Spring and OSGi

Martin Lippert
akquinet agile GmbH
lippert@acm.org

Bernd Kolb
b.kolb@kolbware.de

Gerd Wütherich
gerd@gerd-wuetherich.de

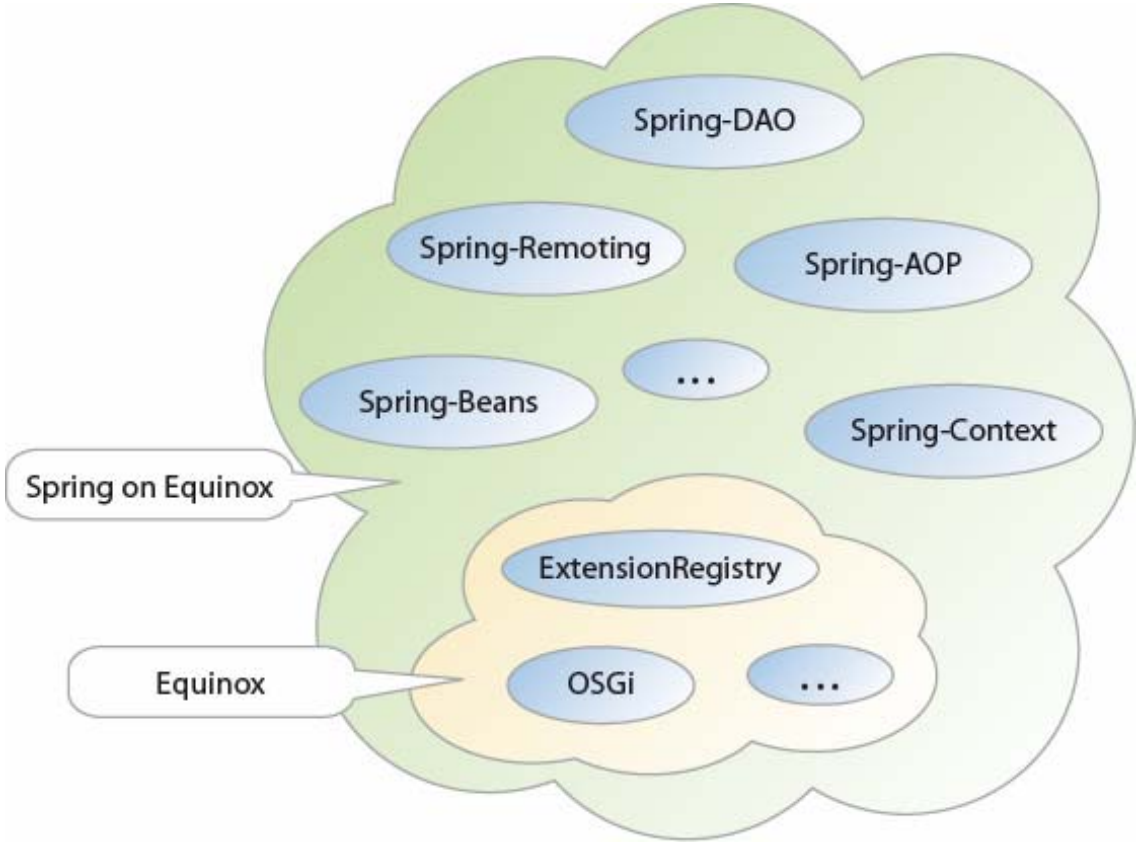
Server-Side Eclipse

- Eclipse OSGi is recognized as platform for server-side use
  - Pure server-side applications (web, etc.)
  - Middle-tier of distributed applications
  - Etc.
- See the many examples from Jeff and others…

Spring

- A widely used framework for enterprise Java applications
  - Not only for the server-side
- Provides:
  - Dependency Injection
  - AOP
  - Many technology abstractions to be used in a POJO like way

Eclipse + Spring



How to combine both worlds?

- With Spring we define beans via application context definitions (XML files)
  - Typically one application context per application
  - Dependencies and injections are defined statically
- Within OSGi we have separate bundles instead of a single application
  - One application context per application does not fit into the OSGi world
  - The dynamic nature of OSGi seems to be problematic for Spring as well

A powerful combination

- The idea:
  - Spring is deployed as a bundle
  - A spring application context definition per bundle
- Spring beans and OSGi services
  - Spring beans can be exported as OSGi services
  - OSGi services can be imported as Spring beans
- Standard Spring proxy mechanism used to deal with dynamically installed and deinstalled bundles

Current state

- The Spring team at Interface21 is working on the OSGi bridge for Spring
  - Take a look at the Google group “spring-osgi”: http://groups.google.com/group/spring-osgi
- The OSGi bridge will be part of the Spring 2.1 release
  - Alpha version available
  - https://svn.sourceforge.net/svnroot/springframework/spring-osgi/trunk

A live demo

- Equinox deployed within a web-app
  - Using the servlet bridge
- Spring deployed as a bundle
  - Into this web-app
- Using the Spring dispatcher servlet for remote services
  - Remoting via HTTP, for example

Advantages

- Modularity using OSGi
- Dependency Injection via Spring
- Spring technology abstractions usable for OSGi-based apps

Challenges 1/2

- Classloading, of course:
  - Spring uses a huge number of open-source libraries
  - Some of them don‘t work together with the OSGi classloading
- Dynamic nature of OSGi:
  - Spring proxy mechanism works fine, but:
  - What happens to other libraries that are used
    - (today Hibernate is not able to deal with dynamically added or removed stuff)

Challenges 2/2

- Scalability
  - Today the Spring OSGi bridge reads and initializes a spring context definition file when a bundle is activated
  - Therefore the Spring OSGi world seems to need eagier activation of bundles to create the universe of available services
  - This does not fit nicely into the idea of scaling the bundle und extension idea up to thousands of plugins

Using the ExtensionRegistry with Spring and OSGi

- Currently the eclipse extension mechanism is **not integrated** in the Spring OSGi bridge
  - Since it is for pure OSGi use
- It would be desirable to support flexible architectures based on Spring specific extension-points and extensions for server-side applications
  - Imagine extension points that create spring definitions automatically
- First Spring & OSGi & extension registry prototypes available.
- Need for further discussion in the community.

Experiences

- A large insurance application build on top of Equinox OSGi
  - Including domain-specific extension points all over the place
  - Adding new insurance products via extensions (including services, business objects, UIs, persistence adaptors, etc.)
  - Those extensions are plugged into the client as well as into the server middle tier
- Spring is used to distribute the domain services
- Client and server middle tier:
  - The same structure, the same technology, the same build