The WebApp Custodian

as presented by Sture Andersen in Berlin at the EDUC 2016 conference.

european dataflex users' conference - berlin, germany
EDUC
2016
Dias 1
Præsentationsværtsvisning
Afslut

The WebApp Custodian

by Sture Andersen, Sture ApS

The WebApp Custodian is DataFlex 18.2 webapp that is used to update other DataFlex webapps running on the same server, and to do so via a browser ui.

It offers a convenience over having to establish a remote desktop session and doing an update manually closing it down with WebApp Administrator and copying files over with Windows Explorer.

With WebApp Custodian, you can get very short update cycles. You can update your development server or test server or even a production server - from your development machine within a few seconds.

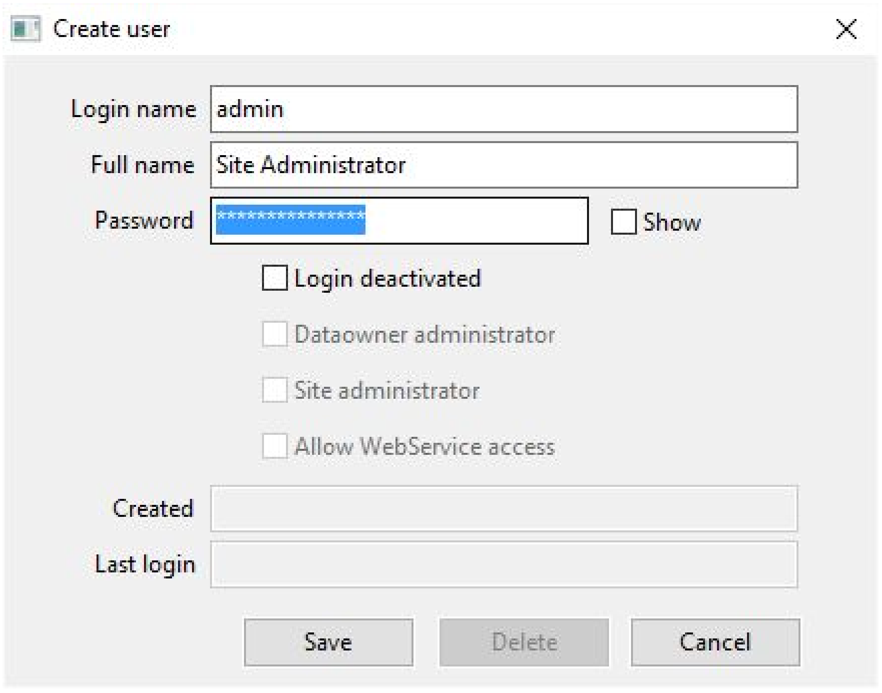
You can NOT use the Custodian to restructure the database as part of an update. If there is a need for that it's right back to the remote desktop strategy. But in reality a great majority of updates do not involve updates to the tables themselves.

If you find this interesting you may proceed to download the Custodian webapp deploy folder at ftp.stureaps.dk:

- Download from ftp.stureaps.dk/software/dataflex/webappcustodian/

- Copy deploy folder to desired location on server
- Reindex tables to server collate (dbbldr.exe)
- Create 1 user with CustodianSetup.exe
- Register the application with WebApp Administrator

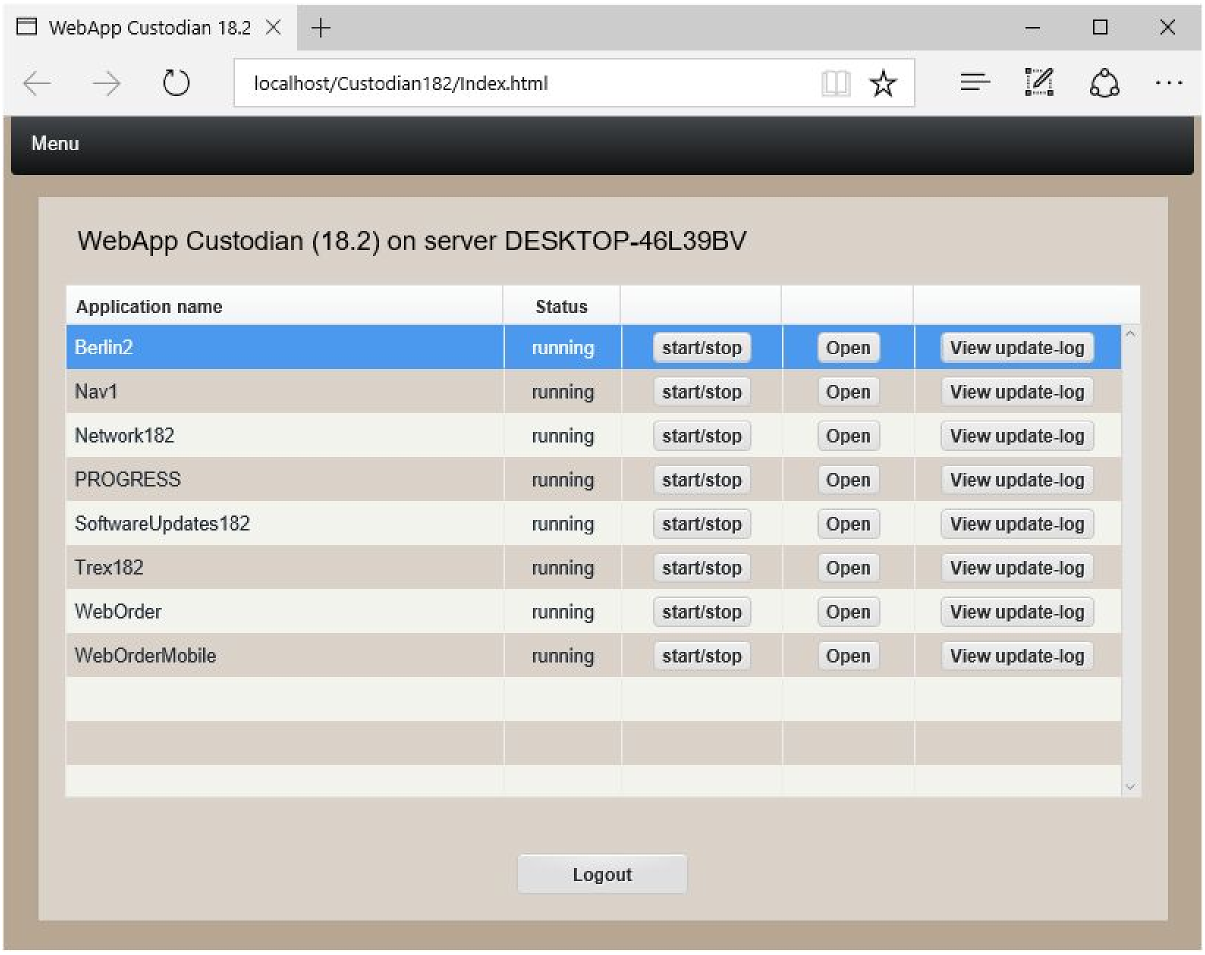
You should create the user with a healthy password (CustodianSetup.exe):


Create user

Login name	admin
Full name	Site Administrator
Password	***************
Created	
Last login	

You may now access the custodian by this url: http://{servername}/Custodian182/ and start administrating the updates of the DataFlex 18.2 applications running on that server.

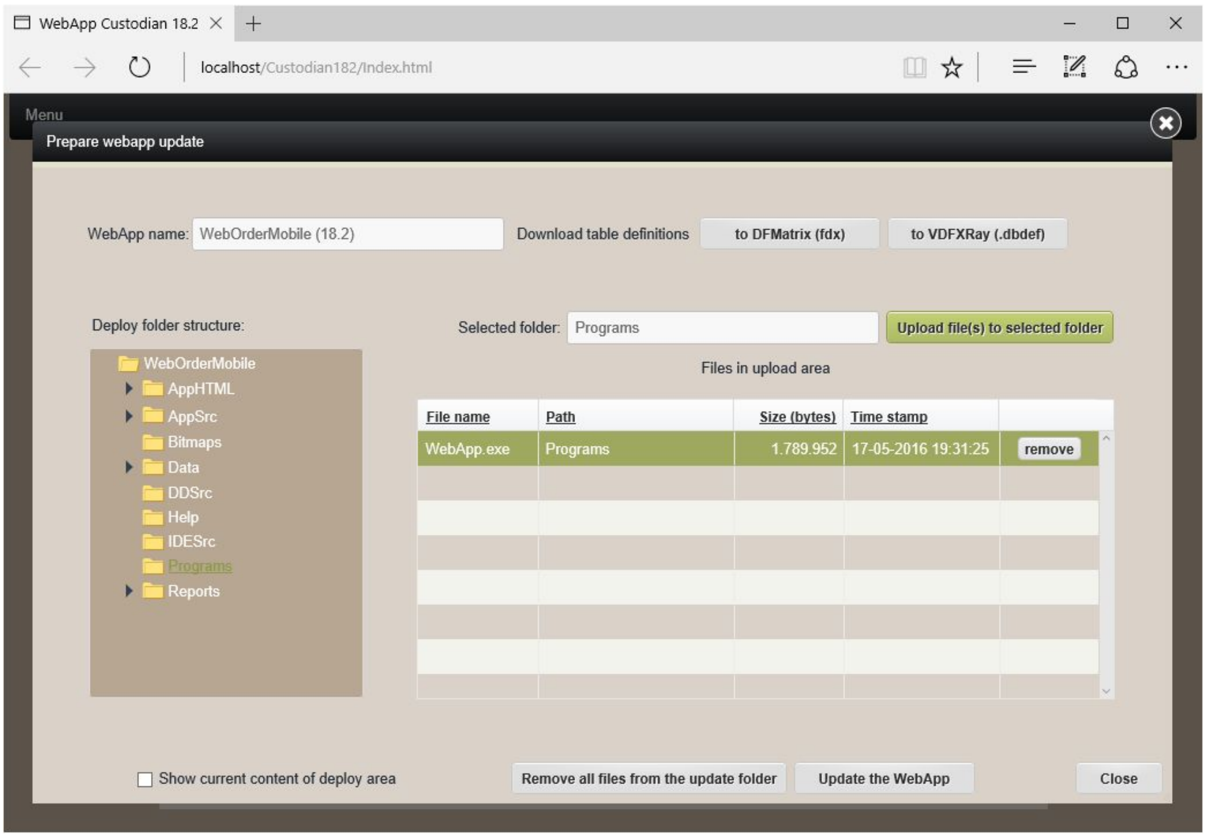
From the opening panel you can start and stop each webapp as in WebApp Administrator.



WebApp Custodian (18.2) on server DESKTOP-46L39BV

Application name	Status			
Berlin2	running	start/stop	Open	View update-log
Nav1	running	start/stop	Open	View update-log
Network182	running	start/stop	Open	View update-log
PROGRESS	running	start/stop	Open	View update-log
SoftwareUpdates182	running	start/stop	Open	View update-log
Trex182	running	start/stop	Open	View update-log
WebOrder	running	start/stop	Open	View update-log
WebOrderMobile	running	start/stop	Open	View update-log

Or you can click the “open” button and get this panel:


WebApp Custodian 18.2

localhost/Custodian182/Index.html

Prepare webapp update

WebApp name: WebOrderMobile (18.2)

Download table definitions: to DFMatrix (fdx) | to VDFXRay (.dbdef)

Deploy folder structure:

- WebOrderMobile
  - AppHTML
  - AppSrc
  - Bitmaps
  - Data
  - DDSrc
  - Help
  - IDESrc
  - Programs
  - Reports

Selected folder: Programs

Upload file(s) to selected folder

Files in upload area

File name	Path	Size (bytes)	Time stamp	
WebApp.exe	Programs	1.789.952	17-05-2016 19:31:25	remove

Show current content of deploy area

Remove all files from the update folder | Update the WebApp | Close

This is a screenshot from a development machine so there are more folders present than one would expect from a deployed app (which would be something like *apphtml*, *data* and *programs)*.

The idea is that you select a folder to the left and then upload files to that folder by clicking the button labelled *Upload file(s) to selected folder*. In the screenshot a new *webapp.exe* has been uploaded to the *programs* folder.

Once the necessary number of updated files have been uploaded to the Custodian you may click the *Update the WebApp* button.

You will be prompted for the reason for the update and then ...

In a split second the Custodian will take down the *WebOrderMobile* application (in this case), copy over the files you have uploaded and take the application back online.

> **Note:** For each update performed on an application a backup folder is created containing the files that were overwritten. This folder may be copied back to the deploy folder to fall back one version. Windows Explorer will ask you to confirm each file of the fallback since it replaces a file already there.
>
> The Custodian creates subfolder of your deployed applications folder called *CustodianRepository.* The undo folders are found in there. They have impossible names so they should be sorted and identified by creation date-time.

Down to the left of the panel there is a checkbox that can be used to change the operating mode. If checked, it displays files currently present in the actual deploy area. The files can then be downloaded. However, if you want to download a .dat file, you should probably stop the application first.

A Consideration

There is an issue that is related to updating webapps in general, not just with the Custodian. And that is related to how any active sessions will react to the updated server system.

You may ignore this issue during test and development, but for a production system, it should at least be considered.

When a new program version is put into production, sessions that are already active will become “out of synch” with the server, in a sense. At least some of the knowledge that the javascript application got from the server when the session loaded, may no longer be accurate. The javascript application simply carries on to run on outdated premises.

The programmer may have changed the name of a web-object. The browser app will become very confused if can no longer “talk” the server object. Web-properties could have different meanings etc.

For the application and its data to remain in synch we must force all active sessions to reload when the webapp.exe is updated.

> **Note.** The same sort of consideration could be applied to session-data stored on the server, for example in the *WebAppSession.dat* record belonging to the session. This implies that we should not only force the browser to reload. We should also invalidate the current session record and thus force a new login.

While this applies no matter how you update your web-application, the issue is accentuated when using the Custodian because the update only takes the system off-line for a very very short time. Sessions are highly unlikely to notice.

(If they “notice”, the user will just get one of those http 500 errors and upon refresh he’s good again because his browser reloads the app, once it is available again.)

A scent of a Strategy for handling

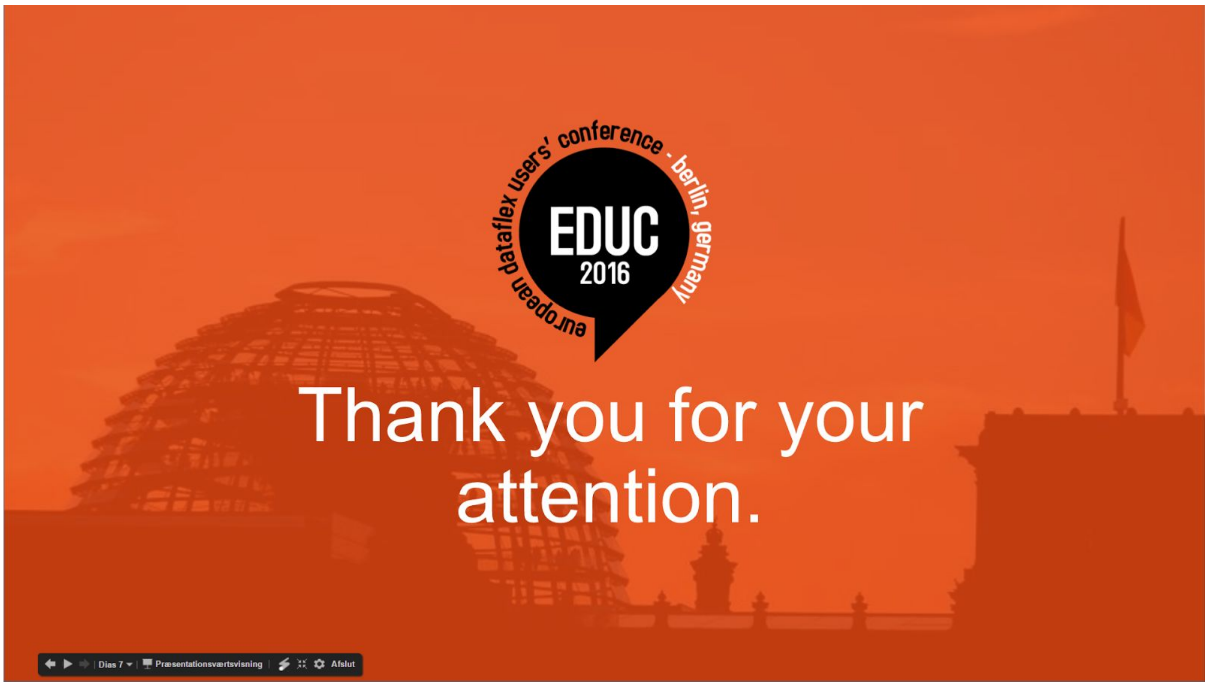
If we want to handle this situation gracefully, we must stamp each new session with the timestamp of the webapp.exe file that created it. *WebAppSession.WebappCompileTimeUTC* is used for that purpose in the code below.

Then, in the *ValidateSession* function of the *ghoWebSessionManager* object we must have code that looks something like this included:

```
If (not(IsDebuggerPresent())) Begin // Not while working in the Studio!
    If (FindCurrentWebAppRecord(ghoWebApp)) Begin // Think of this as a system table
                 // (the webapp has stored its own exe-file creation time in that table)
        // Check if the webapp has been updated:
        If (awWebApp.WebappCompileTimeUTC<>WebAppSession.WebappCompileTimeUTC) Begin
            Reread WebAppSession
                Move "N" to WebAppSession.Active // Close the session
                SaveRecord WebAppSession
            Unlock
            Error 950 "Your application has been updated. The browser app will reload."
            // refresh the WebApp at the client (because we closed the session
            // this triggers a login)
            Send NavigateRefresh of ghoWebApp
            Function_Return False
        End
    End
    Else Begin
        Error 950 "The application has not self-registered for automatic updates."
        // refresh the WebApp at the client (triggers a login)
        Send NavigateRefresh of ghoWebApp
        Function_Return False
    End
End
```

The first line makes sure that the code does not trigger when run under the debugger. In that case we most likely have just compiled our webapp, and we do not want the browser to trigger an extra reload every time we do that.

And this finishes the resume of the EDUC presentation. I hope it brought you something.



EDUC 2016

european dataflex users' conference - berlin, germany

Thank you for your attention.

Sture Andersen
May 2016