DARWINO

Developer's Guide for Domino

Table of Contents

Darwino User's Guide for Domino Developers

Installing Darwino on a Domino server

Darwino is able to synchronize with a variety of database systems, Domino included. It does this by means of a synchronization service running on the Domino, supported by database adapters that define the field mapping between replicating databases.

There are various support considerations, primarily to do with replication. The darwino.sync service runs on the Domino server to handle inbound replication requests and outbound scheduled replications. It makes Domino appear to the outside world like a Darwino server. It will provide the list of changes since last replication with the requesting server, and then handle the necessary Domino <-> JSON data transformations.

To support this, there is a set of Darwino-enablement plugins (found in the Domino repository in the Darwino space on GutHub. It contains:

- The NAPI, which is the C API bindings
- The replicator code
- Supporting libraries
- Connections for an XPages application to deal with the replicator and Darwino databases generally

This is installed in basically the same way that an application would be installed. In the Domino server’s Update Site database, choose “Import Local Update Site…” and select the site.xml file. Once the import is complete, restart the HTTP task.

Creating and configuring the Synchronization database

In the repository is a darwinosync.ntf file. The Darwino replicator expects there to be a database based on this template, and named darwinosync.nsf, in the root of the Domino data directory. The default title for this database is “Darwino Sync Admin”.

There is no proper Notes UI for this database; it is intended to be maintained via its XPages UI. This database serves several purposes:

- It acts as a repository for the replication history stubs. When Darwino replicates with Domino, a stub document will be created here to record the adapter name, foreign server identity, database, and replication time. These documents are keyed by server name, allowing the database to be replicated among Domino servers without confusion as to the identity of the replicating Domino servers’ identities.
- It holds the replication adapter definitions. Adapters map the incoming Darwino field definitions to Domino fields. Beyond mapping fields one-to-one, it defines the rules for transforming data from one database system to another. For example, JSON lacks the concept of a date, and so dates coming into Domino need to be recognized as such and stored in a Notes DateTime field. It is also possible to do arbitrary transformations.
- It defines the replication schedule used by the Darwino replicator service when replicating out from Domino to Darwino. For outbound replication, the service is controlled by Replication Schedule documents that will be familiar to anyone who ever configured replication in a Domino Connection document.

(We should have a screen shot of a Replication Schedule document, but only after the UI is settled.)

Customizing the data transformation

The database adapters are written in Groovy, using a DSL (Domain Specific Language). It is essentially a set of hooks that are provided to the Groovy environment and that are executed to build the adapter. This allows the adapters to defined in a natural and flexible way. There are several benefits to using Groovy for the adapter definitions:

- Groovy is a straightforward scripting language that runs on top of the Java virtual machine. It was designed from the start to be a friendlier way to write Java, and it lends itself to writing Domain Specific Languages. In this case, everything about the adapter scripting language is focused on the writing of database adapters.
- Groovy's closures, with their names parameters, are ideal for creating easy-to-read, concise, and extensible data definitions.

```
def commonDoc = {
  field “from”, type:NAMES
  field “text”, flags:MULTIPLE
  field “body”, type:RICHTEXT
}
form(“Topic”, commonDoc)
```

There are two ways to deploy the adapters:

- Install them as you would install any other plugin on Domino (Screenshot of a Adapter Definition in Eclipse here)
- Define them in the Synchronization database by creating an Adapter Definition document (Screenshot of a Adapter Definition document here)