DRAFT 03 - Getting Started with EclipseRT

Dec. 11th, 2009

Overview

Eclipse is well known as a source of trusted development tools, but it also provides a wide selection of runtime containers, middleware and enterprise frameworks. EclipseRT is the name given to this portfolio of Eclipse runtime projects. Developers can use the EclipseRT technology to build rich desktop, web, enterprise, and embedded applications, as well as service-oriented architectures (SOA) and applications platforms.

EclipseRT is based on the concept of component-oriented development and assembly, or CODA. A key aspect of CODA and EclipseRT is the ability to assemble and extended different components to meet the immediate needs of a specific application type. In this document we will show how EclipseRT can help you apply the principles of CODA to some of the more popular application architectures: desktop applications, web applications, SOA, enterprise applications, platforms and mobile/embedded systems.

Desktop Applications

Desktop applications provide a rich, rapid-response graphical interface for end users. This type of application also allows for easier access to the desktop resources and allow for an off-line/disconnected mode.

The Eclipse Rich Client Platform (RCP) provides a modern desktop framework with the rich and powerful features required for high productivity applications. For example, RCP has been used by companies like IBM to deliver its highly dynamic and network-dependent applications in its Lotus productivity portfolio used as a core business tool daily in many companies.

Extending the capabilities of RCP, the Riena project provides a complete framework for creating next generation client/server applications. In addition to making GUI development with RCP easier, Riena provides the communications back-end needed to create and run connected desktop applications. The Eclipse BIRT reporting framework is commonly used in desktop applications for enterprise grade reporting. For creating applications with graphical editors, the Graphical Editing Framework (GEF) and Eclipse Modeling Framework (EMF) are valuable frameworks for accelerating development of these types of applications.

The p2 provisioning platform in Equinox provides the runtime updating framework needed to support frequent feature delivery. More than just providing a place to click and download an installer for a new version of the application, Equinox-driven desktop applications can update select components with p2 over the network resulting in easier to manage update cycles.

Web Applications

Web applications are clearly one of the favorite delivery models for consumer oriented applications. While web applications are quick and easy to create, as they grow the demands to extend and maintain the application can quickly overwhelm the team. EclipseRT provides reliable, and flexible frameworks for web user interfaces, back-end connectivity, and the tools that support a healthy software development process.

Eclipse provides a light-weight, high-performance web server, called Jetty. Widely used in many domains, such as Google, HP, Cisco, and Atlassian, as a web container in development and production. During development, Jetty's easy configuration management and fast start-up time enables a rapid feedback loop code-build-deploy. In fact it is so fast, we eliminate the "build" and "deploy" parts and just run right out of the workspace for test cycles, allowing developers to stay focused on the immediate task. In production, Jetty provides a diverse array of performance profiles for different scenarios, providing a complete runtime for web applications. Jetty integrates with popular frameworks such as Spring, JBoss, Glassfish, ActiveMQ, and standard Java web technologies. And, of course, Jetty can be used with the rest of EclipseRT, such as the Eclipse Rich Ajax Platform and Equinox.

The Eclipse Rich Ajax Platform (RAP) is a runtime for running web applications and a complete framework and tool-set for developing Ajax applications. RAP is tuned to work with Jetty, Tomcat, or any standard web container. In addition to web applications, RAP builds on SWT and the Eclipse UI framework so you can write the application UI once and have it rendered as web or desktop UI.

A rich UI is vital for any contemporary web application, but access to all types of data and back-end processes is required as well. EclipseLink can be used for integrating data mapping into a web applications, Eclipse Communication Framework (ECF) can be used for REST-based access to back-end services and Swordfish can be used for integration with SOA infrastructures.

SOA and REST

EclipseRT includes a SOA framework called Swordfish. Built on proven open source components such as Apache ServiceMix and Apache CXF, Swordfish provides an extensible framework that allows application developers and system integrators to build their own ESB that can be tailor-made to their requirements. Other components include a process engine provided by Apache ODE for BPEL support, monitoring, a service registry, and the configuration stores required to run an ESB-driven SOA. The components in Swordfish come together to provide a extensive platform for creating and then running service oriented architectures.

Developers can also use EclipseRT to implement a lightweight REST-based infrastructure.

The Eclipse Communication Framework (ECF) provides for discovering, accessing, and implementing REST-based services, asynchronous/messaging-based services, and full support for the OSGi 4.2 Enterprise Experts Group remote services standard. ECF's transport-independent architecture allows a variety of protocols to be used (e.g. http, JMS, XMPP, multicast IP, zeroconf, SLP, others) without modification of application code that uses the framework.

EclipseLink SDO offers the reference implementation of Service Data Objects allowing services to easily pass structured data between them crossing service and programming language boundaries. This infrastructure is leveraged with the Swordfish project but could be used in any SOA development efforts.

There are two key advantages of using EclipseRT to build SOA and REST-based applications: 1) tight integration with Eclipse tools makes it easier to build-test-debug new services and 2) the component nature of EclipseRT means you can easily integrate and use other EclipseRT components like BIRT, EclipseLink as you develop the specific services.

Enterprise Applications

EclipseRT's component-driven nature makes it extremely good for delivering and maintaining large, modern enterprise applications. IT departments are expected to deliver applications more frequently, delivering on the promise of agile software development. As business requirements evolve at a rapid clip, software must change and move at the same fast pace. At the same time, development teams must provide the same stability, controls, and forward looking design that comes from good enterprise architecture.

Starting with Equinox, an OSGi runtime framework, EclipseRT provides the foundation for an enterprise architecture, allowing you to specialize the enterprise stack to your application's needs. This component-oriented approach allows you to spend less time carving down bulky runtimes, instead focusing on the application at hand.

The EclipseRT portfolio of capabilities allows developers to easily extended and assembled different frameworks to satisfy the infrastructure needs of the enterprise applications. EclipseRT includes the following capabilities:

- UI layers with RCP, RAP, or Rienna.
- Runtime containers such as Jetty and Equinox
- Persistence services with EclipseLink
- Data reporting support from BIRT.
- Data modeling with Eclipse Modeling Framework (EMF)
- Remote communication and distributed OSGi with ECF
- SOA integration with Swordfish
- Identity integration and management with Higgins
- Access to unstructured information in the enterprise via search solutions using **SMILA**.

The combination of the EclipseRT projects and Eclipse tooling make it the ideal choice for developing and deploying enterprise applications.

Platforms

Larger organizations and teams working together become more efficient when they consolidate their architecture, services, and code around a shared platform. These platforms then become a shared set of services and frameworks used by others to build applications and allow developers to focus on their business logic rather than the general purpose underpinnings.
Eclipse itself is an example of a platform for creating tools and runtime frameworks. As such, it's little wonder that many of the early adoptors of EclipseRT have built platforms that allow them to rapidly deliver applications to their customers and end-users.

For example, NASA, the US space agency, uses EclipseRT for its Ensemble platform. Ensemble is used by the different science teams to write the applications needed to control and run the experiments on their Mars rover operations. Instead of each team writing their software from the ground up, they build on-top of the EclipseRT-based Ensemble platform. This results in an an acceleration of their software development efforts, a consistent end-user view of these applications and an interchange point between applications.

The component-oriented nature of EclipseRT and the OSGi standard creates a technology base that can be designed to be a platform. Architects can use the design and policy enforcement aspects available in EclipseRT to architect proper extension points and application development in their organization. Because the platform is built on-top of Eclipse technologies, developers can use their familiar tools and frameworks when developing for the platform instead of being forced to use custom platform tools. The deployment services available in p2 gives IT the ability to deploy and then maintain applications built on the platform.

Mobile & Embedded Systems

While EclipseRT is often used for traditional software projects, it works well for mobile and embedded software. These types of software systems have a different set of challenges, due to the limited resources, yet they are increasingly expected to be as dynamic and functional as regular applications.

Once again, the close attention to a high performance, component-oriented architecture provides the foundation for EclipseRT's success in embedded software, helping to support everything from physical security management to bomb sniffing. The embedded Rich Client Platform (eRCP) provides a lightweight footprint for application delivery to many embedded scenarios.

Device connections to back-end systems can be integrated with the Eclipse Communications Framework (EC), integration with back-end data stores and orchestration services can be provided with technology such as EclipseLink and Swordfish. The p2 updating and management mechanisms mean that embedded devices can be updated dynamically, adding new functionality as needed across the life-time of the device.

A key point of the EclipseRT approach is that developers can use: 1) use the same OSGi component model across embedded, mobile, desktop and server applications and 2) use the same Eclipse tool-chain across the same applications types, making developers more

productive by cutting down context-switching costs and helping unify development teams instead of splitting them based on the delivery model.

Conclusion - Eclipse Runtimes

Eclipse has excelled at providing cross-platform tooling for many years, and now with EclipseRT now provides a high quality foundation for all types of applications and services.

[Once main content is nailed down, the conclusion will tie together with a theme like "RT enables agile/rapid application delivery no matter what the delivery mode," or RT as a diverse, general purpose development and production environment." A key part will be emphasizing again that RT is only partly about development, but is also largely about production runtimes.]