DRAFT - Getting Started with Eclipse Runtime

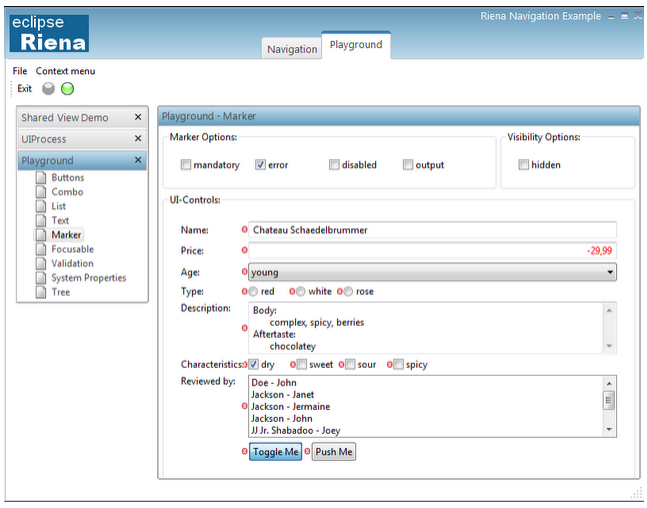
Nov 24th, 2009

Overview

How can you start using The Eclipse Runtime projects for your development needs? The Runtime (or "RT") project is a combination of several Eclipse projects oriented around developing and running many types of software application, services, and even embedded software.

While individual projects and even the component-oriented nature of RT have been covered in depth elsewhere, this white paper gives an overview of how RT addresses the needs of different types of software projects: desktop applications, web applications, SOA, enterprise applications and services, platforms, and embedded systems.

Desktop Applications



[Placeholder for Riena application example screenshot]

The expectations for desktop applications have dramatically changed: users expect applications that are connected to networked services (like the web) and that are frequently updated. Desktop applications with these attributes gives users the ability to tweak existing

business models and use new ones, such as social networking, rather than being held back by their aging software. Legacy desktop frameworks focus on single user, disconnected scenarios and do not contain the SDK and functionality to deliver the connected, frequently updated desktop.

Starting with RCP, Eclipse Runtime provides a modern desktop framework with rich networking and updating support. RCP has been used by companies like IBM to deliver its highly dynamic and network-dependent Lotus Connections social networking portfolio.

Extending the success 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 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. Without solving the problem of frequent, small updates as p2 does, agile delivery models just create more desktop management hassle for IT.

The tooling to support all of this comes from the many years Eclipse has spent at the center leading IDE and development tool-chains. More than being sourced by one vendor or used by one community, the Eclipse tool-chain has been honed and perfected by many eyes over many years, resulting in a tooling foundation that is well known by many developers.

Web Applications

Web applications are clearly one of the favorite delivery models for many development teams. 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. The Eclipse Runtime project provides the reliable, but flexible frameworks for web user interfaces, back-ends, and the tools that support a healthy software development process.

The Eclipse Rich Ajax Platform (RAP) is not only a runtime for running web applications, but a complete framework and tool-set for developing Ajax applications. In addition to web applications, RAP builds on the SWT Eclipse UI framework meaning that it can be used to single source web and desktop UIs for teams that are pressured to do both.

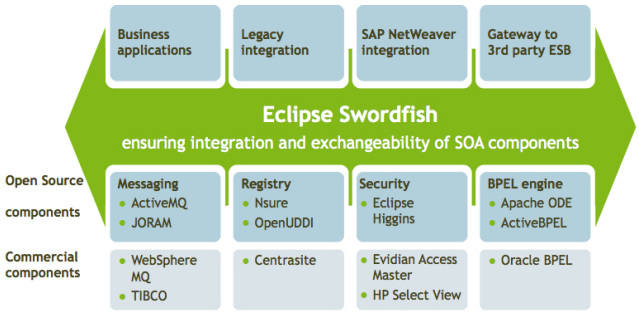
A rich UI is vital for any contemporary web application, but access to all types of data and processes is required as well. Many web applications are backed by databases and SOA infrastructural. Both EclipseLink for data mapping and Swordfish for SOA are part of the Eclipse Runtime stack, along with SMILA which can be used to interact with unstructured data such as documents.

Eclipse even provides a light-weight, high-performance web server, Jetty. During development, Jetty's easy configuration management and fast startup time enables rapid feedback loop code-build-deploy-test cycles, allowing developers to stay focused in their development environments. In production, Jetty's provides a diverse array of performance profiles for different scenarios.

Over the years, the Eclipse community has developed plugins for developing just about

every type of web application including state of the art Ajax for both web and mobile delivery. Because these web development plugins build on and integrate with all Eclipse tooling, developers and teams don't have to sacrifice a fully integrated tool-set and process framework for functionality.

SOA



[Place holder diagram]

The Eclipse Runtime project contains a comprehensive, production-hardened SOA framework called Swordfish. The project originated from one of the world's largest post services, Deutsche Post, and has since been extended and deployed in many more environments. Swordfish is built on a proven ESB platform integrating with several Eclipse and non-Eclipse projects, services, and frameworks. Additionally, Swordfish .Net provides native .Net integration ensuring that the resulting SOA is in no way siloed by platform, language, or vendor.

Key to Swordfish's success has been the tight integration with the Eclipse tool chain and developer environment. Rather than having to learn new tools and then switch between those and their Eclipse IDE, Swordfish lets developers start and then stay in the environment they know and work in best.

In addition to the Swordfish project, the Eclipse Communications Framework (ECF) contains libraries that provide asynchronous point-to-point and publish-and-subscribe messaging. Along with the Jetty web server, ECF can be used to create light-weight SOA or RESTful end-points and consumers when a more comprehensive SOA framework would be overkill. [tk: is this true about ECF?]

[May delete for space:] Building in the proper abstractions and separations of concerns is key for the long-term health and flexibility of any service-oriented architecture. For SOA design concerns, the component-oriented nature of the Eclipse Runtime project (supported by Equinox) helps divide the system into smaller sub-systems that teams can more easily create and then manage over multiple releases.

Enterprise Applications

[Customer quote here would be ideal]

More so than ever, IT departments are expected to deliver applications more frequently, delivering on the promise of agile software development. IT departments are now required to provide new offering and change existing ones to match ever changing consumer desires

at a frightening frequency. As business offers and services evolve at a rapid clip, software must change and move at the same fast pace.

The collection of tools and production frameworks in the Eclipse Runtime project offer the foundation needed for agile software delivery. Starting with Equinox's component-driven model, the UI and back-end projects, and the comprehensive tooling available in the Eclipse ecosystem, enterprise developers have can create applications whose functionality can be iteratively built and dynamically updated as new features and modifications are requested.

Riena, RAP, and RCP provide UI-tier development frameworks and runtimes that helps focus teams on rapidly delivering applications. Projects like Swordfish, EclipseLink, and SMILA are equally effective at integrating with enterprise back-ends and the services required for most enterprise applications. For enterprise grade reporting, the widely used BIRT project fits into the Eclipse RT-driven development chain.

[Can delete for space/repetition]: Finally, many enterprise tool-chains are already based on and use Eclipse technologies. Enterprise developers are likely to be immediately familiar with the RT tools and those day-to-day IDE skills will easily translate to other popular enterprise technologies.

Platforms

> "The One Bench plus Update Manager combination lets us develop and deploy quickly while reducing our risk."
> --Paul Sampat, Vice President, IB Technology Exotics & Hybrids group, JPMC

Development teams are often tasked with building a "platform": a shared set of services and frameworks to build applications for use by themselves or others. The Eclipse Foundation itself is a provider of platforms and most of the projects, including the Eclipse Runtime project, are build to be such platforms. As such, it's little wonder that many of the early users of the Eclipse Runtime project have built platforms that allow them to rapidly deliver applications to their customers and end-users.

For example, JP Morgan Chase used Equinox [tk: true?] and RCP to build its successful One Bench platform, relied on by the banking giant to rapidly build and deploy financial applications to [tk: JPMC employees - but what type? analysts, bankers, quants, traders?]. JPMC has successfully built a platform that allows them to consolidate disparate applications onto one, shared platform, laying a solid foundation for security, auditing, scalability, interoperability, and reusability.

The component-oriented nature of the Eclipse Runtime project - provided by the OSGi-based Equinox model and complimented by sibling projects that build on Equinox - creates a technology base that's built to be extended, to be a platform. Architects can use the design and policy enforcement aspects available in various Eclipse Runtime projects to architect proper avenues for extension and application development in their organization. Developers can use their familiar tool-chains when developing for the platform instead of being forced to use custom platform tools. The management services available in Eclipse Runtime such as p2 and the orchestration services in Swordfish gives IT the ability to deploy and then maintain applications built on the platform.

Embedded Systems

> "Equinox is absolutely essential for our applications going forward. It makes reliable development and deployment of loosely coupled, but highly cohesive applications possible in very short timeframes."
> --John Cunningham, President, Band XI

While the Eclipse Runtime projects is often used for traditional software projects, it works well for embedded software as well. Embedded software faces a different set of challenges than traditional software: limited resources require high performance, low-profile frameworks and yet embedded devices are increasingly expected to be as dynamic and functional as regular applications.

Once again, the close attention to a high performance, component-oriented architecture lays the foundation for Eclipse Runtime's success in embedded software. The embedded Rich Client Platform (eRCP) provides a field-tested layer for application delivery, helping support everything from physical security management to bomb sniffing. And with a 10 meg runtime [tk: is this right?], eRCP is a small enough profile for many embedded scenarios.

Connections to back-ends can be designed with the Eclipse Communications Framework [tk: true in the embedded space?], and back-end data stores and orchestration services can be provided with the extensive server-side projects such as EclipseLink and Swordfish. The updating and management mechanisms in Eclipse Runtime mean that embedded devices can be updated dynamically, adding new functionality as needed across the life-time of the device, even at runtime [tk: true?].

Developers can also use the same tool-chain between embedded, desktop, server, and even mobile development with the familiar Eclipse working environment, 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 - Snappy Phrase]

[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.]