User Preferences

Genuine leverages the Preferences API provided by Java by adding infrastructure to easily set up a preferences dialog with individual preference pages. This infrastrucure contains of a preferences module and a preferences service.

The preferences dialog and the preferences service are set up only if the preferences module is added to the application configuration data. Here is an example of how this module may be configured:

```
<module id="PreferencesModule"
class="net.sf.genuine.preferences.PreferencesModule">
  <properties>
    <property key="PreferencesRoot"
value="net/sf/genuine/example/crm/"/>
    <propertyset key="PreferencesPages">
      <propertysetvalue
value="net.sf.genuine.preferences.general.AppearancePreferencesPage"/>
    </propertyset>
  </properties>
  <menubar>
    <menu id="edit.menu">
      <menuitem id="preferences" separator="infront"/>
    </menu>
  </menubar>
  <services>
    <service interface="net.sf.genuine.preferences.PreferencesService"/>
  </services>
  <visualcomponents>
    <visualcomponent id="preferences.component" modal="true">
      <size width="600" height="400"/>
    </visualcomponent>
  </visualcomponents>
</module>
```

You may customize this configuration data by changing the position of the menu item and by changing the size of the preferences dialog. Two properties are shown; both of them are mandatory:

- `PreferencesRoot`: The root `java.util.prefs.Preferences` node for the application.
- `PreferencesPages`: A property set that has to contain the fully expanded class names of all preferences pages that should be shown on the preferences dialog.

All preferences pages have to inherit from the abstract class `PreferencesPage`. This class is based on `JPanel` so that you may create any kind of GUI to visualize your preferences. You should create a preferences page for each set of related preferences. The

preferences dialog shows all defined preferences pages and allows the user to switch between them. Each preferences page is responsible to restore its current preferences settings and store any changes into a given `Preferences` object.

A preferences page is identified by a path, for example "general/appearance". Genuine adds this identification path to the root node path given in the configuration data to determine the `Preferences` node that corresponds to a preferences page.

To check a preference inside your code, you should use the preferences service to fetch the appropriate `Preferences` instance. Because you cannot get a reference to a `PreferencesPage` object, you might provide static methods in your classes as in this example:

```
/**
 * The identification path for this preferences page. Use it to get the
 * preferences object from outside this class using the
PreferencesService.
 */
public static final String IDENTIFICATION_PATH = "general/appearance";

/**
 * Key for the preference setting that controls whether text labels are
shown
 * in the tool bar.
 */
 public static final String PREF_TOOLBAR_BUTTONS = "toolbarButtons";

 public static final int PREF_TOOLBAR_BUTTONS_ICONS = 0;
 public static final int PREF_TOOLBAR_BUTTONS_TEXT = 1;
 public static final int PREF_TOOLBAR_BUTTONS_ICONSTEXT = 2;

 /**
  * Returns the currently set preference for toolbar buttons by querying
the
  * preferences service as long as that service has been created and
registered by
  * the preferences module. Otherwise, the default preference setting is
returned.
  *
  * @return A integer that encodes the preference. See the constants of
this class.
  */
public static int getToolbarButtonsPreference(ServiceBroker
serviceBroker) {

  PreferencesService service = (PreferencesService)
    serviceBroker.getService(PreferencesService.class);
  if (service == null) {
    return PREF_TOOLBAR_BUTTONS_ICONS;
  }

  Preferences preferences = service.getPreferences(IDENTIFICATION_PATH);
  int result = preferences.getInt(PREF_TOOLBAR_BUTTONS,
PREF_TOOLBAR_BUTTONS_ICONS);

  return result;
```

```
}
```

Genuine supplies some default preferences pages that may be added to every application. These classes may serve as examples of how preferences should be stored and restored by a preferences page. These preferences pages are:

- General appearance preferences: `net.sf.genuine.preferences.general.AppearancePreferencesPage`