

Real World SOA

Sean A Corfield
CTO, Scazu Inc.



What is this about?

- SOA = Service-Oriented Architecture
 - Expose common business entities and operations to multiple clients across the enterprise
- Adobe Hosted Services “Stanza” project
 - Created services for Adobe Acrobat Connect, Adobe Document Center and Kuler
 - Lessons learned, what works, what doesn't
- If I could do it all again...

Caveat!

- This is a code-free presentation
 - Well, almost code-free...
- I no longer work for Adobe
- I no longer have access to the code
- I cannot show you code I don't have access to

Who am I?

- Freelance Consultant
- CTO, Scazu Inc.
- Previously
 - Manager, Hosted Services, Adobe
 - Senior Architect, Macromedia
- Blogger – An Architect's View – since 2002
- ColdFusion developer since 2001
- Web Developer since 1997

What is SOA?

- SOA = Service-Oriented Architecture
- More than just Services...
- Data Dictionary
- Service Directory
- SaaS

Why SOA?

- Based upon current trends, IDC predicts a compound growth rate of 20% per annum for SaaS, set against the overall software market, which is only growing at around 6% per annum. This leaves IDC in no doubt that there is a fundamental shift toward SaaS as a delivery mechanism, and its use within the notion of Web 2.0, and the convergence of SOA, Web 2.0, and SaaS.
 - 2007 IDC report on Software as a Service

SaaS and SOA

- **Software-as-a-Service**
 - Subscription based (or free) online system
 - Supplements or replaces desktop / enterprise s/w
 - Often exposes an API for 3rd party applications
- **SOA**
 - A way to structure and build systems that naturally exposes those APIs
 - Building blocks for your software
 - Building blocks for 3rd party software

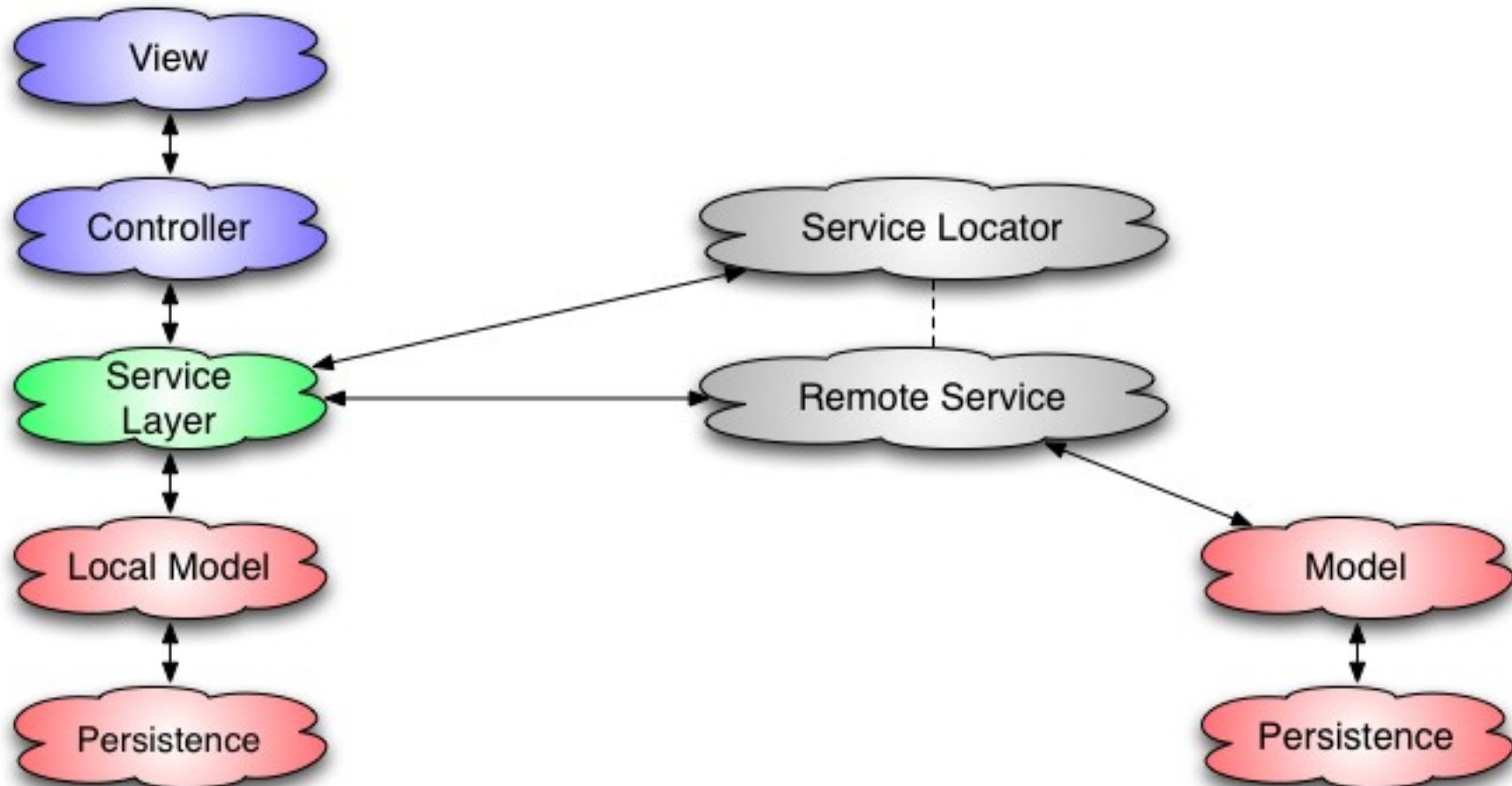
SOA Applications

- Just like regular applications except that Model is combination of:
 - Local Code
 - Remote Code
- Separation of concerns / N-tiers
 - Typically MVC or MVP (Model-View-Presenter)
 - Rely on a variety of web service technologies
 - Typically use a ServiceLocator of some sort

Sounds like regular MVC/MVP?

- View talks to...
- Controller/Presenter talks to...
- (Service layer talks to...)
- (Business) Model layer talks to...
- Data (Persistence) / System layers

Typical SOA Application



It's all about the model

- Not going to cover consumption of services
- One or more services in front of your model
- Lots of objects hidden / managed by a facade
- Data storage for those objects

SOA and ColdFusion

- ColdFusion is a great choice for SOA
 - As a producer of services
 - As a consumer of services
- Web Services
 - SOAP
 - REST
 - JSON (native in Scorpio!)
- Flex / Flash Remoting

How SOA is different

- Client-Server by definition
- Clients are varied (and can be “dumb”)
 - Authentication and sessions
 - Data formats
 - Published API definitions
 - Error handling
- Performance and scalability

Session Management (I)

- Remote clients often have no sense of session
 - Often can't / don't handle cookies at all
- Don't rely on **session**, **cookie** or **client** scope
 - **application** and **server** scope are OK
- Turn **session** scope off to save memory!
 - `<cfset this.sessionmanagement = "false" />`

Session Management (II)

- Simplest solution is token passing
- `authenticate()` methods returns token
- Subsequent calls pass token back
- Manage the tokens in the database
 - Subsequent calls may come to another server!
 - Schedule a task to clean up “old” tokens
- Or use a self-validating token (not as secure)
 - Two-way encrypted containing timestamp, ID etc

Call / Return Formats

- Consider client technology
 - Query is native to ColdFusion
 - Untyped “struct” does not always map well
 - WDDX can be a pain for some clients
- Simple data types are good
- Typed “struct” – i.e., value object CFC is good
- XML is good for most clients too
- JSON is becoming popular

Typed struct / CFC

```
<cfcomponent>
```

```
  <cfproperty name="firstName" type="string"/>
```

```
  <cfproperty name="lastName" type="string"/>
```

```
  ...
```

```
  <cfset this.firstName = ""/>
```

```
  <cfset this.lastName = ""/>
```

```
  ...
```

```
</cfcomponent>
```

Value Objects

- Pass simple value objects back and forth
- Typed struct works well for SOAP and AMF
- For every {vobject}.cfc, have {vobject}.as
- Can work for REST too – we chose Spring-like XML definitions

```
<bean class="dotted.path.to.type">  
  <property name="foo"><value>bar</value></property>  
</bean>
```

REST Calling Format

- REST Adapter (open source, available from my blog) wraps remote CFC API in simple XML
- POST XML packet

```
<path.to.cfc operation="method">
```

```
  <arg1name><value>42</value></arg1name>
```

```
  <arg2name><list>...</list></arg2name>
```

```
</path.to.cfc>
```

Remote Facade

- Might be more than just access="remote"
 - Your service layer / model API is not always a great API for clients of your service
 - Flex API and Web Service API may differ too
- We hand-crafted all our remote APIs
 - Allows you to publish an API but decouple your underlying service / model APIs from that

Technology may drive API

- SOAP, REST, JSON, AMF
 - May (will!) require different API designs
 - Multiple remote facades with different APIs
 - Underlying model remains the same
- CFMX makes it easy to expose a single CFC as all of SOAP, JSON (Scorpio), AMF
 - That does not mean you **should!**

APIs, Clients and Versions

- Lots of API clients – different assumptions
- Applications “out there” that depend on your services working in a particular manner
- Unit Testing and regression testing is important!
- Evolving API – cannot break existing clients
 - Need strategy to create / manage versions of APIs

Exceptions (I)

- Exceptions often make no sense to clients of your services!
 - 500 error for REST
 - AxisFault for SOAP
 - fault handler in Flash/Flex
- Better to return an object containing:
 - Success / failure indicator
 - If success, the result
 - If failure, details about the failure

Exceptions (II)

- How you return success / result may depend on the client technology
 - REST – XML with 0 or 1 result and 0 or 1 fault data
 - AMF – boolean with Object for either result or fault
 - JSON – similar approach
 - SOAP – return extended result object that also contains the success flag and optional fault data

REST Exceptions

<fault>

<type>authentication_failure.invalid_password</type>

<message>Invalid password</message>

<detail>The password you provided was
incorrect</detail>

</fault>

Clustering and Caching

- Caching is great for performance
- Each member of a cluster has its own cache
- Need to keep caches in sync – or not cache

- Web service to refresh cache elements
- Considered moving to JMS to notify instances of cache content changes

- This a hard problem to solve!

Tools / Frameworks

- ColdSpring
- ORM
 - Transfer
 - Reactor, objectBreeze etc
- Testing
 - cfcUnit
 - ab, httpperf, jMeter

ColdSpring

- **Manages all the service-like CFCs**
 - Each remote web service facade has one or more corresponding ColdSpring-managed model CFCs
- **Manages all configuration data**
- **Manages Transfer factory**

Directory Structure (I)

- /Application.cfc
- /svcA
 - /facade.cfc
- /svcB
 - /facade.cfc
- /svcC
 - /facade.cfc

Directory Structure (II)

- /config
 - /coldspring.xml
 - /svcA
 - /coldspring.xml
 - /svcB
 - /coldspring.xml
- /model
 - /svcAmanager.cfc
 - /svcBmanager.cfc
 - /svcCmanager.cfc

Application.cfc

```
<cffunction name="onApplicationStart">  
  <cfset var cs = createObject("component",  
    "coldspring.beans.DefaultXmlBeanFactory").init() />  
  <cfset cs.loadBeans(  
    expandPath("/config/coldspring.xml") ) />  
  <cfset application.cs = cs />  
</cffunction>
```

coldspring.xml (I)

```
<beans>
```

```
  <bean id="serverConfiguration" ...>...</bean>
```

```
  <bean id="transferConfiguration" ...>...</bean>
```

```
  <bean id="transferFactory" ...>...</bean>
```

```
  <import resource="/config/svcA/coldspring.xml" />
```

```
  <import resource="/config/svcB/coldspring.xml" />
```

```
  <import resource="/config/svcC/coldspring.xml" />
```

```
  ...
```

```
</beans>
```

coldspring.xml (II)

```
<beans>
```

```
  <bean id="svcA" class="/model/svcAmanager.cfc>
```

```
    ...
```

```
  </bean>
```

```
  ...
```

```
</beans>
```

Simple remote facade method

```
<cffunction name="performAction" ...>  
  ... map arguments ...  
  <cfset result = application.cs.getBean("svcA")  
    .doSomething(someArguments) />  
  ... map result ...  
  <cfreturn result />  
</cffunction>
```

Transfer

- Injected into model by ColdSpring
- Manages (nearly) all persistence and queries
 - TQL makes “all” possible, not just “nearly”
- Caching is judiciously used
 - Cache currency
 - Balance update frequency vs read frequency
 - Discard dirty objects across the cluster
 - `transfer.discardByClassAndKey(className,key)`

cfcUnit

- Write an extensive suite of tests
 - For low-level model components
 - For the remote facades
 - Use mocks to provide “canned” responses
- Automate your testing
 - Use the cfcUnit ant task and Eclipse's “builder”
 - Use the new CFUnit plugin and my cfcUnit facade
- Use TestSuites to aggregate TestCases
 - Make sure you have “AllTests” for regressions!

AllTests.cfc

```
<cffunction name="suite"  
    returntype="org.cfcunit.framework.Test"  
    access="public" output="false">  
    <cfset var suite = createObject("component",  
        "org.cfcunit.framework.TestSuite").init("SvcA")>  
    <cfset suite.addTest( createObject("component",  
        "basicsuite").suite() )>  
    <cfset suite.addTestSuite( createObject("component",  
        "extratest") )>  
    <cfreturn suite/>  
</cffunction>
```

What went well?

- Unit testing and regression testing
 - cfcUnit, ant
- Managing CFCs and persistence
 - ColdSpring, Transfer
 - We didn't do anything complex with Transfer tho'
- Automated builds
 - CruiseControl

What wasn't so good?

- Value object structure
 - Matched to “legacy” internal system
 - Should have designed new model from scratch
- Services were too granular
 - Multiple endpoints for similar services
- We probably didn't write enough unit tests :)

Resources

- ColdSpring - <http://coldspringframework.org>
- Transfer - <http://compoundtheory.com/transfer>
- An Architect's View - <http://corfield.org>
 - REST Adapter

Questions?



Sean A Corfield
sean@corfield.org
<http://corfield.org>