ApacheCon Europe

Stuttgart 2005

Tom Riley - FlyMine

Axis2 - history

- First there was Apache SOAP
  - Experimental. Slow (DOM).
- Then there was Apache Axis
  - Allowed access to headers. Faster (SAX).
  - Handlers. Better API.

Axis2 - improvements

- Document centric interactions as opposed to request / response.
- Asynchrony - long running tasks.
- StAX - more memory efficient.
- Can include binary in messages.
- Some support for REST.
- Support for alternative transport mechanisms. SMTP, JMS etc.

REST

(Representational State Transfer)

- SOAP
  - Grew out of the RPC world - not a good start.
  - Choose endpoint URI.
  - Encapsulate method and params in message.
  - Get back some XML.

REST

(Representational State Transfer)

- The REST way:
  - Use the methods / operations of the web.
    - GET, PUT, POST and DELETE
  - The URI represents the data / document and arguments.
- Get back some XML.

REST - example

- GET http://www.mybank.com/accounts/38712/balance
- PUT http://www.mybank.com/accounts/38712/deposit
- DELETE http://www.mybank.com/accounts/38712/direct-debit/23432

REST

- Often chosen over SOAP given the choice (see Amazon).
- Can create richer APIs with SOAP.
- Use JAXB / XMLBeans and schema to create Java XML bindings.

Cocoon & Spring

Spring

- Not too much said about Spring.
  - IoC - dependency injection. Simpler than J2EE approach.
  - O/R mapping support, JDBC,
  - Aspects - e.g. transactions.
- Does not offer:
  - Logging, Pooling, it's own O/R mapping.

Cocoon & Spring

Cocoon

- XML Pipelines.
- Continuations. Tries to solves the state problem.
  - FlowScript. (also see StrutsFlow).
  - Stack, variables stored for each request.
- Forms. Widgets described with XML.

MyFaces

- Only free open source implementation of Java Server Faces.
- Supports tiles (via struts.jar!)
- JSF not quite a full framework?

Oracle ADF

- Large set of JSF components plus framework improvements.
- AJAX (‘Partial Page Rendering’).
- **Adds processScope**.
- ADF will soon migrate to the Apache incubator.
- Access webapp over other transports, telnet, AIM etc. Cool! Proprietary, I think.

Ant 1.7

- Didn’t learn much in the talk - we’re doing it all already!
- Ant was never intended for work-flows, deployment or very long running tasks.
- <import> task - they didn’t know what they’d done until they’d done it! Some issues addressed.

Ant 1.7

- Dependency handling.
  - They don’t want to do anything about it.
  - I sensed they felt they might have to.
- Maven 2 ant tasks.
  - Very immature (I’ve tried them).
  - Missing jars (xerces). Security still an issue.

Ant 1.7

- Large projects used to require a ‘Make guy’.
- Ant allowed allowed all developers to maintain build files. Low barriers.
- Complex projects require complex use of Ant.
- Now we require an ‘Ant guy’ (me).

smartfrog.org

- HP
- Ant’s deployment counterpart.
- Not XML - hooray!
- Configuration, deployment, liveness.
- Template driven (macros sort of).
- Capable of very large-scale deployments.

Gump

- Continuous integration over development versions (checks out TRUNK).
- Can run gump yourself (non-trivial) or add projects to the Apache Gump build cycle.
- Half a dozen or so builds a day.
- 842 projects.
- Looks to be good at diagnosing problems.

Gump

- Sets Ant's build.sysclasspath and then all other classpaths are ignored.
  - Can have side-effects (classpath order).
- Can run unit tests or whatever you want.