GOPHER Evolution

GOPHER has provided a great beginning in automating HTML GUI testing.

Not just for DANY but for any GUI.

It has few compiled dependencies and has a data provider which uses XLS files as a backing store.

The provider delivers arguments to test cases based on rows in an XLS file, one row per test run.

Rows that are being read are also being updated.

This works pretty well for serial test runs but is not thread safe for concurrent runs.

XLS as a data store is limited.

We need to be able to take advantage of relational database features – relations, row/column locking, transactions, queries etc.

So, I think that it is time for GOPHER to experience rebirth as a relational DB driven critter.

This may mean a new approach to driving test suites.

Instead of TestNG driving suites, I am thinking that advanced GOPHER drives the suites.

This might mean first writing the XML files required by TestNG and then using these to drive the suite as usual. I would like to prototype this.

I also want to consider running test cases directly as part of DB driven suites where test cases are running individually but would require more work and I want to maintain as much existing code as possible.

Here are some thoughts –

* GOPHER configuration data is bootstrap information for the framework. Some examples are hub IP and port, nodes IP and port, logging levels and destinations, etc. This store contains one-time parameter values required by the framework. It does not contain test specific data. It should be backed by a properties or XML file.
* Test metadata defines suites, cases and descriptions of the test case arguments. This data is best represented using relations.
* Test expected data is used in comparison with actual element properties obtained from the HTML. This data may be backed by file as long as it never changes per test case, otherwise it should be relational. It should be kept separately from configuration data for clarity.
* Test argument data varies per test run. . This includes target platforms and browsers for the cases, login credentials and form input in general. This data is best represented as relational.

In TestNG the mechanism for providing run time values to a test case involves implementing a data provider. I would like to see if we can have a single data provider backed by MySQL.

This provider must deliver all run-time data necessary for the test case.

Note that this need not be an explicit set of atomic data the way it is currently.

It could be just the minimal information required to start the test run, for example a primary key into the proposed test run table.