eclipse rich ajax platform (rap)

Jochen Krause
CEO Innoopract
Member of the Board of Directors Eclipse Foundation
jkrause@innoopract.com

outline

- rich ajax platform
  - project status and background
  - technology overview

eclipse rich ajax platform project

rap aims to enable developers to build rich, AJAX-enabled web applications by using the eclipse development model, plug-ins and a java-only api

eclipse plays a significant role in the rich client world

- provides advanced concepts and technologies that can be easily built upon
- „enforces“ solid architecture (e.g.: promotes creation of apis, loose coupling)

the goal of the project is to extend the reach of the eclipse platform to the web

project status: approved in June 2006, now in validation phase

- large part of code contribution from Innoopract has been approved (now in CVS)
- no builds yet, small code contribution still waiting for approval

eclipse rich ajax platform project - continued

rap did not start from scratch, the code contribution brings:

- w4t, a widget toolkit that allows development of ajax web ui's in plain java
- technlology has proved to be stable, e.g. with the eclipse download configurator service http://yoxos.com/ondemand - handling 500 concurrent users easily
- the project has received the

jax INNOVATION AWARD 2006 winner

- The award honours and recognises the most remarkable and outstanding european contributions in the world of Java and Eclipse.

current trends in application development

- the most commonly applied technology for developing user interfaces in the past decade, templating for (simple) HTML, is getting replaced by two new major trends:
  - rich client applications (with concepts for keeping the client up to date)
  - rich internet application, with a strong focus on ajax technologies
- eclipse has succeeded in delivering a state of the art rich client framework, but the rich client camp is getting pressure from ajax enabled webapps
- the ajax world to date is very colorful, with many very promising technologies and projects. Most of the effort seems to be focused on providing client-side widget toolkits and a communication layer to the server.

why rap? ajax suffers from dev. complexity

- although ajax is a promising vision, the development complexity is very high
- better tools can help
  - e.g. eclipse atf http://eclipse.org/atf/
  - better javascript editors are desperately needed (this can be an area for collaboration)
- frameworks and toolkits can deal with the low level stuff
  - qooxdoo js gui framework
  - Kabuki Ajax Toolkit
  - Dojo
  - OpenRico

how does rap compare to google gwt?

google gwt is a cool technology
- provides a java api
- running on an emulated java engine in the browser (needs javascript to work)
- javascript is in charge of „drawing“ the user interface
- eventhandling in GWT is on the client side (+ RPC calls to the server to access data)
- GWT enables a "standalone SWT" comparable approach
- can scale to 100 thousands of concurrent users

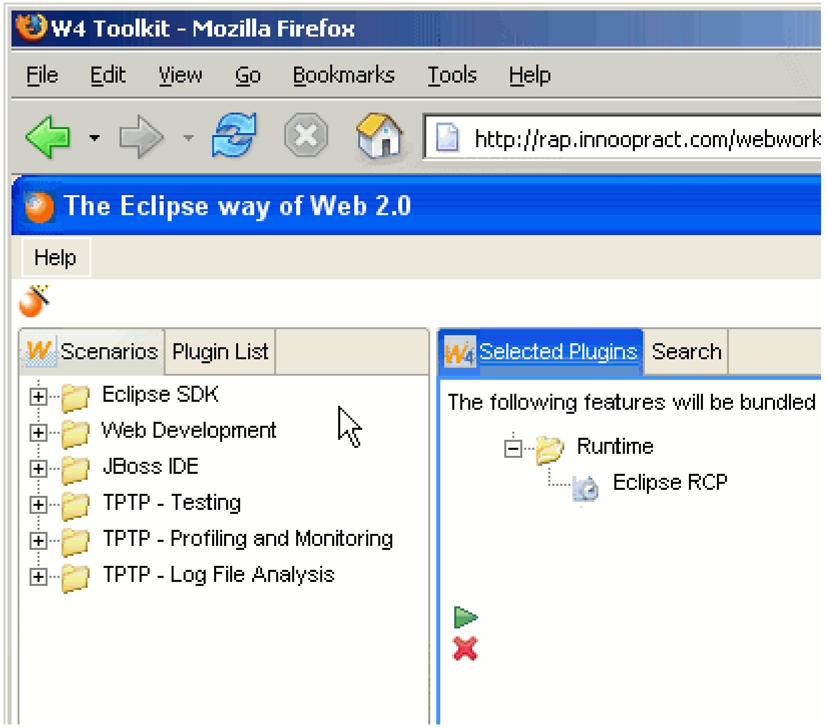
rap is a cool technology, too
- provides a java api
- runs standard html and javascript in the browser (can work with javascript disabled)
- the browsers rendering engine „draws“ the ui, refreshes happen through transfer of html snippets
- RAP relays most client-side events to the server for processing (solely ui related events can be processed on the client).
- running mainly on the server it can access the full java api enable the full usage of the eclipse plugin model
- can scale to thousands of concurrent users

outline

- rich ajax platform
  - project status and background
  - an eclipse platform strategy

rap leverages and extends the eclipse platform

- rap enables developers to employ the eclipse concepts for developing ajax applications, leveraging the advanced eclipse programming model
- **plugin concept** – based on osgi, implemented by Eclipse Equinox
- **workbench concept** – a powerful UI metaphor that facilitates providing a consistent user experience
- a **widget toolkit** encapsulates all ajax technologies behind Java objects and rendering kits
- only developers who want to create their own widgets need to deal with javascript and ajax
- eclipse as a platform becomes an attractive alternative for ajax develop-ment – not only for ajax tooling



a brief example

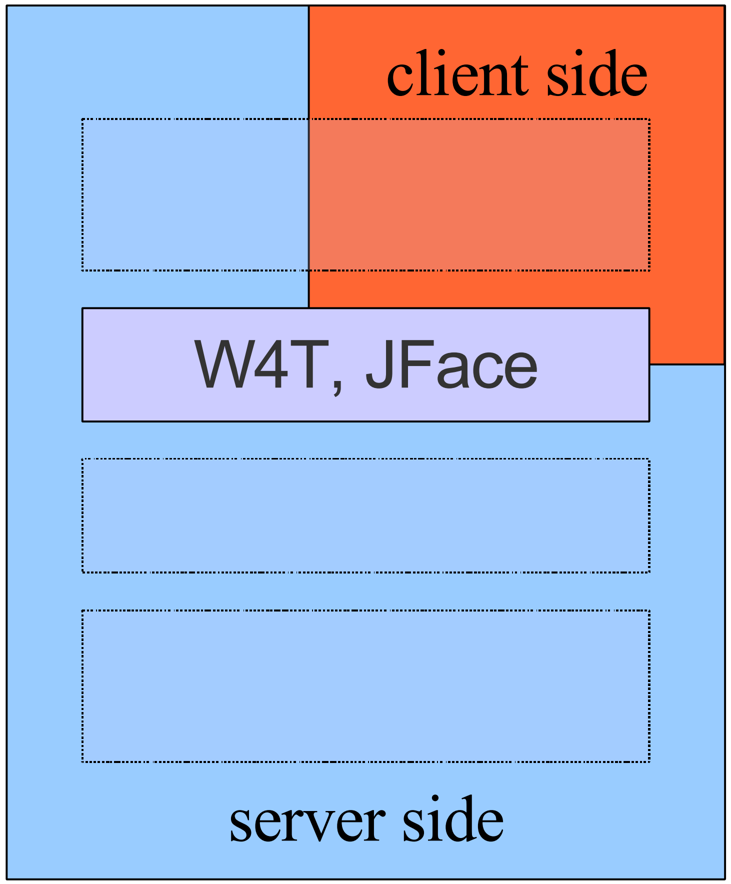
- webworkbench – look & feel of the eclipse workbench in a browser
  - adding type ahead search

- implementation is not yet based on the eclipse workbench model
  - „hand-coded“ workbench, like creating the workbench look & feel directly in swt
  - showcasing feasibility, performance, look & feel

DEMO

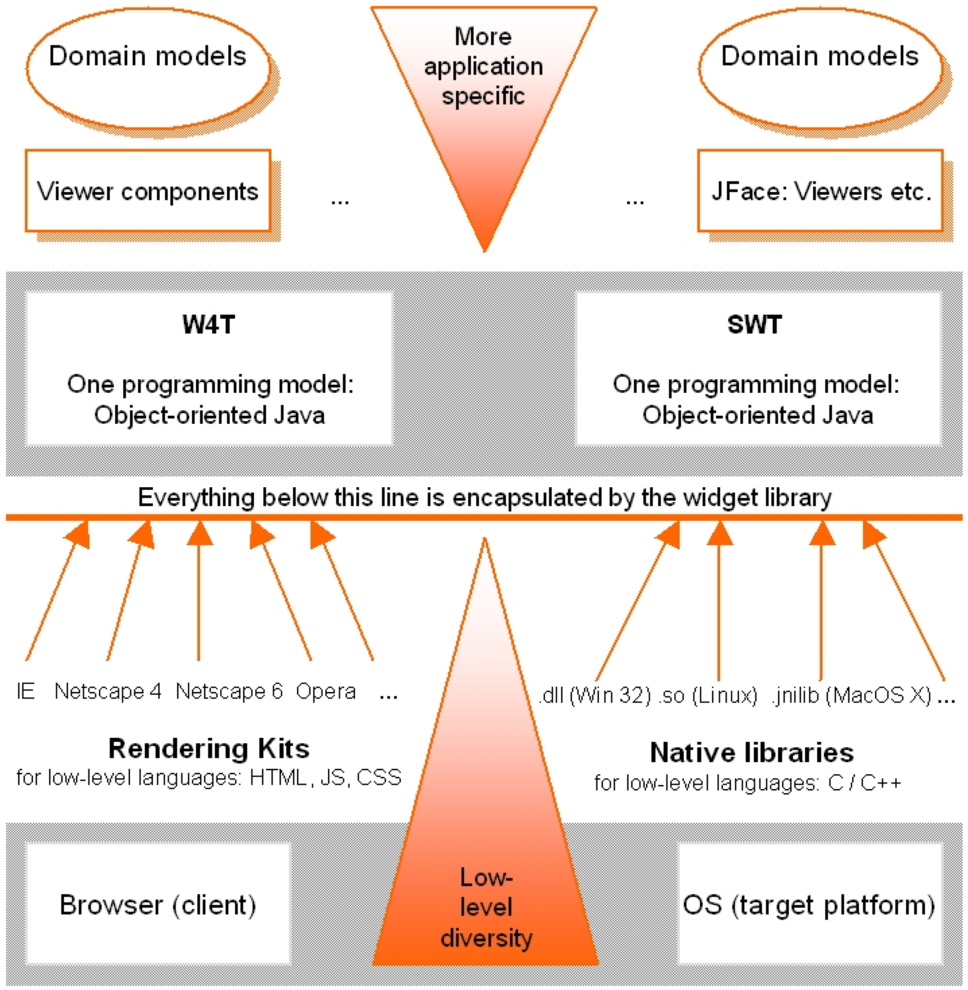
see http://yoxos.com/ondemand/

rap architecture overview



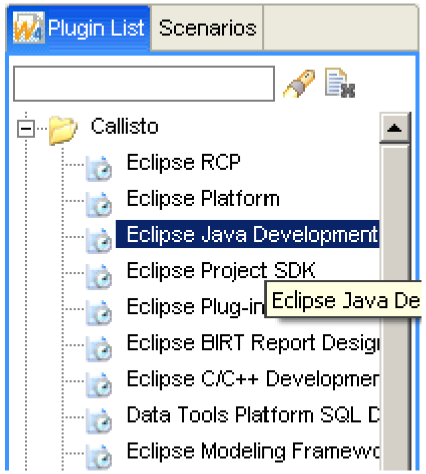
➔ widget toolkit, mvc, handling of distributed environment

www widget toolkit explained

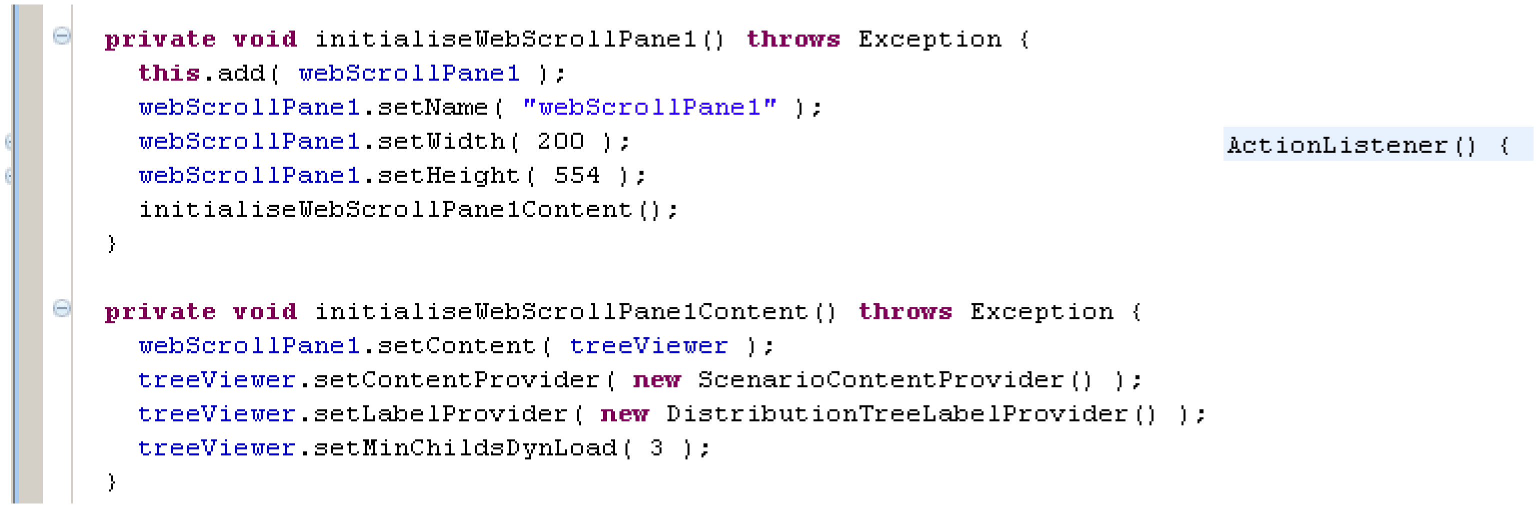

Domain models
More application specific
Domain models
Viewer components
...
...
JFace: Viewers etc.
W4T
One programming model: Object-oriented Java
SWT
One programming model: Object-oriented Java
Everything below this line is encapsulated by the widget library
IE
Netscape 4
Netscape 6
Opera
...
.dll (Win 32)
.so (Linux)
.jnilib (MacOS X)
...
Rendering Kits
for low-level languages: HTML, JS, CSS
Native libraries
for low-level languages: C / C++
Browser (client)
Low-level diversity
OS (target platform)

wigdet toolkit – www windowing toolkit (w4t)

- OO – programming interface
- composition of wigets into a component tree
- event driven ui
- lifecycle management of the request
- AJAX engine
- rendering kits
- userdefined components



```
Callisto
    Eclipse RCP
    Eclipse Platform
    Eclipse Java Development
    Eclipse Project SDK
    Eclipse Plug-in  Eclipse Java De
    Eclipse BIRT Report Desig
    Eclipse C/C++ Developmer
    Data Tools Platform SQL D
    Eclipse Modeling Framewc
```

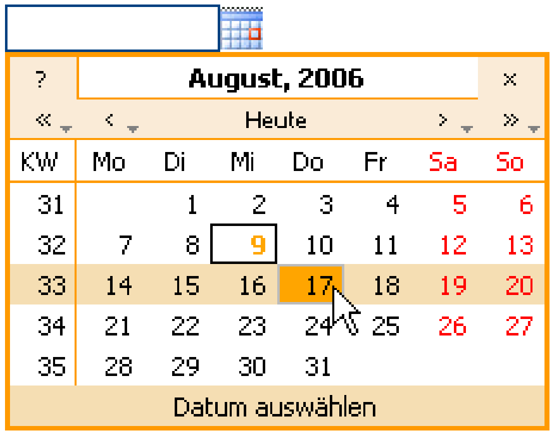


```
private void initialiseWebScrollPane1() throws Exception {
  this.add( webScrollPane1 );
  webScrollPane1.setName( "webScrollPane1" );
  webScrollPane1.setWidth( 200 );
  webScrollPane1.setHeight( 554 );
  initialiseWebScrollPane1Content();
}

private void initialiseWebScrollPane1Content() throws Exception {
  webScrollPane1.setContent( treeViewer );
  treeViewer.setContentProvider( new ScenarioContentProvider() );
  treeViewer.setLabelProvider( new DistributionTreeLabelProvider() );
  treeViewer.setMinChildsDynLoad( 3 );
}
```

```
ActionListener() {
```

extensible – user defined components

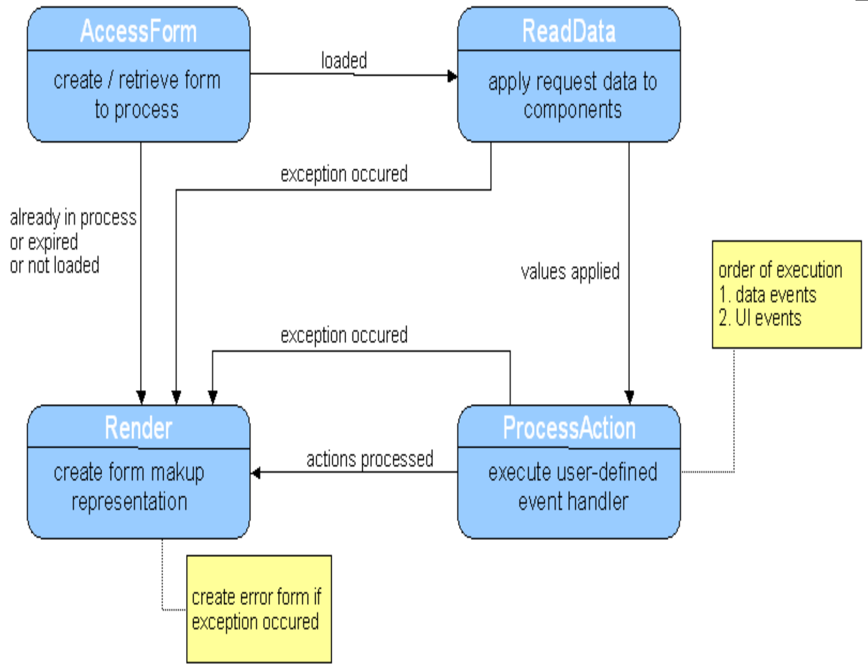


KW	Mo	Di	Mi	Do	Fr	Sa	So
31		1	2	3	4	5	6
32	7	8	9	10	11	12	13
33	14	15	16	17	18	19	20
34	21	22	23	24	25	26	27
35	28	29	30	31			

Datum auswählen

- extend WebComponent
- implement rendering kits for markup generation
- reuse of existing js libraries

life cycle (request management)



AccessForm
create / retrieve form to process
loaded
ReadData
apply request data to components
exception occured
already in process or expired or not loaded
values applied
order of execution 1. data events 2. UI events
exception occured
Render
create form makup representation
actions processed
ProcessAction
execute user-defined event handler
create error form if exception occured

- 4 phases of request handling
  - access form
    versioning of the WebForm
  - read data
    read request data,
    apply data to model
  - process action
    process user action
  - render
    create markup, update ui components
    that represent model data

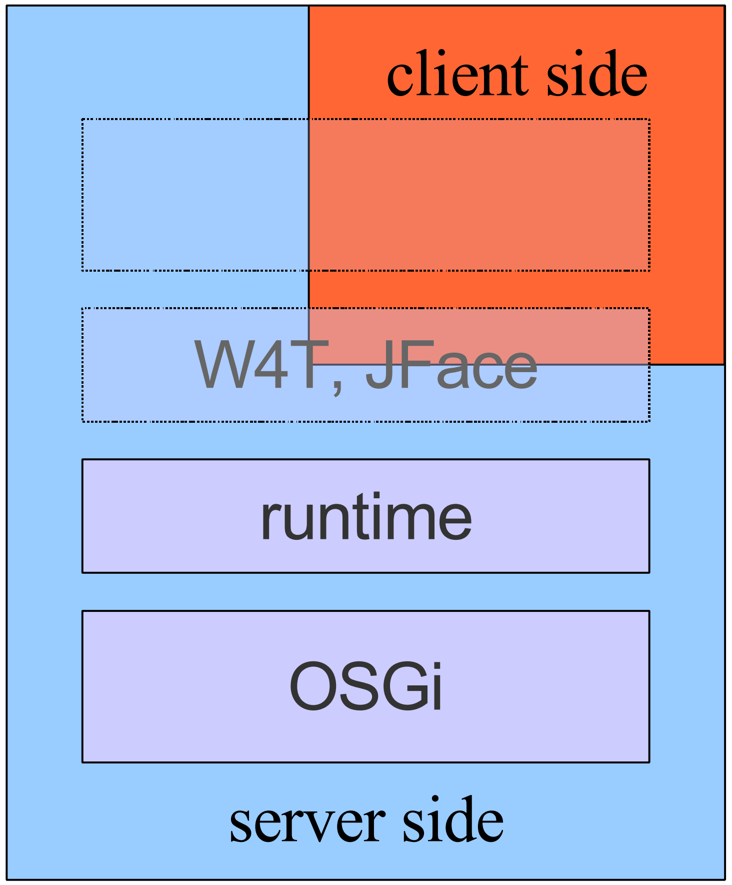
ajax engine for partial ui updates

- send request (client-side)
  - collect form data and submit via XmlHttpRequest
- detect changed components (server-side)
  - hash code based algorithm to trace component state
  - renders only markup for widgets that need to be updated
- apply response (client-side)
  - received HTML fragments are applied to document
- transparent for application developer

rendering kits

- targeted output for a variety of browsers (ie, firefox, opera, safari)
- AJAX renderer for partial page updates
- script renderer for browsers with AJAX-functionality turned off
- noscript Render as fallback for browsers with strict security settings
- dynamic loading based on namespaces

rap architecture overview



➔ extension points

➔ modularity, dependency management (bundles / plugins) **based on standard jee technology**

eclipse OSGi on the server side

- equinox is providing an „incubator“ for running eclipse inside a web app and interacting with a servlet
  - server side integration - main problems have been solved and are part of Eclipse 3.2
  - embedding in a servlet container
    - war file to demo is available – starting an eclipse platform server side http://www.eclipse.org/equinox/incubator/server/eclipse_serverside_integration.php
  - rap is mainly reusing equinox technology and act as a client for this project
- a rap sample application using equinox is available for download
  - can be launched with equinox launcher using a equinox http bundle
  - shows reuse of a common core plugin between rcp and rap
  - http://wiki.eclipse.org/index.php/RapExamples

eclipse runtime - on the server side

- late bindings
- declarativ
- loose coupling
- contributing
- extending existing implementations

- ONCE per web application (alternatively running osgi as a server with http service)

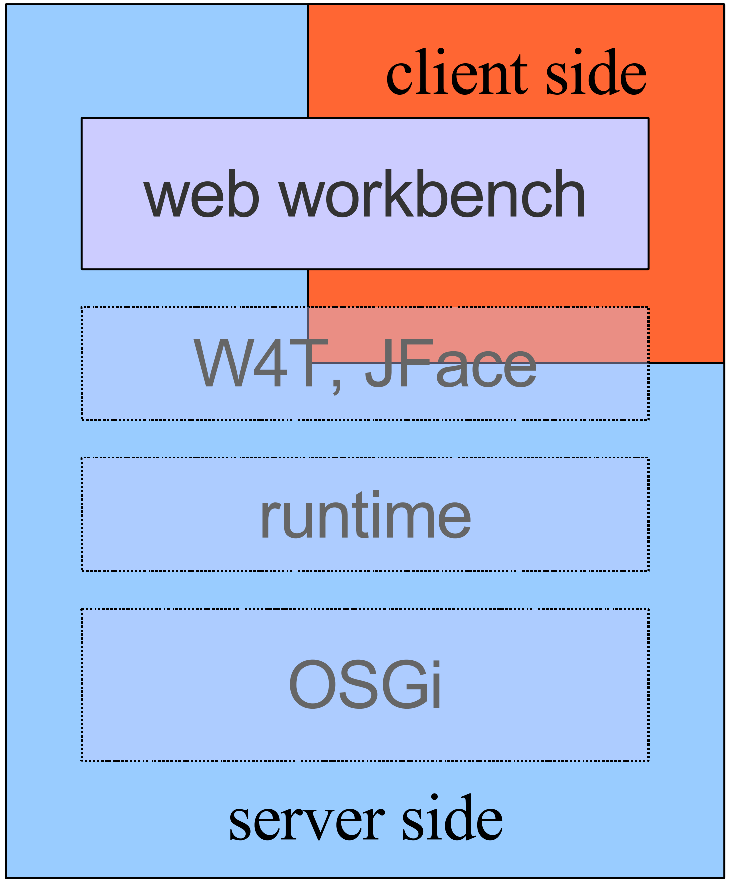
taking plugins to web applications

- eclipse plugin concept is enabled on the server side inside a web app
- everything is a plugin (server side)
  - core plugins can be reused if they are stateless
  - ui is assembled by contributions (server side) providing a well thought out development model

DEMO

see http://wiki.eclipse.org/index.php/RapExamples

rap architecture overview



➔ selection service, action sets, viewparts

➔ widget toolkit, mvc, handling of distributed environment

➔ extension points

➔ modularity, dependency management (bundles / plugins)
based on standard jee technology

workbench

- strong coupling between workbench, swt and jface in rcp
- need to reimplement core apis for rap to align with widget toolkit (swt api under exploration)

challenges ahead:

- workbench
  - session vs. application scope
  - memory considerations
  - multi user / logins
- layouts
  - absolute positioning, formLayout
- integration with existing web applications

the (right) balance of server and client side

- framework provides client-side handling for workbenchparts (open, close, resize)
- widgets can provide client side event processing (e.g. expanding a tree)
- other event processing happens (mostly) server side
  - implementation in java
  - ui changes are calculated on the server side, client will get partial updates
- data binding happens on the server side (jface is the eclipse standard)

collaboration with other ajax approaches is important

rap benefits from simple integration of widgets based on common ajax frameworks

- rap's widget toolkit is extensible
    - server side java api (might be moving to swt)
    - rendering kits provide implementation (html, css, js)
    - a canvas can be filled with client side life
    - server side needs info about client state

problems to avoid:

- possible incompatibilities between different libraries
- different versions of libraries

rap plan

tentative planing:

- 2006-06 - 2006-09 initial code contribution: Java component library for UI development
- 2006-09 M1: OSGi running exemplary inside web applications on selected open source servers
- 2006-10 M2: Moving widget toolkit to org.eclipse packages, re(de)fine widget toolkit api (get involved: https://bugs.eclipse.org/bugs/show_bug.cgi?id=158930)
- 2007-01 M3: Basic WebWorkbench implementation running on OSGi
- 2007-03 M4: Provide all API for Release 1.0
- 2007-05 RC1: Code freeze for 1.0

conclusion

- ajax is here to stay, but it has yet to overcome some obstacles
- ajax does not need to be in contradiction with rich clients – the technologies can complement each other
- shielding ajax complexities is one of the hottest topics today – a java api (swt) has proved to work in rich ui development, but there is also a strong movement to build javascript libraries

- give rap a try - http://eclipse.org/rap/

references

- Eclipse RAP project http://eclipse.org/rap/
- Eclipse Rich Client platform http://eclipse.org/rcp/
- Eclipse ATF project http://eclipse.org/atf
- Google Web Toolkit http://code.google.com/webtoolkit/
- qooxdoo – JavaScript GUI framework http://qooxdoo.org

Q&A

Contact info:

http://eclipse.org/rap/

Jochen Krause

jkrause@innoopract.com