The propbox package
Boxes with Properties[*]

Sean Allred[†]

Released 9999/12/31

This is some cool stuff. The basic idea is this: have a sequence of boxes that you can use (all with generated names using `\newsavebox`) and then have their properties stored as an additional token list of nearly the same name. Then, when it comes time to retrieve the boxes, just iterate through the sequence and pick some that satisfy a given filter.

1 Properties

1.1 Defining New Properties

`\DeclareBoxProperties`
Updated: 2013/08/11

`\DeclareBoxProperties {⟨properties⟩}`

Declares the set of box properties as a comma-separated list.

`\NewBoxProperty`
Updated: 2013/08/11

`\NewBoxProperty {⟨property⟩}`

Adds a *single* key to the property list.

`\RemoveBoxProperty`
Updated: 2013/08/11

`\RemoveBoxProperty {⟨property⟩}`

Removes a *single* key from the property list.

`\ClearBoxProperties`
Updated: 2013/08/11

`\ClearBoxProperties`

Removes *all* keys from the property list.

[*]This file describes v999, last revised 9999/12/31.
[†]E-mail: seallred@smcm.edu

2 propbox implementation

```
⟨*initex | package⟩
⟨@@=propbox⟩
⟨*package⟩
\ProvidesExplPackage
  {\ExplFileName}{\ExplFileDate}{\ExplFileVersion}{\ExplFileDescription}
\__expl_package_check:
⟨/package⟩
```

2.1 Debugging

`\__propbox_debug_mode_bool` This controls almost all log output, and will certainly clutter your log file if set. When set, propbox will show the contents of internal sequences and keys throughout the compile. It is probably hard to follow, so the usefulness of macros dependendent on this remains dubious.

```
\bool_new:N \__propbox_debug_mode_bool
```

(*End definition for* `\__propbox_debug_mode_bool`. *This variable is documented on page* **??**.)

2.2 Boxes

`\__propbox_boxes_seq` Stores all of the boxes by name. Remember that `\newsavebox` needs two parameters; it needs a single ⟨*cs name*⟩ and its contents. It is this ⟨*cs name*⟩ that we are storing into this sequence.

```
\seq_new:N \__propbox_boxes_seq
```

(*End definition for* `\__propbox_boxes_seq`. *This variable is documented on page* **??**.)

2.3 Properties

`\__propbox_properties_clist` This list contains all of the properties that are used for boxes, along with their default values. Thus, it is a bona-fide l3keys specification. Since each property is *supposed* to be a rather simple identifier, a comma-separated list seemed appropriate to store this set.

```
\clist_new:N \__propbox_properties_clist
```

(*End definition for* `\__propbox_properties_clist`. *This variable is documented on page* **??**.)

`\__propbox_property_parse` Parses a `key=value` token list into `key` and `token`, storing them into `\l_tmpa_tl` and `\l_tmpb_tl` respectively.

```
\def\propbox_property_parse#1=#2{
  \tl_set:Nn \l_tmpa_tl { #1 }
  \tl_set:Nn \l_tmpb_tl { #2 }
}
```

(*End definition for* `\__propbox_property_parse`. *This function is documented on page* **??**.)

`\propbox_new_box:nn` When creating a new box, care must be taken so that the new box and its properties are somehow associated with each other. This function does so by generating a name for the box (whose content is given as `#1`) and then using this generated name (say, `\gn`) to store the property list, `#2`, as a token list in `\gn_properties`. Thus, when retrieving the properties for a box, one only needs to append `_properties` to the control sequence to retrive that box's properties.

```
\cs_new_protected:Npn { \g_propbox_new_box } #1 #2
 {
```

Store the generated name into a local temporary token list. (It gets very tedious, not to mention inefficient, to type out all the time.)

```
  \tl_set:Nn \l_tmpa_tl { propbox_box_number_\seq_count:N \__propbox_boxes_seq }
```

Store the properties for this box in its own token list. This way, the keys can be set easily and simply to determine whether or not to select the box.

```
  \tl_new:cn { \l_tmpa_tl _properties } { #1 }
```

Create a new save box of the name `\propbox_box_number_N` where N is the current number of items in the collection. ($N \in \{0\} \cup \mathbf{N}$.) The `minipage` environment is used here to force page-wise sequential output only; this package grew out of a question on the TeX Stack Exchange site, and this seemed appropriate at the time.

```
  \newsavebox { \use:c { \l_tmpa_tl } }
  \savebox { \use:c { \l_tmpa_tl } } {
    \begin{minipage}{\linewidth}
      #2
    \end{minipage}
  }
```

Put the handle for the box into the collection of boxes.

```
  \seq_push:Nv { \__propbox_boxes_seq }
               { \use:c { \l_tmpa_tl } }

 }
```

(*End definition for* `\propbox_new_box:nn`. *This function is documented on page* **??**.)

2.4 Debugging Utilities

`\__propbox_debug_mode_bool` Create a toggle to turn on/off debug mode.

```
\bool_new:c { propbox_debug_mode_bool }
```

(*End definition for* `\__propbox_debug_mode_bool`. *This variable is documented on page* **??**.)

`\propbox_debug_msg:n` If `\propbox_debug_mode` is set, then `\typeout` the argument, `#1`.

```
\cs_new:Npn \propbox_debug_msg:n #1 {
  \bool_if:NTF \__propbox_debug_mode_bool
               { \typeout{#1} }
               { } }
```

(*End definition for* `\propbox_debug_msg:n`. *This function is documented on page* **??**.)

⟨/initex | package⟩