RESTful SCA with Apache Tuscany

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

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

Agenda

- What is Apache Tuscany
- What is Apache Wink
- RESTFul SCA Overview
- Usage and architecture walk-through
  - Including a look at various scenarios
- What’s next
- Getting involved

What is Apache Tuscany ?

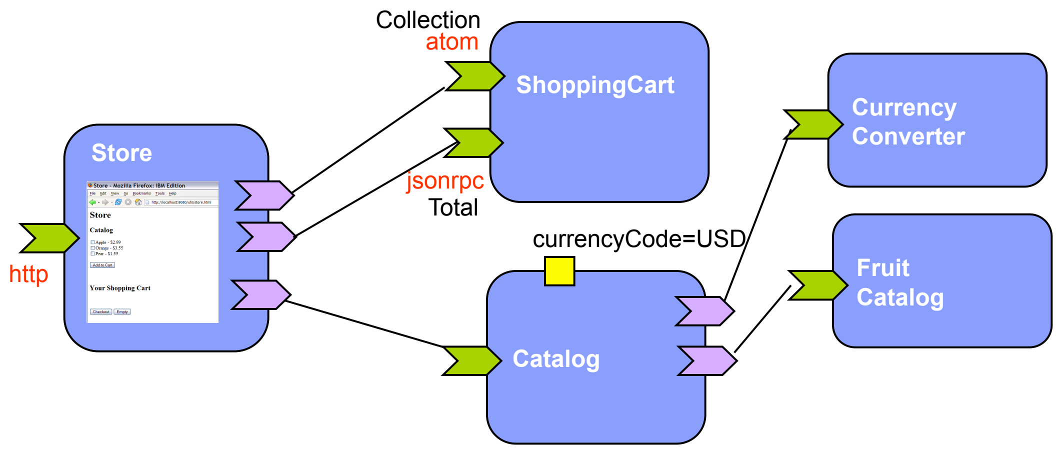
Apache Tuscany

- Apache Tuscany provides a component based programming model which simplifies development, assembly and deployment and management of composite applications.

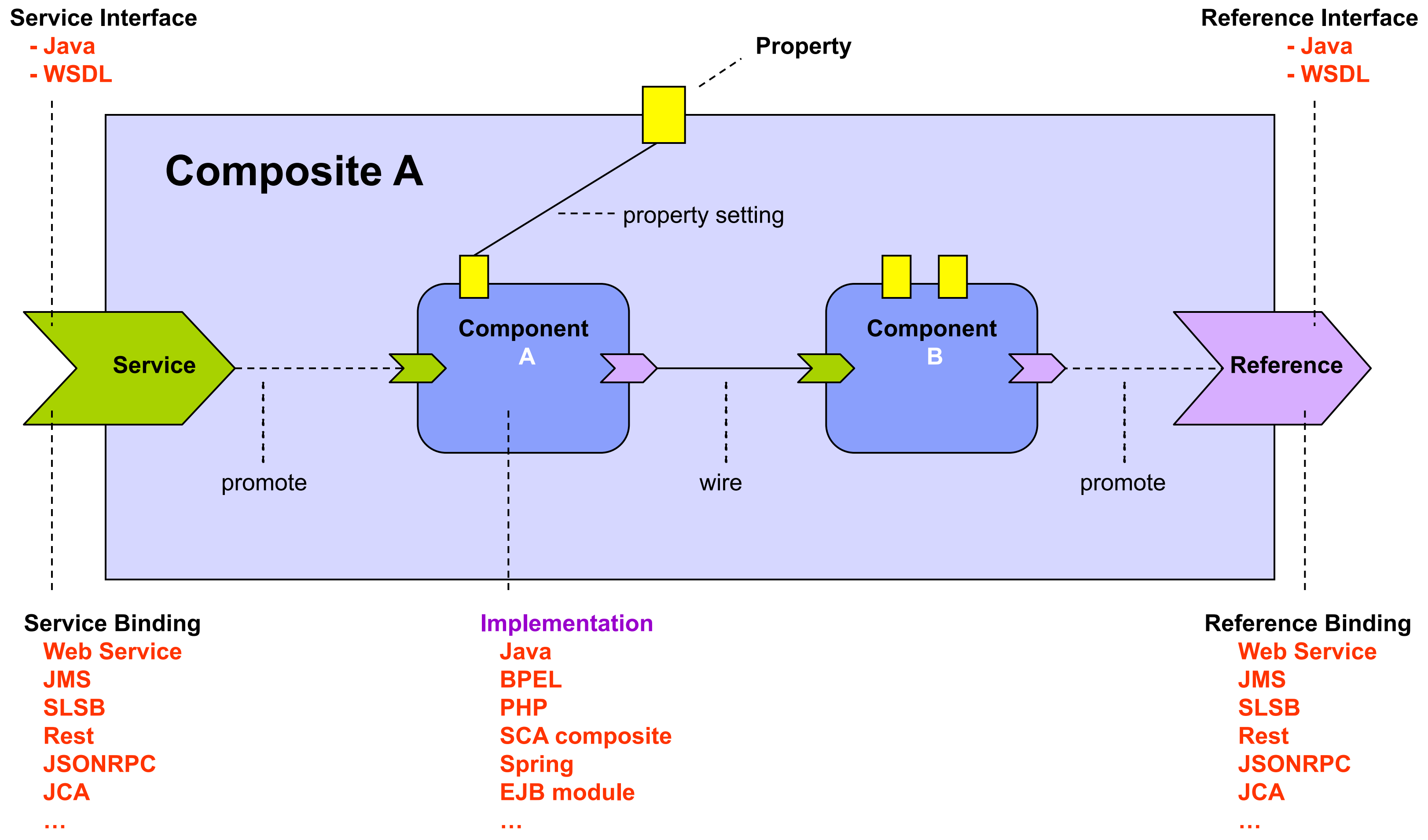
- Apache Tuscany implements SCA standards defined by OASIS OpenCSA and extensions based on real user feedback.

Building Applications using SCA

- Business functions are defined as SCA Components
  - That expose services
    - Using different communication protocols (bindings)
  - And have dependencies on other services through References



SCA Overview



What is Apache Wink?

Apache Wink

- Apache Wink is a simple yet solid open source framework for building RESTful Web services. It is comprised of a Server module and a Client module for developing and consuming RESTful Web services.
- The Wink Server module is a complete implementation of the JAX-RS v1.0 specification. On top of this implementation, the Wink Server module provides a set of additional features that were designed to facilitate the development of RESTful Web services.
- The Wink Client module is a Java based framework that provides functionality for communicating with RESTful Web services. The framework is built on top of the JDK HttpURLConnection and adds essential features that facilitate the development of such client applications.

RESTful SCA
Overview

REST-related user stories

- As a developer, I want to expose new or existing services over HTTP in REST styles with the flexibility to use different wire formats including (but not limited to) JSON, XML.
- As a developer, I want to allow RPC services to be accessed via HTTP GET method in REST styles to take advantage of HTTP caching.
- As a developer, I want to configure a service exposed using REST to use different cache control mechanisms to maximize performance of static resources and or cache dynamic content.
- As a developer, I want to invoke existing RESTful services via a business interface without dealing with the HTTP client APIs.
- As a developer, I want to compose services (both RESTful and non-RESTful) into a solution using SCA.

A win-win situation using SCA and JAX-RS

- SCA gives us the power to declare, configure and compose services in a technology neutral fashion.

- REST is an important aspect of the Web 2.0 world. Building RESTful services can be a challenge as REST is just an architectural style. JAX-RS emerges as the standard REST programming model.

- A balanced middle-ground to leverage the power of declarative services using SCA and annotation based REST/HTTP mapping using JAX-RS (without tying business logic to the technology specifics) (try to avoid JAX-RS APIs directly).

Tuscany's offerings for REST

- The Tuscany Java SCA runtime provides the integration with REST services out of the box via several extensions.
  - Tuscany REST binding type (binding.rest)
    - Leverage JAX-RS annotations to map business operations to HTTP operations such as POST, GET, PUT and DELETE to provide a REST view to SCA services.
    - Support RPC over HTTP GET
    - Allow SCA components to invoke existing RESTful services via a JAX-RS annotated interfaces without messing around HTTP clients.
  - Tuscany JAX-RS implementation type (implementation.jaxrs)
    - JAX-RS applications and resources can be dropped into the SCA assembly as JAX-RS implementation (implementation.jaxrs).
  - Tuscany also enrich the JAX-RS runtime with more databindings to provide support for data representations and transformation without the interventions from application code.

Runtime overview

- Related Tuscany modules
  - interface-java-jaxrs (Introspection of JAX-RS annotated interfaces)
  - binding-rest (binding.rest XML/Java model)
  - binding-rest-runtime (runtime provider)
  - implementation-jaxrs (implementation.jaxrs XML/Java model)
  - implementation-jaxrs-runtime (runtime provider)
- Tuscany uses Apache Wink 1.1.1-incubating as the JAX-RS runtime
  - Contributions were made to the Wink runtime to facilitate embedding Wink and it’s going to be available in Wink 1.1.2 which should be available in the near future

Usage and architecture walk-through

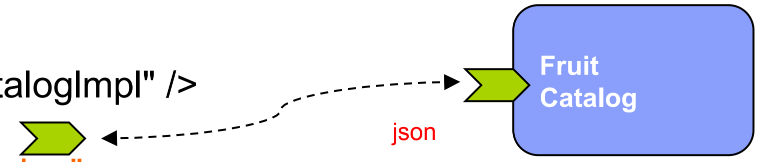
Use Case #1:
Exposing an SCA service to HTTP using REST

- We have an existing SCA service and want to make it RESTful
  1. Define a Java interface with JAX-RS annotations
     - Provide URI
     - Map between business operations and HTTP methods
     - Map Input/Output (Path segments, query parameters, headers, etc)
     - Configure wire formats
  2. Each method should have a compatible operation in the SCA service

REST Services

- Supports exposing existing SCA components as RESTFul services
- Exposing a service as RESTful resource

```
<component name="Catalog">
   <implementation.java class="services.FruitsCatalogImpl" />
   <service name="Catalog">
      <t:binding.rest uri="http://localhost:8080/Cartalog" >
         <t:wireFormat.json />
         <t:operationSelector.jaxrs />
      </t:binding.rest>
   </service>
</component>
```

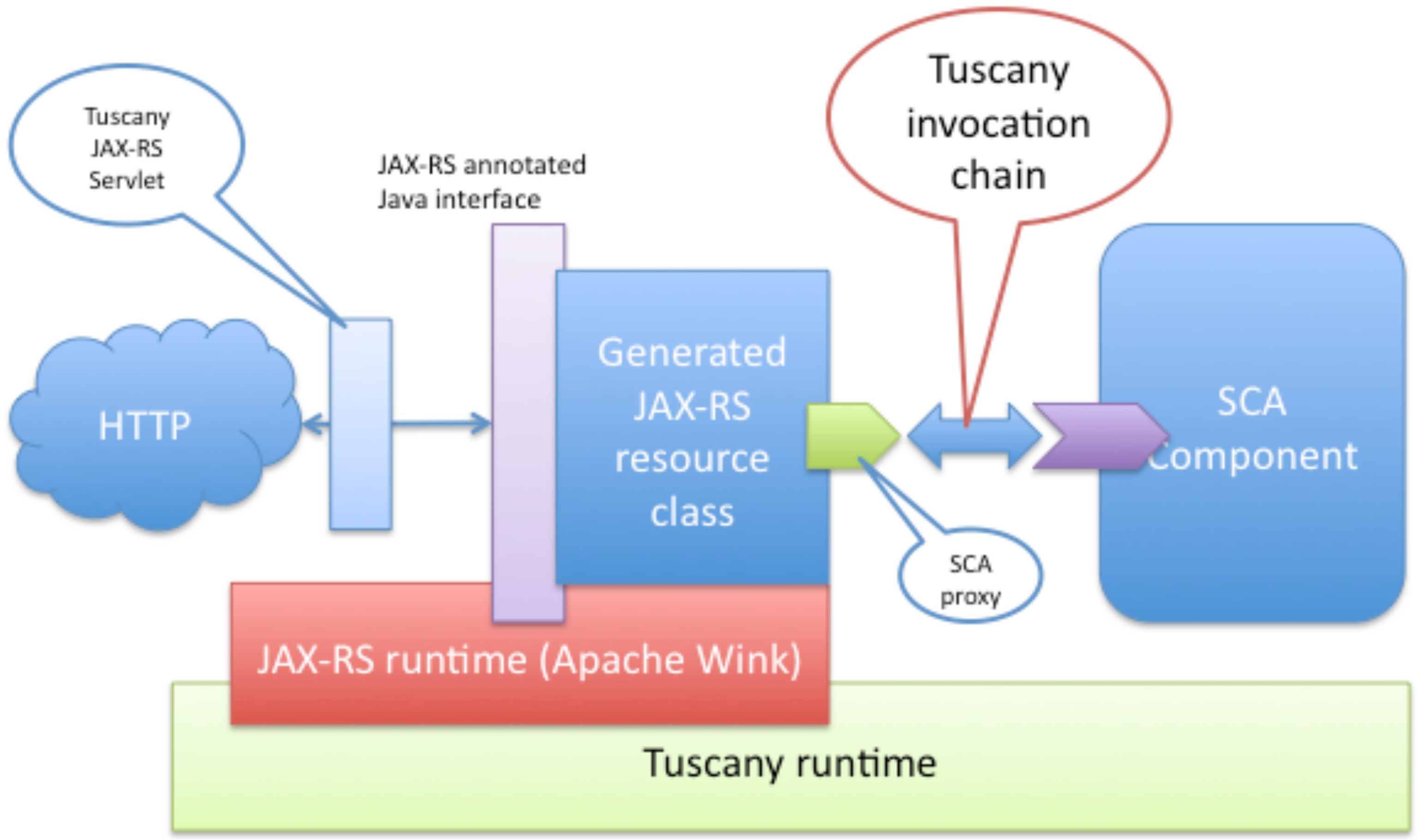


- Consuming REST Services
  - Multiple JavaScript frameworks such as Dojo (XHR)
  - Regular Web browsers
  - Regular utilities such as cUrl, wGet, etc
  - SCA component

Mapping business interfaces using JAX-RS

```
@Remotable
public interface Catalog{

    @GET
    Item[] getItem();

    @GET
    @Path("{id}")
    Item getItemById(@PathParam("id") String itemId);

    @POST
    void addItem(Item item);

    @PUT
    void updateItem(Item item);

    @DELETE
    @Path("{id}")
    void deleteItem(@PathParam("id") String itemId);

}
```

REST service binding runtime architecture


Tuscany JAX-RS Servlet
JAX-RS annotated Java interface
Tuscany invocation chain
HTTP
Generated JAX-RS resource class
SCA Component
SCA proxy
JAX-RS runtime (Apache Wink)
Tuscany runtime

Use Case #2:
Allowing HTTP GET access to an SCA RPC service

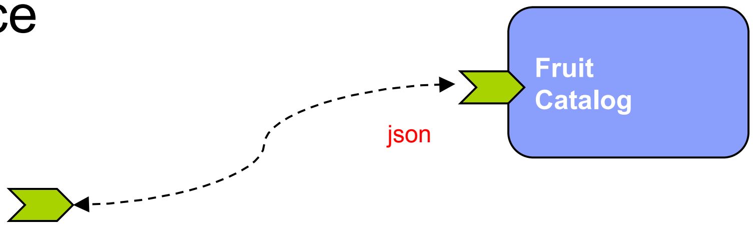
- We have an existing RPC SCA service and want to allow remote accesses using HTTP GET
  - No standard JAX-RS client is defined by the JAX-RS spec
  - We need to figure the URI, parameter passing (positional or name/value pairs, headers, etc)

RPC Services over HTTP GET

- The REST binding provides mapping your RPC style calls over HTTP GET

- Exposing a service as RESTful resource

```
<component name="Catalog">
   <implementation.java class="services.FruitsCatalogImpl" />
   <service name="Catalog">
      <t:binding.rest uri="http://localhost:8080/Cartalog" />
        <t:wireFormat.json />
        <t:operationSelector.rpc />
      </t:binding.rest>
   </service>
</component>
```



- Client Invocation
  - http://localhost:8085/EchoService?method=echo&msg=Hello RPC
  - http://localhost:8085/EchoService?method=echoArrayString&msgArray=Hello RPC1&msgArray=Hello RPC2"

Mapping queryString to RPC method parameters

```
@Remotable
public interface Echo {

    String echo(@QueryParam("msg") String msg);

    int echoInt(int param);

    boolean echoBoolean(boolean param);

    String [] echoArrayString(@QueryParam("msgArray") String[] stringArray);

    int [] echoArrayInt(int[] intArray);

}
```

Design note

- As of today, we have a special path to handle the RPC over HTTP GET in the REST binding runtime (operationSelector.rpc). Potentially, we can unify this approach with the runtime design that implements the logic for Use case #1.
  - Imaging that you are writing a JAX-RS resource method that supports the RPC style (GET with a list of query parameters, one of them is the "method")

- For a service method that is not annotated with JAX-RS http method annotations, we will generate a JAX-RS resource class with a mapping so that:
  - @GET
  - @Path
  - @QueryParam for each of the arguments

Use Case #3:
Access external RESTful services using SCA

- We want to access external RESTful services from an SCA component using SCA references (w/ dependency injection) instead of calling technology APIs such as HTTP client

Tuscany SCA client programming model for JAX-RS

- Model as an SCA reference configured with binding.rest in the client component
- Use a JAX-RS annotated interface to describe the outbound HTTP invocations (please note we use the same way to handle inbound HTTP invocations for RESTful services exposed by SCA)
- A proxy is created by Tuscany to dispatch the outbound invocation per the metadata provided by the JAX-RS annotations (such as Path for URI, @GET for HTTP methods, @QueryParam for parameters, etc).
  - If no HTTP method is mapped, we use the RPC over HTTP GET

Sample configuration

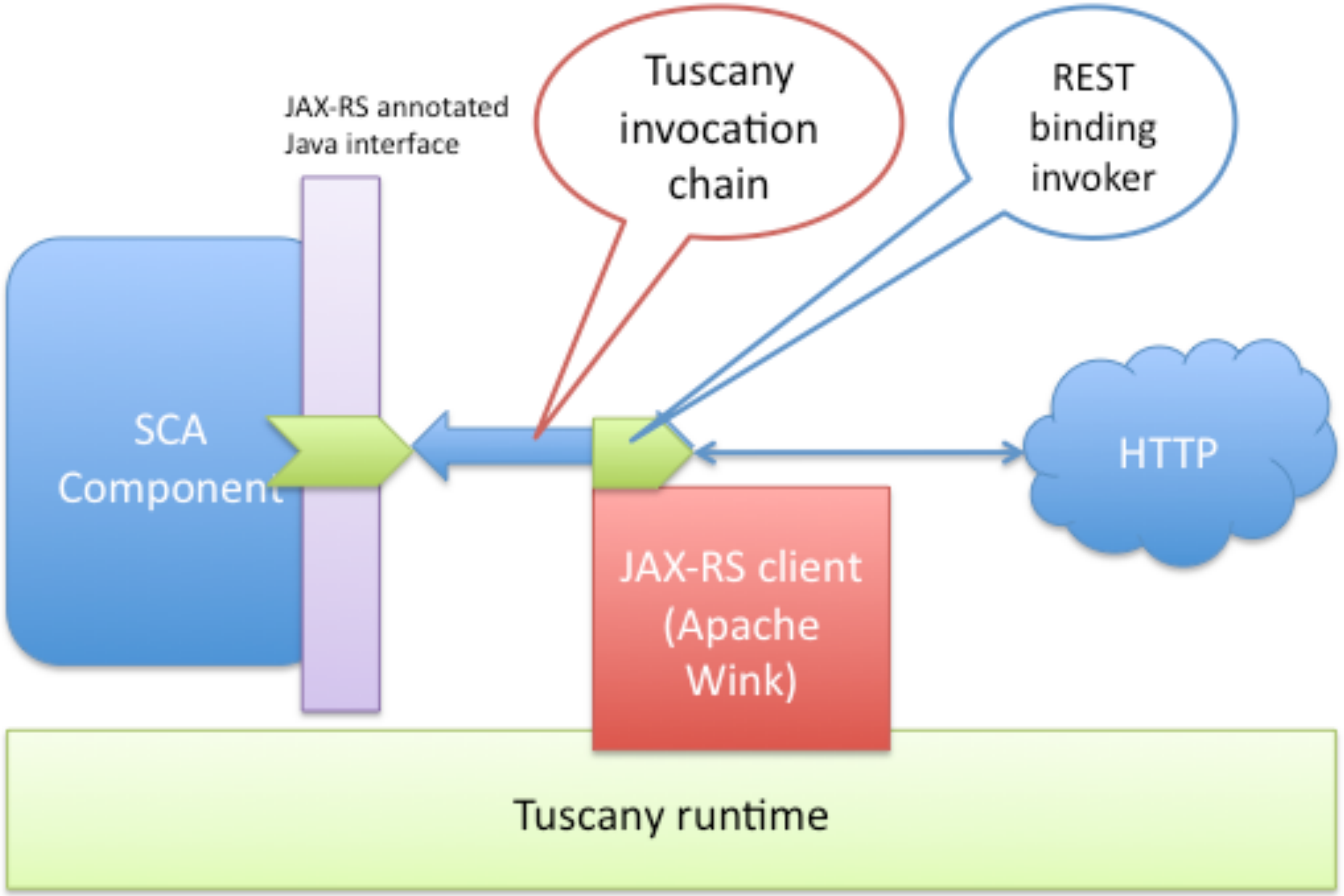
- To invoke a RESTful resource such as getPhotoById(String id):

```
@GET
@Path("/photos/{id}")
InputStream getPhotoById(@PathParam("id") String id);
```

- SCA Reference

```
<reference name="photoService">
   <interface.java interface="…"/>
   <tuscany:binding.rest uri="http://example.com/"/>
</reference>
```

REST reference binding runtime architecture



Use case #4:
Drop in a JAX-RS application into SCA

- We already have a JAX-RS application that is written following JAX-RS programming model (to take full advantage of JAX-RS for the HTTP protocol beyond just business logic).

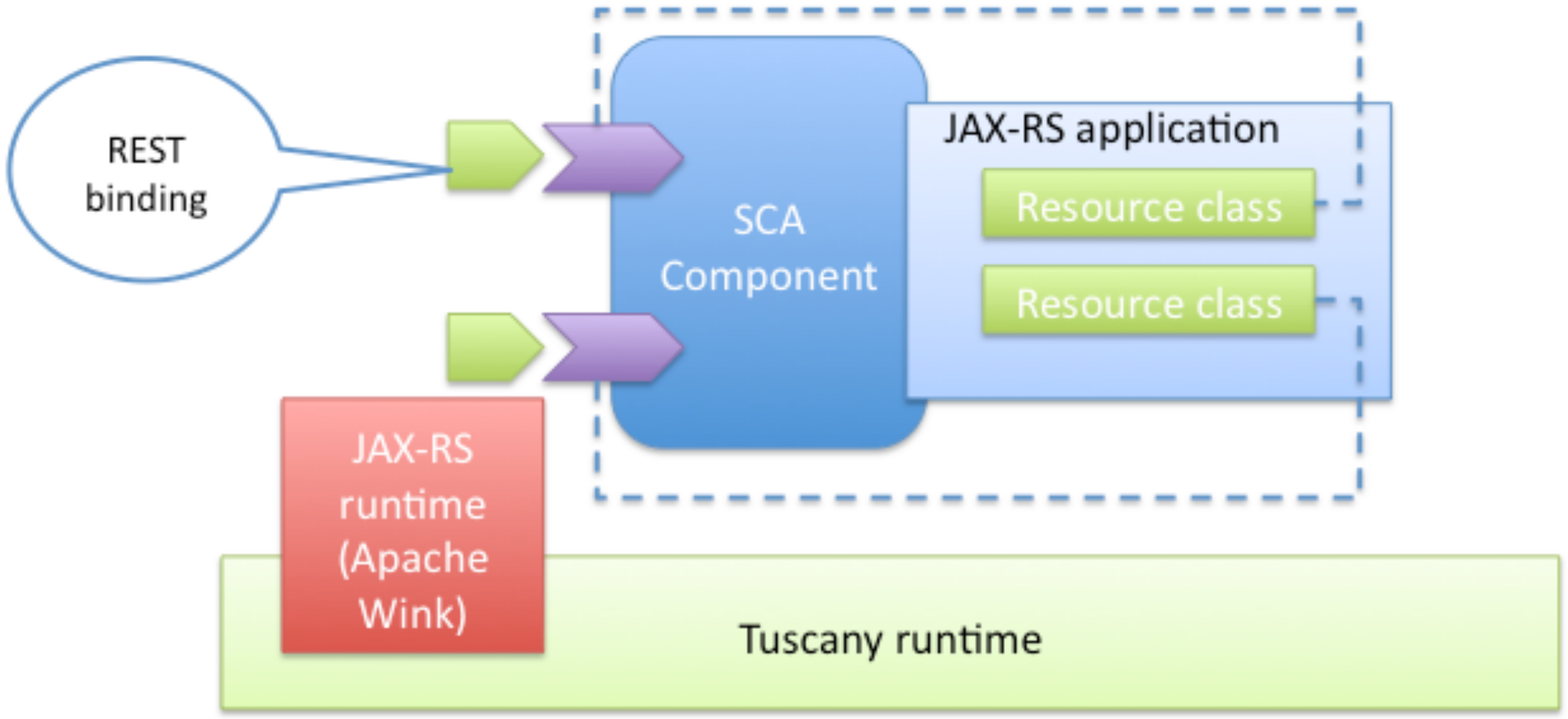
- We want to encapsulate it as an SCA component so that it can be used in the SCA composite application. (Potentially be able to use SCA @Reference or @Property to inject service providers or property values).

Tuscany implemention.jaxrs

- Tuscany introduced an implementation type (under tuscany namespace) to provide out-of-box support for component using JAX-RS

implementation.jaxrs runtime archirecture

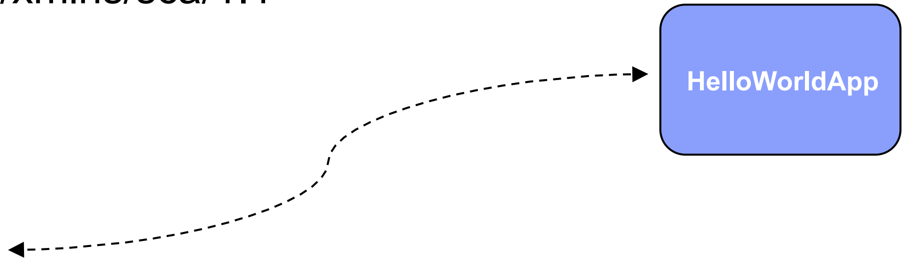
- Each root resource is translated into an SCA service with binding.rest



Sample configuration for implementation.jaxrs

- Composite

```
<composite xmlns="http://docs.oasis-open.org/ns/opencsa/sca/200912"
    xmlns:tuscany="http://tuscany.apache.org/xmlns/sca/1.1"
    targetNamespace="http://sample/jaxrs"
    name="HelloWorld">

    <component name="HelloWorldApp">
        <tuscany:implementation.jaxrs application="helloworld.jaxrs.HelloWorldApp"/>
    </component>

</composite>
```



Summary

Summary

- What do we support today ?
  - Expose SCA Services as RESTFul services
  - Expose RPC SCA Services via HTTP
  - Declaratively configure RESTFul services
    - Cache Controls Headers and general HTTP Headers
  - Consume RESTFul services as SCA References
  - Re-use JAX-RS Resources into your composite applications

What's Next ?

What’s next ?

- What’s on my “next” todo list ?
  - WADL Support via service endpoint ?wadl
  - Support for partial response and partial updates
- What’s on yours ?
  - Submit your suggestions to our user / development list

Getting Involved

Resources

- **Apache Tuscany**
  - http://tuscany.apache.org
- **Getting Involved**
  - http://tuscany.apache.org/getting-involved.html
- **Tuscany SCA Java Releases**
  - http://tuscany.apache.org/sca-java-2x-releases.html
- **Tuscany SCA Java Documentation**
  - http://tuscany.apache.org/java-sca-documentation-menu.html
- **Apache Wink**
  - http://incubator.apache.org/wink
- **Getting Involved**
  - http://incubator.apache.org/wink/community.html
- **Apache Wink Downloads**
  - http://incubator.apache.org/wink/downloads.html

Thank You !!!