ASYNC UTILITIES

1. registered on load with configuration
2. async initialization / finalization
3. code invocation

```
- provides: guillotina_cms.interfaces.IWorkflowUtility
  factory: guillotina_cms.utilities.workflow.WorkflowUtility
  settings: {}
```

```
from guillotina.interfaces import IAsyncUtility

class IWorkflowUtility(IAsyncUtility):
    pass

@configure.utility(provides=IWorkflowUtility)
class WorkflowUtility:

    def __init__(self, settings={}, loop=None):
        self.loop = loop

    async def initialize(self, app):

    async def finalize(self, app):
```

```
from guillotina_cms.interfaces import IWorkflowUtility

self.utility = get_utility(IWorkflowUtility)
```

ASYNC UTILITIES

1. registered on load with configuration
2. async initialization / finalization
3. code invocation

```
from guillotina_cms.interfaces import IWorkflowUtility

self.utility = get_utility(IWorkflowUtility)
```

VIEW

```
@configure.service(
    context=IContainer, method='GET',
    permission='guillotina.AccessContent', name='@types/{type_name}',
    summary='Read information on available types',
    responses={
        "200": {
            "description": "Result results on types",
            "schema": {
                "properties": {}
            }
        }
    })
class Read(Service):
    async def __call__(self):
        self.request
        self.context
```