Symbolic Link Stage

from: DC2 Management -> Persistence Framework .

For DC2, it is desirable to have the directories of input images and templates symbolically linked into the run-specific directory hierarchy. This simplifies the process of comparing inputs and outputs.

The symbolic linking needs to happen based on the exposures that are being processed. It is best for this to occur as events are generated and sent to the pipelines, not when the pipelines and run-specific directory is being created, since that step does not depend on exposures at all. Accordingly, a separate pipeline stage is used to create these links before the first InputStage.

This SymLinkStage is defined in the Python module `lsst.dps.SymLinkStage` (see source).

Behavior

SymLinkStages are configured by a Policy or a PolicyPtr. Currently, this policy must be passed to the constructor (`__init__` method) of the stage, along with the stage's identifier. The policy may contain various items, as documented below.

additionalData

The mechanism used by the persistence framework for passing execution context information to formatters in order to enable generation of things like database table names or column values is the `additionalData` parameter to `persist()` and `retrieve()`. This parameter is a DataProperty. The concept of this DataProperty is reused in the SymLinkStage.

Four keys are predefined in this DataProperty and are set by the SymLinkStage: `sliceId`, the MPI rank of the slice process; `universeSize`, the total number of MPI processes; `runId`, the string identifying the run; and `ccdId`, currently `sliceId` plus one, with leading zero if required.

Other keys may be set with values extracted from the stage's input queue's clipboard. To do this, the policy key "AdditionalData" is defined as an array of strings. Each string has the form "*additionalData-key=clipboard-key*". The *clipboard-key* is used to retrieve an arbitrary value from the clipboard. If the *clipboard-key* contains a dot ('.'), the part preceding the dot is used as the name of a DataProperty on the clipboard; the part after the dot is used as a key within that DataProperty. This is typically used to retrieve values from within events on the clipboard, where the event topic is the name on the clipboard. The *additionalData-key* is the name that the value will have within the `additionalData` DataProperty parameter.

In particular, paths often will include the `exposureId` key from a triggering event, so that key should typically be specified in AdditionalData. Nevertheless, AdditionalData need not be specified.

If the *clipboard-key* is not present on the clipboard, an exception will be raised.

Links

An array of sub-policies named "Links" must be specified. Each sub-policy contains "sourcePath" and "destPath" keys. Each of those keys specifies a string pattern for the relevant path in the symlink command.

These patterns are Python format strings that are filled in using the Python dictionary created for the `additionalData` in the previous section.

Directories in the `destPath` will be created if they do not exist.

RunMode

It is sometimes desirable for symbolic links to occur in the `preprocess()` or `postprocess()` methods of a Stage instead of the normal `process()` method. In particular, this is true if the dataset is to be used in the master pipeline process instead of in the slice processes. Specifying the key "RunMode" with a value of "preprocess" or "postprocess" will cause this to occur. Any other value, or not specifying the RunMode at all, will cause the I/O to happen in `process()`.

Sample Policy

A sample SymLinkStage policy might look like:

```
AdditionalData: "exposureId=triggerVisitEvent.exposureId" "filterName=triggerVisitEvent.filterName"
Links: {
        sourcePath: "/lsst/images/becker/D4/%(filterName)s/output/%(ccdId)s/
%(exposureId)d"
        destPath: "/share/lsst9/DC2root/%(runId)s/ipd/input/%(exposureId)d/%(ccdId)s"
}
```