

High Performance Cloud-Friendly SCA Runtimes

Luciano Resende
lresende@apache.org
<http://lresende.blogspot.com>



Jean-Sebastien Delfino
jsdelfino@apache.org
<http://jsdelfino.blogspot.com>



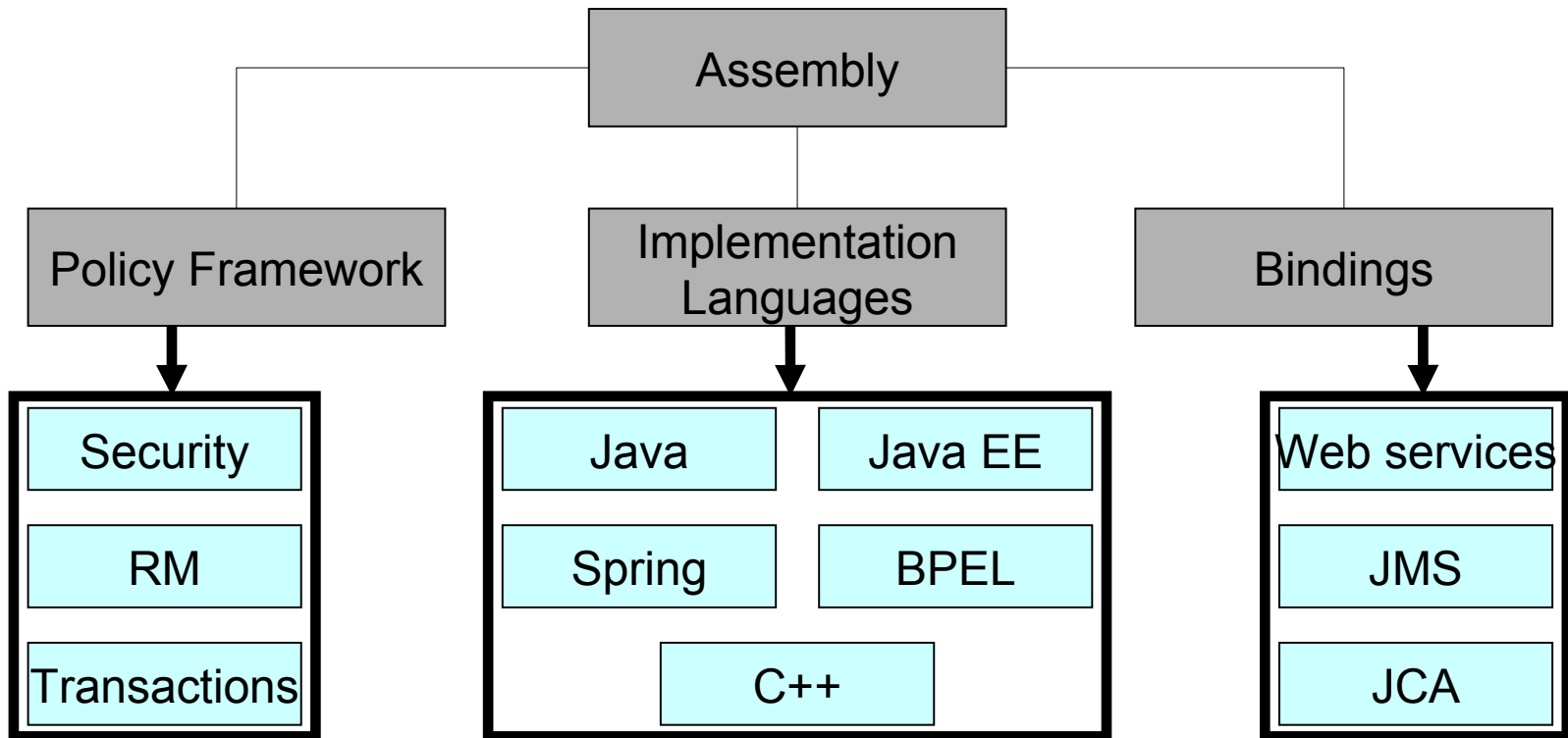
Agenda

- **SCA – Not just one Runtime**
- **Apache Tuscany and Sub-Projects**
- **SCA C++ Runtime**
 - Usage Scenarios
 - How It Works
 - Utility Components
 - Demo
- **SCA Python Runtime**
 - Usage Scenarios
 - How It Works
 - Demo
- **Getting Involved**

SCA – Not Just One Monolithic Runtime

SCA Specifications

- **SCA is going through a formal standardization process at OASIS OpenCSA (<http://www.oasis-opencsa.org>)**



OASIS Open CSA - <http://www.oasis-opencsa.org/>



Navigation

Members

About

The **OASIS Open Composite Services Architecture (CSA) Member Section** advances open standards that simplify SOA application development. Open CSA brings together vendors and users from around the world to collaborate on the further development and adoption of the Service Component Architecture (SCA) and Service Data Objects (SDO) families of specifications.

Steering Committee

23 March 2007 - 1:07pm — [jeff.mischkinsky](#)

Open CSA activities are managed by a Steering
Open CSA membership in an open process. The

- Graham Barber - IBM
- David Burke - TIBCO
- Patrick Leonard - Rogue Wave Software
- Mark Little - Red Hat
- Jeff Mischkinsky - Oracle
- Sanjay Patil - SAP
- Michael Rowley - BEA Systems

Committees

Several technical committees are affiliated with Open CSA:

OASIS Service Component Architecture / Assembly (SCA-Assembly) TC

Defining core SCA composition model to simplify SOA application development

OASIS Service Component Architecture / Policy (SCA-Policy) TC

Defining an SCA policy framework to simplify SOA application development

OASIS Service Component Architecture / Bindings (SCA-Bindings) TC

Standardizing bindings for SCA services and references to communication protocols, technologies and frameworks

OASIS Service Component Architecture / BPEL (SCA-BPEL) TC

Specifying how SCA component implementations for SOA can be written using BPEL

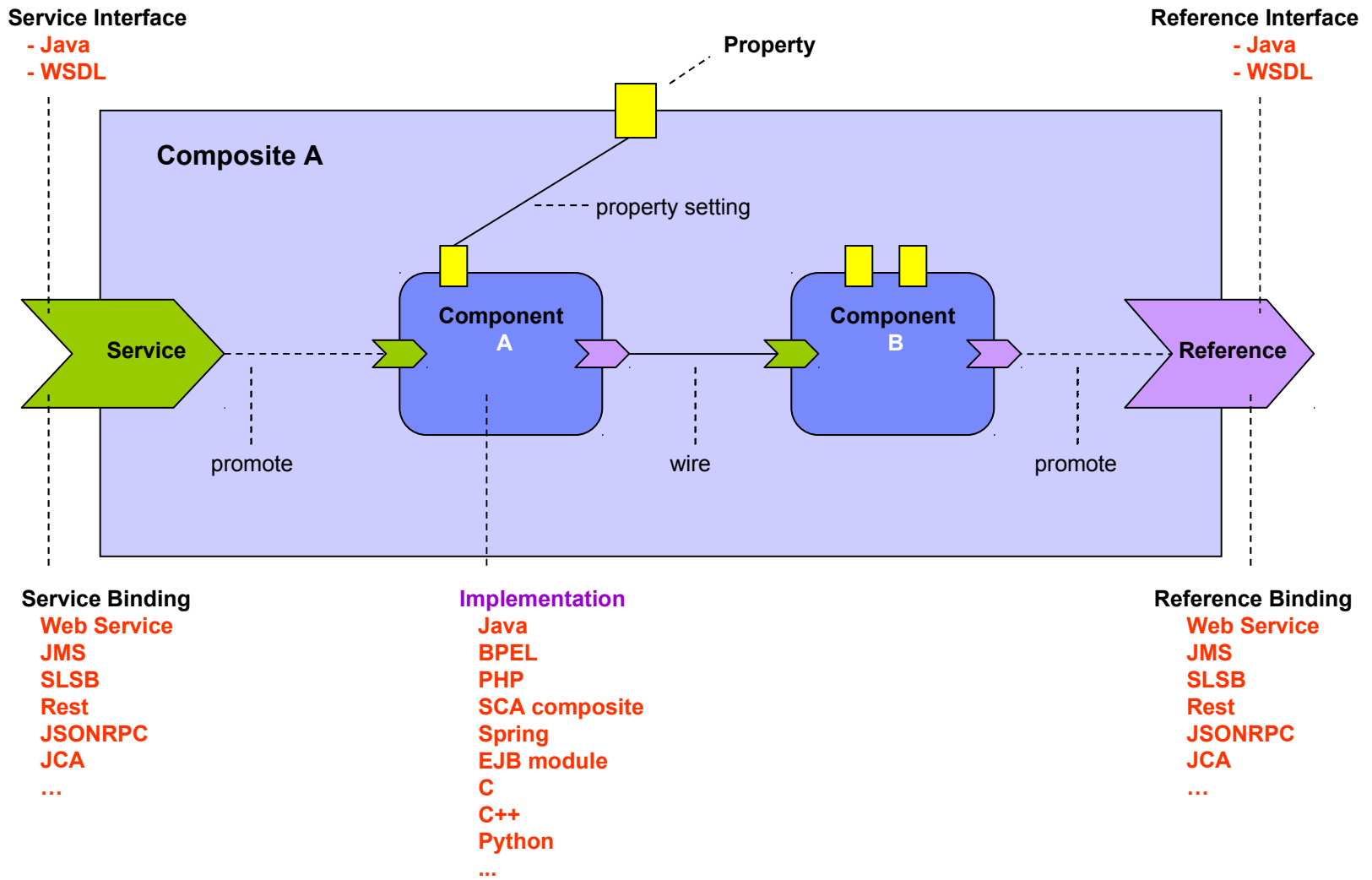
OASIS Service Component Architecture / C and C++ (SCA-C-C++) TC

Standardizing C and C++ use within an SCA domain for SOA

OASIS Service Component Architecture / J (SCA-J) TC

Standardizing Java (tm) use within an SCA domain for SOA

SCA Assembly Model



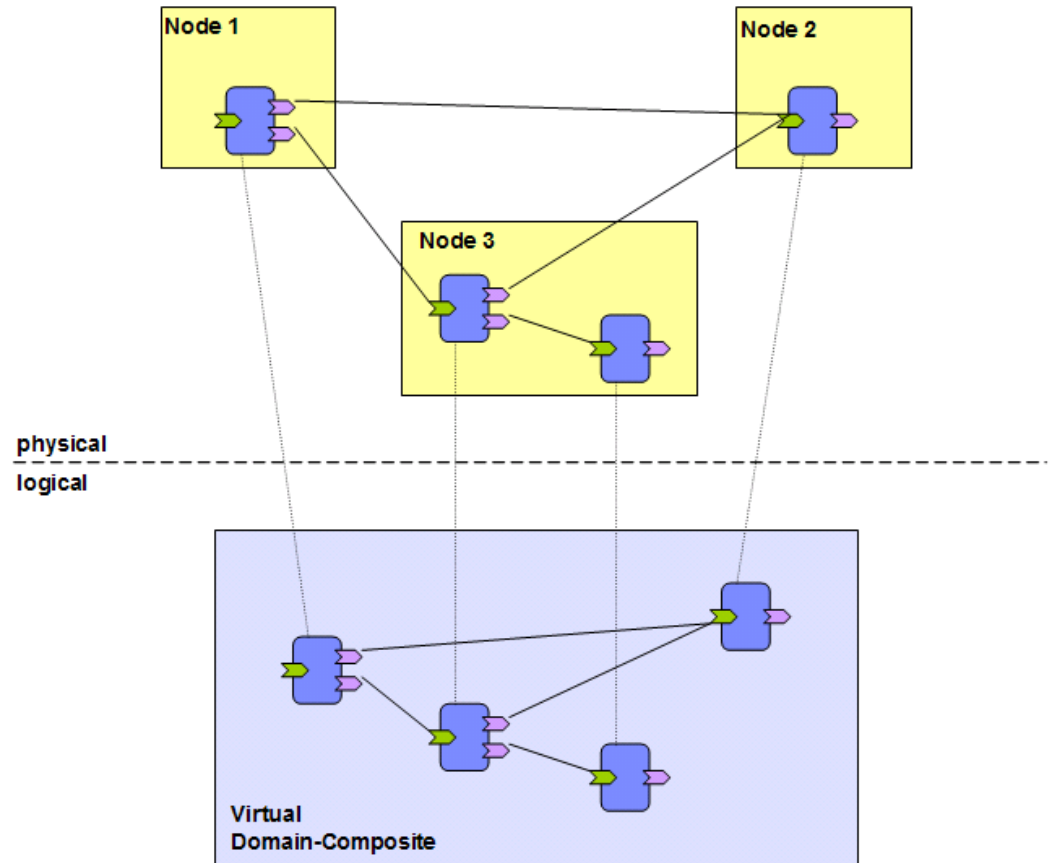
SCA Domain

A common assembly model

A distributed deployment of the assembly

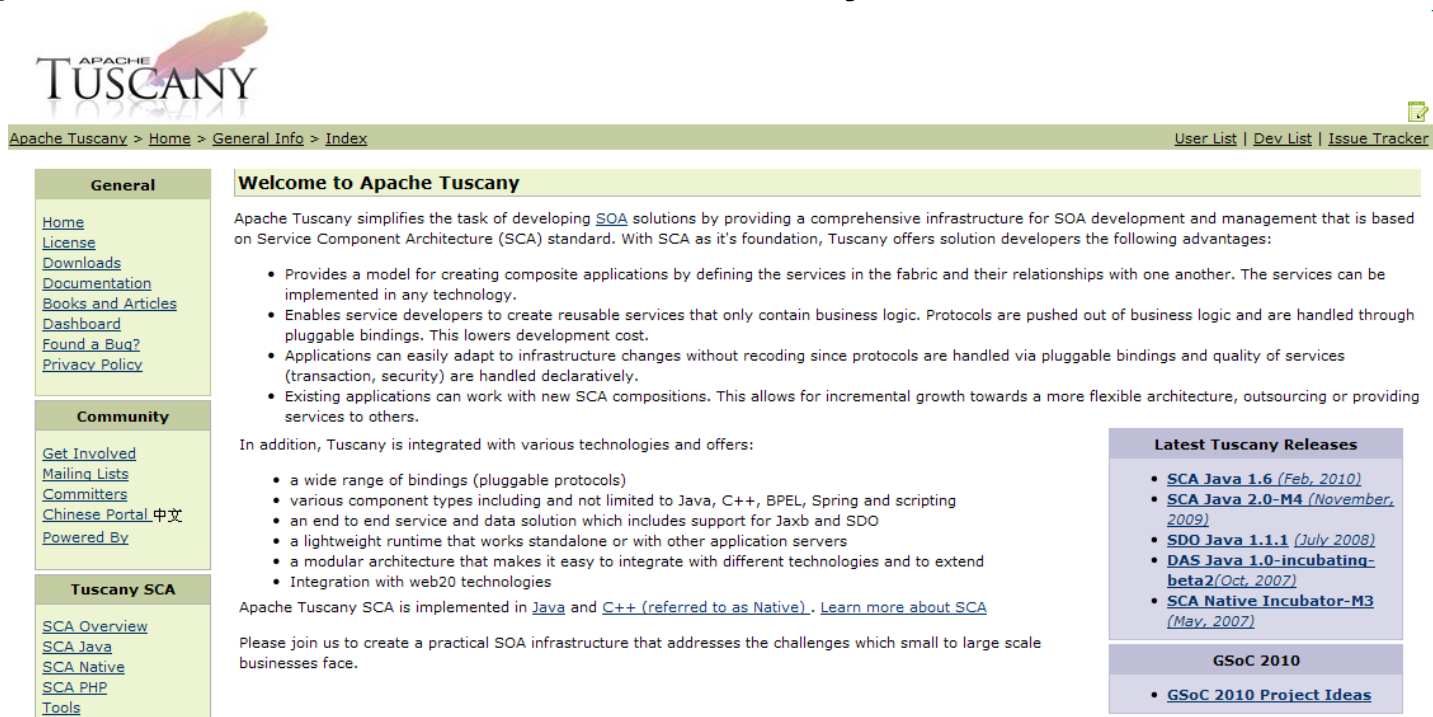
SCA nodes can use different specialized runtimes, optimized for specific component types

Java
JEE
BPEL
C / C++
Python
PHP
etc



Apache Tuscany

- Apache Tuscany provides a component based programming model which simplifies development, assembly and deployment and management of composite applications.
- Apache Tuscany implements SCA standards defined by OASIS OpenCSA + extensions based on community feedback.



The screenshot shows the Apache Tuscany website homepage. At the top left is the Apache Tuscany logo. Below it is a navigation bar with links: Apache Tuscany > Home > General Info > Index. On the right side of the navigation bar are links for User List, Dev List, and Issue Tracker. The main content area is divided into three columns. The left column has three sections: 'General' with links for Home, License, Downloads, Documentation, Books and Articles, Dashboard, Found a Bug?, and Privacy Policy; 'Community' with links for Get Involved, Mailing Lists, Committers, Chinese Portal (中文), and Powered By; and 'Tuscany SCA' with links for SCA Overview, SCA Java, SCA Native, SCA PHP, and Tools. The middle column has a green header 'Welcome to Apache Tuscany' followed by a paragraph describing the project's purpose and a bulleted list of advantages. Below this is a paragraph about integration with various technologies and another bulleted list of features. At the bottom of the middle column is a paragraph about SCA implementation and a call to action. The right column has a blue header 'Latest Tuscany Releases' followed by a bulleted list of recent releases, a blue header 'GSoC 2010', and a bulleted list of project ideas.

General

- [Home](#)
- [License](#)
- [Downloads](#)
- [Documentation](#)
- [Books and Articles](#)
- [Dashboard](#)
- [Found a Bug?](#)
- [Privacy Policy](#)

Community

- [Get Involved](#)
- [Mailing Lists](#)
- [Committers](#)
- [Chinese Portal 中文](#)
- [Powered By](#)

Tuscany SCA

- [SCA Overview](#)
- [SCA Java](#)
- [SCA Native](#)
- [SCA PHP](#)
- [Tools](#)

Welcome to Apache Tuscany

Apache Tuscany simplifies the task of developing [SOA](#) solutions by providing a comprehensive infrastructure for SOA development and management that is based on Service Component Architecture (SCA) standard. With SCA as it's foundation, Tuscany offers solution developers the following advantages:

- Provides a model for creating composite applications by defining the services in the fabric and their relationships with one another. The services can be implemented in any technology.
- Enables service developers to create reusable services that only contain business logic. Protocols are pushed out of business logic and are handled through pluggable bindings. This lowers development cost.
- Applications can easily adapt to infrastructure changes without recoding since protocols are handled via pluggable bindings and quality of services (transaction, security) are handled declaratively.
- Existing applications can work with new SCA compositions. This allows for incremental growth towards a more flexible architecture, outsourcing or providing services to others.

In addition, Tuscany is integrated with various technologies and offers:

- a wide range of bindings (pluggable protocols)
- various component types including and not limited to Java, C++, BPEL, Spring and scripting
- an end to end service and data solution which includes support for Jaxb and SDO
- a lightweight runtime that works standalone or with other application servers
- a modular architecture that makes it easy to integrate with different technologies and to extend
- Integration with web20 technologies

Apache Tuscany SCA is implemented in [Java](#) and [C++](#) (referred to as [Native](#)). [Learn more about SCA](#)

Please join us to create a practical SOA infrastructure that addresses the challenges which small to large scale businesses face.

Latest Tuscany Releases

- [SCA Java 1.6](#) (Feb, 2010)
- [SCA Java 2.0-M4](#) (November, 2009)
- [SDO Java 1.1.1](#) (July 2008)
- [DAS Java 1.0-incubating-beta2](#) (Oct, 2007)
- [SCA Native Incubator-M3](#) (May, 2007)

GSoC 2010

- [GSoC 2010 Project Ideas](#)

Tuscany SCA Native - C++ Runtime



SCA Native Runtime – Usage Scenarios

- **Components written in Python or C++**
- **Integrated with HTTPD**
- **High Throughput and Scalable**
 - Stateless components
 - Load balancing (mod_proxy_balancer)
- **Small disk and memory footprint**
 - no JVM, not a lot of code, most dependencies optional
- **Built-in Security**
 - HTTPS, OpenID, Oauth 1+2, mod_auth*
- **HTTP-based Protocol Bindings**
 - JSON-RPC, ATOM + RSS
- **Easy install + configuration + reconfiguration**
- **Multi-tenant**
 - currently using HTTPD mass dynamic virtual hosting

A Useful Subset of SCA

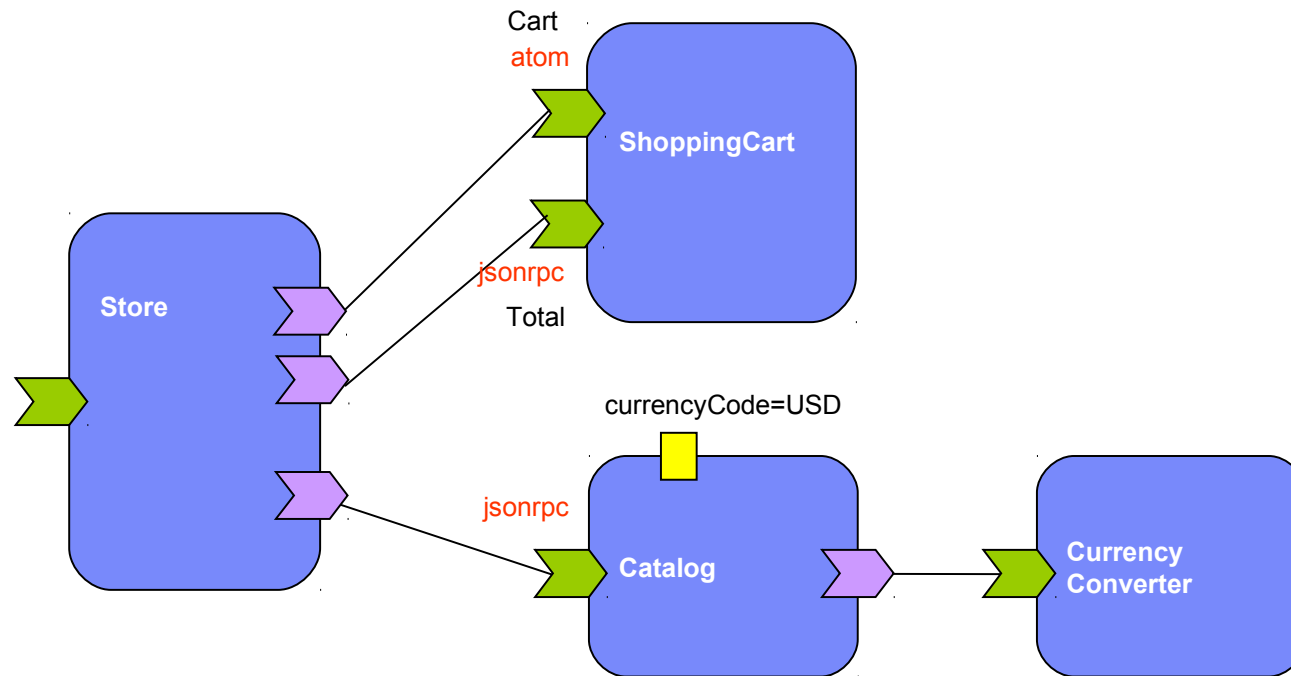
➤ **Subset of SCA**

- Simple one level assembly (no recursive composition)
- Simple wiring (no autowiring, no multiplicity)
- String-only properties
- No Policies
- No SCDL Includes
- No SCDL validation
- Stateless components, parameter injection only
- One SCA contribution directory per Runtime Instance

➤ **Still Useful and Usable**

- Runs apps similar to Tuscany Store / ShoppingCart Tutorial
- Same Python components run on
 - Tuscany / Java
 - Tuscany / C++
 - Tuscany / Python

Sample Online Store App



Sample Composite

```
<composite xmlns="http://docs.oasis-open.org/ns/opencsa/sca/200912" ... >
...
  <component name="Catalog">
    <t:implementation.python script="fruits-catalog.py"/>
    <property name="currencyCode">USD</property>
    <service name="Catalog">
      <t:binding.jsonrpc uri="catalog"/>
    </service>
    <reference name="currencyConverter" target="CurrencyConverter"/>
  </component>

  <component name="ShoppingCart">
    <t:implementation.python script="shopping-cart.py"/>
    <service name="ShoppingCart">
      <t:binding.atom uri="shoppingCart"/>
    </service>
    <service name="Total">
      <t:binding.jsonrpc uri="total"/>
    </service>
    <reference name="cache" target="Cache"/>
  </component>

  <component name="CurrencyConverter">
    <t:implementation.python script="currency-converter.py"/>
    <service name="CurrencyConverter">
      <t:binding.jsonrpc uri="currencyConverter"/>
    </service>
  </component>
</composite>
```

Sample Python Components

fruits-catalog.py

```
def items(converter, currency):
    def convert(price):
        return converter.convert("USD", currency, price)
    symbol = converter.symbol(currency)
    return (
        (("name", "Mango"), ("currency", currency), ("currencySymbol", symbol), ("price", convert(2.99))),
        (("name", "Passion"), ("currency", currency), ("currencySymbol", symbol), ("price", convert(3.55))),
        (("name", "Kiwi"), ("currency", currency), ("currencySymbol", symbol), ("price", convert(1.55))))
```

store.py

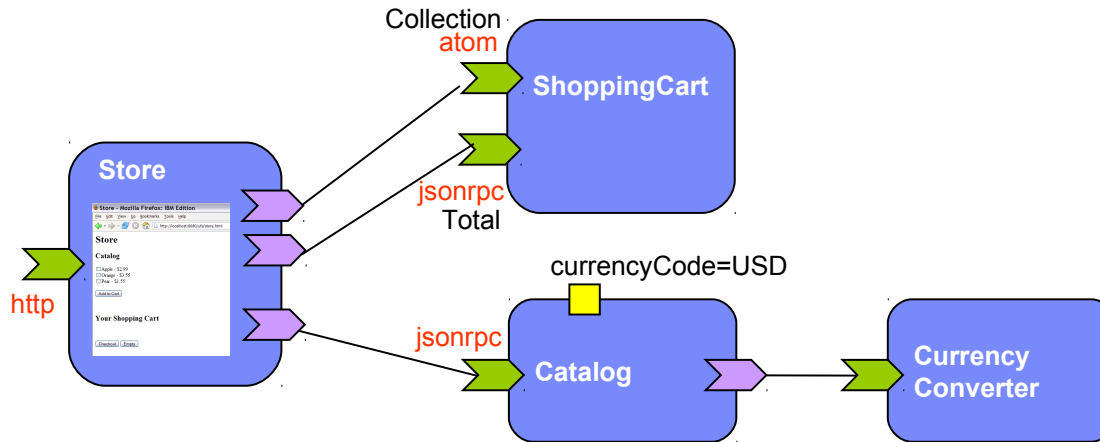
```
def getall(catalog, shoppingCart, shoppingTotal):
    return shoppingCart.getall()

def get(id, catalog, shoppingCart, shoppingTotal):
    return shoppingCart.get(id)

def items(catalog, shoppingCart, shoppingTotal):
    return catalog.items()

def total(catalog, shoppingCart, shoppingTotal):
    return shoppingCart.gettotal()
```

How It Works



mod_wiring

mod_eval_python
mod_eval_cpp

curl

HTTPD apr libxml2 json oauth openid

Utility Components

- **All these components are optional**
- **You just drop them in your SCA assembly as needed**
- **Simpler than having more built-in stuff in core SCA?**

- **Cache component using Memcached**
- **Log component using Facebook Scribe**
- **Queue component using Apache Qpid/C**
- **SQL DB component using PostgreSQL**
- **Web Service component using Axis2/C**
- **Key / value store component**
- **XMPP Chat component**
- **...**

Demo

➤ SCA Native Project Layout

➤ Samples

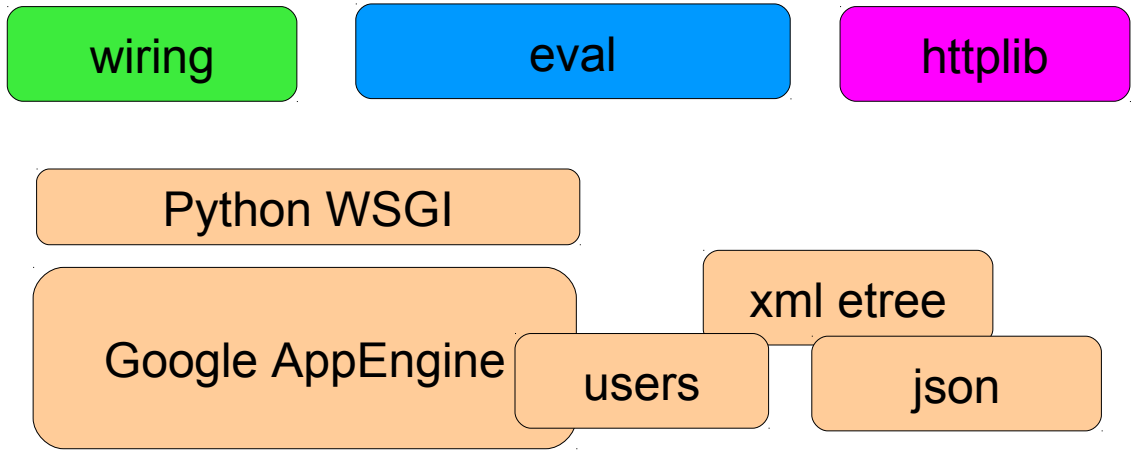
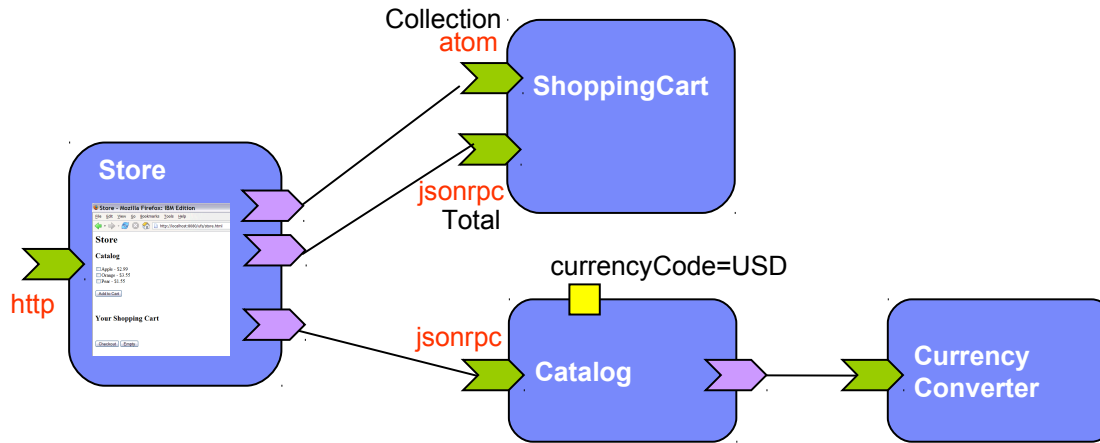
- C++ Store / Shopping Cart app
- Python Store / Shopping Cart app
- Multi-tenant app

SCA Python Runtime – Usage Scenarios

- **Components written in Python**
- **Run on Google AppEngine**
- **Run on a Python WSGI Server**
- **Lightweight with very little code**
- **Stateless**
- **JSON-RPC, ATOM, RSS Bindings**

- **Same Programming Model as SCA Native runtime**

How It Works



Demo

- **Store / Shopping Cart app on Google AppEngine**

Getting Involved with Apache Tuscany

SCA - Resources

➤ **Good introduction to SCA**

- http://www.davidchappell.com/articles/Introducing_SCA.pdf

➤ **OASIS Open CSA – <http://www.oasis-opencsa.org>**

➤ **V1.1 level specs**

- <http://www.oasis-opencsa.org/sca>

➤ **Open CSA Technical Committees**

- <http://www.oasis-opencsa.org/committees>

➤ **OSOA**

- <http://osoa.org/display/Main/Home>

➤ **More information on that site**

- <http://osoa.org/display/Main/SCA+Resources>

Apache Tuscany Resources

➤ **Apache Tuscany**

- <http://tuscany.apache.org>

➤ **Getting Involved**

- <http://tuscany.apache.org/getting-involved.html>

➤ **Tuscany Dashboard**

- <http://tuscany.apache.org/tuscany-dashboard.html>

Getting Involved with Apache Nuvem

Apache Nuvem Resources

➤ Apache Nuvem

- <http://incubator.apache.org/nuvem/>

➤ Getting Involved

- <http://incubator.apache.org/nuvem/nuvem-getting-involved.html>

Thank You !!!

