

Context and Objectives

In an Oracle APEX application, there is possibility to enter Help text at development stage. The help text may be used to provide page level context sensitive help.
In almost every component, there is a specific property (`Help Text`) aimed to receive the text.

Usually, the developer is responsible for this task and that has drawbacks :

- The text can only be provided in the Developer interface.
- There is no rich text editor support at this level
- The developer himself is not, most of time, the best qualified person to handle this job

Given these above considerations:

- We need a more confortable tool for authoring Help Text
- It’s better to delegate this job to a writer who will have better user sensitivity.
- the text needs to be continually improved as it is used, without doing back and forth in the development console.

Let’s consider a possible solution to achieve our goal.

In this paper, we’ll focus on Help Text at page level only and we’ll not consider the other level of component (items) neither `Inline Help Text`.

Global approach

One solution is to add an application table wich contains specific help text. The authors will need a regular access to application in order to modify the help text through a dedicated form page.

In standard, the help texts are accessible through an APEX view: `APEX_APPLICATION_PAGES`

Access key is : APPLICATION_ID + PAGE_ID
text: HELP_TEXT

We build a new table which contains customized help texts. Let's say : PLA_HELP

```
CREATE TABLE PLA_HELP
( APP_ID NUMBER,
  PAGE_ID NUMBER,
  HELP_TEXT CLOB,
  CREATED_BY VARCHAR2(50),
  CREATED_ON DATE,
  CONSTRAINT PLA_HELP_PK PRIMARY KEY (APP_ID,PAGE_ID)
  USING INDEX ENABLE
)
```

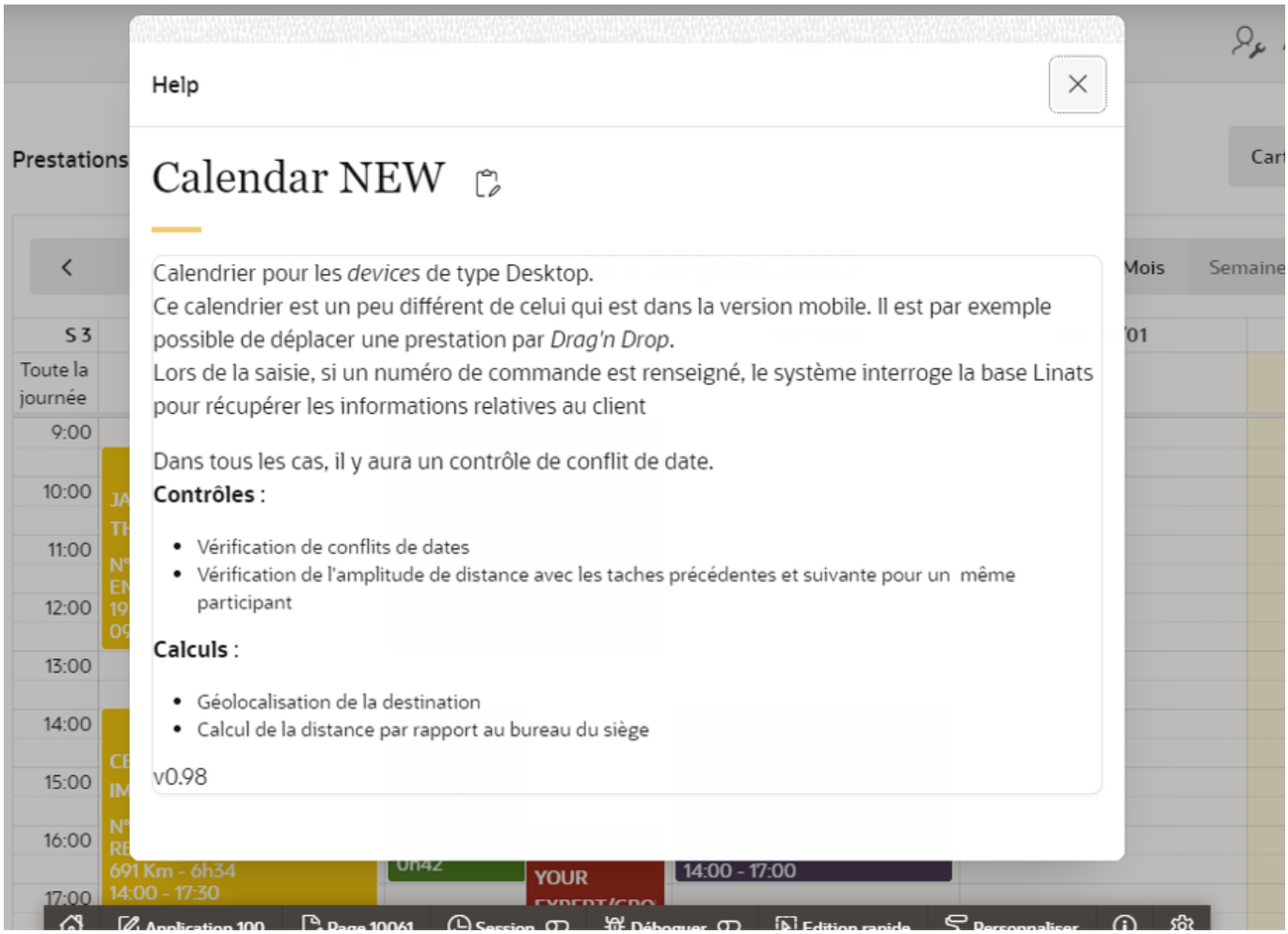
We build a view `PLA_HELP_V` that superimposes help text derived from repository and help text coming from our specific table.
Priority is given to Customized Help Text.

```
CREATE OR REPLACE FORCE EDITIONABLE VIEW PLA_HELP_V (APP_ID, PAGE_ID,
CREATED_BY, CREATED_ON, PAGE_TITLE, HELP_TEXT, IS_FACTORY) AS
```

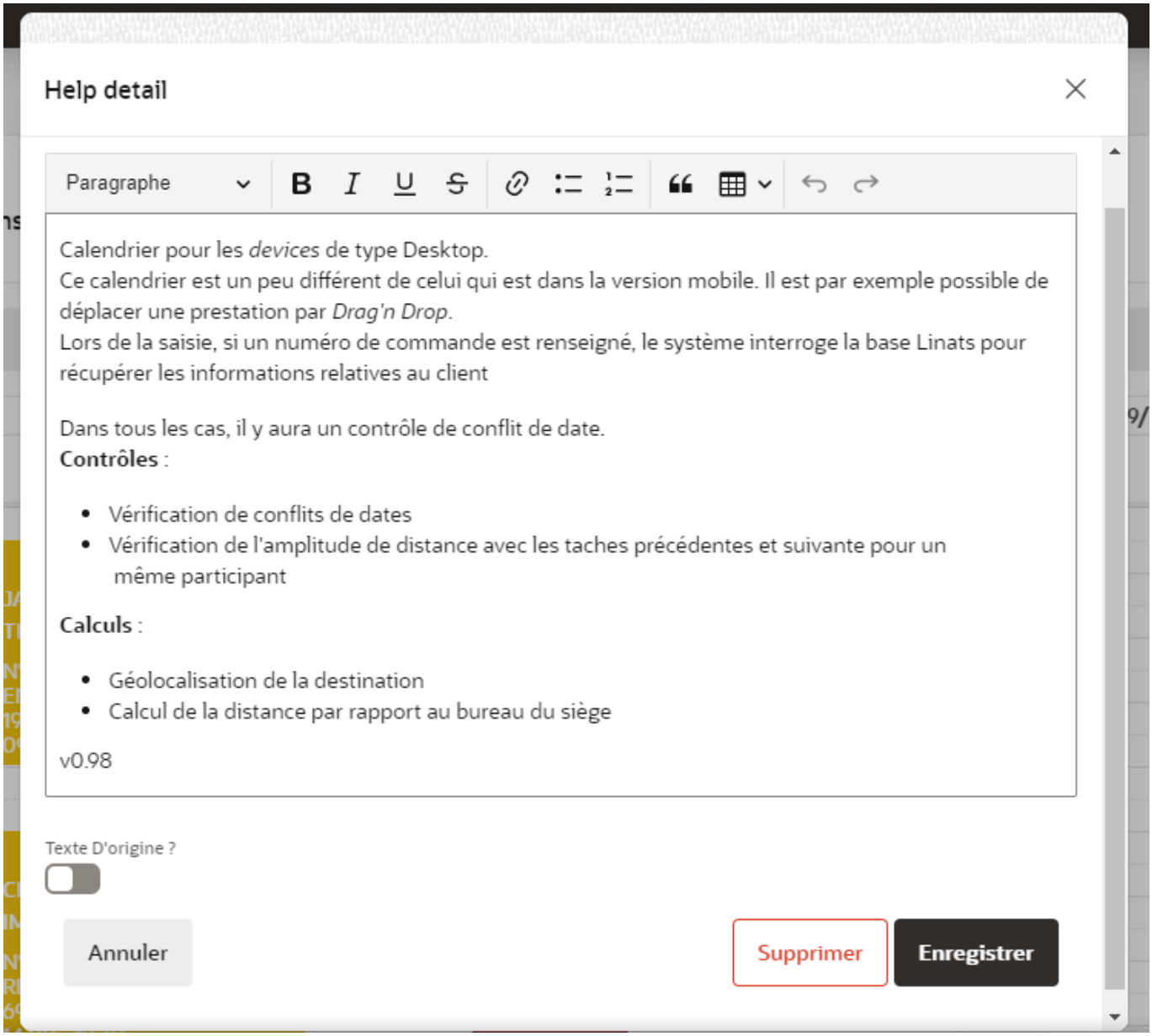



```
select
  p.APPLICATION_ID,
  p.PAGE_ID,
  h.CREATED_BY,
  h.CREATED_ON,
  p.PAGE_TITLE,
  case when h.HELP_TEXT is null
     then p.HELP_TEXT
     else h.HELP_TEXT
  end HELP_TEXT,
  case when h.HELP_TEXT is null
     then 'Y'
     else 'N'
  end IS_FACTORY
from APEX_APPLICATION_PAGES p,
     PLA_HELP h
where p.APPLICATION_ID = h.APP_ID(+)
and p.PAGE_ID = h.PAGE_ID(+)
```

Displaying help text needs a specific page dedicated to that.
When we launch the wizzard, the page and navigation icons are automatically created if a page at least is added, but it’s not the case otherwise.

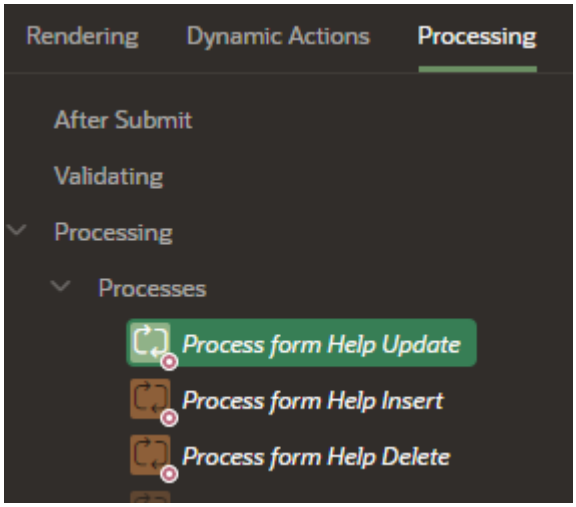


Help about a page. A edit icon allows to navigate to edit mode for authorized users.



Edit Help text page.

My first choice has been to create a I*nstead of Trigger* but this way reached an obstacle: I got an -1031 error *insufficient privileges*, because of presence of a system APEX view. I could workaround by granting some privileges directly to user, but I gave this approach up and I put all the code in the APEX form.



In the `process section`, we remove the default form DML process and replace it by three processes, each dedicated to an action (Update, insert or delete)

ie: the code for update below:

```
BEGIN
 update PLA_HELP
     set
       HELP_TEXT = :P35_HELP_TEXT,
       CREATED_ON = :P35_CREATED_ON,
       CREATED_BY  = :P35_CREATED_BY
     where APP_ID = :P35_APP_ID and
           PAGE_ID = :P35_PAGE_ID;
 COMMIT;
END;
```

Conclusion

The new system provides a more flexible solution for providing accurate information to users.

We could imagine to refine our help system by adding multiple levels of reading, depending the user profile.
This solution can be improved, as well, by merging additional infos provided later by developper (because of fixing bugs or adding features) with an already existing customized text.

Author

- 

  Patrick

  GPM Factory