Couchdb Plugin

Description

The CouchDB plugin enables lightweight access to database functionality using CouchDB and jcouchdb. This plugin does **NOT** provide domain classes nor dynamic finders like GORM does.

Installation

The current version of **griffon-couchdb** is **0.6.1**
To install just issue the following command

```
griffon install-plugin couchdb
```

Usage

Upon installation the plugin will generate the following artifacts at `$appdir/griffon-app/conf`:

- CouchdbConfig.groovy - contains the database definition properties.
- BootstrapCouchdb.groovy - defines init/destroy hooks for data to be manipulated during app startup/shutdown.

A new dynamic method named `withCouchdb` will be injected into all controllers, giving you access to a `org.jcouchdb.db.Database` object, with which you'll be able to make calls to the database. Remember to make all calls to the database off the EDT otherwise your application may appear unresponsive when doing long computations inside the EDT.
This method is aware of multiple couchdbs. If no couchdbName is specified when calling it then the `default` dataSource will be selected. Here are two example usages, the first queries against the default couchdb while the second queries a couchdb whose name has been configured as 'internal'

Method Signatures

- Object withCouchddb(Closure closure)
- void withCouchddb(RunnableWithArgs runnable)
- Object withCouchddb(String databaseName, Closure closure)
- void withCouchddb(String databaseName, RunnableWithArgs runnable)

Sample usage

```
class SampleController {
    def someAction = {
         withCouchdb { databaseName, db ->
             assert databaseName == 'default'
             // do something with db
         }
    }
}
```

These methods are also accessible to any component through the singleton `griffon.plugins.couchdb.CouchdbConnector`. You can inject these methods to non-artifacts via metaclasses. Simply grab hold of a particular metaclass and call `CouchdbConnector.enhance(metaClassInstance)`.

Configuration

Dynamic method injection

Dynamic methods will be added to controllers by default. You can change this setting by adding a configuration flag in `Config.groovy`

```
griffon.couchdb.injectInto = ["controller", "service"]
```

Multiple DataSources

The config file `CouchdbConfig.groovy` defines a `default` database block. As the name implies this is the database used by default, however you can configure named databases by adding a new config block. For example connecting to a database whose name is 'internal' can be done in this way

```
databases {
    internal {
        host = "localhost"
        port = 5566
        username = "private"
        password = "secret"
        datastore = "myapp"
    }
}
```

This block can be used inside the `environments()` block in the same way as the default database block is used.

Events

The following events will be triggered by this addon

- **CouchdbConnectStart[config, databaseName]** - triggered before connecting to the datastore
- **ConfigureCouchdbJSONParser[config, databaseName, parser]** - triggered before the JSONConfig is built and applied to the datastore
- **CouchdbConnectEnd[databaseName, database]** - triggered after connecting to the datastore
- **CouchdbDisconnectStart[config, databaseName, database]** - triggered before disconnecting from the datastore
- **CouchdbDisconnectEnd[config, databaseName]** - triggered after disconnecting from the datastore

Full source code for this sample application can be found at https://github.com/aalmiray/griffon_sample_apps/tree/master/persistence/couchdb

History

Version	Date	Notes
0.6.1	10-21-11	Release sync with Griffon 0.9.4
0.6	09-30-11	Upgrade to Griffon 0.9.3; multiple datasource support
0.5	01-08-11	Fix GRIFFON-320
0.4.1	11-08-10	Fix a metaclass problem when injecting dynamic methods
0.4	10-27-10	Release sync with Griffon 0.9.1
0.3	08-24-10	Fixed typo as noted in the comments; upgraded to jcouchdb-0.11.0-1
0.2	07-22-10	Release sync with Griffon 0.9
0.1	03-26-10	Initial release