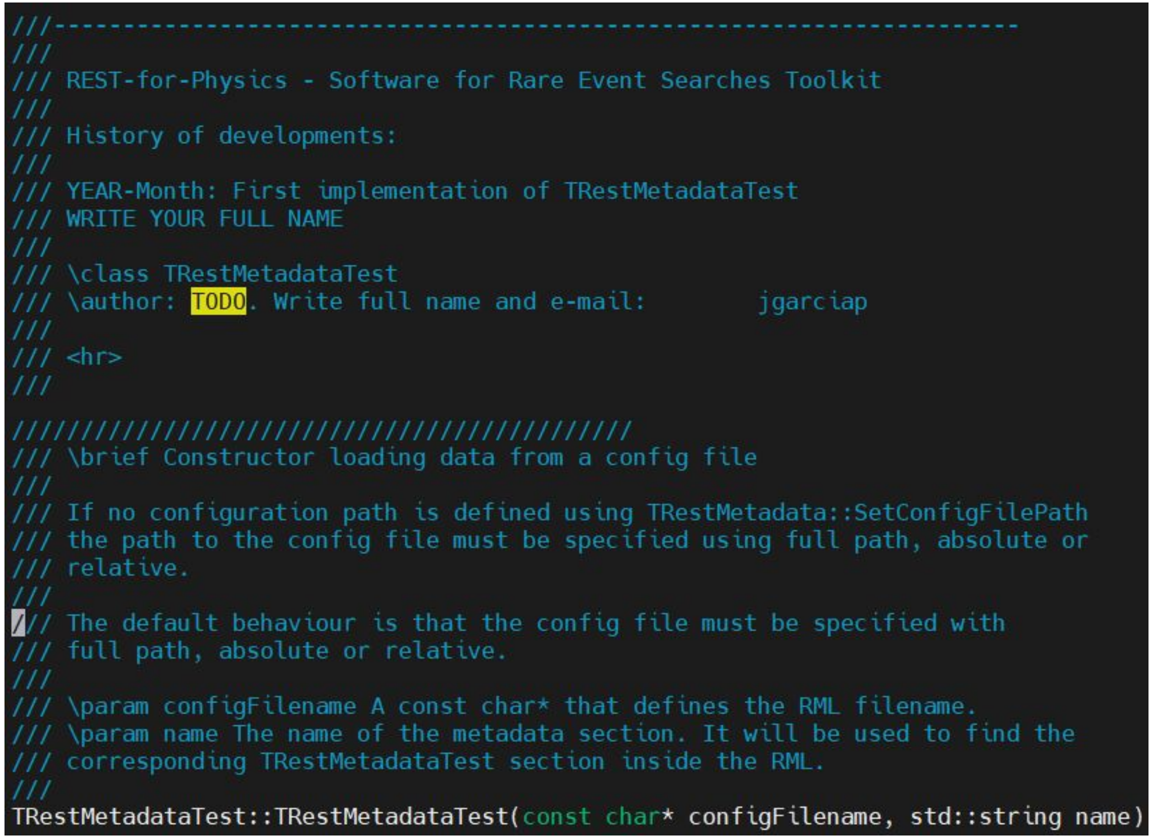
Description of the class and the different parameters.

Give examples using rml description.

Add as many information as possible.

REST-for-Physics documentation

Template for documentation (source file):



```
///-----------------------------------------------------------------------
///
/// REST-for-Physics - Software for Rare Event Searches Toolkit
///
/// History of developments:
///
/// YEAR-Month: First implementation of TRestMetadataTest
/// WRITE YOUR FULL NAME
///
/// \class TRestMetadataTest
/// \author: TODO. Write full name and e-mail:        jgarciap
///
/// <hr>
///

/////////////////////////////////////////////
/// \brief Constructor loading data from a config file
///
/// If no configuration path is defined using TRestMetadata::SetConfigFilePath
/// the path to the config file must be specified using full path, absolute or
/// relative.
///
/// The default behaviour is that the config file must be specified with
/// full path, absolute or relative.
///
/// \param configFilename A const char* that defines the RML filename.
/// \param name The name of the metadata section. It will be used to find the
/// corresponding TRestMetadataTest section inside the RML.
///
TRestMetadataTest::TRestMetadataTest(const char* configFilename, std::string name)
```

Keep track of the changes on the class.

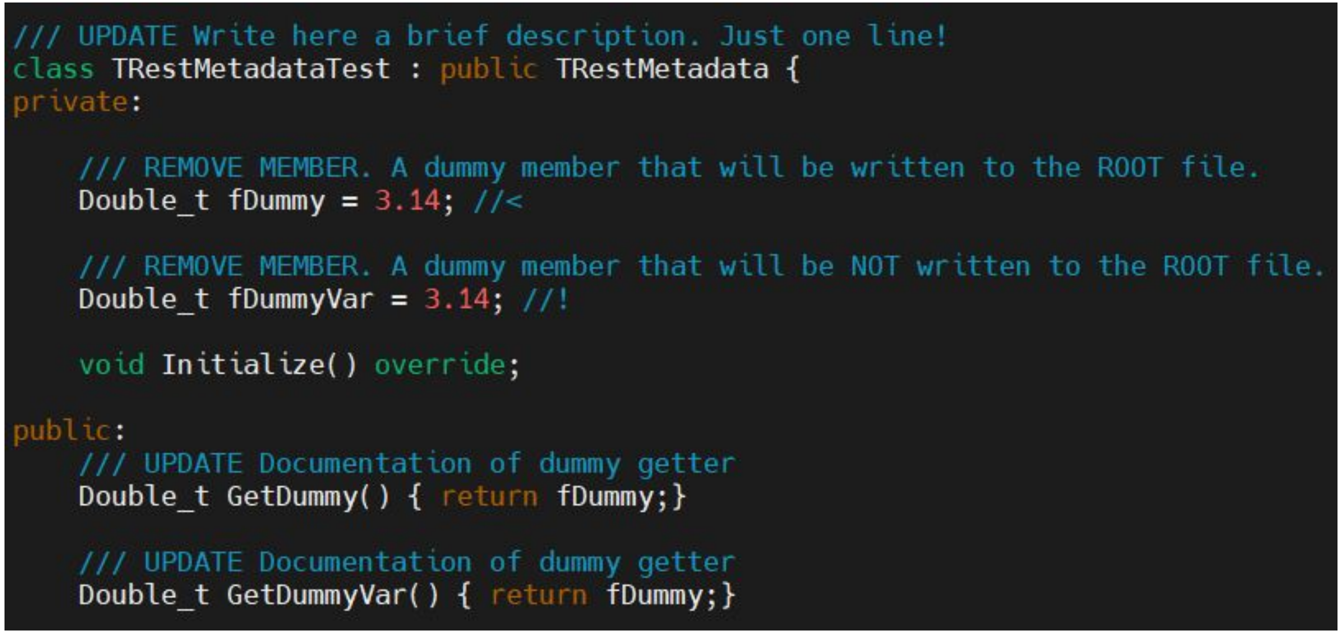
Add author and date.

Start the description of a function with ***\brief***

Add as many information as possible

REST-for-Physics documentation

Template for documentation (header file):



```
/// UPDATE Write here a brief description. Just one line!
class TRestMetadataTest : public TRestMetadata {
private:

    /// REMOVE MEMBER. A dummy member that will be written to the ROOT file.
    Double_t fDummy = 3.14; //<

    /// REMOVE MEMBER. A dummy member that will be NOT written to the ROOT file.
    Double_t fDummyVar = 3.14; //!

    void Initialize() override;

public:
    /// UPDATE Documentation of dummy getter
    Double_t GetDummy() { return fDummy;}

    /// UPDATE Documentation of dummy getter
    Double_t GetDummyVar() { return fDummy;}
```

Brief description of the class.

Describe your data members if any.

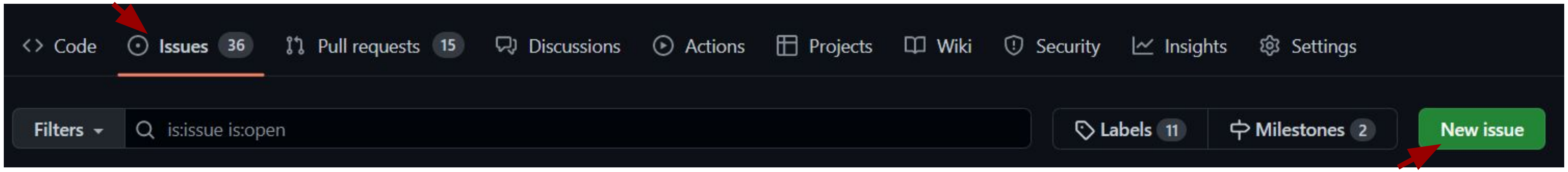
Document your inline functions.

Creating issues in GitHub

When to create an issue:

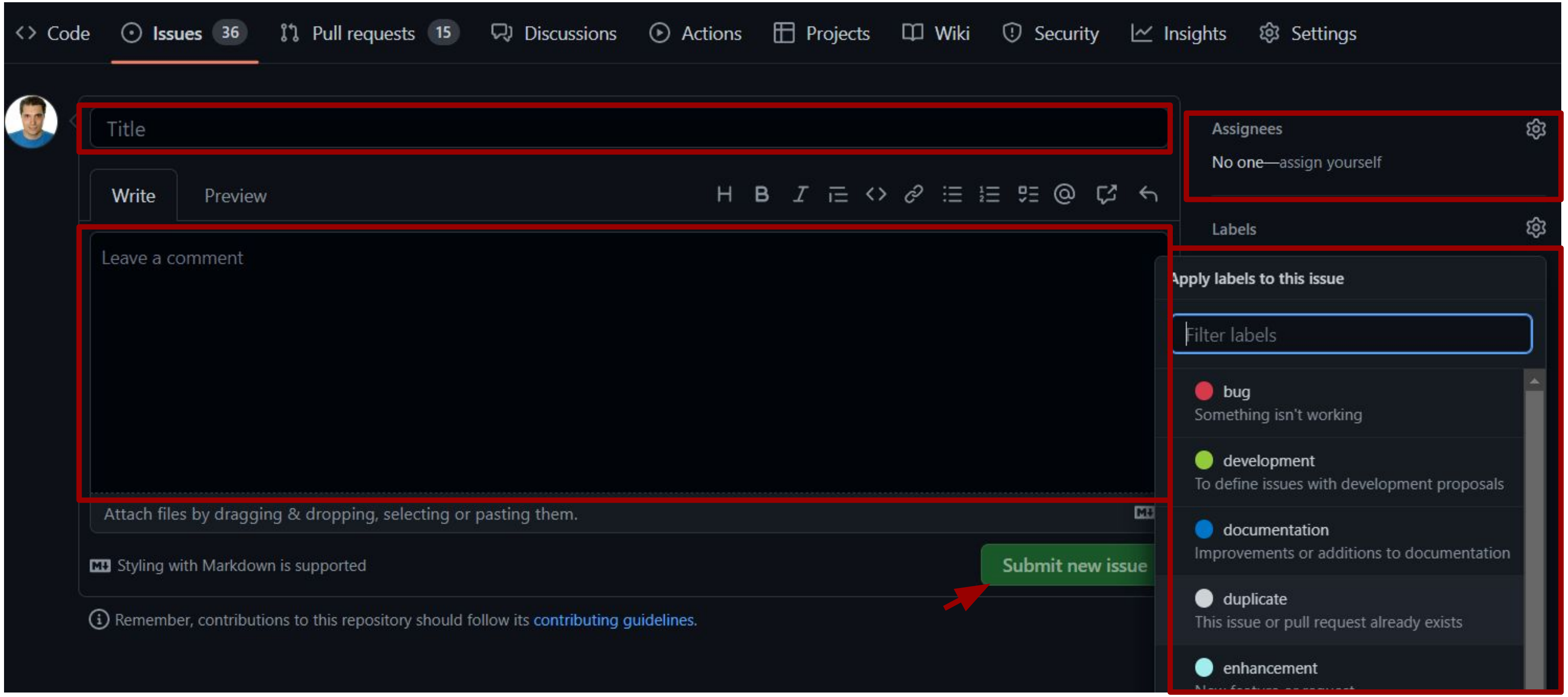
- Bug: something is not working as expected.
- Feature request: I have an idea that I would like to implement.
- Keep track: Track things that are not done due to lack of time.

How to create an issue



Creating issues in GitHub

How to create an issue

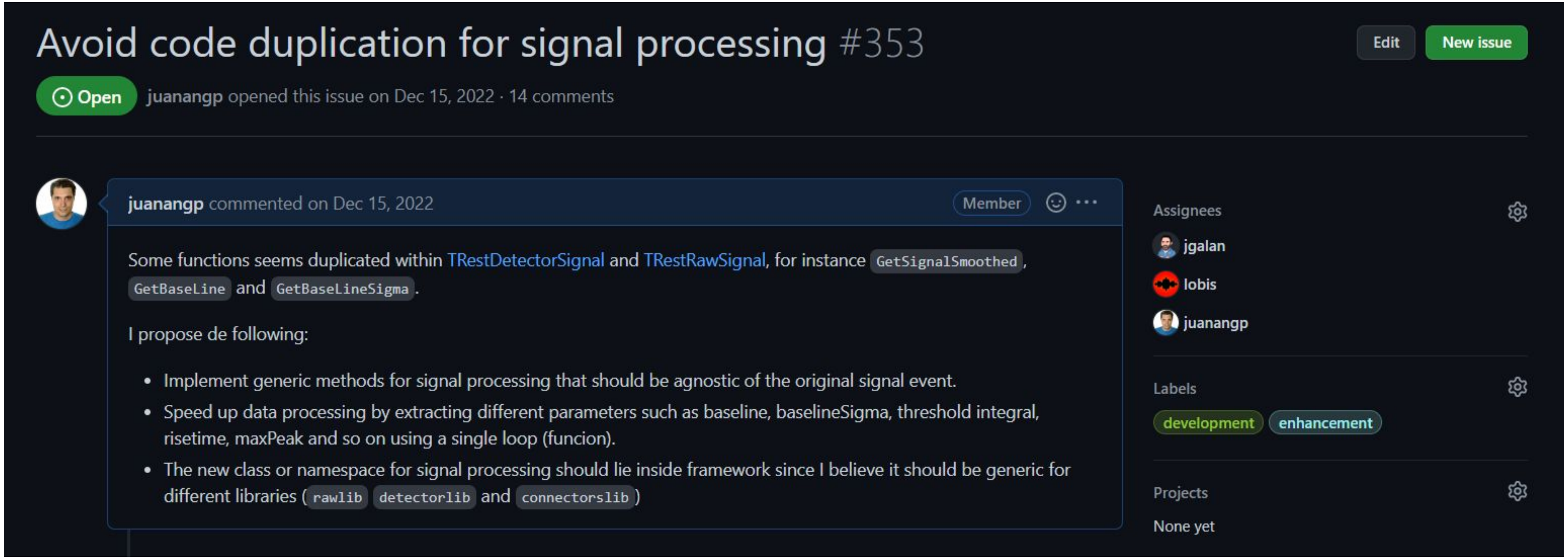


Remember, contributions to this repository should follow its contributing guidelines.

bug	Something isn't working
development	To define issues with development proposals
documentation	Improvements or additions to documentation
duplicate	This issue or pull request already exists
enhancement	

Creating issues in GitHub

Example of an issue



Avoid code duplication for signal processing #353

juanangp opened this issue on Dec 15, 2022 · 14 comments

juanangp commented on Dec 15, 2022

Some functions seems duplicated within TRestDetectorSignal and TRestRawSignal, for instance `GetSignalSmoothed`, `GetBaseLine` and `GetBaseLineSigma`.

I propose de following:

- Implement generic methods for signal processing that should be agnostic of the original signal event.
- Speed up data processing by extracting different parameters such as baseline, baselineSigma, threshold integral, risetime, maxPeak and so on using a single loop (funcion).
- The new class or namespace for signal processing should lie inside framework since I believe it should be generic for different libraries (`rawlib` `detectorlib` and `connectorslib`)

Assignees

jgalan

lobis

juanangp

Labels

development enhancement

Projects

None yet

Creating issues in GitHub

Issues are particularly important:

- Keep track of bugs or new features
- Issues might require several iterations until a quorum is reached
- An issue can be split between the different developers
- Any developer can report any issue although the reporter is not necessarily the responsible of fixing it
- Some issues are suitable for getting started in REST-for-Physics development, and are tagged as

How to close an issue:

- Issue should be closed with a dedicated Pull Request in which the issue is properly tagged e.g. Fixes #353
- Issue can be reopened e.g. in case a bug is not properly resolved

REST-for-Physics naming convention

General C++ rules:

- The first character in the class name must be in upper case
- Use upper case letters as word separators, and lower case for the rest of the word in the class name.
- Digits may be used in a variable name but only after the alphabet.
- No special symbols can be used in variable names except for the underscore('_').

Naming convention:

- REST-for-Physics inherit naming convention from ROOT which follows Taligent rules.
  - Classes begin with *TRest*
  - Data members begin with *f*
  - Getters and setters begin with *Get…*, *Set…* or *Is…*
  - Macros starts with *REST_*

Good practises for contributing

Clang-format:

- For code readability it is highly recommended to format the code in the same way.
- clang-format is a tool to automatically format C/C++/Objective-C code.
- REST-for-Physics repository provides a script under ***/scripts/reformat-clang.sh*** to apply the appropriate format to the different files.
- Work is on-going to automatically apply clang-format after commit.

Good practises for contributing

Coding style:

- REST-for-Physics try to follow Google C++ style.
  - Code should target C++17, use STL iterators and structure bindings when possible
  - All header files should have #define guards to prevent multiple inclusion
  - Avoid using forward declarations where possible.
  - Define inline functions only when they are small <10 lines
  - Use ***nullptr*** for pointers, and ***'\0'*** for chars

Good practises for contributing

Standard output:

-