

Eclipse RCP as a foundation for J2EE based application clients in an enterprise environment

Eclipse Summit 2006 - Position Paper

Stefan Bohm
IT-Designers GmbH
bohm@it-designers.de

Introduction

The IT-Designers GmbH is involved in the development of integrated application platforms for a large car manufacturer. The J2EE application platform that is part of this product suite provides the corporate standard for server-side J2EE-based applications of our customer. The platform supports as part of its Client Container component the development of rich clients based upon Java Swing and Java Eclipse RCP technology.

Our customer's Client Container is based on the J2EE Client Container specification and represents the runtime environment for Java application clients. Application clients implement the presentation layer of the application while the business logic is implemented as Java Enterprise Beans (EJB) on the server. In order to support this scenario, a J2EE Client Container provides basic services (e.g. configuration, EJB lookup) for the application client, similar to an EJB or servlet container for EJBs or servlets, respectively. Additionally, the J2EE Client Container is responsible for the authentication process, so that application clients are able to access protected EJBs on the server side.

Our customer's Client Container extends the J2EE specification by providing customer-specific features that guarantee a seamless integration into the J2EE application platform. The most prominent features are:

- J2EE-like security in application client (isUserInRole)
- Fine-grained authorization based on resources assigned to a user or a user-group
- Start-up permission for application clients
- Session-cookie based single sign-on (SSO) support
- Remote logging
- Offline authentication and operation support
- Remote method invocation via HTTP(S)

To support Eclipse-based rich clients, an Eclipse plug-in has been developed that provides the services of the Client Container to Eclipse RCP applications. This plug-in encapsulates the existing Client Container implementation.

Requirements

In order to migrate existing Client Container functionality to a standardized, stable and well-known framework, we propose the following features for the Eclipse RCP.

Security Integration

The RCP should be able to integrate existing server-based security solutions. This means that Eclipse RCP should be able to defer authentication and authorization decisions to third-party security providers.

Authentication and Authorization

The RCP should be able to provide authentication and authorization functionalities based upon existing server-side security solutions, as stated above.

- Only authenticated users may start the application. There may be the need for certain applications to use 'unauthenticated' guest users. The guest user can be switched by a 'logon' to an authenticated unique user. An explicit logoff for a user is supported as well as a user-switch, which is similar to a logoff and subsequent logon.
- Each user must be capable of having multiple roles bound to the applications.
- An application, for which a user has no access rights, must not be accessible.
- Support of authorization decisions based on declarative and programmatic security.
- Support for JAAS to enrich the user's subject with additional attributes and information.

Online and Offline Mode

An application has to be able to start in online mode or offline mode. This has either to be transparent to the user (e.g. the client is started in offline mode if a network connection is not available) or the user must be capable to explicitly select the mode. A user-triggered switch from offline to online mode and vice versa has to be possible. Each application needs to be aware of the mode to disable certain functionality that is not appropriate for the current mode.

Local Credential Store

To support the operation of the applications in offline mode, the necessary credentials, access control definitions, user attributes, etc. need to be available on the client itself. This requires that the RCP is capable to download the related information in online-mode and store a local copy in a secure manner.

Local Data Store

The RCP should provide means for storing business data locally. Though this can be pushed to each individual application, the RCP should provide a common capability for the applications to rely on an overall mechanism for local data storage.

Storing data locally imposes several further requirements:

- The data should be stored per user to support multi-user operation.
- The locally stored data needs to be secured similar to a server-side database (access control, separation between individual applications, etc.)
- The data should be encrypted to allow confidential data to be stored on the client. To use the data store in both offline and online mode, the encryption/decryption key cannot be obtained from a server. It must be generated on the client.

Single-Sign-On support for Web Browsers

Browser-based thin-client applications deployed in a corporate environment often require the usage of security cookies from third-party security providers (e.g. CA SiteMinder). In order to integrate such thin-client applications via an embedded browser, the application must be able to share security tokens with the embedded browser.

Further Requirements

Currently, the following requirements exhibit a "nice-to-have" priority for our customer:

Multi-Application Capability

The upcoming IBM technology WebSphere Everyplace Deployment (WED) technology shows remarkable steps into the direction to client-side portals. Several new concepts are introduced here, such as personalization or the Rich Client Mark-up Language (RCPML). Especially, the possibility to run multiple applications under the hood of a single framework instance is of interest for our customer's J2EE application platform. Supporting multiple applications imposes the following requirements:

- Each application must be able to run in its individual state. This means that unless otherwise required by the applications, each application can run individually from other applications, does neither know about the other applications, nor gets affected by them.
- Each application runs individually in its own thread (multi-threading across integrated applications).
- The framework does not impose limitations to the applications with respect to multi-threading. An application can be multi-threaded on its own to allow multiple concurrent streams of data processing.