The Exit Service

Genuine provides an internal service that handles the termination of an application. If you do nothing special, this service asks the user for confirmation when the user tries to exit the application.

You may may provide your own service to implement a different behavior. The service is defined by the interface `net.sf.genuine.services.ExitService`. You therefore need to provide your own implementation of this service and register it in a module of your application.

The example application provides an example of such an implementation. The following XML code defines a module whose only responsibility is to register an exit service:

```
<!-- Registers application-specific exit service -->
<module id="ExitModule"
class="net.sf.genuine.example.crm.exit.ExitModule">
  <services>
    <service interface="net.sf.genuine.services.ExitService"/>
  </services>
</module>
```

The `initialize` method of the module creates an instance of the exit service and registers it so that the framework may fetch it:

```
CrmExitService exitService = new CrmExitService(moduleManager);
registerService(ExitService.class, exitService);
```

By this registration, the framework calls the exit service whenever the user tries to terminate the application. It is the responsibility of the service to perform all checks that are necessary and to actually terminate the application. Here is what the example exit service does:

```
public void exitApplication() {

  Channel exitCheckChannel =
moduleManager.getMessageBus().getChannel("exitcheck");
  Message message = new Message("mayExit");

  boolean mayExit = true;
  String reason = "";
  try {
    exitCheckChannel.sendVetoableMessage(message);
  }
  catch (MessageVetoException e) {
    mayExit = false;
    if (e.getExplanatoryResourceItem() != null) {
      reason = e.getExplanatoryResourceItem().getText();
    }
  }

  if (!mayExit) {
    String result =
      helperService.showMessage(CrmResources.EXIT_CONFIRM_UNSAVEDDATA,
                                new Object[]{reason});
    if (result.equals(DefaultResources.CANCEL)) {
      return;
    }
  }

  System.exit(0);
}
```

The service does not perform the checks by itself but asks the modules to perform checks. This is done by sending a message over a channel called `exitcheck` that has been defined in the application configuration file.

All modules that would like to perform a check have to register at the channel and receive messages from it. Whenever the message `mayExit` is received, a module may check whether the application may be terminated from its own point of view. If a module would like to prevent the termination, it throws a `MessageVetoException`.

In this case, the service asks the user whether the user would still like to terminate the application. If that is the case, the service calls `System.exit(0)`, otherwise it returns without any further actions.