



Spring and Eclipse RCP

Martin Lippert

akquinet agile GmbH

lippert@acm.org

Equinox Incubator Committer

Agenda

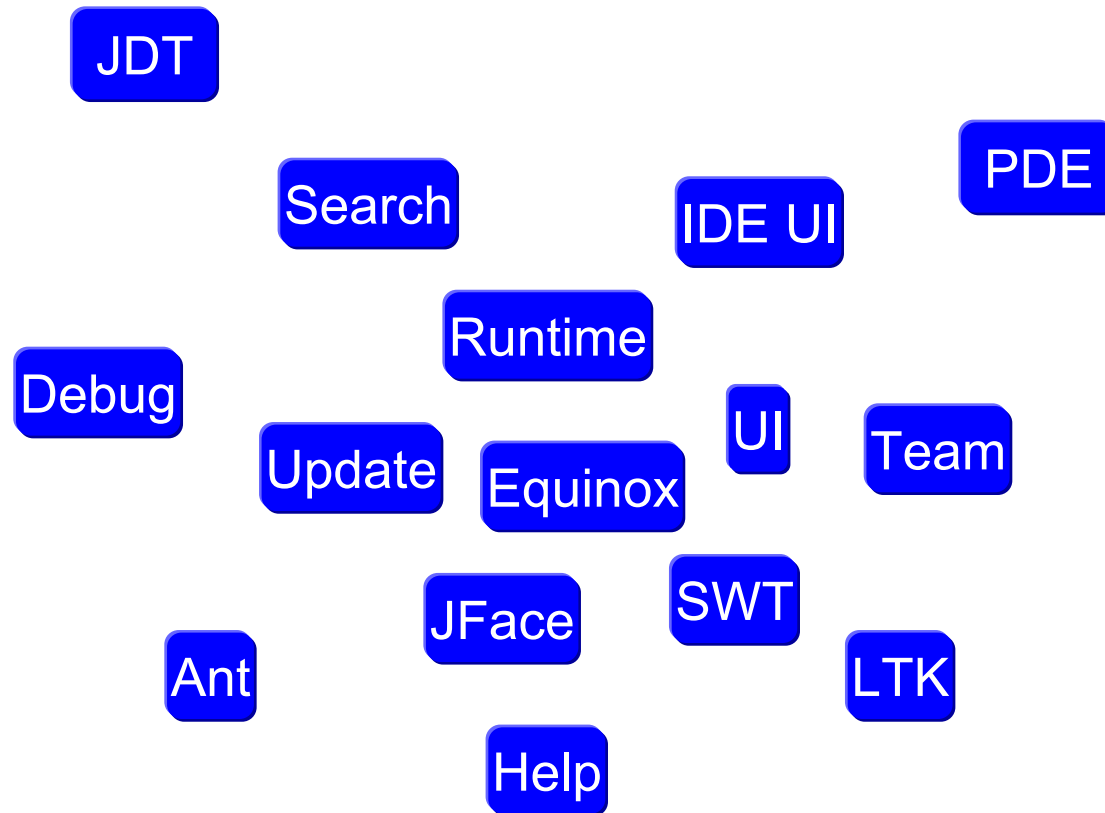


- What is the Eclipse Rich Client Platform?
- Spring and Eclipse RCP
 - Spring Backends for Eclipse RCP
 - Eclipse RCP + Spring on Client and Server
 - OSGi and Spring everywhere
 - More Spring on the Client
- Conclusions



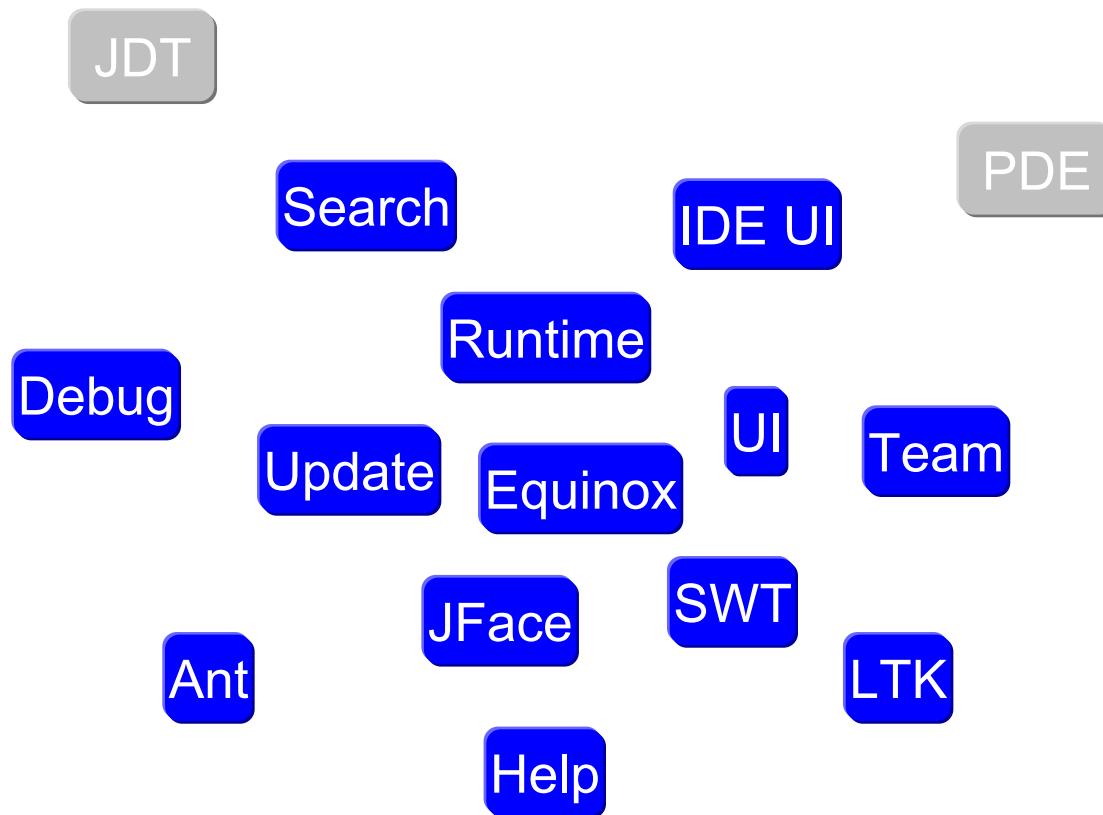
What is Eclipse Rich Client Platform?

Eclipse is a Composition of Components



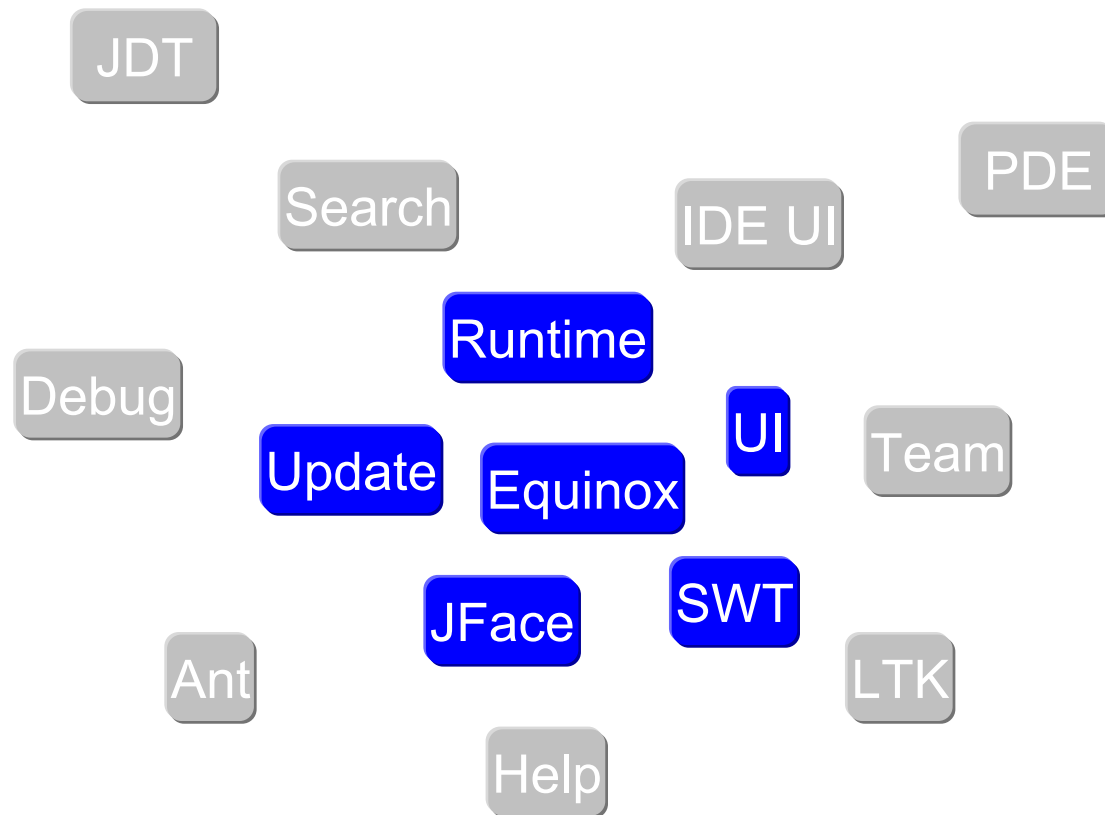
Eclipse SDK/Java IDE

Generic IDE Components



Eclipse IDE

Eclipse Rich Client Platform

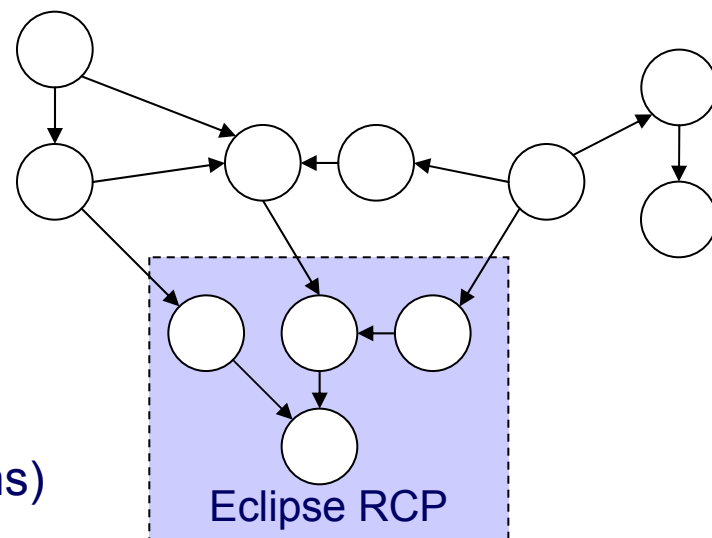


Why Use Eclipse Rich Client Platform?



- 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)
 - Pervasive extensibility – Extension registry
 - Update Manager
 - 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

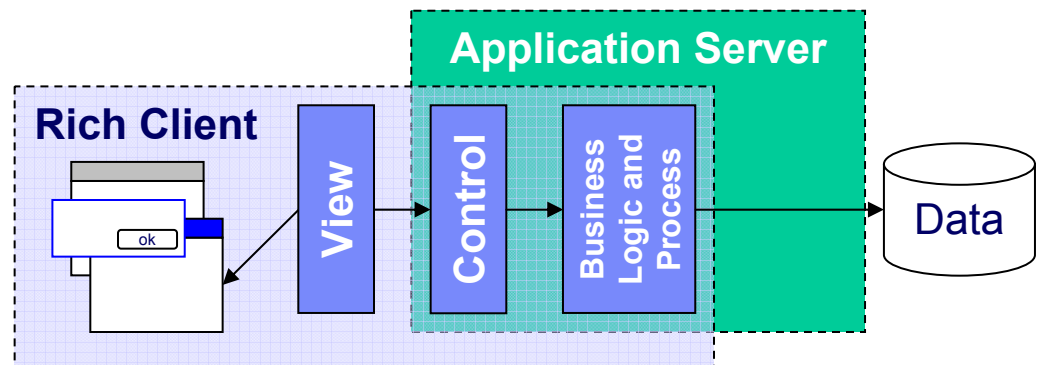
- Equinox is the Eclipse component model
 - Based on OSGi R4 specification
 - Standard Java lacks an explicit notion of components
- Components == Bundles == Plug-in
 - Versioned
 - Defined declaratively
 - Dynamically loadable/unloadable
 - Support dynamic update and install
- Explicitly define
 - Dependencies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



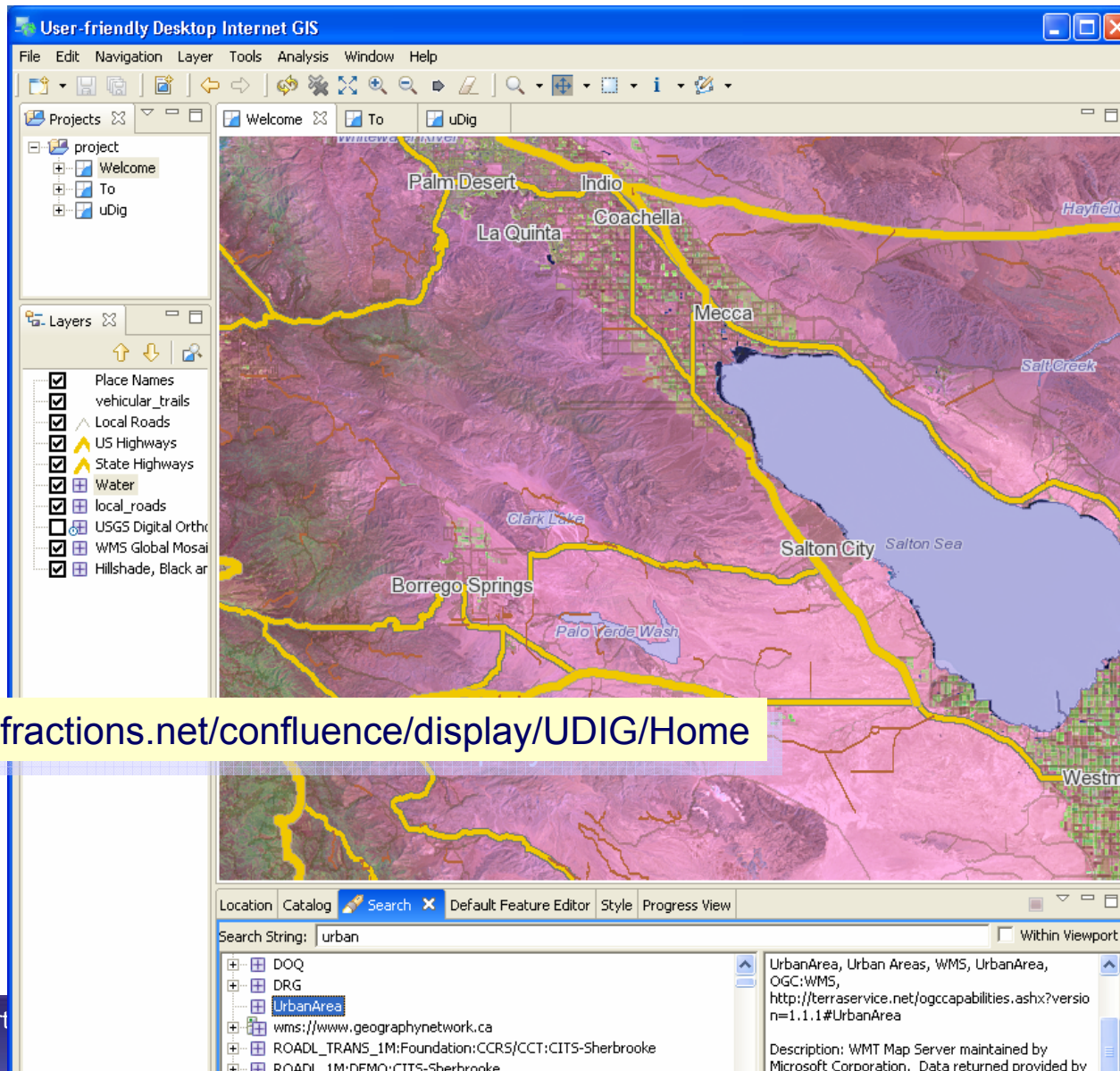
The typical RCP Application



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



Example: GIS

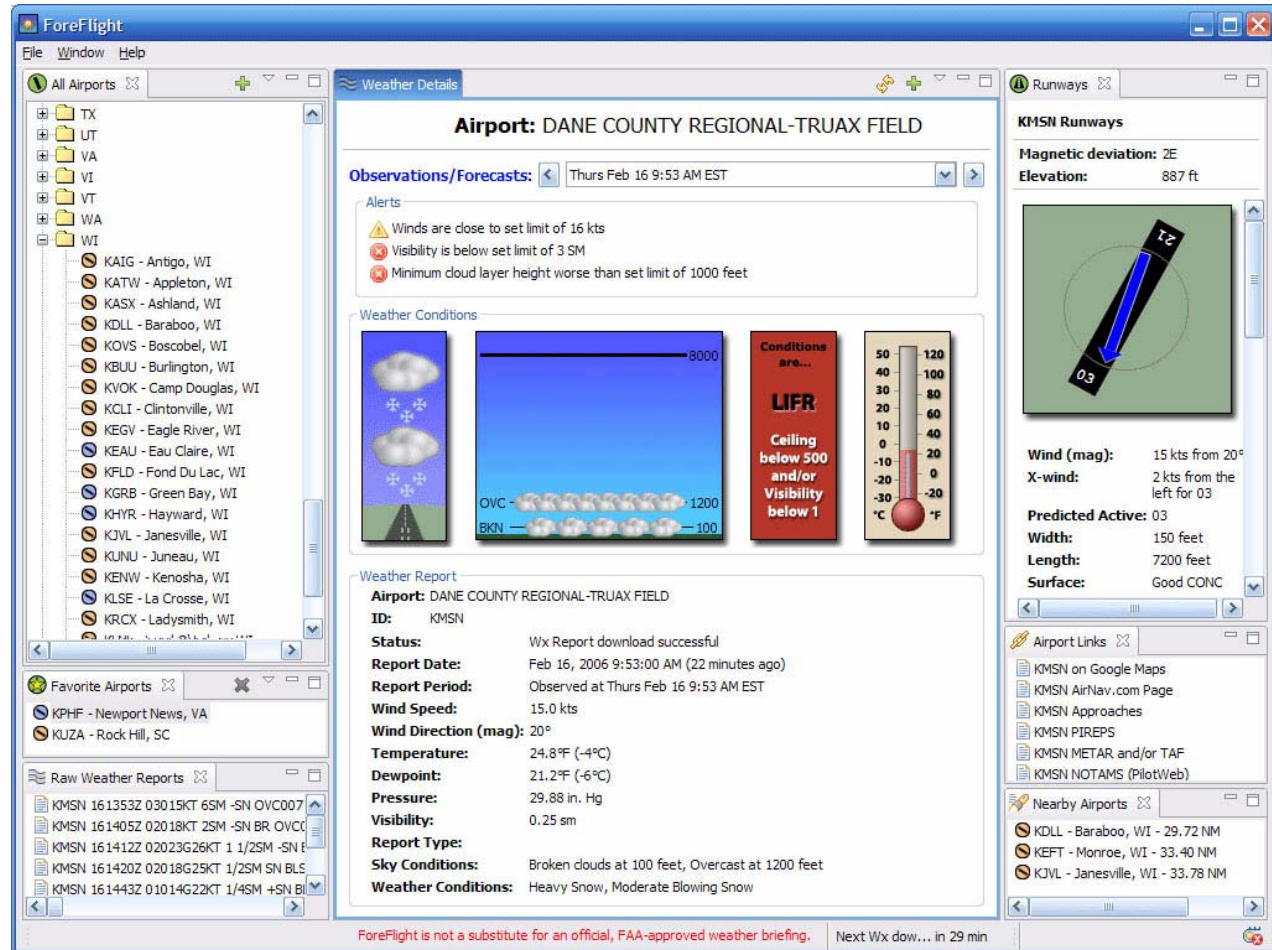


<http://udig.refrations.net/confluence/display/UDIG/Home>

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”



Sam Curman - Inbox - IBM Lotus Notes

File Edit View Action Tools Window Help

Welcome Mail 4 Activities 5 Calendar 2 Contacts Sales Leads 2

Go to More

Search: [] New Reply Forward Collaborate Follow Up Organize By Date

Sam Curman

- Inbox (12)
- Drafts
- All Messages
- Sent
- Follow Up
- Junk Mail
- Trash
- Views
- Folders
- Tools

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 Today, 10:02AM

From: Lukas Geiger
To: Sam Curman, Pierre Dumont, Anna Bauer, Laura Klein
cc: Monifa Shani

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

Related Activities

- OP Tools deal
- Sales Lead: OP Tools
- More on OP Tools deal
- Competitive products
- 5/01/05 Sam Curman...

Contacts

Available

- Related Contacts (1)
- Pierre Dumont
- Anna Bauer
- Monifa Shani
- My Contacts (26)

Related RSS Feed

- Reuters Business Headlines
- Sales Discussion
- Commercial Development N...

Calendar

Today is May 01, 2006

- Related meetings
- Today 1:00 OP Deal Meeting
- May 05 1:30 Meeting with Monifa
- May 19 3:30 OP Tools Meeting
- Today's meetings
- 9:30 Business Discussion
- 1:00 OP Deal Meeting

Online

Example: RSS Solutions



- Advanced planning and scheduling (APS) solutions



Example: Maestro – NASA Space Mission Management



Maestro

File Edit Window Downlink Help

Orbital View

EDR Search View

Sol Range Start End

Instruments FRONT_HAZCAM_LEFT FRONT_HAZCAM_RIGHT Use Selected Orbital Region

Go

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
2F135149189EDN2700P1141L0M1	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

Image View Image View Image View Image View

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

The image shows the Maestro software interface. The main window displays a grayscale image of the Moon's surface with various sites marked by yellow labels (Site: 1 through Site: 33) and connected by a green line. A blue box highlights a specific area on the map. The right-hand side features an 'EDR Search View' panel with a 'Sol Range' dropdown, a list of instruments ('FRONT_HAZCAM_LEFT' and 'FRONT_HAZCAM_RIGHT'), and a table of product IDs. The bottom right corner shows a small 'Image View' window displaying a close-up of the lunar surface.



Spring Backends for Eclipse RCP

Spring-based Backends

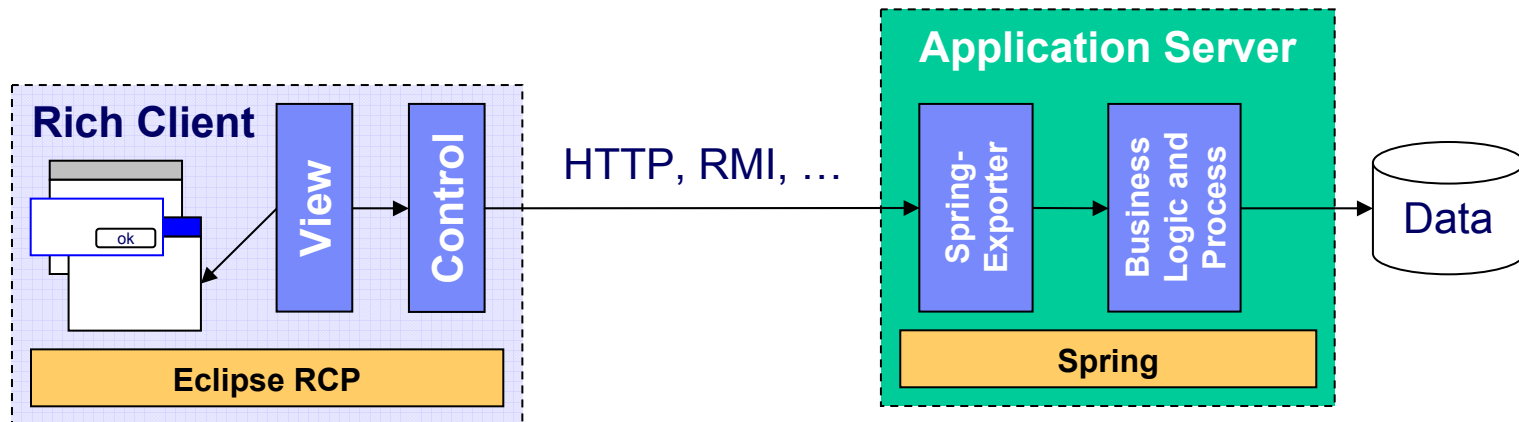


- Spring is great for implementing the backend:
 - Dependency injection for the implementation
 - AOP for cross-cutting and interceptor-based features
 - Easy transaction handling
 - Easy integration of other technology like O/R mapping, security, and so on...
- *It is a natural choice to implement the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

Pure RCP Client for the Spring Backend



- Client: Pure RCP
- Server: Pure Spring
- Ways to communicate, for example:
 - Server provides RESTful/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



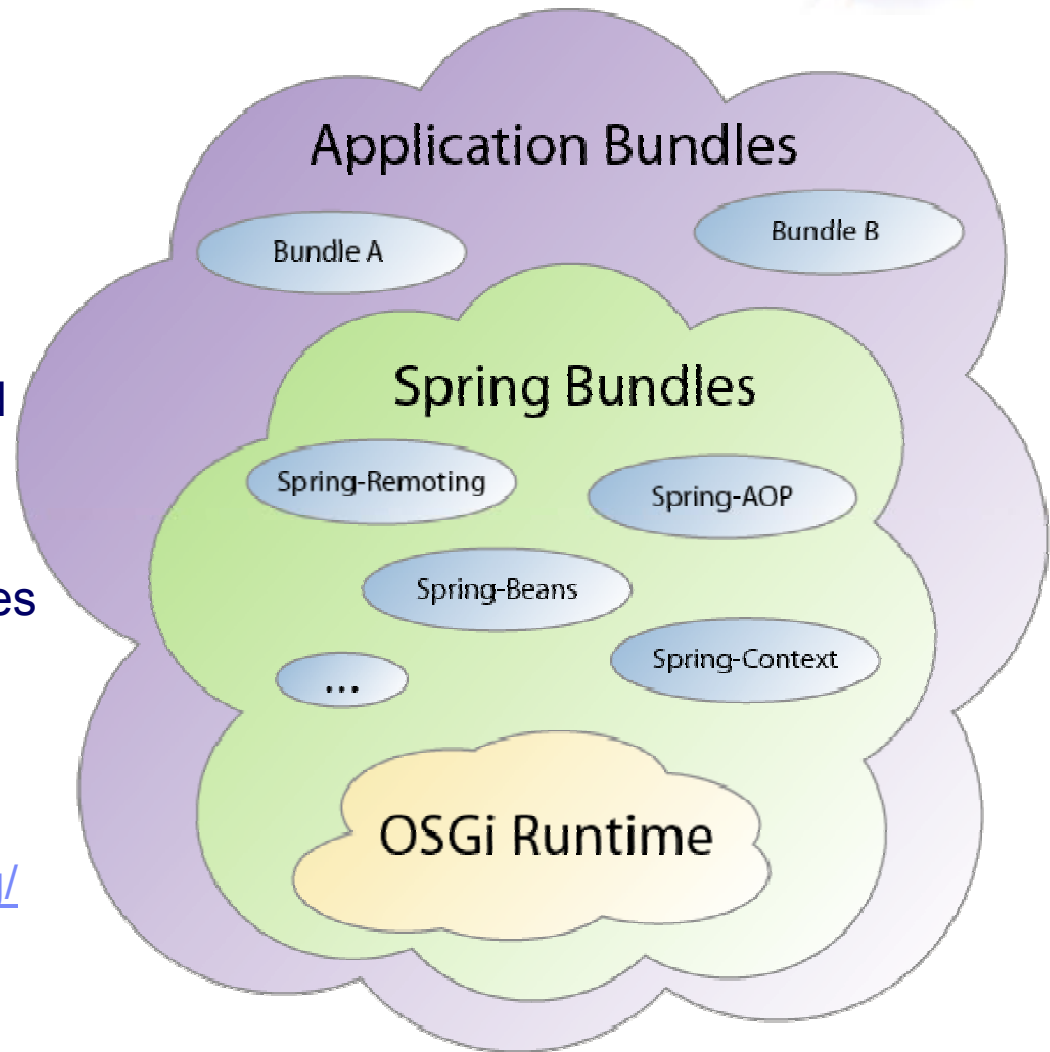
The Spring-OSGi bridge

- 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



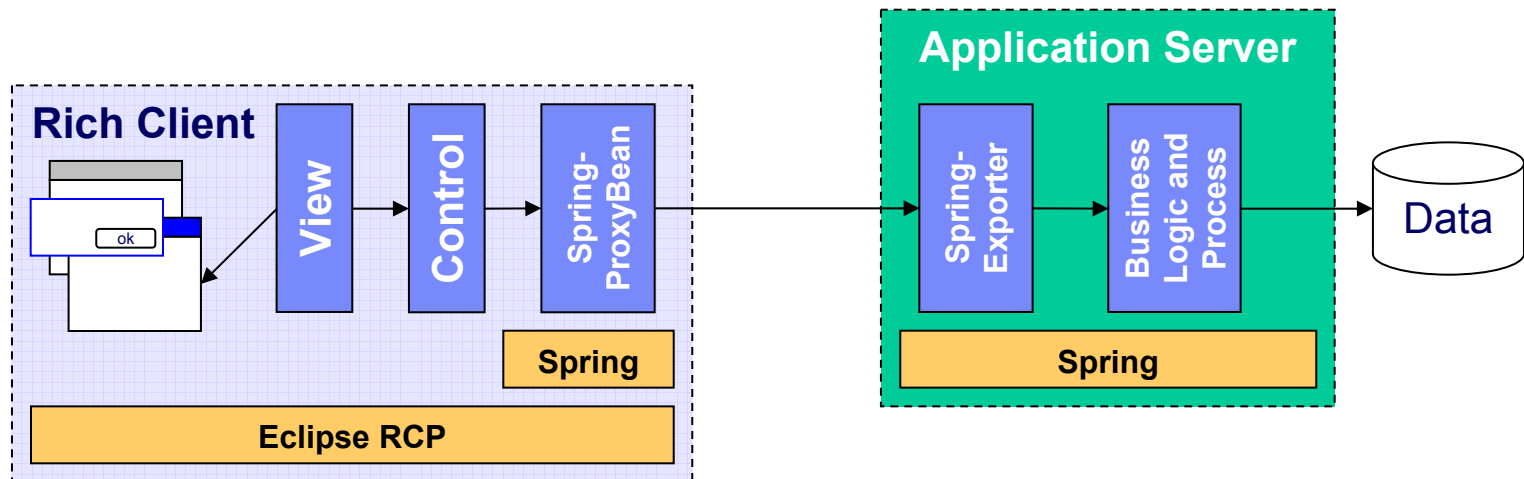
- Easy to use
- But it is just the beginning, the base infrastructure
- The interplay with the Spring libraries need to be investigated in the future
- Classloading could cause problems with third-party libraries that are used by Spring
- Detailed information:
<http://www.springframework.org/osgi/>



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

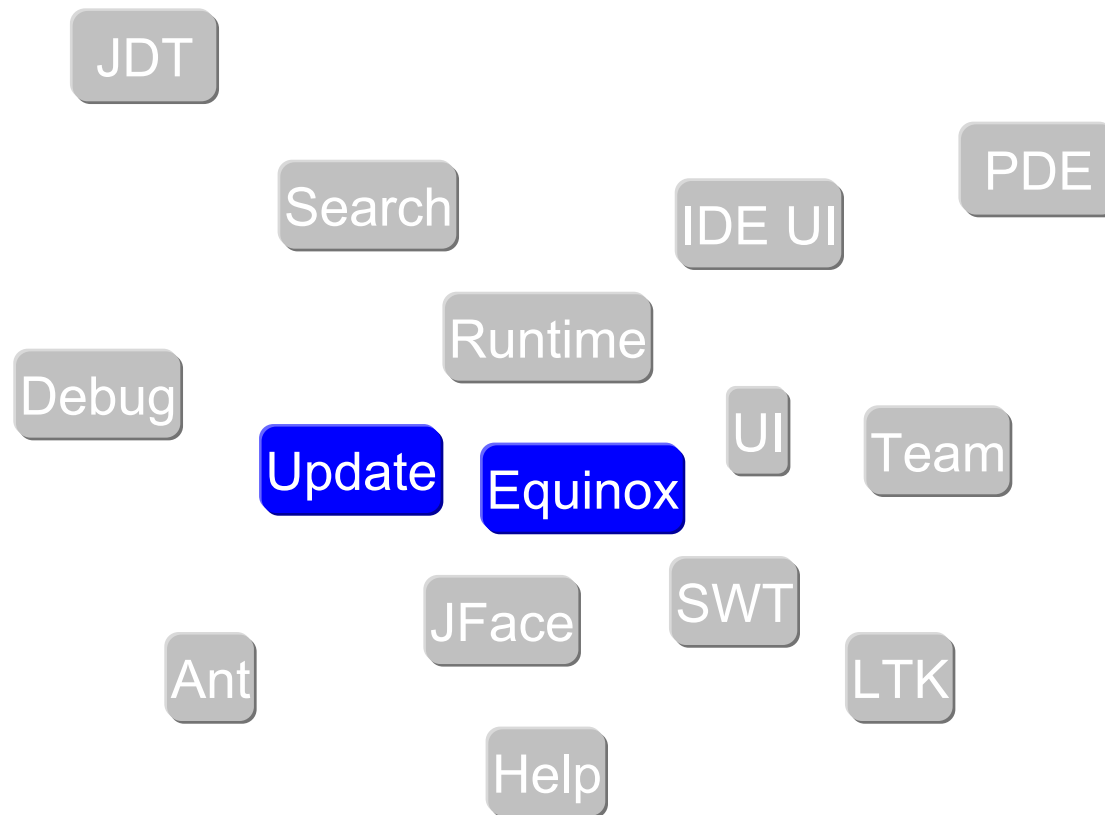


- Unrestricted usage of Spring on the client **and** the server
- Unrestricted usage of RCP on the client
- Easy remote communication via Spring/Remoting on both sides
- Still different deployment and programming models
(OSGi bundles on the client, typical WAR file on the server)
 - Although most likely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



OSGi and Spring everywhere

Eclipse Rich Server Platform (RSP)



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

Server-side Equinox/OSGi is well accepted...



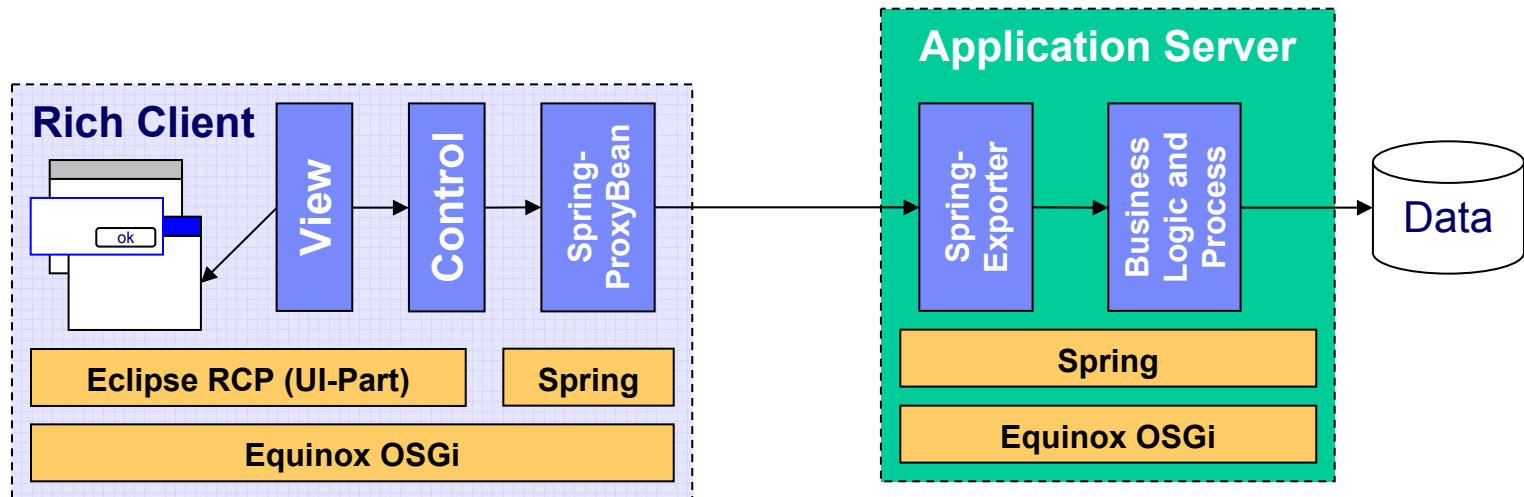
- WAS 6.1
- Adobe Version Cue
- Apache Harmony
- Eclipse Rich AJAX Platform
- ...



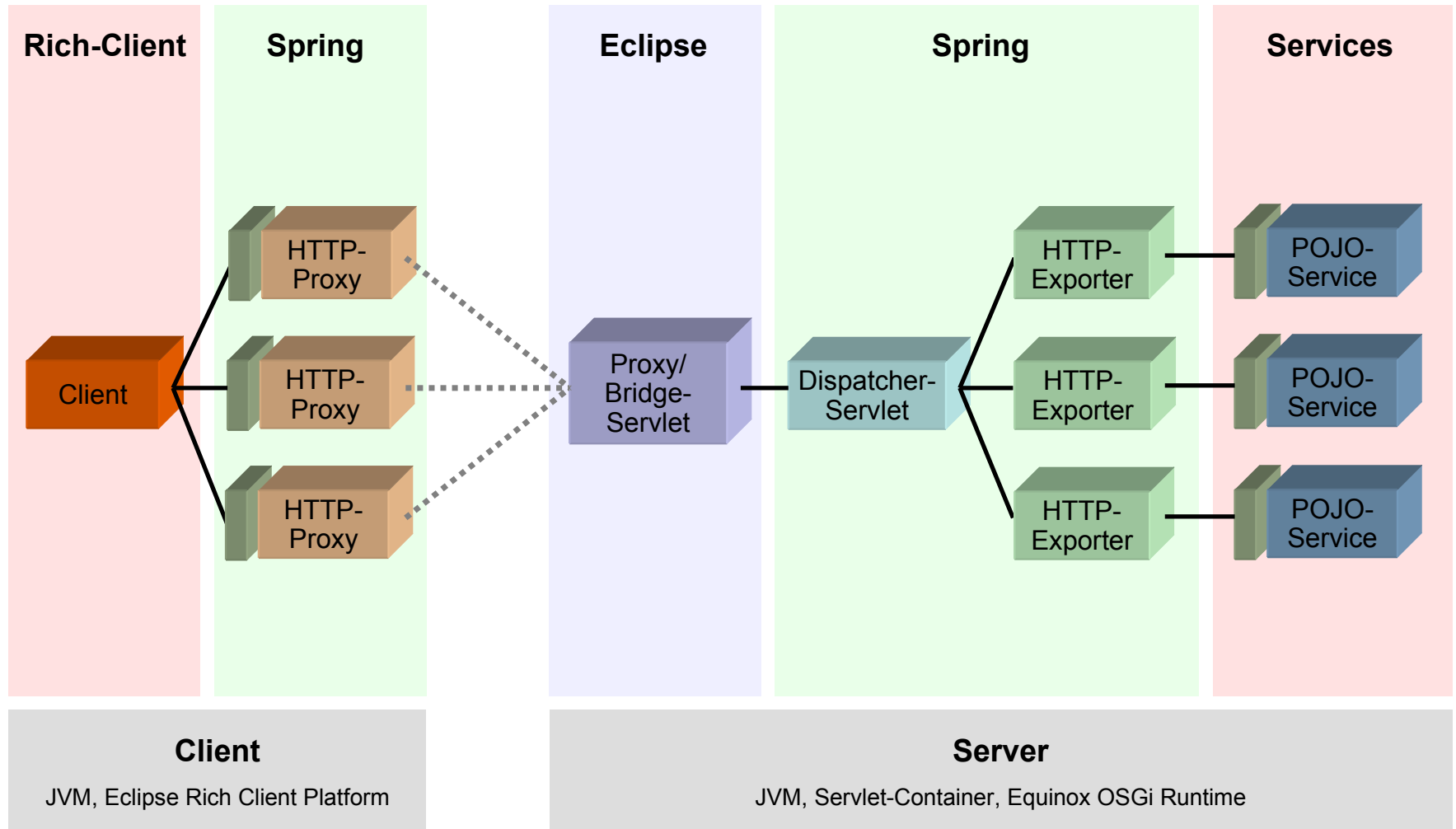
Middle-tiers on Equinox



- 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)



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

- Dependency Injection:
 - Designed to de-couple classes
 - No metadata, not designed for scalability



Conclusions

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? ;-)***

Thank you for your attention!



- Questions welcome !!!

Special thanks to Jeff McAffer for feedback and material

Martin Lippert

lippert@acm.org

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

