Developing RESTful Services using Rational Software Architect

Sandeep Kohli
Sandeep Katoch

IBM Software
Innovate2011
The Premier Event for Software and Systems Innovation

Software. Everyware.

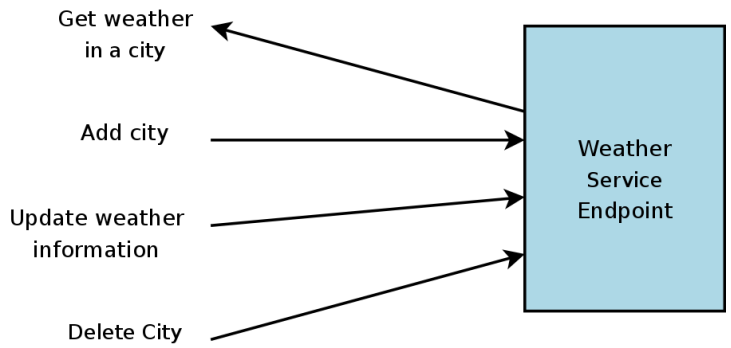
August 9-11, Bangalore | **August 11,** Delhi

Agenda

- What is REST?
  - REST Concepts
  - Why Model REST Services?
- Modelling Support for REST in RSA
  - REST Service Profile and tooling support
  - Customized Sequence Diagram
  - REST Service Model Report Generation
- JAX-RS Support
  - JAXRS Modelling
  - Code Generation & Reverse Engineering

REST : REpresentational State Transfer

- REST defines a set of architectural principles for designing Web services
  - Focus on resources, including how resource states are addressed and transferred over HTTP.
- A simpler alternative to SOAP- and Web Services Description Language (WSDL)-based Web services
- Has gained widespread acceptance across the Web
  - Adoption of REST by mainstream Web 2.0 service providers—including Yahoo, Google, and Facebook



- REST Web service follows four basic design principles:
  - Use HTTP methods explicitly.
  - Be stateless.
  - Expose directory structure-like URIs.
  - Representation of resource state

REST Concepts

- One-to-one mapping between create, read, update, and delete (CRUD) operations
  - To create a resource on the server, use POST.
  - To retrieve a resource, use GET.
  - To change the state of a resource or to update it, use PUT.
  - To remove or delete a resource, use DELETE.
- Web services be stateless
- Expose directory structure-like URIs
  - hierarchical, rooted at a single path, and branching from it are subpaths
  - *http://www.bookmarkservice.come/bookmarks/users/{john}*
- Resource Representation
  - A resource representation typically reflects the current state of a resource
  - Has to do with the format of the data that the application and service exchange in the request/response payload or in the HTTP body

Roy T. Fielding (in a discussion on RESTFul)

- A RESTful system progresses from one steady-state to the next, and each such steady-state is both a potential start-state and a potential end-state. I.e., a RESTful system is an unknown number of components obeying a simple set of rules such that they are always either at REST or transitioning from one RESTful state to another RESTful state. Each state can be completely understood by the representation(s) it contains and the set of transitions that it provides, with the transitions limited to a uniform set of actions to be understandable. The system may be a complex state diagram, but each user agent is only able to see one state at a time (the current steady-state) and thus each state is simple and can be analyzed independently. A user, OTOH, is able to create their own transitions at any time (e.g., enter a URL, select a bookmark, open an editor, etc.).
- ....Roy

Why Model REST Services?

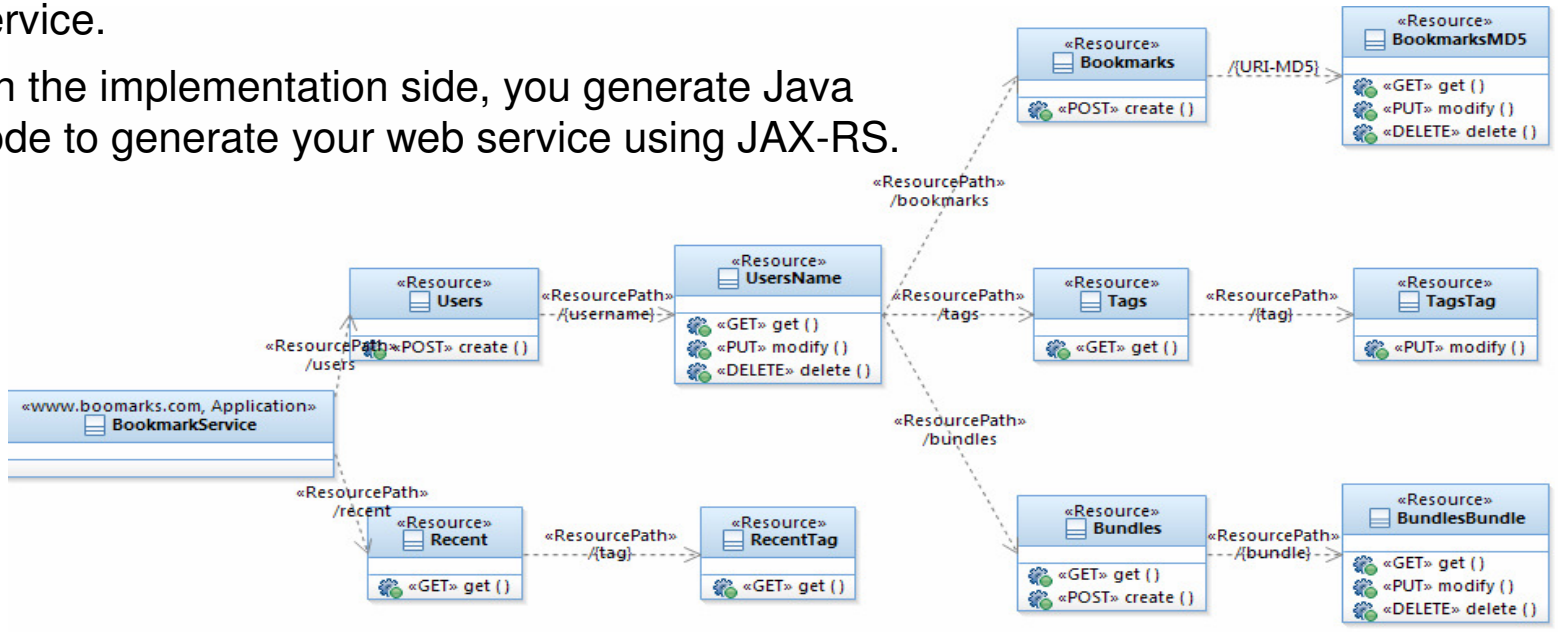
- Traditional Approach
  - Describe the design on REST based Services in terms of URIs, Resource, HTTP methods and their representations
  - Publish as documentation to enable its implementation and enable the clients of the service
  - Lack of any formal notation

Resource	URI	HTTP Methods Supported	
Users	/users	GET	getListofUsers
		POST	createUser
User	/users/{username}	GET	getUser
		DELETE	deleteUser
Bookmarks	/users/{username}/bookmarks	GET	getListofBookmarks
		POST	createBookmark
Bookmark	/users/{username}/bookmarks/{bookmarkId}	GET	getBookmark
		DELETE	deleteBookmark

- How do you design your RESTful Web Service?
- How do you implement this design?
- How do you publish your RESTful services to consumers?
- How do you evolve this design and implementation?

Why Model REST Services?

- MDD-based Approach
  - Rational Software Architect v8.0.3 supports modeling and implementation of RESTful Web Services.
  - The modeling support enables you to create UML models for your web service to describe your web service.
  - On the implementation side, you generate Java code to generate your web service using JAX-RS.

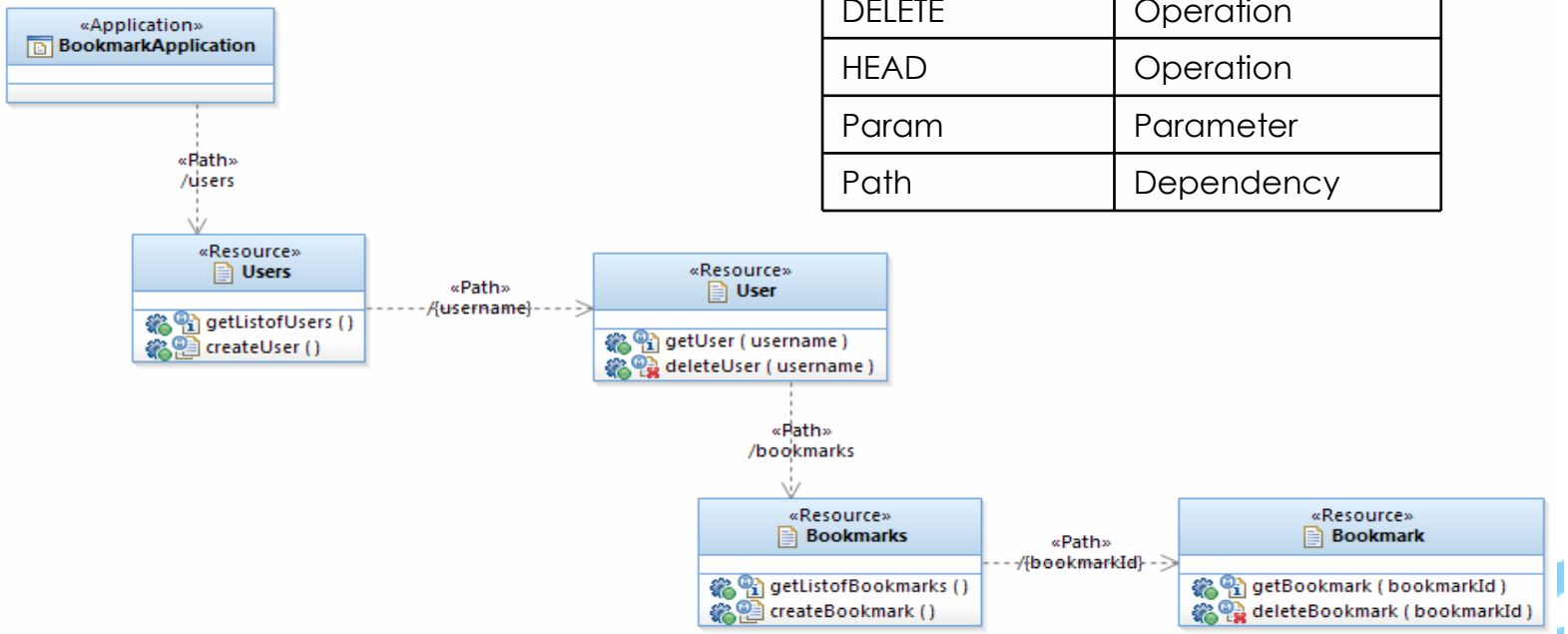


Modelling REST Services

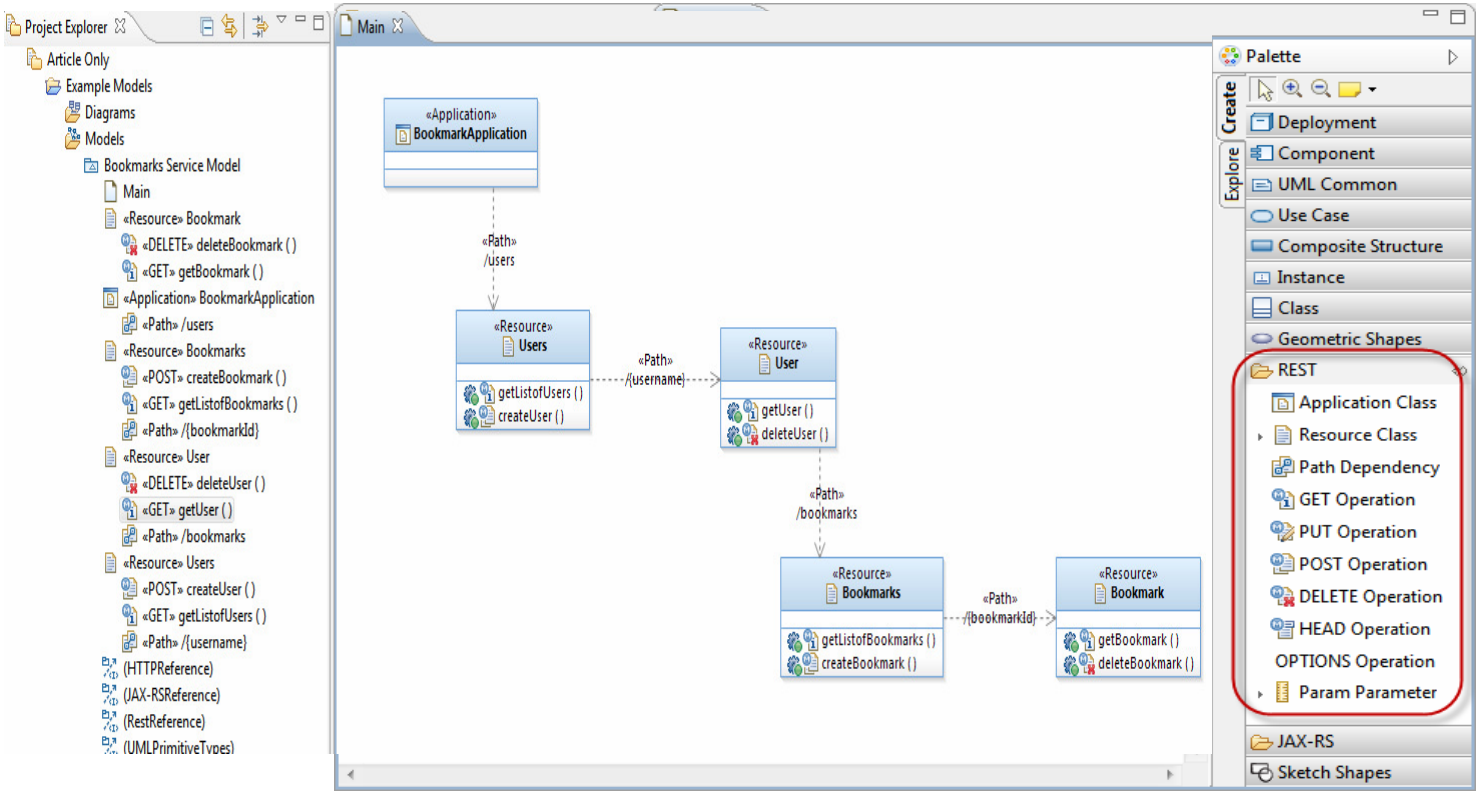
The key modeling elements:

- **RESOURCE**
- Resource **Path**
- Resource Methods
- Resource **Input/Output** types
- Param Types

Rest Stereotype	UML Element
Resource	Class, Interface
Application	Class
GET	Operation
PUT	Operation
POST	Operation
DELETE	Operation
HEAD	Operation
Param	Parameter
Path	Dependency

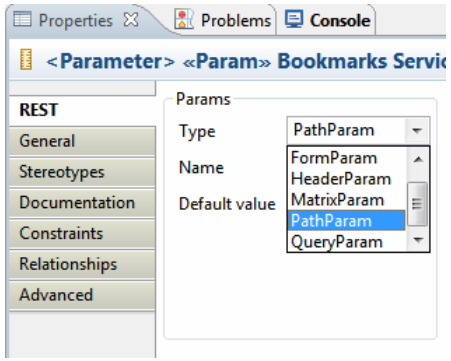


RESTful Service Modeling – Palette Support

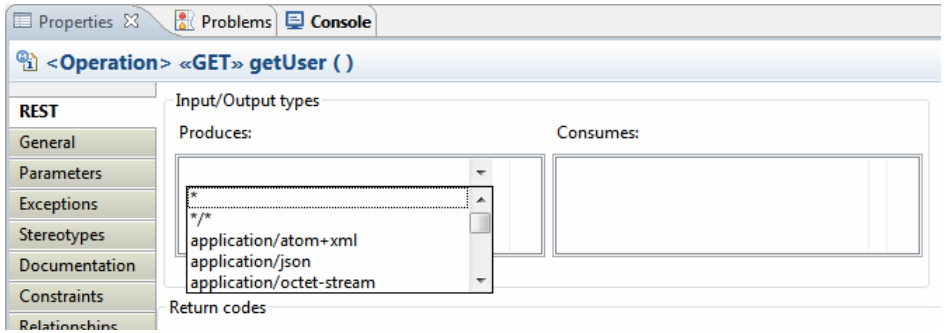


RESTful Service Modeling

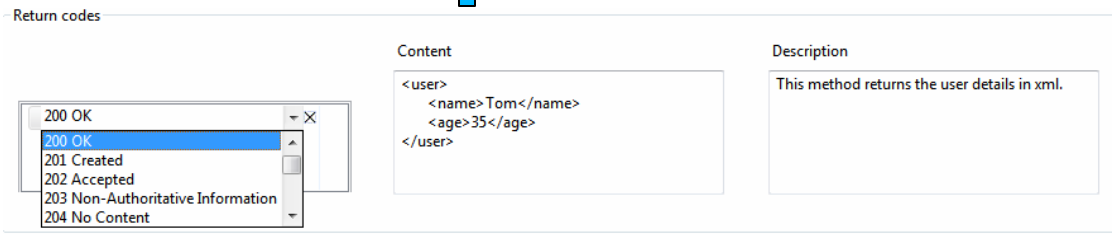
- ✓ Capture parameter types



- ✓ Capture input/output types for resources/resource methods



REST Modeling Tooling Support

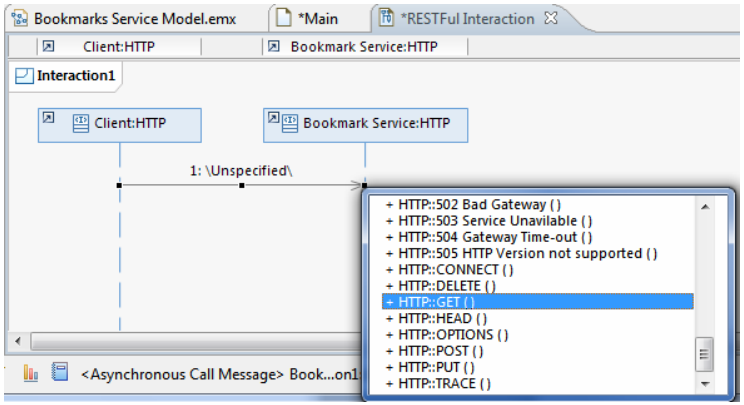


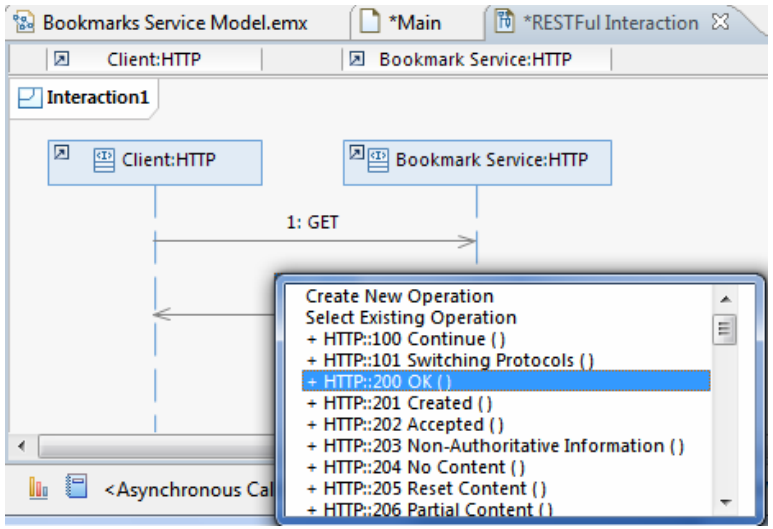
- ✓ Capture return codes for resource methods

Modeling RESTful interactions in Sequence Diagrams

- You can model the typical interactions with the clients of your RESTful Web Service using the sequence diagrams.

HTTP Request on Sequence Diagram

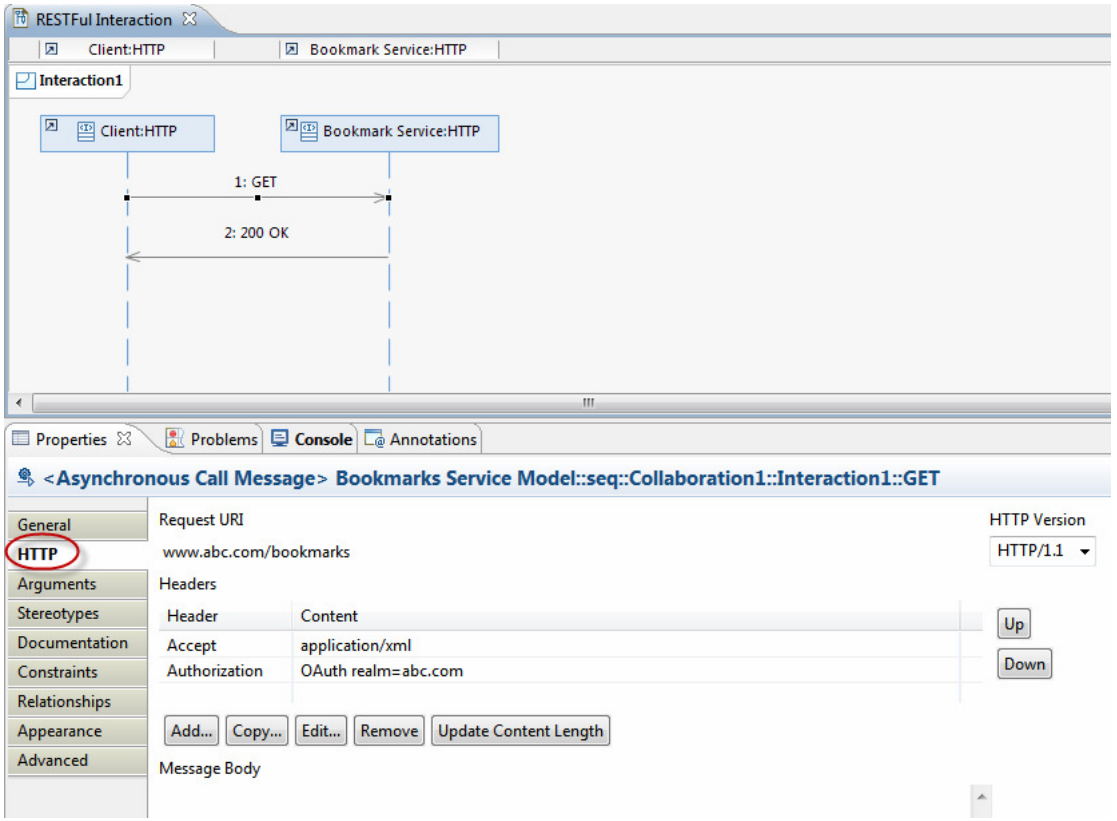




HTTP Response on Sequence Diagram

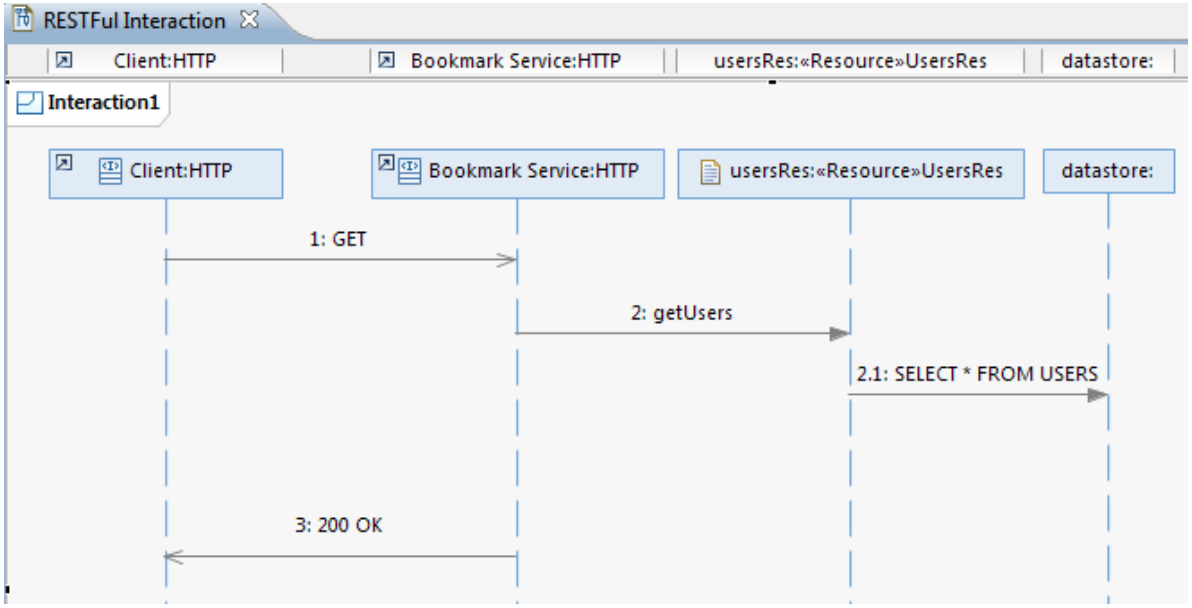
Modeling RESTful interactions in Sequence Diagrams

- You can also detail each request or response in terms of the URI, headers and content using the HTTP properties tab for a message



Modeling RESTful interactions in Sequence Diagrams

- To detail implementation side, you can further details the sequence diagram with calls to actual Resource classes
- Simply drag your Resource classes on the sequence diagrams and draw messages to it



BIRT reports for REST services

- Generating documentation using BIRT Reports

REST Resource Report

Resource	User
URL	/users/{username}
Description	This Resource can be used to access individual users.

Method	***Description***
DELETE	Delete a user.

Produces

Consumes

Parameters

Name	***Type***	***Default Value***
username	PathParam	

Return Codes

Code	***Content***	***Description***
200 OK		Deletes a user.

Method	***Description***
GET	Get details for a user.

Produces application/xml

Consumes

Parameters

Name	***Type***	***Default Value***
username	PathParam	

Return Codes

Code	***Content***	***Description***
200 OK	<user> <name>Tom</name> <age>35</age> </user>	This method returns the user details in xml.

JAX-RS Support

JAX-RS: The Java API for RESTful Web Services

- JAX-RS: Java API for RESTful Web Services provides Java API for creating REST Services

- JAX-RS uses annotations to simplify the development and deployment of web services
  - @Path, specifies the relative path for a resource class.
  - @GET□@PUT, @POST□@DELETE, specifies the HTTP request type of a resource method.
  - @Produces, specifies the returned MIME media types etc
  - ….

```
@Path("widgets")
@Produces("text/plain")
public class WidgetsResource {

        @GET
        @Path("offers")
        public WidgetList getDiscounted() {

        }

        @Path("{id}")
        public WidgetResource findWidget(@PathParam("id") String id) {
                return new WidgetResource(id);
        }
}

public class WidgetResource {
        public WidgetResource(String id) {    }
        @GET
        public Widget getDetails() {}
}
```

JAX-RS: The Java API for RESTful Web Services

Sub-Resource Method

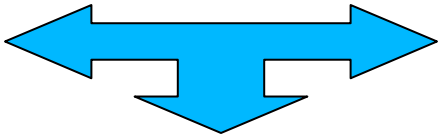
```
public class World{
      @PUT
      public void create(){…}

      @Path({"continents"})
      @GET
      public void get(){…}

      @Path({"continents"})
      @PUT
     public void modify(){…}

      ……..
}
```

JAXRS Domain Concepts



Sub-Resource Locator

```
public class Res1{

    @Path({"subRes"})
    public SubResource getSubResource(){…}

}

public class SubResource{

        @GET
        public void Operation1(){…}

}
```

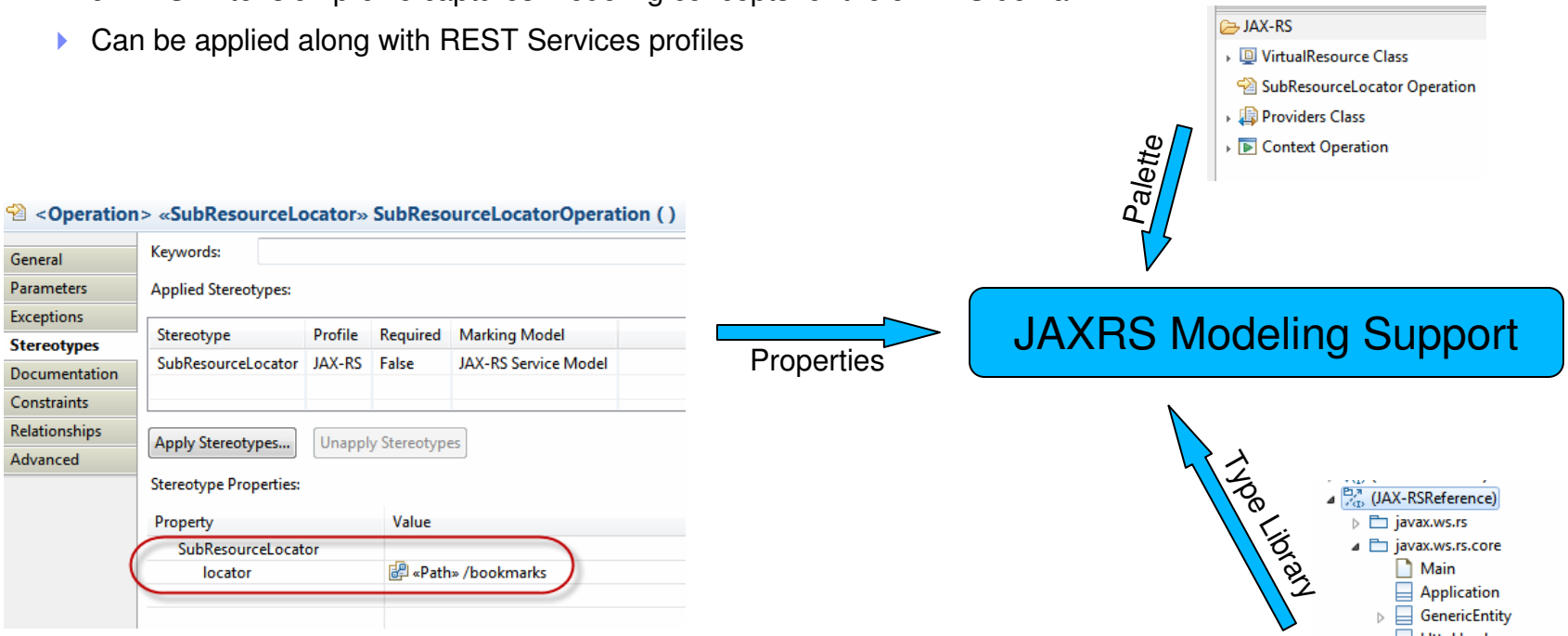
```
@Provider
@Produces(application/xml)
public class WorldProvider
    implements MessageBodyWriter<World>{


}
```

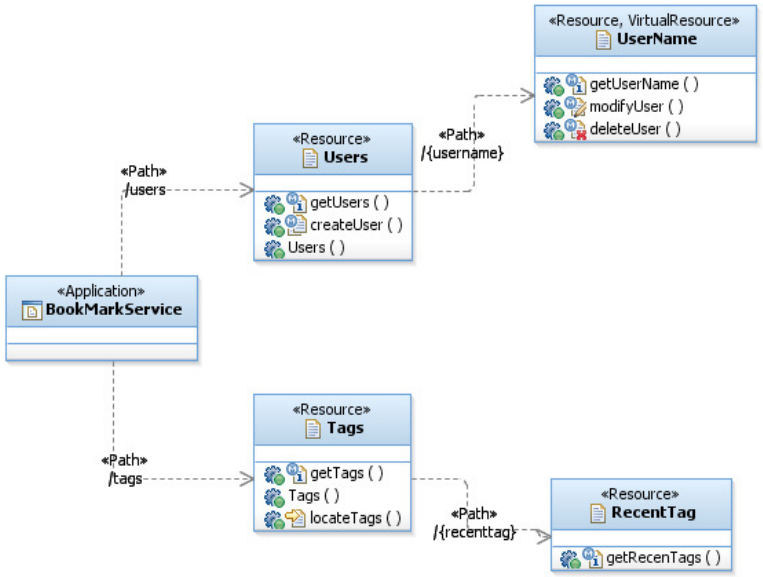
Provider for Type Conversion

JAX-RS Modelling

- REST Profile is independent of any target platform
  - JAXRS Extension profile captures modeling concepts for the JAXRS domain
  - Can be applied along with REST Services profiles



JAX-RS Code Generation




```java
@Path("/users")
@Produces("application/xml")
public class Users {

    public Users() {
        // TODO Auto-generated constructor stub
    }

    @GET
    @Produces("text/html")
    public List getUsers(){
        return null;
    }

    @PUT
    @Consumes("text/plain")
    public void modifyUsers(String name){

    }

    @Path("{username}")
    public UserName getUserName(){
        return new UserName();
    }

}
```

Reverse Engineering

- Allows reverse transforming JAXRS code into REST service model
- Complete RTE support for incremental development

Demo

Bookmark Service

Resource	URL
Users	\users
User	\users\{name}
Bookmarks	\users\{name}\bookmarks
Bookmark	\users\{name}\bookmarks\{id}

www.ibm/software/rational

www.ibm/software/rational