maxims Documentation

Release 0

the maxims authors (see AUTHORS)

December 02, 2013

Contents

Contents:

CHAPTER 1

Items

Items are just commonly reusable Axiom items. They can represent objects such as credentials, endpoints...

1.1 API

Things that can be persisted as Axiom items share a common API. The item has an `instantiate` method taking no arguments, giving you a new, independent instance of the object represented by that item.

```
>>> storedEndpoint = endpoints.ServerEndpoint(description="tcp:0")
>>> storedEndpoint.instantiate()
<twisted.internet.endpoints.TCP4ServerEndpoint object at ...>
```

Alternatively, many items will allow you to use them directly as if they implemented the interface of the object you'd usually have; for example, that same `ServerEndpoint` implements the `listen` method:

```
>>> someFactory = protocol.ServerFactory()
>>> d = storedEndpoint.listen(someFactory)
```

1.1.1 Endpoints

You can persist Twisted server and client endpoints. They are stored and created using their string descriptions. You can create the regular endpoint objects by calling their `instantiate` methods, but the preferred API is to call `listen` or `connect` just as if they were regular endpoints.

Examples

Here's a server endpoint:

Here's a client endpoint:

1.1.2 Credentials

You can persist Twisted Cred credentials.

CHAPTER 2

Creation time logging

You can log the creation time of any item by calling `logCreation` on it. The creation time can later be queried by calling `creationTime`.

Please note that it's pretty annoying to do more complex queries on the creation times of classes annotated this way; it's really only useful for tacking on creation times. If creation time is an important part of the item, it should be added as an attribute separately. You may also want to do both: `creationTime` will remember when the object was first added to the store, but the attribute on the item will remember the time as defined by the business domain.

2.1 Automatic logging for an item class

You can automatically log the creation time of all instances of an item class with the `@creationLogged` decorator. This decorator causes the item instance to have its creation time logged when the `stored` callback fires: that is, when an item is added to the store for the first time.

```
>>> @creation.creationLogged
... class Logged(item.Item):
...     dummyAttribute = attributes.boolean()
>>> logged = Logged(store=store.Store())
>>> creation.creationTime(logged)
extime.Time.fromDatetime(datetime.datetime(...))
```

Doing this when the item is first added to the store is generally what you want, since:

- Items are generally put into a store at creation or very soon after creation
- It's done using a documented callback instead of a hack into `Item`'s internals
- Creation time logging requires item references to work. While references work without a store, there's no obvious way to transport all of the items into a store afterwards.

Note that manual creation time logging works fine with or without stores.

CHAPTER 3

Indices and tables

- *genindex*
- *modindex*
- *search*