Eclipse Key Bindings

Patrik Suzzi

Extend Eclipse Key Binding

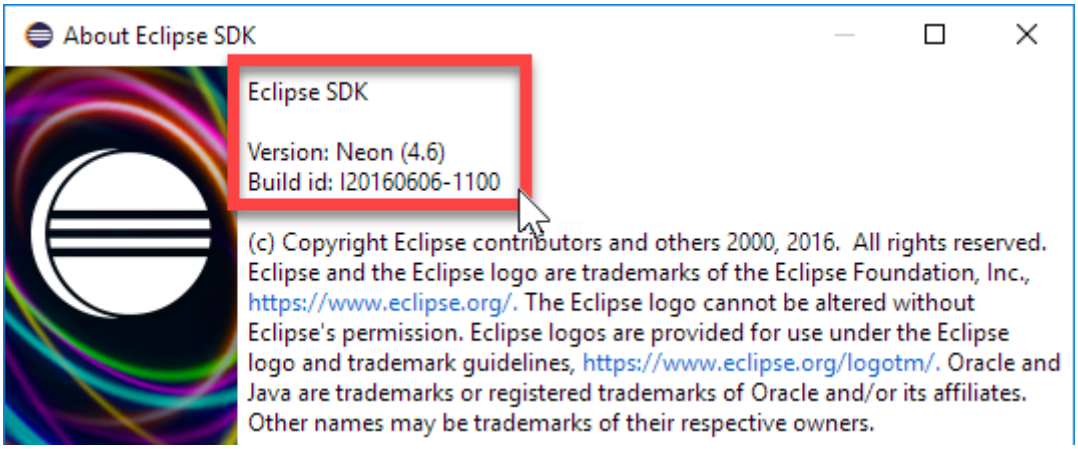
Eclipse Key Bindings

Case study: Bug 496319.

Reuse `Ctrl+C` key binding in About Dialog, to copy the build ID.

Override Key Binding in Child Context

Reuse `Ctrl+C` to implement a specialized command in Eclipse. The command will copy part of the build information visible in the about dialog. The `Ctrl+C` binding will override the default copy command, using a *Context* that is working only in the *About Dialog*.



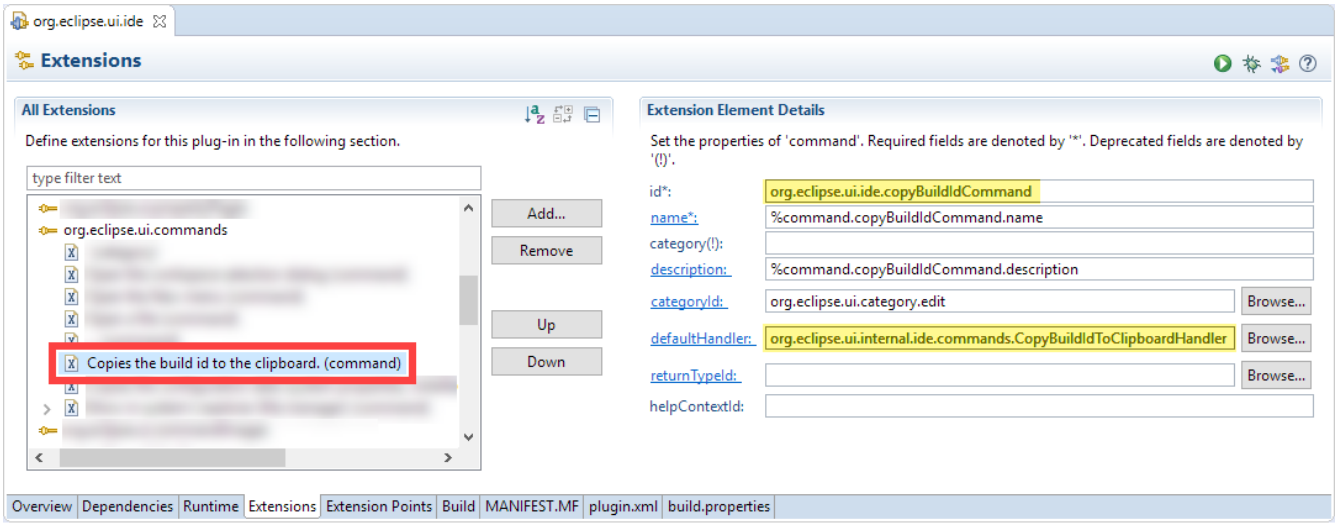
These are the steps to implement the solution:

1. Define a command: add or find the declarative command to be invoked on KeyPress
2. Add an handler: implement the copy build id operation as requested
3. Add a binding in its context: The Ctrl+C binding will be in a specific context, to avoid overlap with default `Ctrl+C`.
4. Activate context programmatically: In lecacy dialog, we need to activate the context programmatically.

Command - Copy Build Id

Eclipse already defines the "copy build id" command. So we take note of

- command id: `org.eclipse.ui.ide.copyBuildIdCommand`
- handler class: `CopyBuildIdToClipboardHandler`



Handler, copy Build Info

The existing handler needs to be modified, to copy the build information provided in the about dialog. The relevant code, within in `CopyBuildIdToClipboardHandler#execute(.)`, is below.

```
  // get the product
  final IProduct product = Platform.getProduct();
  // get about text and split lines
  String aboutText = ProductProperties.getAboutText(product);
  String lines[] = aboutText.split("\\r?\\n"); //$NON-NLS-1$

  // format interesting lines only
  String toCopy = String.format("%s%n%s%n%s", lines[0], lines[2], lines[3]); //$NON-
NLS-1$

  // copy to clipboard
  Clipboard clipboard = new Clipboard(null);
  try {
    TextTransfer textTransfer = TextTransfer.getInstance();
    Transfer[] transfers = new Transfer[] { textTransfer };
    Object[] data = new Object[] { toCopy };
    clipboard.setContents(data, transfers);
  } finally {
    clipboard.dispose();
  }
```

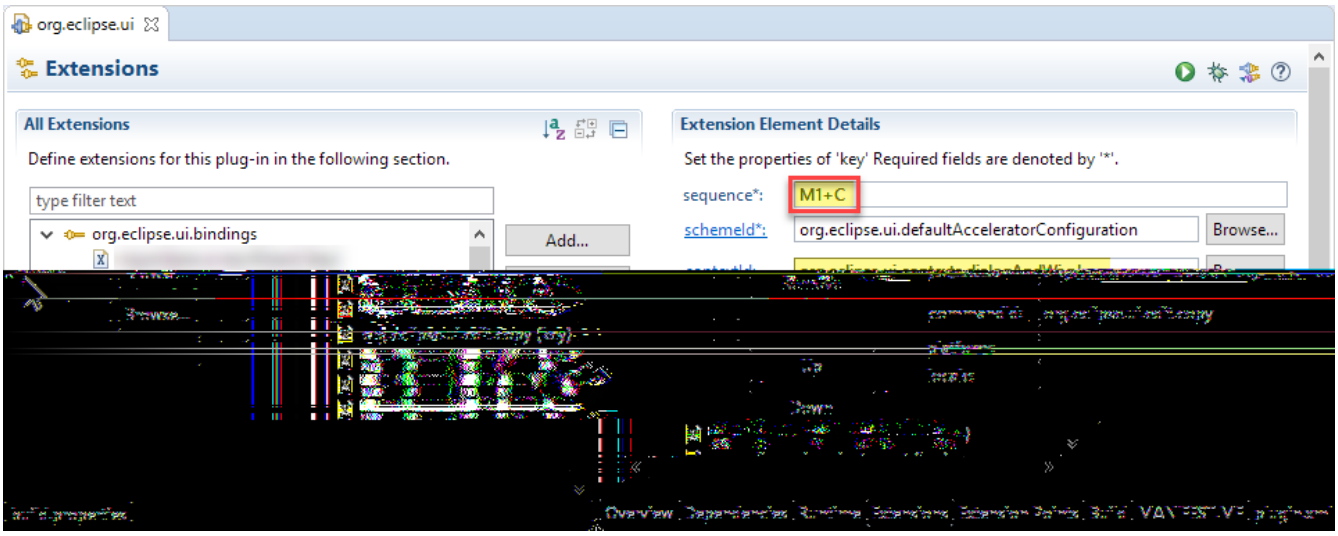
Now the copy build id command provides all the needed information.

Add Binding

Now, we want to execute the `copyBuildIdCommand` when the user has `AboutDialog` active, and presses `Ctrl+C`.

Problem: The `M1+C` Key Binding is already defined, to execute the **Copy** *Command*, in the

`org.eclipse.ui.contexts.dialogAndWindow` context.



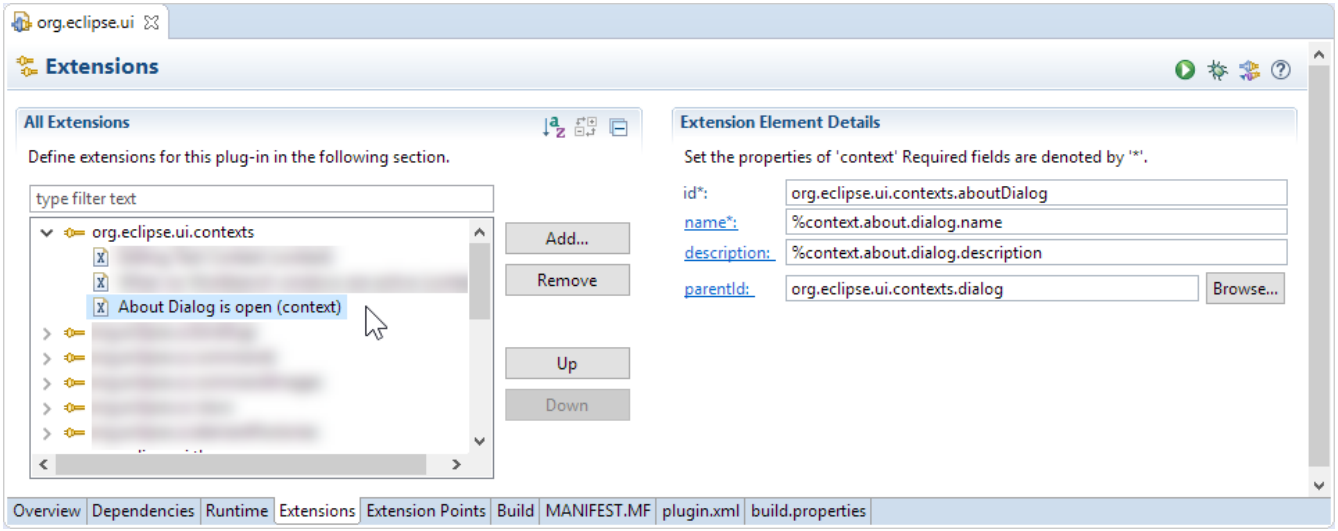
Solution: To execute a different command with the same Key Binding, we need to:

- define a new context for About Dialog
- add the binding with the new context
- activate the new context

Add new context for About Dialog

Under the `org.eclipse.ui.contexts` node, add the new context with

- *id*: `org.eclipse.ui.contexts.aboutDialog`;
- *parentId*: `org.eclipse.ui.contexts.dialog` (default context for dialogs).

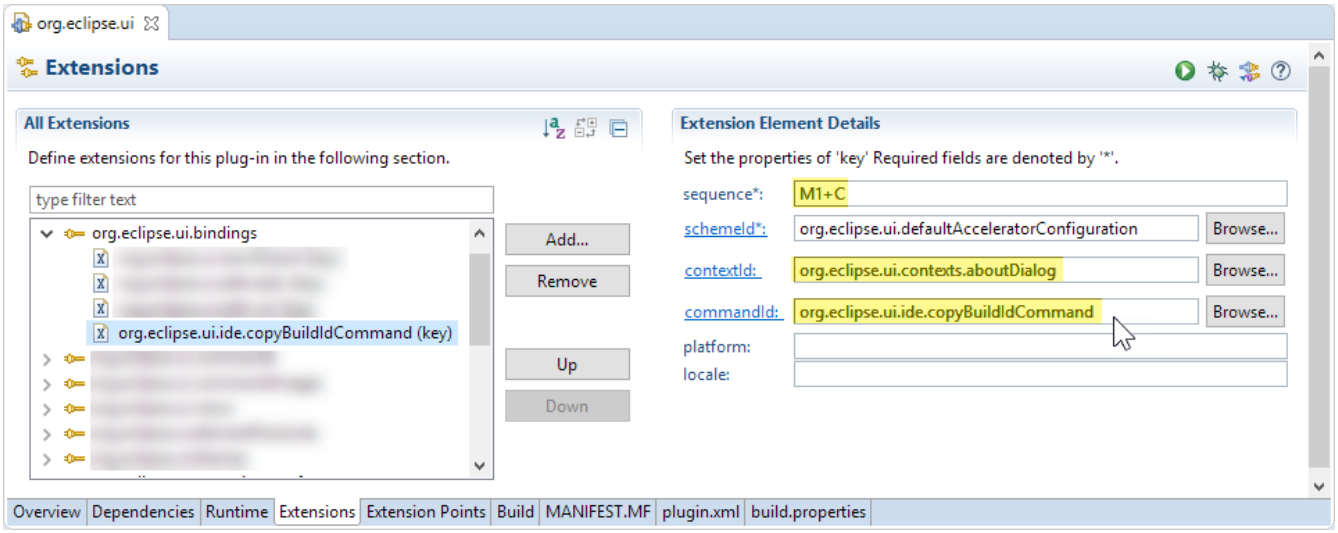


Add new Key Binding for about context

Add a new **Key** under the `org.eclipse.ui.bindings`, with

- *sequence*: `M1+C`, a keybinding corresponding to `Ctrl+C`.
- *contextId*: `org.eclipse.ui.contexts.aboutDialog`, the new context defined

- *sequence*: `org.eclipse.ui.ide.copyBuildIdCommand`, the command to be used



Activate the context

Usually, in E4 Applications, we activate context by associating them to dialog or windows, using the E4 Model editor. However, in this case we have a *Legacy Dialog*, and we need to proceed differently.

Activate new context programmatically

Now each aspect is defined, except the activation of the context. The `AboutDialog` is a *Legacy Dialog*, not defined in an E4 model. So, we can not associate the context to the dialog through the E4 model editor. We need indeed a *programmatical* activation for the context.

The relevant code to activate the dialog programmatically, located in `AboutDialog`, is visible below:

```
  /** Id for the context associated to this dialog */
  private static final String ID_CONTEXT = "org.eclipse.ui.contexts.aboutDialog";
//$NON-NLS-1$

  // represents the activated context
  private IContextActivation contextActivation;

  @Override
    protected void configureShell(Shell newShell) {
        super.configureShell(newShell);
    // ...

    // gets an instance of the context service
        final IContextService contextService = PlatformUI.getWorkbench().getService
(IContextService.class);

        // Listens to activate/deactivate events, setting context id accordingly
        final Listener listener = e -> {
            if (SWT.Activate == e.type) {
                // activate context
                contextActivation = contextService.activateContext(ID_CONTEXT);
            } else if (SWT.Deactivate == e.type) {
                // deactivate context
                contextService.deactivateContext(contextActivation);
            }
        };

        newShell.addListener(SWT.Activate, listener);
        newShell.addListener(SWT.Deactivate, listener);
        newShell.addListener(SWT.Dispose, e -> {
            // deactivate context and remove listeners
            contextService.deactivateContext(contextActivation);
            newShell.removeListener(SWT.Activate, listener);
            newShell.removeListener(SWT.Deactivate, listener);
        });
    }
```

Notes

- Listening to `SWT.Activate` / `SWT.Deactivate` is needed to deactivate the context when new dialogs are opened and in foreground w.r.t. the about dialog
- Listening to `SWT.Dispose` is needed to deactivate the context before close and remove listeners.

References

To implement the solution I observed how the EGit overrides the `M1+C` in some contexts, and I read some references, that you can see below:

Links on the web

- dhemery.com, Key Bindings In Eclipse/RCP Applications (2008)
- wiki.eclipse, FAQ How do I provide a keyboard shortcut for my action?

Related Bugs

- Bug 496319 - Add Ctrl+C to About box to copy the build ID

Related keywords

- "binding infrastructure" eclipse

Written with AsciiDoc. See ref, also man.