Eclipse Summit Europe - Server-side Eclipse - 11-12 October 2006

Rich Server Platform (RSP) - Position Paper

Introduction

Eclipse Rich Client Platform (Eclipse RCP) has been successfully established as an application platform for the client-side.

With the recent acceptance of large parts of the OSGi specifications as JSR 291 (www.jcp.org), widespread application of OSGi for server-side applications has become possible.

The Eclipse/OSGi development model, applied to the server-side, allows the creation of truly pluggable, componentized, server-side applications. Refer to http://www.infonoia.com/en/content.jsp?d=inf.05.07 for a discussion of possibilities.

Plug-ins can communicate at a logic layer with each other using established and tested methods, such as extension points and OSGi services. What is needed for the UI layer is a UI composition framework which we propose in the form of Rich Server Platform (RSP).

Rich Server Platform (RSP) will allow developers to freely combine the most suitable rendering technologies for versatile web applications. They will increase code reuse, enjoy a flexible and lean assembly of just the components and services they need, and enhance their productivity in composing a versatile web application. Rich Server Platform (RSP) adresses core needs in the area of "service-oriented clients", where this architecture allows developers to build better applications that are more targeted to user needs, tasks, roles and workflows.

A demo walkthrough with sample code of a basic server-side application can be found at http://www.infonoia.com/en/content.jsp?d=inf.05.09. It uses jsp:include tags as the lowest degree of UI composition. A demo application is in preparation.

Issues, solution and benefits

<table>
<thead>
<tr>
<th>Current Issues</th>
<th>Rich Server Platform solution</th>
<th>Benefits</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Code reuse between client-side and server-side is limited</td>
<td>Server-side UI framework that renders &quot;dual&quot; components</td>
<td>Increase code reuse</td>
</tr>
<tr>
<td>2. Modularizing server-side applications is difficult</td>
<td>Apply Eclipse/OSGi bundle paradigm to the server-side</td>
<td>Enjoy a flexible and lean assembly of just the components and services needed</td>
</tr>
<tr>
<td>3. Integrating UI from sets of rendering technologies is cumbersome</td>
<td>Define (best practices) for UI interactions and composition</td>
<td>Enhance developer productivity and reduce time-to-market and project costs</td>
</tr>
<tr>
<td>4. Server-side world offers a large choice of rendering technologies, but no single best answer as to which set to use for web UI</td>
<td>Offer a large choice of rendering technologies to pick and choose from</td>
<td>Meet the diverse and special requirements of an application efficiently</td>
</tr>
</tbody>
</table>
1. **Code reuse between client-side and server-side is limited** – Many business-critical applications have both a heavy data-entry element for internal use (rich client) and a general-public web interface. Usually, the UI of these will not be the same, however, good parts of the business logic will be. By applying OSGi to the server-side, Rich Server Platform (RSP) will provide developers with a web UI framework that allows to reuse a lot of the business logic (non-UI plug-ins) for both parts of an application. With Rich Server Platform (RSP), developers will be able to reuse "dual" components that can be rendered both in a browser and in a rich client.

2. **Modularizing server-side applications is difficult** – Many web applications have numerous functional modules, such as shopping cart, news feed, or user management. This compares to RCP applications where the Eclipse IDE is also composed of numerous plug-ins. However, the WAR (Web Archive) as the smallest deployment unit of aggregation in J2EE web applications is too coarse grained to allow practical reuse of application "modules". It is not practical to package each module into its own WAR for numerous reasons, including the fact that inter-WAR dependencies are not defined. However, maintaining multiple modules inside a single WAR, with code, and URL resources usually residing in different places, does not provide the degree of isolation that makes such modules easily transferable and reusable in other projects. Rich Server Platform (RSP) will apply the OSGi bundle paradigm to the server-side. So developers will enjoy a flexible and lean assembly of just the components and services they need to compose their application.

3. **Integrating UI from sets of rendering technologies is cumbersome** – Eclipse/OSGi allows inter-plug-in communication at the logic level via extension points and OSGi services. Beyond the OSGi http-service as transport for http resources, however, interactions at web UI level are not defined today, making it cumbersome to "glue components together" at the UI level. Rich Server Platform (RSP) will focus on the definition of UI interactions and UI composition in order to create a framework for easily integrating UI from sets of rendering technologies and frameworks already established in the server-side world. With Rich Server Platform (RSP), developers will enhance their productivity and reduce time-to-market, maintenance efforts and project costs.

4. **The server-side world offers a large choice of rendering technologies, while there is no single best answer as to which set to use for web UI** – The choice will vary with project requirements and developer skills, and may well incorporate multiple approaches. Ideally, a truly componentized web application is able to integrate and display content from plug-ins that use classic web technologies such as, but not limited to, JSP/JSTL, Struts/WebWork framework, Wicket, JSF/Facelets and other applicable widget libraries, Spring and PHP. It should also be able to integrate components that use rich rendering technologies such as LZX/Laszlo. A truly componentized web application should be able to serve the user with a page that combines content from, for example, a plug-in with XML/XSL transformation to display an RSS news feed, a plug-in that uses PHP to render content, another plug-in that uses JSF for rich interactive web components, and a plug-in that renders LZX/Laszlo for interactive components with highest layout requirements. Rich Server Platform (RSP) will help to meet the diverse and special requirements of an application efficiently.

**Scope**

1. Define UI composition framework with API and sample implementations
2. Provide services commonly needed for web applications
3. Address needs ranging from dynamic websites to "single-url" web applications

**1. Define UI composition framework with API and sample implementations**

Making plug-ins run on a server involves the whole application stack; Rich Server Platform (RSP) makes no attempt to cover the whole territory. Instead, it narrowly focuses on defining interfaces and creating sample implementations for UI integration and associated components.

The RCP workbench API will be analyzed and used as the most structured, Java-based API for composing web UI, and leveraged by templating and decoration. We also envisage markup-based composition using a tag library that can be used from JSPs.
Essentially, the same basic abstractions of Eclipse UI will be used. Therefore, PageLayouts will be defined by a combination of Parts. Different part types are EditorPart and ViewPart. Parts will have their trim window built by the workbench framework. This workbench will also provide features to manipulate (maximize, minimize, restore, close, move and dock) views and editors.

A given Part will be associated to ContentProvider. Providers will be the link to different rendering technologies, ranging from script languages such as PHP to fine-grained widgets libraries.

Perspectives are a way to group views for a particular purpose. Plug-ins can define new perspectives or change perspectives defined by other plug-ins. The API used for a plug-in to define a new perspective will be essentially the same as Eclipse’s API. The developer will define an extension to the perspectives extension point and implement the createInitialLayout() method of a new IPerspectiveFactory implementation.

Likewise, the API for defining new views are the same as Eclipse’s (through views extension point).

Rich Server Platform (RSP) aims to ensure that individual rendering technologies can access a shared context (such as request, session and application scope for servlets).

The Apache Tiles framework is a very popular server-side UI composition framework. Given its age of about 5 years, Tiles has limitations in its out-of-the box capacities to render non-JSP/servlet content. Some limitations are those of the content to be integrated. For example, integration of JSF is difficult given that JSF currently does not support page scope. Tiles does not use the notion of workbench contributions, perspectives, views and the like. We aim to resolve these issues in a "from-the ground-up" construction of an extensible and configurable UI composition framework which leverages the Eclipse/OSGi approaches.

We will design interfaces and sample implementations that allow integrating classic web technologies and frameworks in combined web UI. This will usually require an intricate understanding of the workings of the particular technology and framework implementation. Class-loading approaches of such frameworks that differ from the OSGi model will be investigated and resolved to meet OSGi requirements.

Taking PHP as an example, we plan to take a uniform integration approach for scripting languages, with the aim to allow the use of various scripting languages supported by JSR 223 (Scripting for the Java Platform Specification 1.1), such as, but not limited to, Groovy. Scripting integrations will pass context information through namespaces, as proposed in JSR-223.

2. Provide services commonly needed for web applications

With Rich Server Platform (RSP) committers coming from a strong web applications background, part of the project effort will be spent on features and services that developers have been able to rely on in classic J2EE/J2SE web applications, and are useful to be leveraged when writing Eclipse-OSGi server-side applications. These features include, but are not limited to, runtime JSP compilation, user authentication, servlet filters and listeners.

We will also address multiple web-client specific issues, such as client state vs. server state, "bookmarkable URLs", double-submit, redirect after post, client-side validation, windowing, and converting events from such web UI components into events/commands that can be understood by Rich Server Platform (RSP) and relayed to the applicable integrated framework/component. Further integrations with WebWork (Struts 2) and Tapestry are sought.

3. Address needs ranging from dynamic websites to "single-url" web applications

We consider leveraging time-tested principles in established web module “composition” frameworks such as Apache Tiles (Standalone) and SiteMesh to maximize layout design and composition flexibility, and thus facilitate the use of Rich Server Platform (RSP) for any kind of dynamic web application, beyond a specific workbench-style application. Note: actual use and integration of Sitemesh would be subject to approval of a modified Apache license.
FAQ

Does Rich Server Platform (RSP) depend on JSP or other non-Java technologies?

No. A tag library is part of the deliverables, but UI can be composed through Java code only, in an API that will essentially be the same as the Workbench API. In web applications, some URL resources, such as CSS and JavaScript always use non-Java technologies.

Will my web application that uses Rich Server Platform (RSP) have dependencies on JSP-JSTL/Struts/WebWork/Tapestry/Wicket/JSF/Facelets/Spring/PHP?

Not unless you choose so. Rich Server Platform (RSP) aims to allow you to use the rendering framework(s) of your choice; though you will want to use at least one of them to render content in Views and Editors if you want to run more than simple servlets in your web application.

How does Rich Server Platform (RSP) relate to Tiles and SiteMesh?

Apache Tiles (Standalone) allows UI composition of modules via XML templates, while OpenSymphony SiteMesh is strong in combining and decorating arbitrary HTML streams. Both functionalities are very useful in the context of all types of web applications. Therefore, Rich Server Platform (RSP) will build on their composition concepts for Eclipse/OSGi-based web applications. Project committer Cedric Dumoulin is the original creator of Tiles.

Will Rich Server Platform (RSP) work with Spring?

Rich Server Platform (RSP) invites contributions by the Spring community to enable the use of Spring within Rich Server Platform (RSP) applications. Work is ongoing in the Spring community to provide use of Spring with OSGi by Spring Release version 3.
Does Rich Server Platform (RSP) enable my classic web applications to run in RCP without any changes?

Running of arbitrary web applications within RCP is already supported by the Tomcat plug-in, which is also used by the Eclipse help system. Rich Server Platform (RSP) may optionally explore embedding Rich Server Platform (RSP)-integrated components as individual UI contributions within RCP applications. This might be done via the RCP Browser widget and a pass-through bridge in an augmented version of the Tomcat plug-in, which accesses the RCP application extension points. WSRP (Web Services for Remote Portlets) concepts would be studied for this. This item is optional.

Will Rich Server Platform (RSP) require a J2EE-certified server or a Servlet Engine to run?

No. An initial implementation will use the Equinox ServletBridge which runs in a servlet container as a WAR-style application. This provides for maximum compatibility, as this approach allows adding Eclipse/OSGi components to existing web applications, without having to migrate the entire web application to Eclipse/OSGi. However, Rich Server Platform (RSP) aims to be "server-agnostic"; it should be able to run in any OSGi-compliant environment. Given that an OSGi-"server" can itself be composed with bundles, and OSGi class-loading isolation features ensure that different implementations of services can co-exist in the same OSGi environment, as a general rule, Rich Server Platform (RSP) does not depend on particular OSGi container features beyond those specified by applicable OSGi release levels.

Is the mission of Rich Server Platform (RSP) to bridge the gap between OSGi and all of Java EE?

No. However, RSP certainly has the potential to go beyond web applications, and include other parts of Java EE, such as Enterprise JavaBeans, applying the same benefits to EARs as WARS. Rich Server Platform (RSP) is aware of interest in running relevant Java EE features under OSGi, and time will tell in what form and by whom those interests may be addressed. As more and more of the Java development spectrum (Java ME-to-Java EE) is brought into the modular, extensible, RCP-style of definition and architecture, the longer Java's lifespan will be. Should any questions of project scope arise during the project, Rich Server Platform (RSP) will act in accordance with the Eclipse Development Process.

Will Rich Server Platform (RSP) make web development simpler?

We believe so. There is pent-up demand for solutions that are simpler to implement than by playing the full Java EE orchestra. Wicket is one rendering technology approach that helps for simplicity in UI development. The key theme of Rich Server Platform (RSP) is easy composition of smart technologies.


Putting everything imaginable under the heading of RSP into one project would be ambitious. This is why the Rich Server Platform (RSP) project narrowly focuses on UI integrations, and provides a framework, with sample implementations only, whereby ContentProviders can use their technology of choice. It deals with the complexity of the server-side world by providing uniform integration approaches, in hindsight of and knowledge of existing technologies. All project deliverables are assigned to definitive committers (see below).

Rich Server Platform (RSP) does not intend to create a server-side component framework beyond what Eclipse/OSGi already provides. Further, Rich Server Platform (RSP) applies existing concepts such as workbench API as well as established server-side approaches. Rich Server Platform (RSP) aims to support a major strength of the Eclipse platform: its capacity for leveraging cross-technology integration.

Creating a UI composition framework that integrates different URL resources is feasible; Rich Server Platform (RSP) committer Cedric Dumoulin has previously done this (Tiles). Running established web frameworks under OSGi is feasible; Rich Server Platform (RSP) committers from OPS4J and Infonoia have done this. Composing web UI using Eclipse workbench API paradigms is feasible; Rich Server Platform (RSP) committers from Vetta Technologies are doing this. Providing PDE build, run and deployment support is feasible; Rich Server Platform (RSP) committer Gunnar Wagenknecht has PDE experience. Rich Server Platform (RSP) brings these kinds of expertise together in a multidisciplinary team for a successful project.

To avoid scope creep, we will be very happy to spin off new, independent projects for scoped items that may develop their own dimension, such as, but not limited to, scripting. For example, it may be useful to extend scripting integration to JavaScript, Groovy or Rails runtimes, within another project.

Does Rich Server Platform (RSP) dilute Eclipse API?

No. By providing uniform integration approaches, we actually help avoid API dilution. Uniform integration approaches are very important when ContentProviders can use their technology of choice. At the same time, Rich Server Platform (RSP) follows the 2006 Eclipse Roadmap, which states that "Support for compiled languages, scripting languages and "Java like" languages such as JSP, SQLJ and the like are all related areas where work needs to be done".

Contact: Wolfgang Gehner, Infonoia SA, wgehner@infonoia.com - Rich Server Platform (RSP) - 08/28/06  Page 5 of 6
Project organization

Rich Server Platform (RSP) would not be possible without the dedication of a multi-disciplinary team that is convinced of the advantages of open source.

The following 12 committers from 8 countries have joined forces to donate Rich Server Platform (RSP) to the open-source community:

Project lead
Wolfgang Gehner (Infonoia, Switzerland)

Initial committers
- Renan Componez (Vetta Technologies, Brazil)
- Cedric Dumoulin (Université des Sciences et Technologies de Lille, France, original creator of Tiles, co-author of "Struts in Action")
- John R. Fallows (TXE Systems, USA, co-author of "JSF and Ajax")
- Wilson Freitas (Vetta Technologies, Brazil)
- Wolfgang Gehner (Infonoia, Switzerland, co-author of "Struts Best Practices")
- Ricardo Giacomini (Vetta Technologies, Brazil)
- Niclas Hedhman (CodeDragons, Malaysia, JSR 291 Expert Group member)
- Edward Jakop (CodeDragons, Malaysia)
- Makas Lau (CodeDragons, Malaysia)
- Peter Neubauer (Jayway, Sweden)
- Gunnar Wagenknecht (Truition, Canada)
- John Wilson (Zheng Da, China)

Further support has been expressed by:
- Alex Blewitt
- Max Carlson (IDE for Laszlo)
- Grahame Grieve (OHF) and Eishay Smith (OHF), OHF is potential consumer
- Russell Hamm (Mayo Clinic via OHF subproject), potential consumer
- Simon Kaegi (Cognos)
- Peter Kriens (aQute)
- Scott Lewis (ECF), potential consumer
- Martin Lippert (it-agile), potential consumer
- Ed Warnicke (Cisco)

The more input the Rich Server Platform (RSP) project receives, the more cross-integration it can provide. We strongly encourage you to contribute your ideas, vision, know-how and skills in an area that may fall within the scope of this project.

Contact: Wolfgang Gehner, Infonoia SA, wgehner@infonoia.com - Rich Server Platform (RSP) - 08/28/06 Page 6 of 6