You may have read my previous ramblings about the theme of handling keyboard shortcuts im Zimbra Zimlets. To sum up: it's not easy. But that's not actually a Zimbra bug. It's more of an architectural problem. You can imagine Zimbra's shortcut handling as a set of layers. There is a default key handler. That is the global key handler contained in the object „ZmZimbraMail“. It would be nice for this handler to support additional shortcuts, but it just can't as of today. If you do something like an overlay application you can push another default handler on top of the default key handler and if you're done with the overlay, you have to „pop“ (remove) the new default key handler, so that ZmZimbraMail becomes the default key handler again.

On top of this default key handling come controls. The control that is currently active is searched first for a matching key handler. If there isn't found anything, the control's parent is asked and so on. If none of the controls handle the key, the current default handler is asked. The only possibility to leverage this system would be to somehow extend Zimbra's default key handler, but that currently doesn't work (or is way to complicated and error-prone).

But there is another way. You can use Zimbra's event handling system and just add another handler for the „KEYUP“ event. That way you will be noticed, once a key is pressed.

Hmm… Zimbra's key handling (espescially the I18N-part) is quite cool, eh?

Well, some parts can be used and I have created an universal shortcut handler, that simply can be plugged into your zimlet. You can get it in my [[https://github.com/dploeger/attic/raw/master/de_dieploegers_shortcut/de_dieploegers_shortcutHandler.js|github attic repository]].

To use it, follow these steps:

First, include this script into your zimlet's xml definition file:

de_dieploegers_shortcutHandler.js

Add shortcut definitions to your properties file consisting of a prefix, an actionCode and three keys:

* „description“ – Description of the key code (currently not used)
* „display“ – User-friendly display of the key code (currently not used)
* „keycode“ – Keycode of the shortcut. Consists of an optional meta-key label („Shift“, „Alt“, „Ctrl“ or „Meta“) and an ASCII-keycode, separated by „+“. So a key „a“ with a hold „Shift“-

Key would be „Shift+65“

A complete definition could be:

```
SHORTCUT.DO_SOMETHING.description = Do something interesting
SHORTCUT.DO_SOMETHING.display = Shift + a
SHORTCUT.DO_SOMETHING.keycode = Shift+65
```

Please note, that the meta-key label is translated. So for a german properties file for example, use „Strg“ instead of „Ctrl“.

Add a „getKeyMapName“ method to your zimlet's handler prototype returning the prefix of the shortcut definitions in your properties file.

For example:

```
my_cool_zimlethandler.prototype.getKeyMapName = function () {
return „SHORTCUT“;
}
```

Finally, In your zimlet's handler init-method, create an object of this class and specify your zimlet's I18N object. It's typically identical to the name of your zimlet. (for example my_cool_zimlet) Additionally, specify a callback handler (created from AjxCallback), that is responsible to carry out the action, if certain keys are pressed. The callback handler is called with the actioncode as it's only parameter. It is only called, if an actioncode is found for the pressed key sequence.

The script has limitations, though. Currently it doesn't support multiple shortcut sequences, like Zimbra uses for marking messages read for example (first press „m“, after that press „r“). But I accept pull requests.