Eclipse 4.0

Jochen Krause
jkrause@eclipsesource.com
EclipseSource
based on EclipseCon 2008 talk by
Mike Wilson, Jochen Krause, Jeff McAffeer, Steve Northover
e4 – adapting to a changing world
A lot of promise, but how about these good old questions

• How to develop effectively?
• Quick develop / test cycle
• Modular, extensible applications
• Consistent APIs
• API stability
• ....
e4 – adapting to change

• As a tool
  – Java is here and will stay for quite some time
  – Enabling launch on the cloud
  – JS support is getting there

• As a platform
  – RCP is very successful
  – Modularization for core and ui components
  – Need to make it easier
  – We can bring modularity to desktop and web
  – High style
The mission of the e4 project is to build a next generation platform for pervasive, component-based applications and tools.
Themes

• Build a better desktop
  - Identify and fix issues in 3.x that prevent new uses
  - Make it easier to build and deploy plug-ins
  - Improve styling capabilities

• Bring Eclipse to the web
  - “Enable Eclipse-based web application development” not “Take over the world”
  - Work with the larger web community
Commitment to 3.x

• We are committed to protecting your investment in the current SDK and RAP
  ✷ Ongoing development in 3.x for >5 years
    ▪ Targeted enhancements, bug fixes, new platforms
    ▪ e4 items that are backwards compatible
  ✷ 3.x plug-ins that use public API will generally run in „Desktop-Mode“ in e4
  ✷ Co-existence as long as needed
    ▪ Think Apache 1.x and Apache 2
Build a better desktop
Make it easier to build plug-ins

• Architectural improvements, API cleanup
  - More uniform APIs, no singletons, finer-grained plug-ins,
    More flexible resource model
    ▪ But support backwards compatibility
  - Dependency injection
    ▪ Likely use an existing framework (e.g. Spring)

• Ability to implement plug-ins in non-Java languages
  - Initially JavaScript
  - + wiring them together
    ▪ i.e. “plugin.js” versus plugin.xml
  - Support scripting

• Ability to model the workbench (using EMF)
Flexible styling of Eclipse desktop applications

- Separate appearance from content
- Simplifies the Workbench API
  - DOM-style model for the desktop widgets
  - Life-cycle managed by the model
  - Modifying the model causes immediate presentation changes
  - Easier to drive from multiple languages
- More radically and with greater ease modify the look of Eclipse
- Use a separate, pluggable styling engine
  - CSS would allow sharing of styling information between desktop and related web pages

```css
tab {font-family: Verdana; height: 23px; }
tab.active {start-color: #afc0eb; end-color: #7a96df; }
tab.inactive {start-color: #ffffff; end-color: #ece9d8; }
```
Demo build by Kai Tödter based on e4 prototype
Eclipse Application Model

• A well-defined and *documented* set of services
• Captures *needed* functionality from Platform API
  - Selection, D&D, Progress, Jobs, Preferences, Logging, Model Listeners
  - Reasonable size (we say, “the 20 things”)
  - Easy to understand
  - Passed to plug-ins via dependency injection
• Extensible
  - Standard ways to provide additional services in apps built on e4
• RESTful architecture
  - Accessible from Java, JavaScript or http
• Multi-user enabled
Bring Eclipse to the Web
Browsers will become a viable platform
We are not starting from scratch

• RAP has pioneered SWT in a browser
• RWT (RAP widget toolkit) is a strict subset of SWT
   Enables code reuse on SWT level
   BUT: SWT API is targeted to rich client and not all aspects map well to the web, some API is none existent (styling)
• Widgets are “remoted” to the browser
   Eventhandling is mostly done on the server
• Custom widgets allow integration of other JavaScript (e.g. Gmap)

Our joint e4 effort will become more flexible:

Richer client technologies, pure web style development
Demo modeled workbench in a browser
Extend ability to run SWT in a browser

• The web is the new platform(s)
  - JavaScript frameworks (Dojo, qooxdoo, ...)
  - Rich “connected” platforms (AIR, Silverlight)

• If you have SWT code, you should be able to re-use it
  - Widgets are not web pages, but that can be ok

• Take web specifics into account for SWT API

• We will support different ways to solve this:
  - GWT-style, “UI on browser, app on server”, with multiple backends (AIR, Dojo,...)
  - RAP-style, “‘X windows’ remoting to UI and app on server”
Align efforts for SWT4 and RAP

• Styling
   Subset of CSS

radius: 10, 20, 30, 40
width: 3, 10, 20, 5
color: "red", "green", "yellow", "blue"
backgroundColor: "gray"
Align efforts for SWT4 and RAP (cont'd)

• RAP on SWT Browser Edition
  - Mix compiled components (running on the client side) and remoted components / workbench (directed from the server side)
Mashups

• Attempting to solve the same problems as OSGi
  - Secure composition of (UI) components from multiple sources
  - Must be lightweight, simple
  - Existing communities (e.g. OpenAJAX “Hub 1.1 / SMash”)
• The Eclipse version would be...
  - Use Eclipse application model + define “OSGi for the web”
  - Extend p2 to support provisioning to browser
    ▪ “Zero-thought” install
  - Provide standard JFace/DataBinding Table, Tree, etc. equivalents but allow arbitrary web UI technologies to be used
• We don't own this space
  - Must work with existing web community
  - Lots of opportunity for participation
Current state and outlook
Timeline

• e4 was announced at EclipseCon 2008
• An e4 summit was held to define e4 more closely and to get the work started
• Technology evaluations have been going on in the e4 incubator component of the Eclipse Project
  ✷ A modeled workbench (using EMF)
  ✷ CSS styling of a workbench
  ✷ SWT on Flex (Browser Edition)
• Project has been created on October 3, 2008
• There is working code!
Timeline cont'd

• Individual work areas move at own pace
  - Graduate as they become ready
• But have overall “e4” platform builds with regular milestones
  - Need to sync up with changes in 3.x code
    - Equinox team has done this – learn from them
• Checkpoint / re-assess after 1 year
  - Are we working on the right things?
    - Have we made the kind of progress we need?
• Deliver in 2 years
  - BUT: This work will influence the 3.x stream
  - You don't need to wait
Resources

• http://wiki.eclipse.org/E4
• https://dev.eclipse.org/mailman/listinfo/e4-dev
• http://www.eclipse.org/rap/

Demos Howto

• http://wiki.eclipse.org/E4/Running_the_demos