



Spring  ONE

20 – 22 June 2007
Metropolis Antwerp, Belgium

Spring and Eclipse RCP

Martin Lippert
Senior IT Consultant
it-agile GmbH, Germany

Speaker's qualifications

- 🍃 Martin is a senior IT consultant at it-agile GmbH, Hamburg, Germany
- 🍃 With a focus on:
 - Agile software development
 - Eclipse technology, especially OSGi/Equinox
- 🍃 Frequent speaker at conferences
- 🍃 Author of articles and some books
- 🍃 Eclipse Equinox Incubator committer

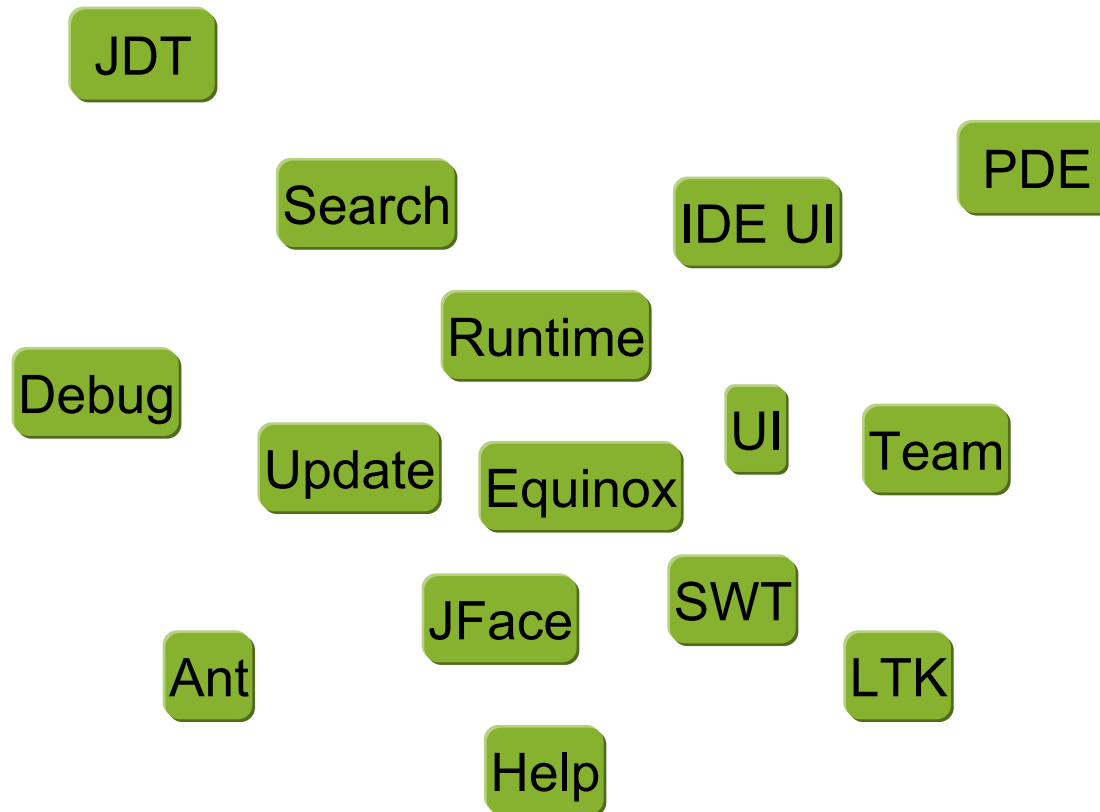
Overall presentation goal

See what Eclipse-RCP is about and why it is
a best fitting buddy for Spring-OSGi

The future of Java programming is
OSGi-based – on the server as well
as on the client

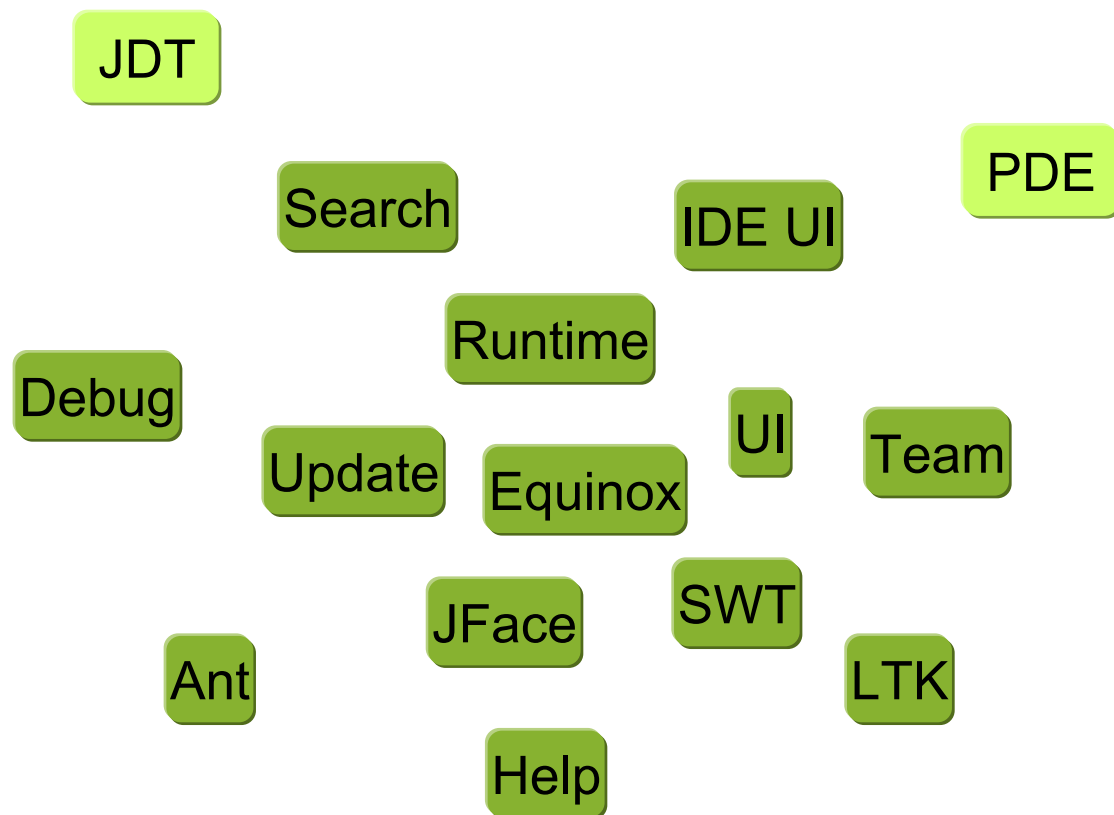
What is the Eclipse Rich Client Platform?

Eclipse: Composition of Components



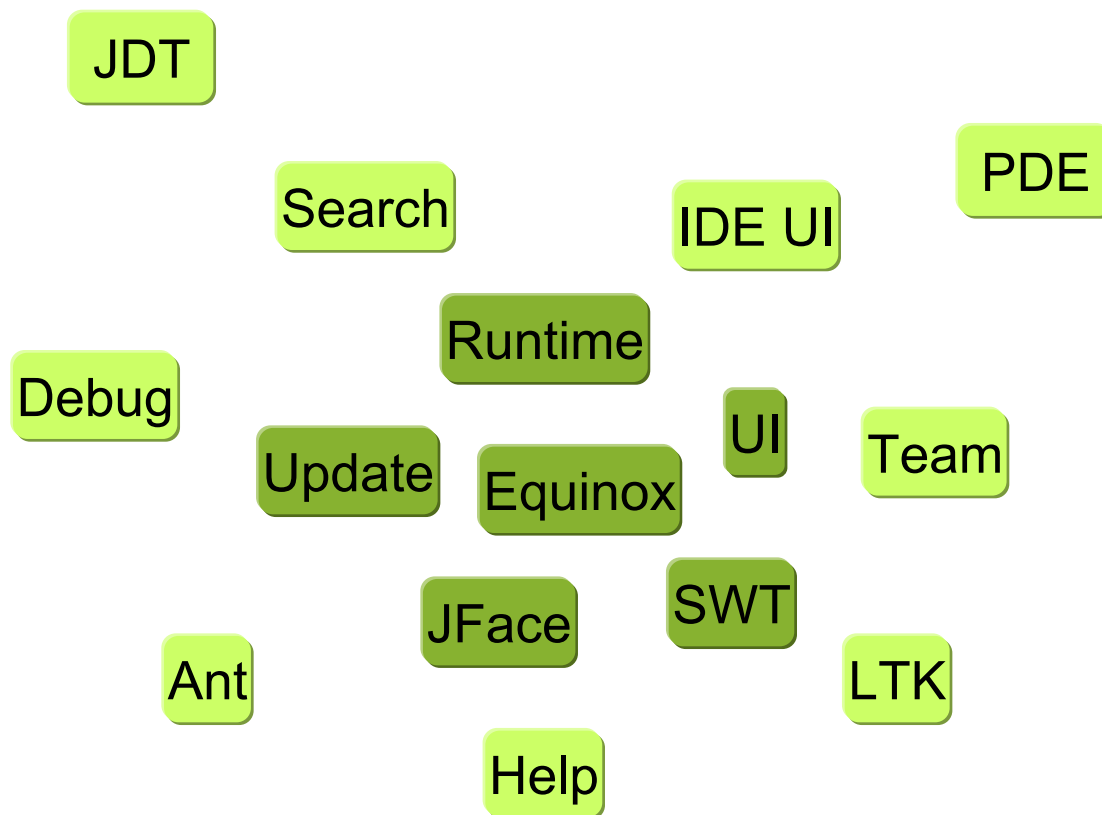
Eclipse SDK/Java IDE

Generic IDE Components



Eclipse IDE

Eclipse Rich Client Platform



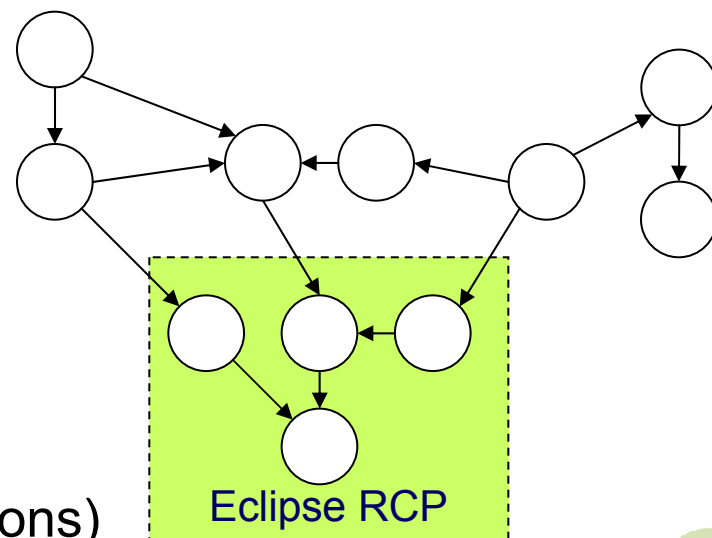
Eclipse Rich Client Platform

Why Use Eclipse Rich Client Platform?

- 🍌 A consistent and native look and feel across applications and features
- 🍌 Provides common application services
 - Native look and feel
 - Window management
 - Standardized component model (Equinox)
 - Help system
- 🍌 First-class development tools
- 🍌 Middleware for building rich client applications!
 - Allows programmers to focus on core application not the plumbing
 - Don't reinvent the wheel

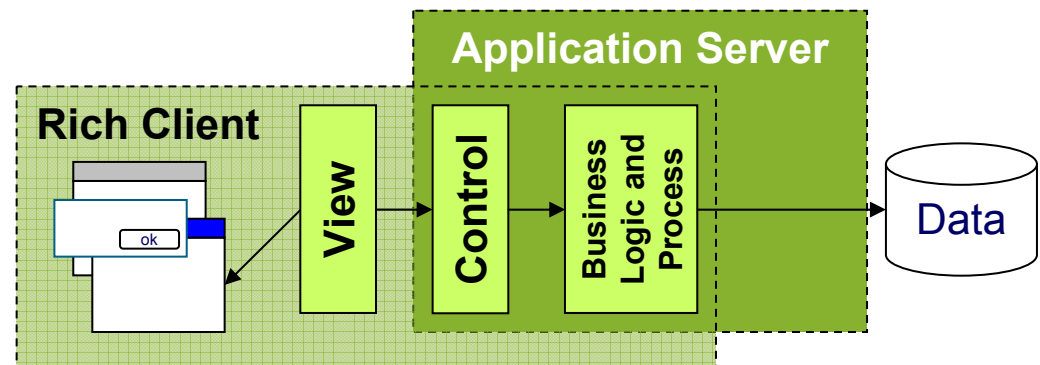
Equinox

- Equinox is the Eclipse component model
 - Based on OSGi R4 specification
 - Standard Java lacks an explicit notion of components
- Components == Bundles == Plug-in
 - Versioned
 - Defined declaratively
 - Dynamically loadable/unloadable
 - Support dynamic update and install
- Explicitly define
 - Dependencies
 - Runtime visibility
 - Interactions (extension points/extensions)

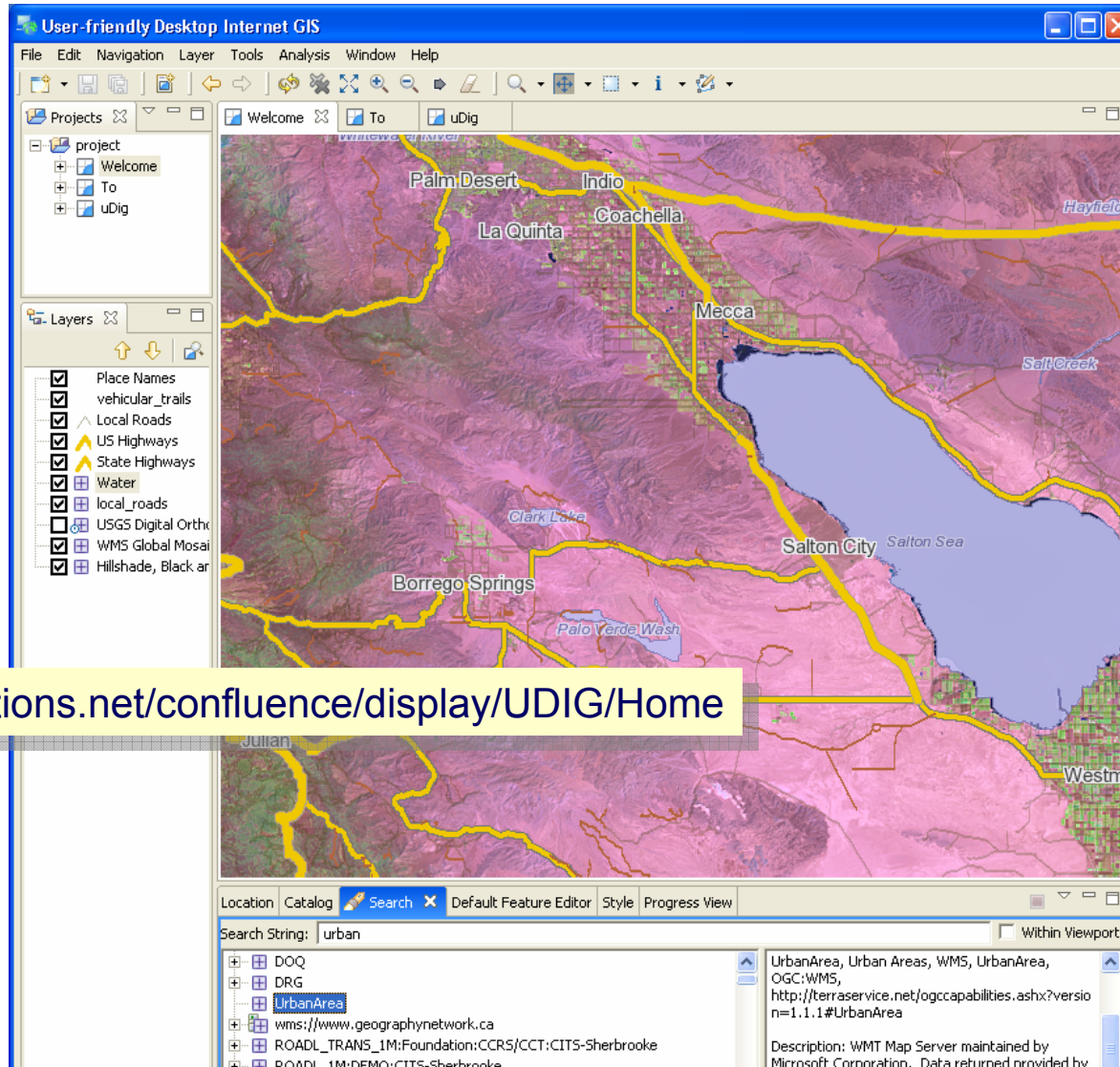


The typical RCP Application

- Rich user experience
- Platform independent
- Component model
- Integrated update mechanism
- Extensible
- Typically (though not necessarily) a client for some backend service**



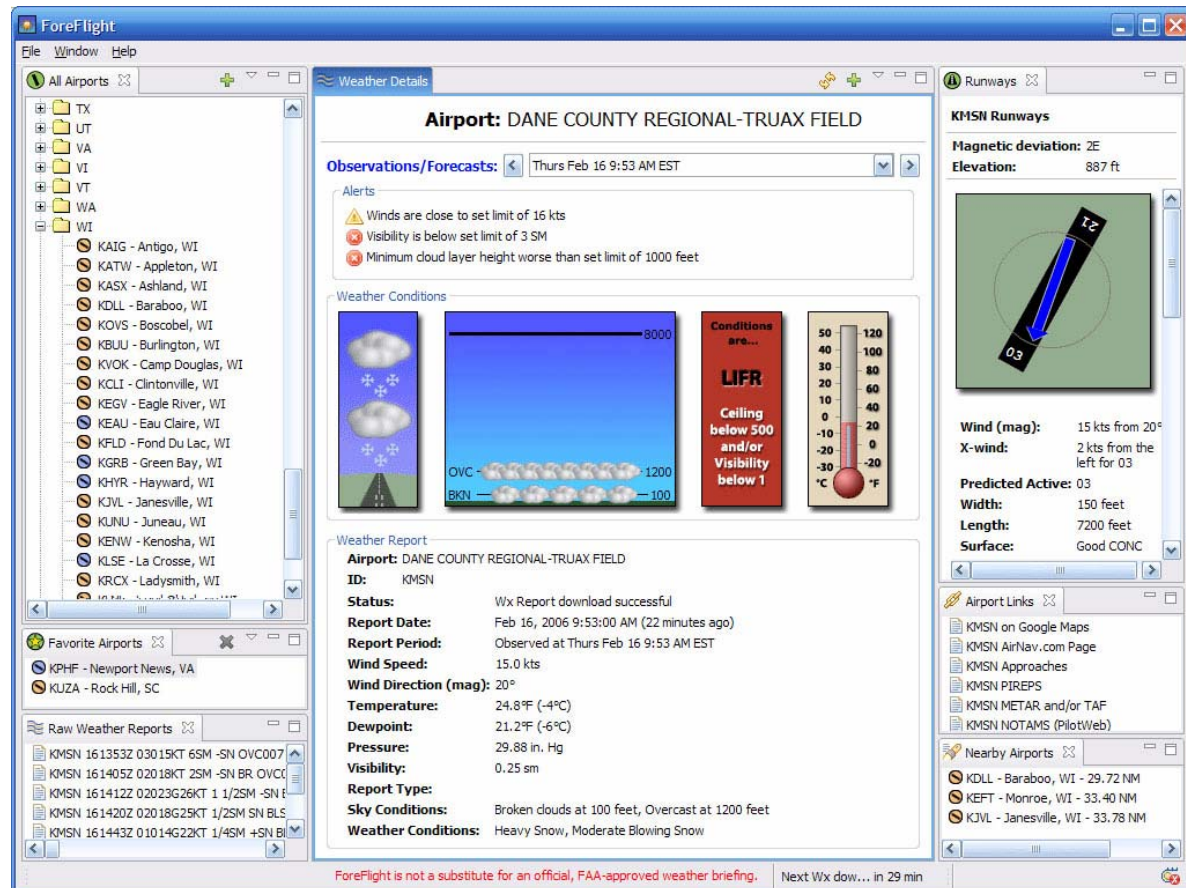
Example: GIS



<http://udig.refrations.net/confluence/display/UDIG/Home>

Example: ForeFlight

- Displays critical information graphically and prominently
- Displays alerts when conditions are near or exceeding the user's preferred limits
- Connects via the web to weather and information services
- Multiple ergonomic views of the weather that affects the go/no-go flight decision



<http://www.foreflight.com/>



Example: Lotus Notes “Hannover”

The screenshot displays the IBM Lotus Notes application interface for user Sam Curman. The main window shows an email inbox with a table of messages. The selected email, titled "More on OP tools deal" from Lukas Geiger, is open, showing its content and a pie chart titled "Total Sales by Line".

| From | Subject | Date | Time | Size |
|----------------|---|----------|---------|---------|
| Betty Zechman | Can we get together? | 05/01/05 | 9:12AM | 46KB |
| Pierre Dumont | Check out these new sales tools | 05/01/05 | 9:15AM | 1,234KB |
| George Bandini | Thoughts on this quarter's results | 05/01/05 | 9:36AM | 1KB |
| Lukas Geiger | More on OP Tools deal | 05/01/05 | 10:02AM | 31.2KB |
| Anna Bauer | Does RNV have an exit strategy for the | 05/01/05 | 10:26AM | 1,345KB |
| Rita Ferrar | Request from Amadou: Please review Pro line | 05/01/05 | 10:45AM | 13KB |
| Lukas Geiger | Idea to boost customer satisfaction next year | 05/01/05 | 11:02AM | 14KB |
| Monifa Shani | Re: More on OP Tools | 05/01/05 | 11:30AM | 156KB |
| Pierre Dumont | Re: More on OP Tools | 05/01/05 | 11:32AM | 356KB |
| Lukas Geiger | Re: Check out these new sales tools | 05/01/05 | 11:42AM | 10KB |
| Anna Bauer | Re: More on OP Tools | 05/01/05 | 12:01PM | 46KB |
| Juan Sanchez | On vacation until next Monday | 05/01/05 | 12:59PM | 1,234KB |
| Lukas Geiger | Fw: New home start projections for next year | 05/01/05 | 1:24PM | 1KB |
| Monifa Shani | Please update projections for this quarter by | 05/01/05 | 1:37PM | 2KB |
| Anna Bauer | Re: Fw: New home start projections for next | 05/01/05 | 1:40PM | 1,345KB |
| Laura Klein | Re: More on OP Tools | 05/01/05 | 1:52PM | 13KB |
| Anna Bauer | Intelligence about change in building | 05/01/05 | 1:57PM | 14KB |

More on OP tools deal
 From: Lukas Geiger
 To: Sam Curman, Pierre Dumont, Anna Bauer, Laura Klein
 cc: Monifa Shani
 Today, 10:02AM

All:

I've been thinking about the OP Tools deal. Take a look at these sales projection charts. Elite line accounts for most of the sales until Q4, when the Pro line really takes off and the Elite line tanks. I think this shows we need to push the Pro line with OP Tools. Thoughts?

Total Sales by Line

1 Attachment, 31.2 KB Save All

The right sidebar contains several panels: "Related Activities" showing "OP Tools deal" and "Sales Lead: OP Tools"; "Contacts" showing "Available" and "Related Contacts (1)" including Pierre Dumont, Anna Bauer, and Monifa Shani; "Related RSS Feed" showing "Reuters Business Headlines", "Sales Discussion", and "Commercial Development N..."; and "Calendar" showing "Today is May 01, 2005" and "Related meetings" including "OP Deal Meeting" at 1:00 and "Meeting with Monifa" at 1:30.

Example: Maestro - NASA Space Mission Management

The screenshot displays the Maestro software interface, which is used for mission management. The main window shows a grayscale image of the Moon's surface with various sites marked by yellow labels (e.g., Site 1, Site 2, Site 3, Site 4, Site 5, Site 6, Site 7, Site 8, Site 9, Site 10, Site 11, Site 12, Site 13, Site 14, Site 15, Site 16, Site 17, Site 18, Site 19, Site 20, Site 21, Site 22, Site 23, Site 24, Site 25, Site 26, Site 27, Site 28, Site 29, Site 30, Site 31, Site 32, Site 33, Site 34, Site 35, Site 36, Site 37, Site 38, Site 39, Site 40, Site 41, Site 42, Site 43, Site 44, Site 45, Site 46, Site 47, Site 48, Site 49, Site 50, Site 51, Site 52, Site 53, Site 54, Site 55, Site 56, Site 57, Site 58, Site 59, Site 60, Site 61, Site 62, Site 63, Site 64, Site 65, Site 66, Site 67, Site 68, Site 69, Site 70, Site 71, Site 72, Site 73, Site 74, Site 75, Site 76, Site 77, Site 78, Site 79, Site 80, Site 81, Site 82, Site 83, Site 84, Site 85, Site 86, Site 87, Site 88, Site 89, Site 90, Site 91, Site 92, Site 93, Site 94, Site 95, Site 96, Site 97, Site 98, Site 99, Site 100). A green line connects these sites, and a blue box highlights a specific area. The right panel shows a table of data with columns for Product ID, Instrument, Sol, and Seq. The bottom right corner displays a small image of the Mars rover.

| Product ID | Instrument | Sol | Seq |
|-----------------------------|-------------------|-----|-------|
| 2F134356767EFF2600P1212L0M1 | FRONT_HAZCAM_LEFT | 90 | p1212 |
| 2F134449644EFF2700P1212L0M1 | FRONT_HAZCAM_LEFT | 91 | p1212 |
| 2F134614869EFF2700P1403L0M1 | FRONT_HAZCAM_LEFT | 93 | p1403 |
| 2F134615161EDN2700P1131L0M1 | FRONT_HAZCAM_LEFT | 93 | p1131 |
| 2F135147950EDN2700P1131L0M1 | FRONT_HAZCAM_LEFT | 99 | p1131 |
| 2F135148174ESF2700P1127L0M1 | FRONT_HAZCAM_LEFT | 99 | p1127 |
| 2F135149109EDN2700P1141L0M1 | FRONT_HAZCAM_LEFT | 99 | p1141 |
| 2F135149794EDN2700P1141L0M1 | FRONT_HAZCAM_LEFT | 99 | p1141 |
| 2F135150380EDN2700P1141L0M1 | FRONT_HAZCAM_LEFT | 99 | p1141 |
| 2F135150997EDN2700P1141L0M1 | FRONT_HAZCAM_LEFT | 99 | p1141 |
| 2F135151610EDN2700P1141L0M1 | FRONT_HAZCAM_LEFT | 99 | p1141 |
| 2F135152416EDN2700P1141L0M1 | FRONT_HAZCAM_LEFT | 99 | p1141 |
| 2F135152602EFF2700P1212L0M1 | FRONT_HAZCAM_LEFT | 99 | p1212 |
| 2F135153765EDN2700P1111L0M1 | FRONT_HAZCAM_LEFT | 99 | p1111 |

<http://www.eclipse.org/community/casestudies/NASAFinal.pdf>

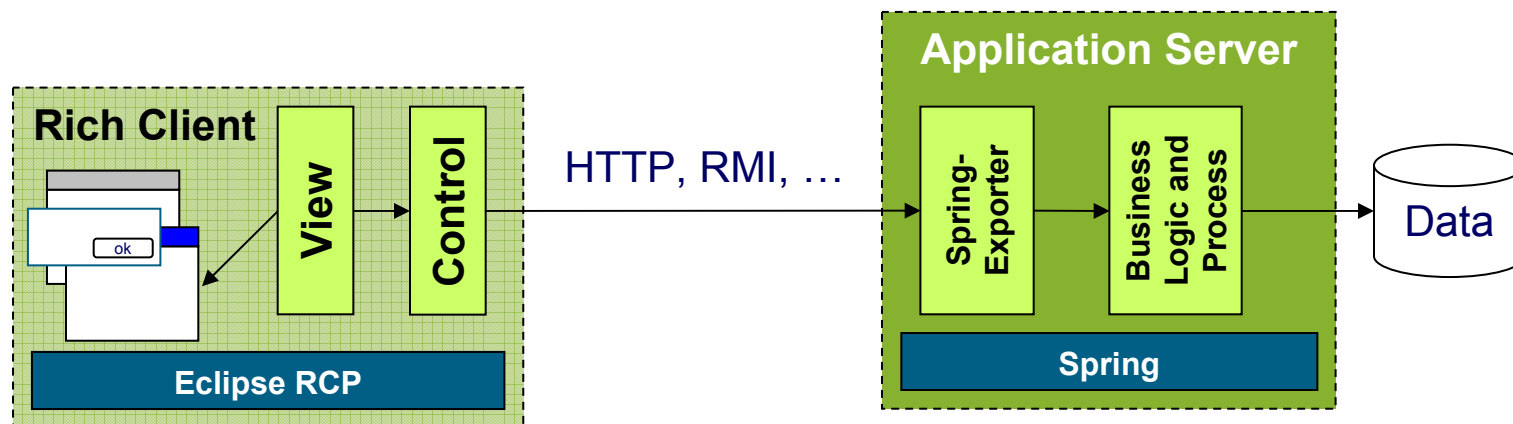
Spring Backends for Eclipse RCP

Spring-based Backends

- 🍌 Spring is great for implementing the backend:
 - Dependency injection for the implementation
 - AOP for cross-cutting and interceptor-based features
 - Easy transaction handling
 - Easy integration of other technology like O/R mapping, security, and so on...
- 🍌 *It is a natural choice to implement the backend using Spring*

Pure RCP Client for the Spring Backend

- Client: Pure RCP
- Server: Pure Spring
- Ways to communicate, for example:
 - Server provides REST/SOAP services, client consumes via HTTP
 - Server provides services via RMI, client consumes via RMI



Evaluation

- 🍌 Unrestricted usage of Spring on the server
- 🍌 Unrestricted usage of RCP on the client
 - Including additional features like data binding support, BIRT, ...
- 🍌 Simple communication protocol (which is good)
 - But difficult for sophisticated remote interfaces
- 🍌 Different deployment and programming models (OSGi bundles on the client, typical WAR file on the server)
 - Good for highly decoupled systems
 - Difficult for more integrated systems

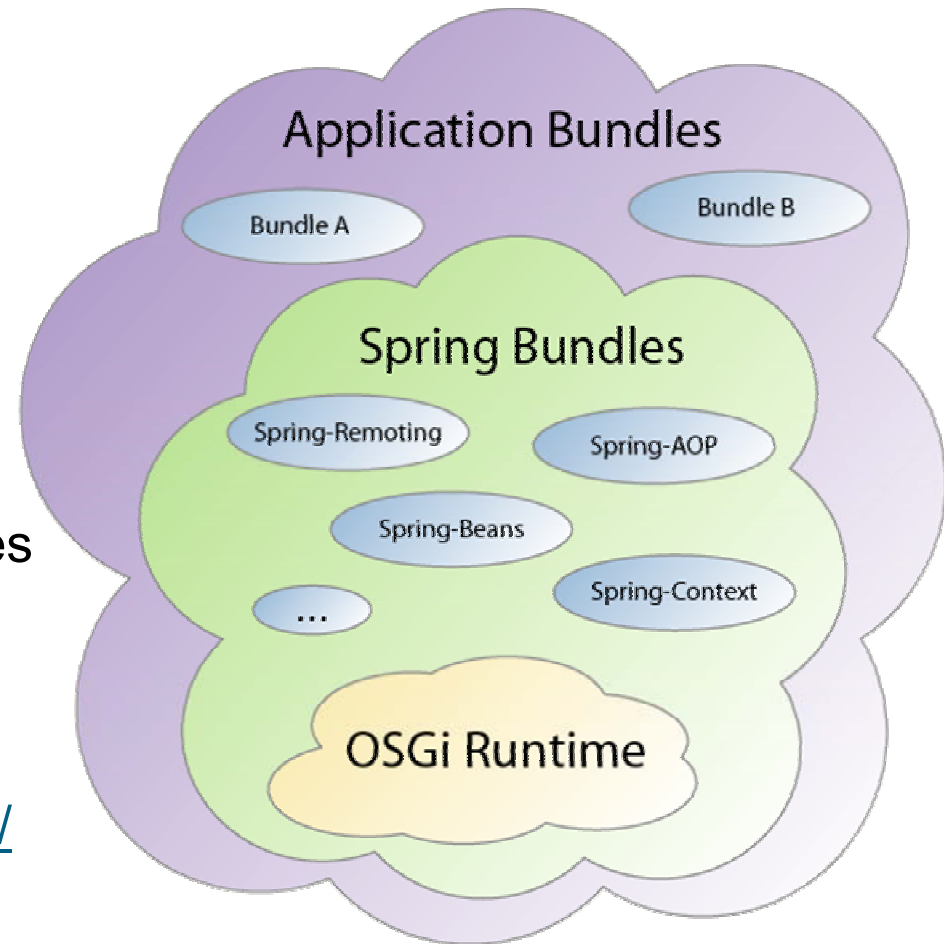
Eclipse RCP + Spring on Client and Server

The Spring-OSGi bridge

- 🍌 Spring-OSGi is an additional Spring project
- 🍌 Allows to use Spring in OSGi applications
 - Per-bundle application context definition
 - Application context initialization at bundle activation
- 🍌 New `<osgi:...>` namespace:
 - Spring-Beans as OSGi-Services and vice versa
 - Dynamic behavior of OSGi via proxies
 - Inter-bundle dependency injection

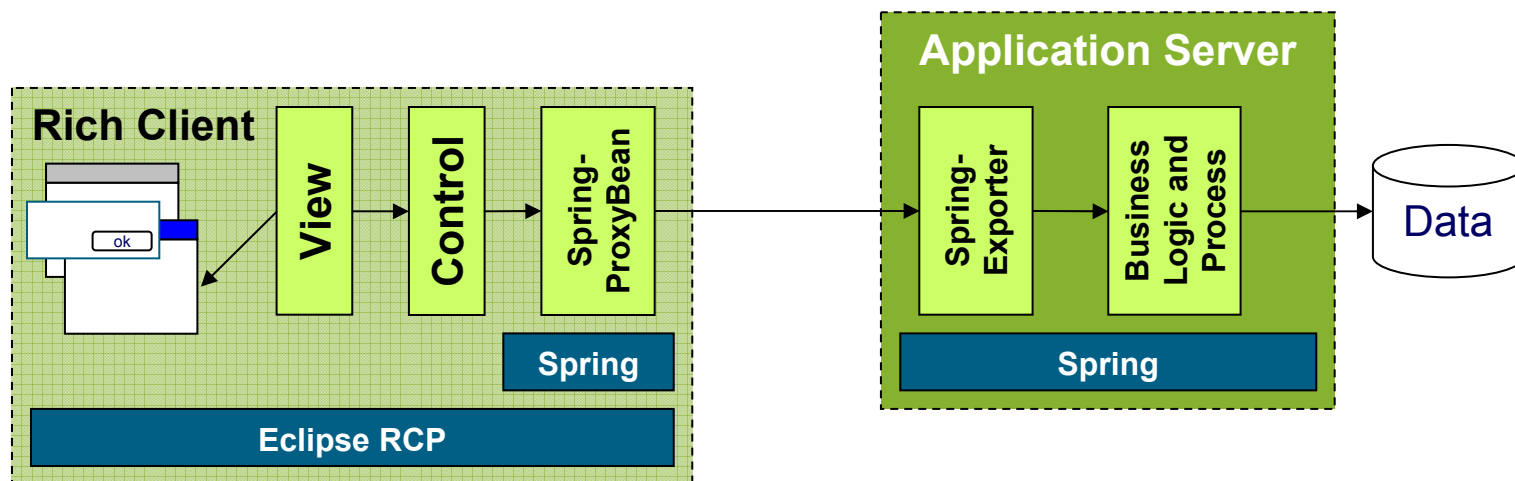
Spring and Equinox combined

- Easy to use
- But it is just the beginning, the base infrastructure
- The interplay with the Spring libraries need to be investigated in the future
- Classloading could cause problems with third-party libraries that are used by Spring
- Detailed information:
<http://www.springframework.org/osgi/>



RCP + Spring on the Client

- Client: Eclipse RCP + Spring/OSGi
- Server: Pure Spring
- Uses Spring/Remoting for remote communication
 - With all the possible variations (RMI, HTTPInvoker, Hessian, Burlap, etc.)



Evaluation

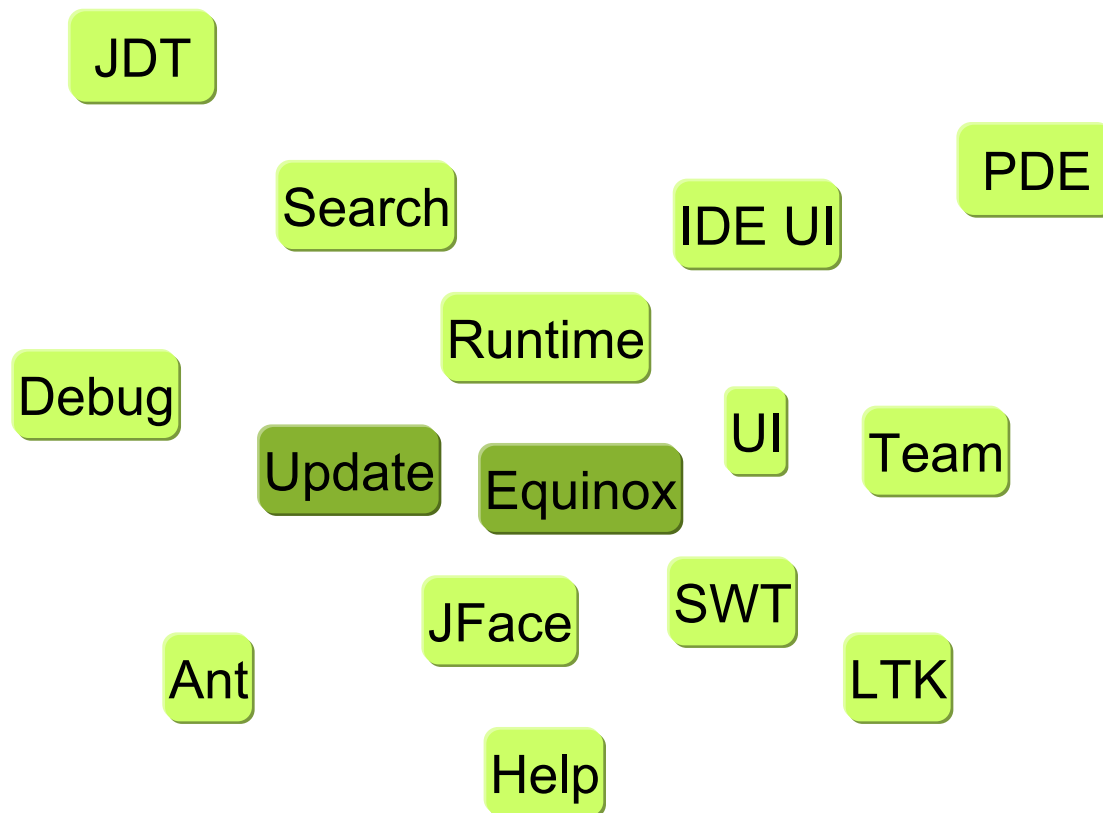
- 🍌 Unrestricted usage of Spring on the client **and** the server
- 🍌 Unrestricted usage of RCP on the client
- 🍌 Easy remote communication via Spring/Remoting on both sides
- 🍌 Still different deployment and programming models (OSGi bundles on the client, typical WAR file on the server)
 - Although most likely classes are shared between client and server

OSGi and Spring everywhere

Server-side Eclipse

- 🍃 Why use the Equinox component technology only on the client side?
- 🍃 Component model
- 🍃 Update mechanism
- 🍃 Extensibility
- 🍃 All interesting for server-side applications as well

Eclipse Rich Server Platform (RSP)



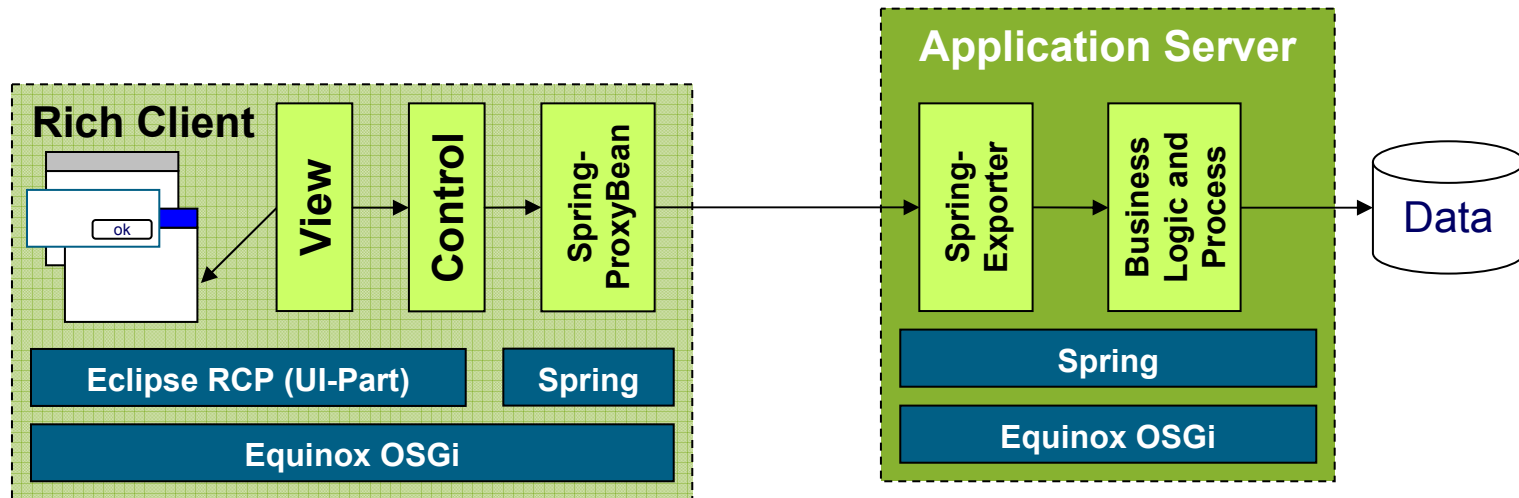
Server-side Equinox/OSGi is well accepted

- 🍌 WAS 6.1
- 🍌 Adobe Version Cue
- 🍌 IBM Rational JAZZ
- 🍌 Apache Harmony
- 🍌 Eclipse Rich AJAX Platform
- 🍌 ...

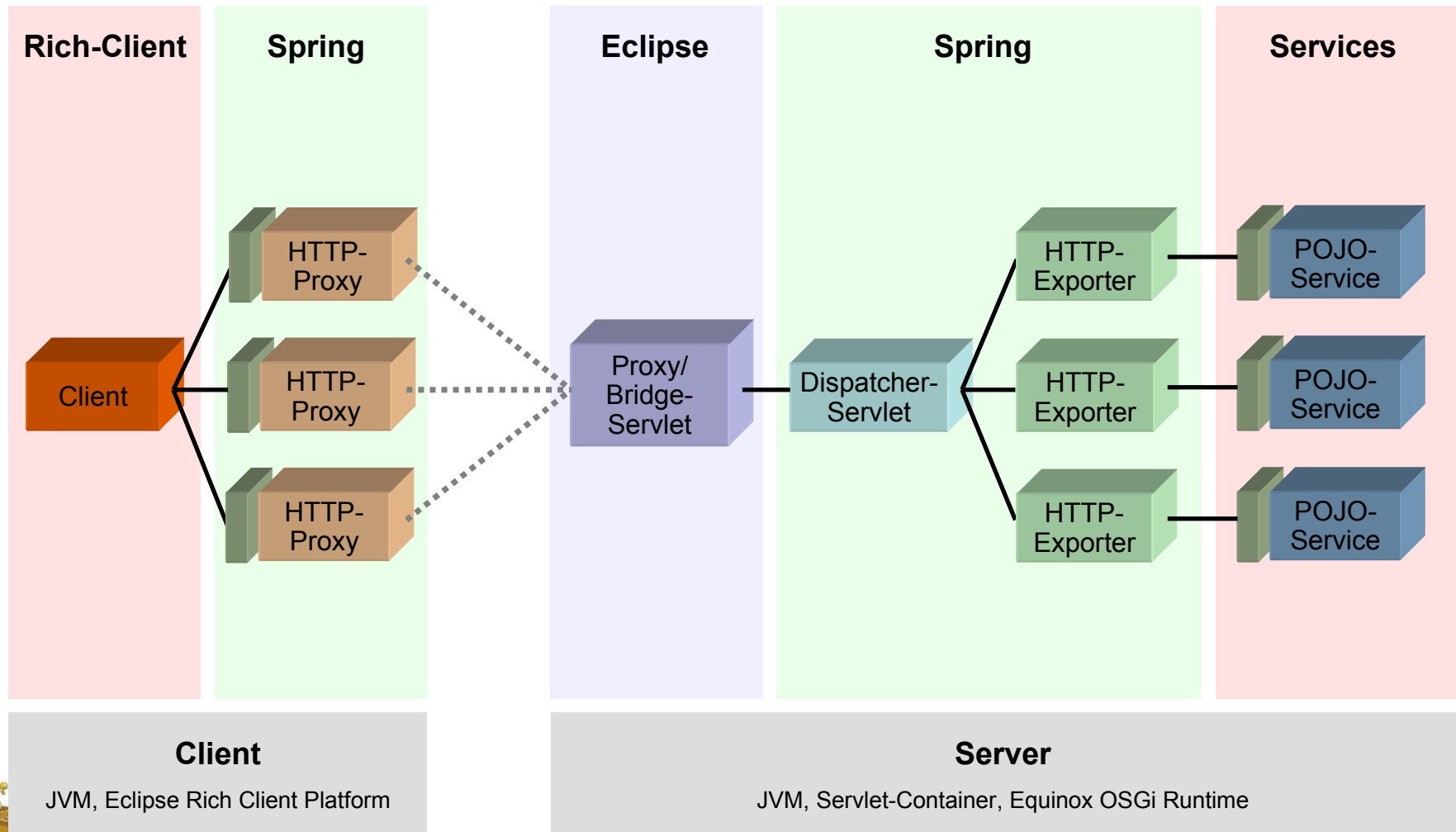


Middle-tiers on Equinox

- Equinox can be used to implement middle-tiers
 - Same component model on both sides
 - Same extensibility for both sides
- Client and server could share the same components
- Integration with web-/app-servers possible



Example: Remoting for POJOs



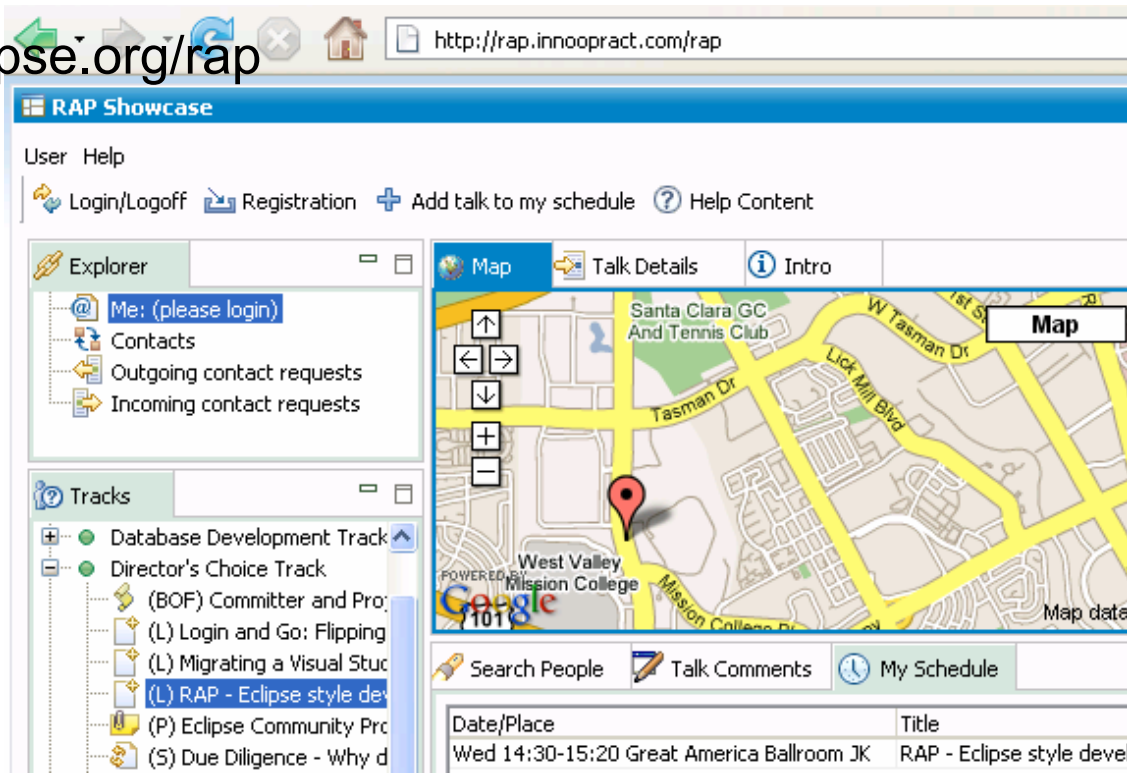
Equinox-based web apps

- 🍌 Equinox can run inside a web app or the web-app can run on top of Equinox
- 🍌 Web-app can be componentized
- 🍌 Web-app can be designed and implemented for extensibility (Extension-Points)

DEMO

Outlook: Eclipse Rich AJAX Platform

- Rich user experience framework for web-apps
 - Based on AJAX technology
 - Provides RCP-like user interface for web-apps (views, editors, dialogs, etc.)
 - <http://www.eclipse.org/rap>



More Spring on the RCP-based Client

More Spring on the Rich Client

- 🍌 How can we benefit from Spring on the client aside from Spring/Remoting?
- 🍌 Dependency injection and all other technology abstractions usable as well
 - Just straight forward using Spring/OSGi
- 🍌 How to incorporate this with the Extension-Registry?
 - For example, inject dependencies into views and editors?

The typical Extension Definition

- 🍌 We define a view via an extension
- 🍌 The view itself is created by the workbench via the extension registry on demand

```
<extension point="org.eclipse.ui.views">  
  <view  
    name="My View"  
    class="org.eclipse.example.rcpspring.MyView"  
    id="org.eclipse.example.rcpspring.view">  
  </view>  
</extension>
```

Defining the View via Spring

- Instead we would like to inject dependencies into the view
- Therefore we define the view “bean” within the Spring context

```
<bean id="injectedView"  
  
    class="org.eclipse.example.rcpspring.MyInjectedView">  
    <property name="businessService"  
        ref="businessService"/>  
</bean>
```


Adapt the Extension Definition

- Instead of the view directly we declare a factory in the extension definition

```
<extension point="org.eclipse.ui.views">  
  <view  
    name="My Injected View"  
    class="org.eclipse.example.rcpspring.  
                                                MyInjectedViewFactory"  
    id="org.eclipse.example.rcpspring.injectedview">  
  </view>  
</extension>
```

Creating an Extension Factory

- 🍌 The Extension-Registry now creates the factory instead of the view and calls `setInitializationData(..)` and `create()`

```
public class MyInjectedViewFactory implements
    IExecutableExtensionFactory, IExecutableExtension {

    public Object create() throws ... {
        return this.view;
    }

    public void setInitializationData(..) throws ... {
        this.view = (MyInjectedView)
            Activator.getAppContext().getBean("injectedView");
        this.view.setInitializationData(..);
    }
}
```



Side Note: Extension-Registry vs. DI

Extension-Registry:

- Designed to open-up specific parts of a component for extension
- Scalable through declarative metadata
- Extensions are tightly coupled to the extension point

Dependency Injection:

- Designed to de-couple classes
- No metadata, not necessarily designed for scalability
- Bean provider not tightly coupled to bean consumer, could have multiple bean consumers

Summary

Conclusion

- 🍌 A big step forward:
 - A homogeneous programming and deployment model through the usage of **Equinox/OSGi** and **Spring** for **Client and Server**
 - Eclipse RCP as UI framework for the rich client
- 🍌 Component model for client and server (through OSGi component model and Spring dependency injection)
- 🍌 Extensibility for client and server (through Extension-Registry)
- 🍌 Technology abstractions for client and server (through Spring)

 ***What else do we need? ;-)***

Eclipse-RCP and Spring are made for each other - you will never would like to work without them any more

Q&A

Martin Lippert, lippert@acm.org

Special thanks to Jeff McAffer for feedback and material

Recommended RCP Reading

- 🍌 Eclipse Rich Client Platform
 - By Jeff McAffer and Jean-Michel Lemieux
 - Addison-Wesley Professional
 - ISBN: 0321334612
- 🍌 SWT : The Standard Widget Toolkit, Volume 1
 - By Steve Northover, Mike Wilson
 - Addison-Wesley Professional
 - ISBN: 0321256638
- 🍌 Contributing to Eclipse: Principles, Patterns, and Plugins
 - By Erich Gamma, Kent Beck
 - Addison-Wesley Professional
 - ISBN: 0321205758

