



University of
Zurich^{UZH}

S3IT

More on workflows

Riccardo Murri <riccardo.murri@uzh.ch>

S3IT: Services and Support for Science IT

University of Zurich

Automatic arrangement of tasks

Want to avoid arranging tasks in parallel- and sequential- task collections? Use a `DependentTaskCollection`!

```
from gc3libs.workflow \
    import DependentTaskCollection

class MyWorkflow (DependentTaskCollection):
    # ...
    def __init__(self, ...):
        DependentTaskCollection.__init__(self)
        app1 = AnApp(...)
        app2 = AnotherApp(...)
        app3 = AThirdApp(...)
        self.add(app1)
        self.add(app2)
        self.add(app3, after=[app1, app2])
```

Usage of DependentTaskCollection

```
from gc3libs.workflow \
    import DependentTaskCollection

class MyWorkflow(DependentTaskCollection):
    # ...
    def __init__(self, ...):
        DependentTaskCollection.__init__(self)
        app1 = AnApp(...)
        app2 = AnotherApp(...)
        app3 = AThirdApp(...)
        self.add(app1)
        self.add(app2)
        self.add(app3, after=[app1, app2])
```

Initialize the base class.

Usage of DependentTaskCollection

```
from gc3libs.workflow \
    import DependentTaskCollection

class MyWorkflow(DependentTaskCollection):
    # ...
    def __init__(self, ...):
        DependentTaskCollection.__init__(self)
        app1 = AnApp(...)
        app2 = AnotherApp(...)
        app3 = AThirdApp(...)
        self.add(app1)
        self.add(app2)
        self.add(app3, after=[app1, app2])
```

... then initialize tasks that you want to run ...

Usage of DependentTaskCollection

```
from gc3libs.workflow \
    import DependentTaskCollection

class MyWorkflow(DependentTaskCollection):
    # ...
    def __init__(self, ...):
        DependentTaskCollection.__init__(self)
        app1 = AnApp(...)
        app2 = AnotherApp(...)
        app3 = AThirdApp(...)
        self.add(app1)
        self.add(app2)
        self.add(app3, after=[app1, app2])
```

... then add tasks to the collection, one by one...

Usage of DependentTaskCollection

```
from gc3libs.workflow \
    import DependentTaskCollection

class MyWorkflow(DependentTaskCollection):
    # ...
    def __init__(self, ...):
        DependentTaskCollection.__init__(self)
        app1 = AnApp(...)
        app2 = AnotherApp(...)
        app3 = AThirdApp(...)
        self.add(app1)
        self.add(app2)
        self.add(app3, after=[app1, app2])
```

... specifying dependencies among them.