Introduction to Web Services (SOAP & REST)

Working with JAX-RS

Lesson Objectives

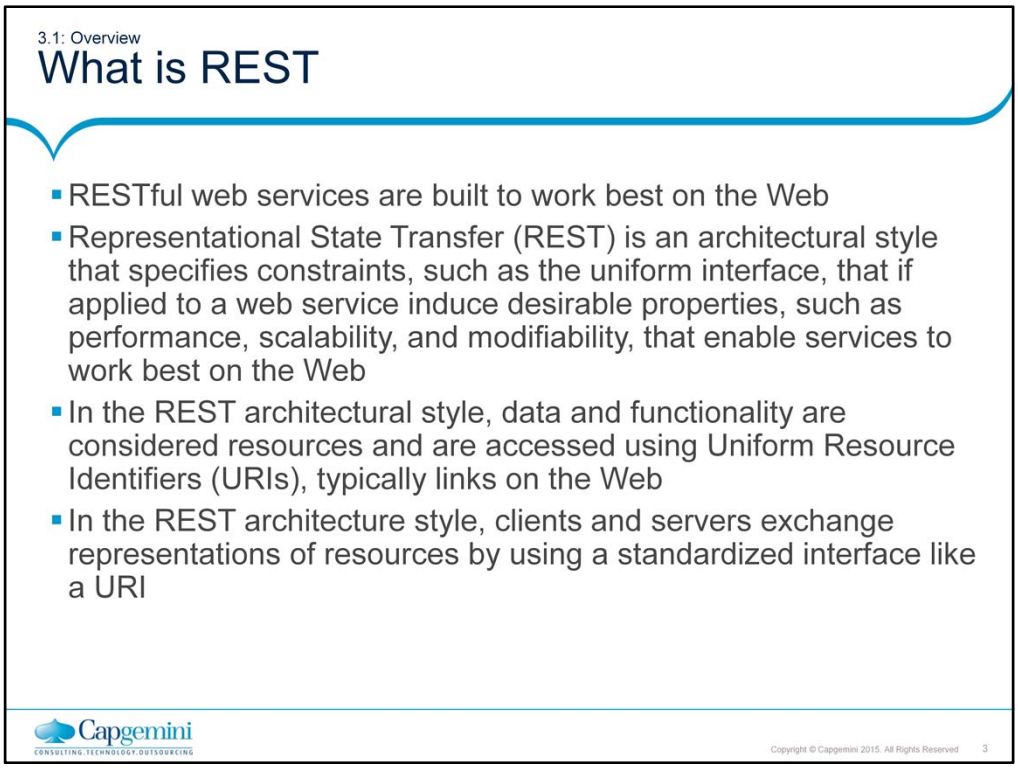
- What is REST
- Working with JAX - RS
- JAX – RS Annotations
- Creating JAX – RS web service
- Consuming RESTful service

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 2

Following contents would be covered:

3.1: What is REST
    3.1.1 : SOAP and REST
3.2: Working with JAX - RS
3.3: JAX-RS annotations
3.4: Creating JAX – RS web service
3.5: Consuming RESTful service



REST is a lightweight alternative to mechanisms like RPC (Remote Procedure Calls) and Web Services (SOAP)

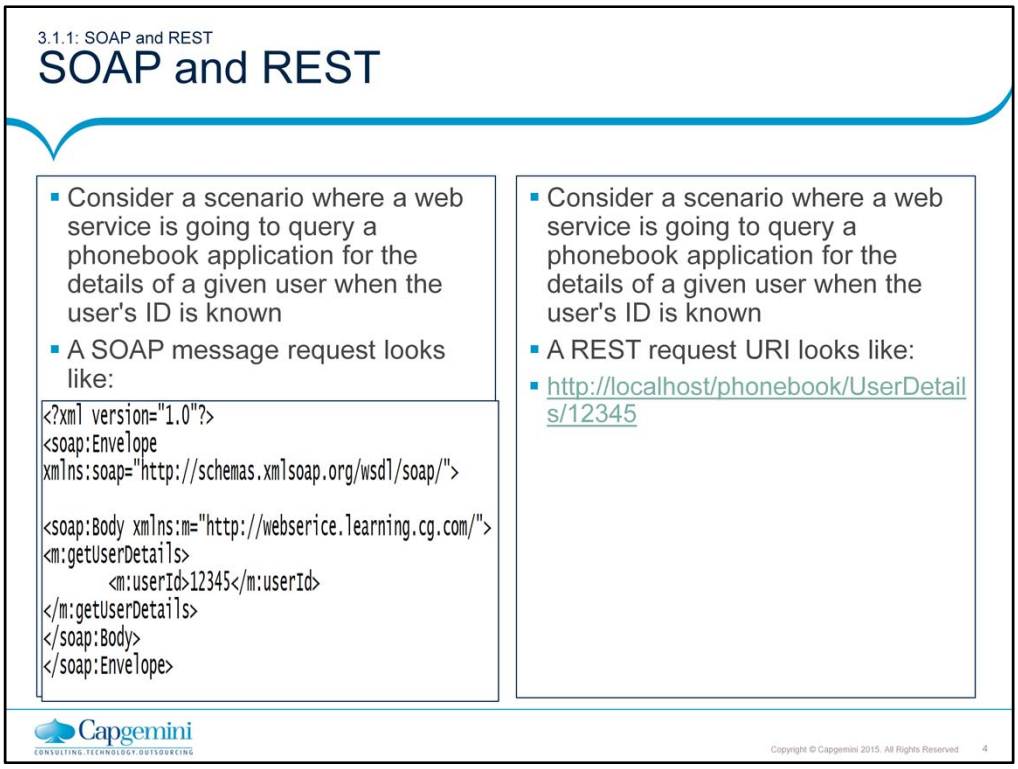
The following principles encourage RESTful applications to be simple, lightweight, and fast:

Resource identification through URI: A RESTful web service exposes a set of resources.
Resources are identified by URIs, which provide a global addressing space for resource and service discovery.

Uniform interface: Resources are manipulated using a fixed set of four create, read, update, delete operations: PUT, GET, POST, and DELETE. PUT updates an existing resource, which can be then deleted by using DELETE. GET retrieves the current state of a resource in some representation. POST transfers a new state onto a resource.

Self-descriptive messages: Resources are decoupled from their representation so that their content can be accessed in a variety of formats, such as HTML, XML, plain text, PDF, JPEG, JSON, and others. Metadata about the resource is available and used, for example, to control caching, detect transmission errors, negotiate the appropriate representation format, and perform authentication or access control.

Stateful interactions through hyperlinks: Every interaction with a resource is stateless; that is, request messages are self-contained. Stateful interactions are based on the concept of explicit state transfer. Several techniques exist to exchange state, such as URI rewriting, cookies, and hidden form fields. State can be embedded in response messages to point to valid future states of the interaction



As seen in above slide same scenario is represented in both forms SOAP and REST
REST turns out to be an easier implementation.

REST vs. SOAP is like mailing a letter:
with SOAP, you're using an envelope;
with REST, it's a postcard. Postcards are easier to handle (by the receiver), waste less paper (i.e., consume less bandwidth), and have a short content.

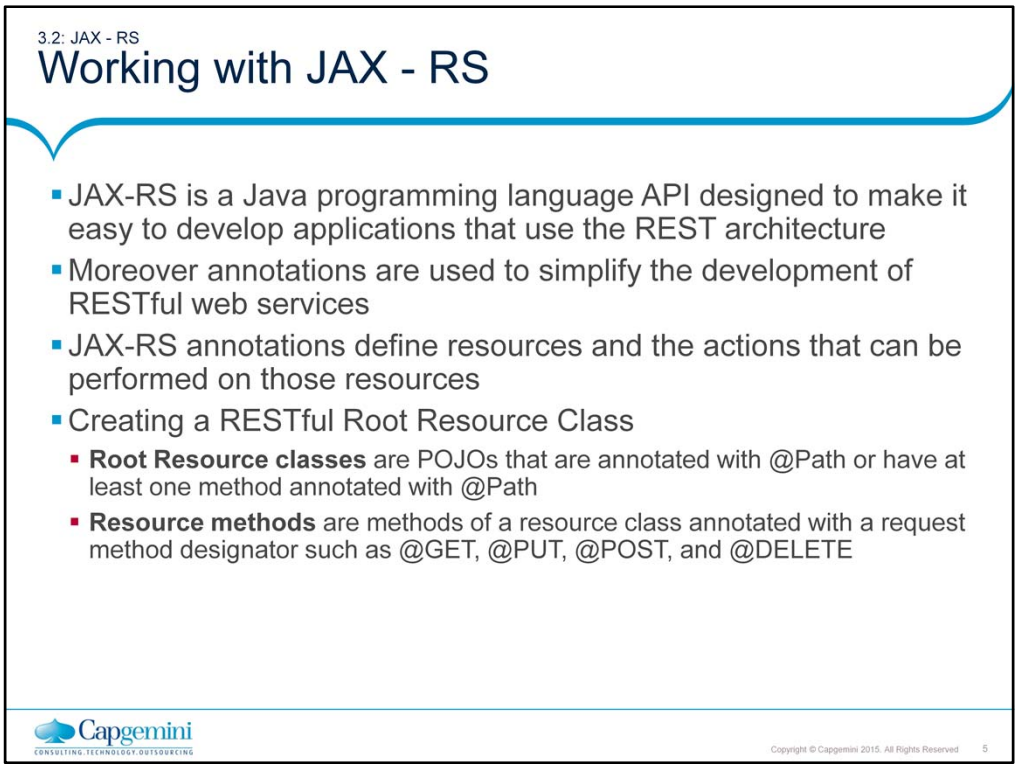
With REST, the semantics are specified entirely by the URI
With REST, the URI is the end
REST: oriented around nouns – Resources

Following examples can be considered:
http://example.com/customer/123
http://example.com/order/555/customer
{POST, GET, DELETE}
REST services are built around Resources

REST services are Stateless and use a Uniform Interface. Resources are manipulated through Representations. Messages are Self-Describing

Differences between SOAP and REST

SOAP	REST
1. Is Simple Object Access Protocol	1. REST is Representational State Transfer
2. SOAP requires more bandwidth	2. REST requires less bandwidth
3. SOAP permits only XML	3. REST permits different data formats like text, XML, JSON (Java script Object Notation) data formats
4. SOAP is Transport layer independent (HTTP). It can be used over SMTP as well.	4. REST requires use of HTTP
5. SOAP is a standard to interact with Web service	5. REST has small learning curve and uses URI to Interact with Web services



There are many implementations of JAX – RS

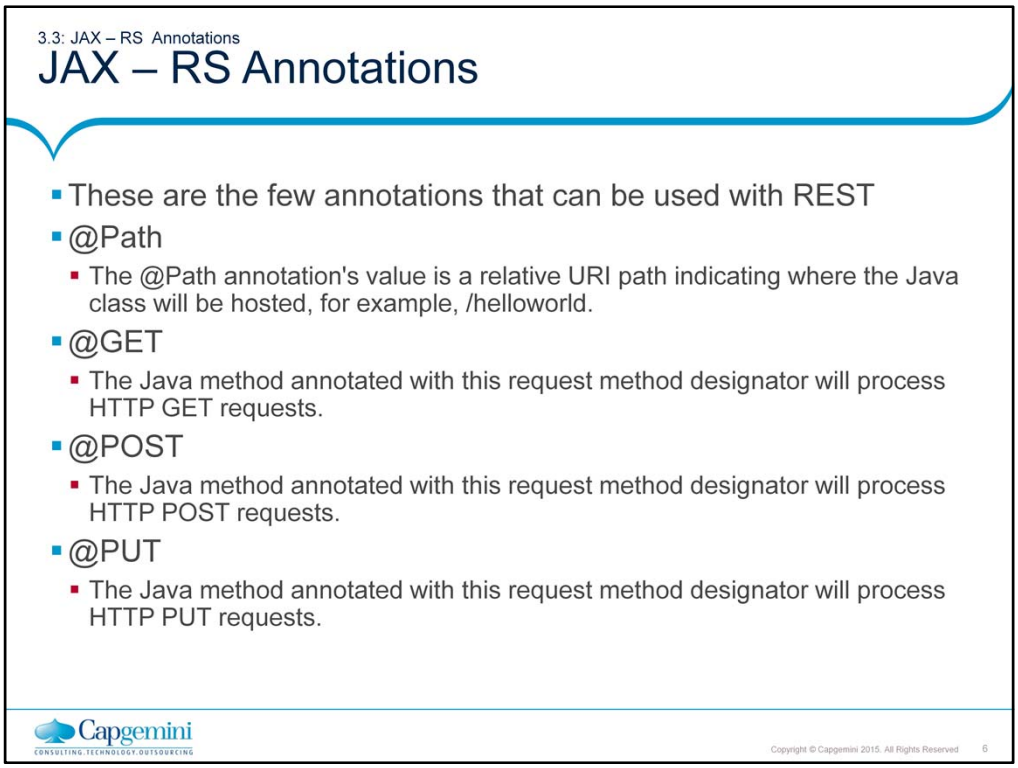
Apache CFX, an Open source Web service framework

Jersey , the reference implementation from Oracle. We would be using Jersey implementation for our REST web services

Jersey is used because it makes development of RESTful web services easier.

RESTful web services created under Jersey can be deployed to any application server.

Moreover JAX – RS is a part of Java EE and can be used with other Java EE technologies.



The @Path annotation's value is a relative URI path indicating where the Java class will be hosted: for example, /helloworld. You can also embed variables in the URIs to make a URI path template. For example, you could ask for the name of a user and pass it to the application as a variable in the URI: /helloworld/{username}.

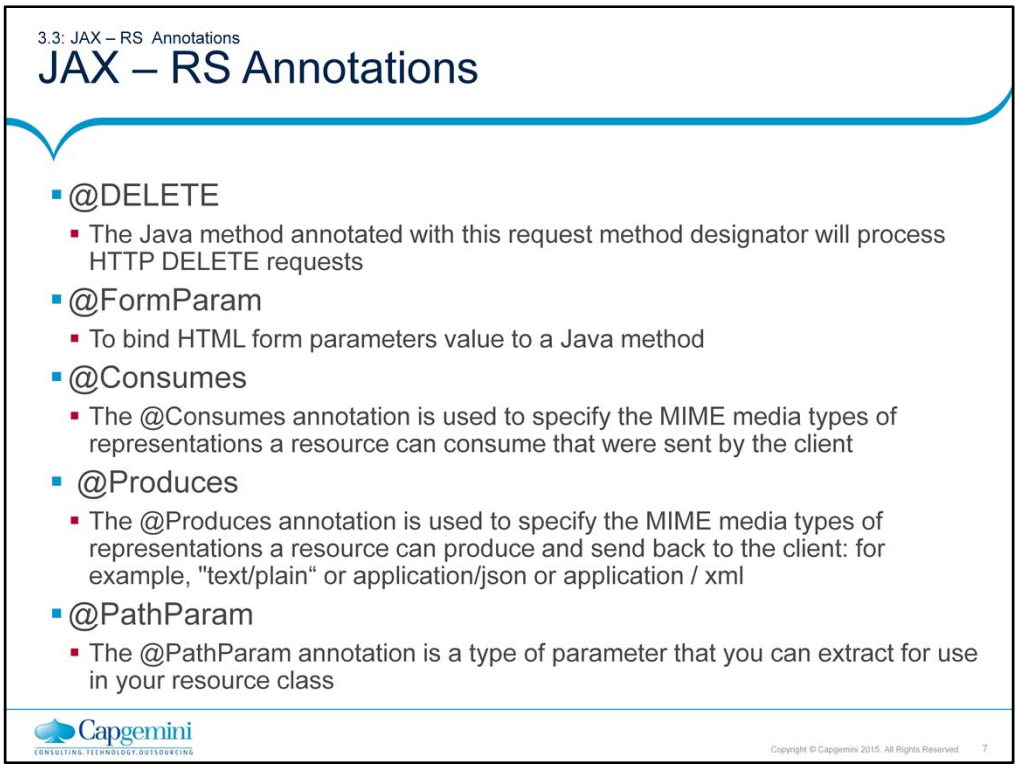
The @GET annotation is a request method designator and corresponds to the similarly named HTTP method. The Java method annotated with this request method designator will process HTTP GET requests. The behavior of a resource is determined by the HTTP method to which the resource is responding.

URI for GET request in JAX-RS-CRUD Demo shared (http://localhost:9090/JAX-RS-CRUD/rest/countries)

The @POST annotation is a request method designator and corresponds to the similarly named HTTP method. The Java method annotated with this request method designator will process HTTP POST requests. The behavior of a resource is determined by the HTTP method to which the resource is responding.

URI for POST request in JAX-RS-CRUD Demo shared (http://localhost:9090/JAX-RS-CRUD/post.jsp) (Here the user enters the details to create a country. The values of which are passed to the controller via @FormParam-discussed in next slide)

The @PUT annotation is a request method designator and corresponds to the similarly named HTTP method. The Java method annotated with this request method designator will process HTTP PUT requests. The behavior of a resource is determined by the HTTP method to which the resource is responding.



The @DELETE annotation is a request method designator and corresponds to the similarly named HTTP method. The Java method annotated with this request method designator will process HTTP DELETE requests. The behavior of a resource is determined by the HTTP method to which the resource is responding

URI for DELETE request in JAX-RS-CRUD Demo shared (http://localhost:9090/JAX-RS-CRUD/rest/countries/delete)

@FormParam: Used to bind HTML form parameters value to a Java method

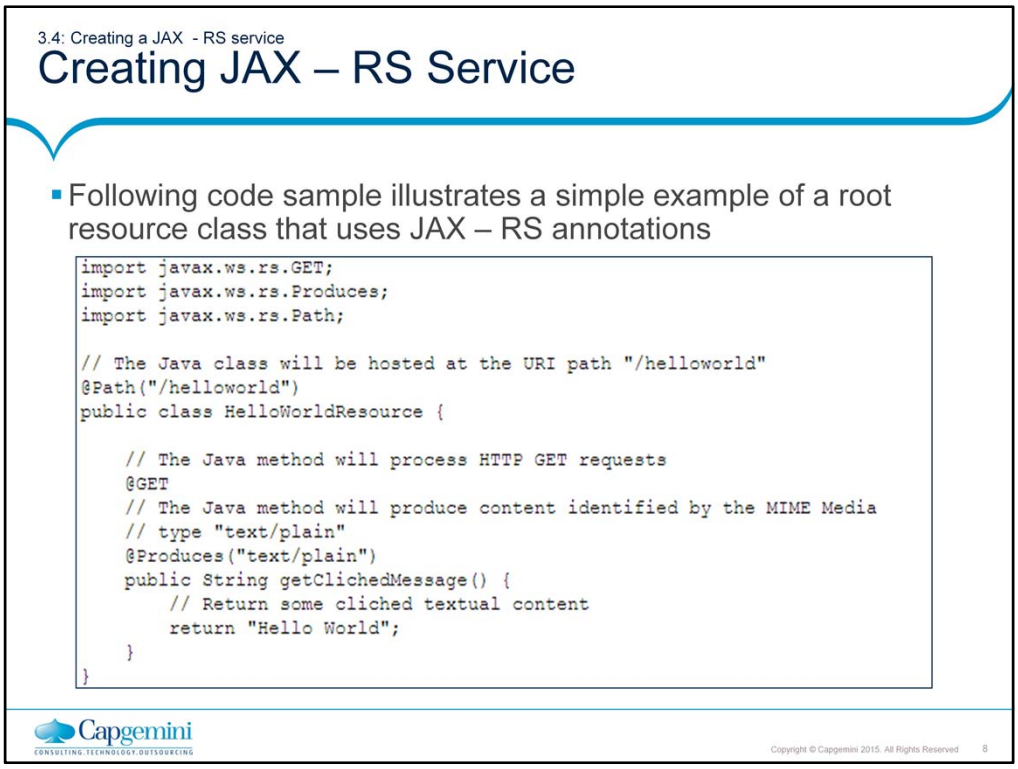
With respect to the JAX-RS-CRUD demo shared - the values entered in post.jsp mentioned in previous slide are passed to controller method. Since we are using HTTP POST method the URI is not appended with parameters.
Following is the URI (http://localhost:9090/JAX-RS-CRUD/rest/countries) to demonstrate that new country is added.

The @Consumes annotation is used to specify the MIME media types of representations a resource can consume that were sent by the client.
@Consumes(MediaType.APPLICATION_JSON) specifies that the controller method will need an input parameter of type object of type JSON

The @Produces annotation is used to specify the MIME media types of representations a resource can produce and send back to the client: for example, "text/plain"..
In JAX-RS-CRUD demo shared - @Produces(MediaType.APPLICATION_JSON) specifies that the controller method will return an object of type JSON

The @PathParam annotation is a type of parameter that you can extract for use in your resource class. URI path parameters are extracted from the request URI, and the parameter names correspond to the URI path template variable names specified in the @Path class-level annotation

URI for GET request with Path parameter in JAX-RS-CRUD Demo shared (http://localhost:9090/JAX-RS-CRUD/rest/countries/3) (3 is the path parameter appended to the URI). Thus details with respect to country Id = 3 is fetched and displayed in browser.



The @Path annotation's value is a relative URI path. In the preceding example, the Java class will be hosted at the URI path /helloworld. This is an extremely simple use of the @Path annotation, with a static URI path.

Variables can be embedded in the URIs. URI path templates are URIs with variables embedded within the URI syntax.

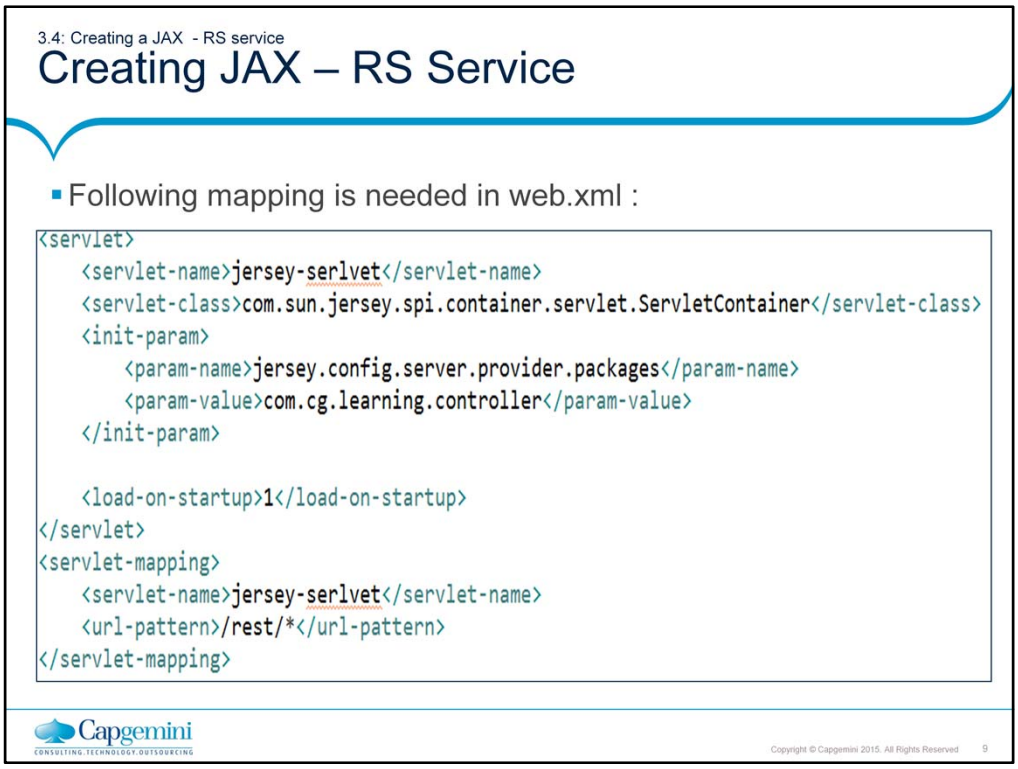
The @GET annotation is a request method designator, along with @POST, @PUT, @DELETE, and @HEAD, defined by JAX-RS and corresponding to the similarly named HTTP methods. In the example, the annotated Java method will process HTTP GET requests.
The behavior of a resource is determined by the HTTP method to which the resource is responding.

The @Produces annotation is used to specify the MIME media types a resource can produce and send back to the client. In this example, the Java method will produce representations identified by the MIME media type "text/plain".

The @Consumes annotation is used to specify the MIME media types a resource can consume that were sent by the client.

The example could be modified to set the message returned by the getClichedMessage method, as shown in this code example:

```
@POST
@Consumes("text/plain")
public void postClichedMessage(String message)
{
                    // Store the message
}
```



From the above mapping all URL patterns will map through URLs with rest being a part of the URI.

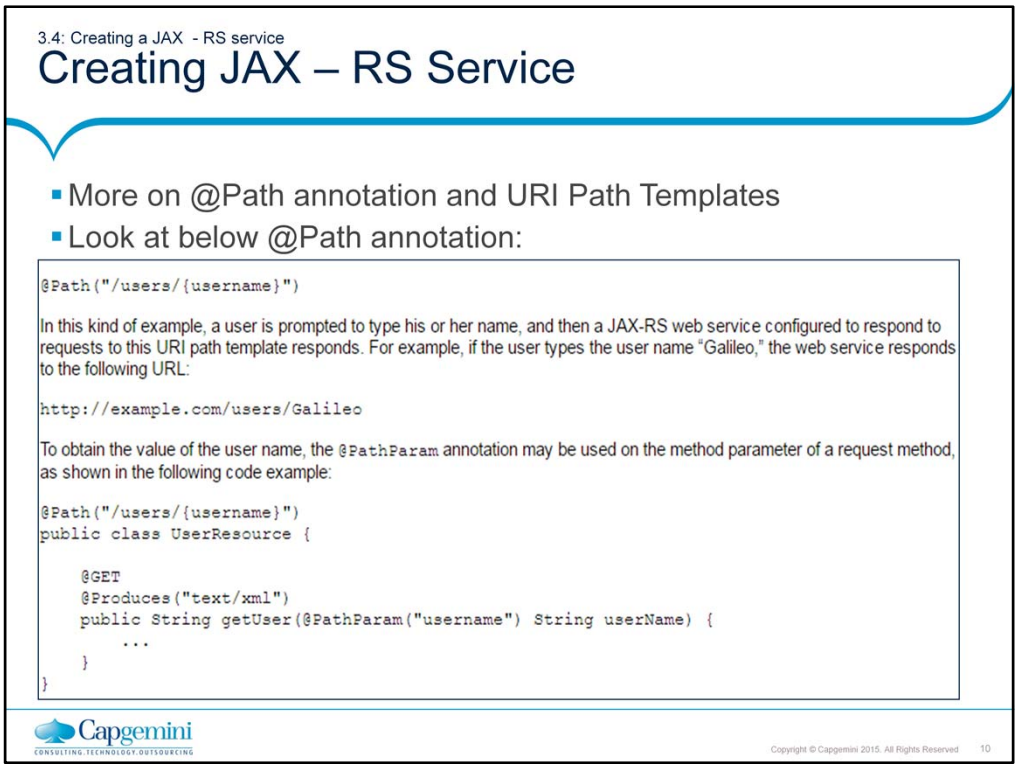
An inbuilt servlet -> com.sun.jersey.spi.container.servlet.ServletContainer is a servlet (Filter) for deploying root resource classes.

Provides support for servlet-based and filter-based Web applications.

The above mapping will deploy Jersey and automatically register any root resource or provider classes present in the directory "/WEB-INF/classes" or jar files present in the directory "/WEB-INF/lib“

The init-param -> jersey.config.server.provider.packages defines one or more packages that contain application-specific resources and providers.

If the property is set, the specified packages will be scanned for JAX-RS root resources and providers.



The @Path annotation identifies the URI path template to which the resource responds and is specified at the class or method level of a resource.

The @Path annotation's value is a partial URI path template relative to the base URI of the server on which the resource is deployed, the context root of the application, and the URL pattern to which the JAX-RS runtime responds.

URI path templates are URIs with variables embedded within the URI syntax. These variables are substituted at runtime in order for a resource to respond to a request based on the substituted URI. Variables are denoted by braces ({ and }). For example, look at the following @Path annotation:
@Path("/users/{username}")

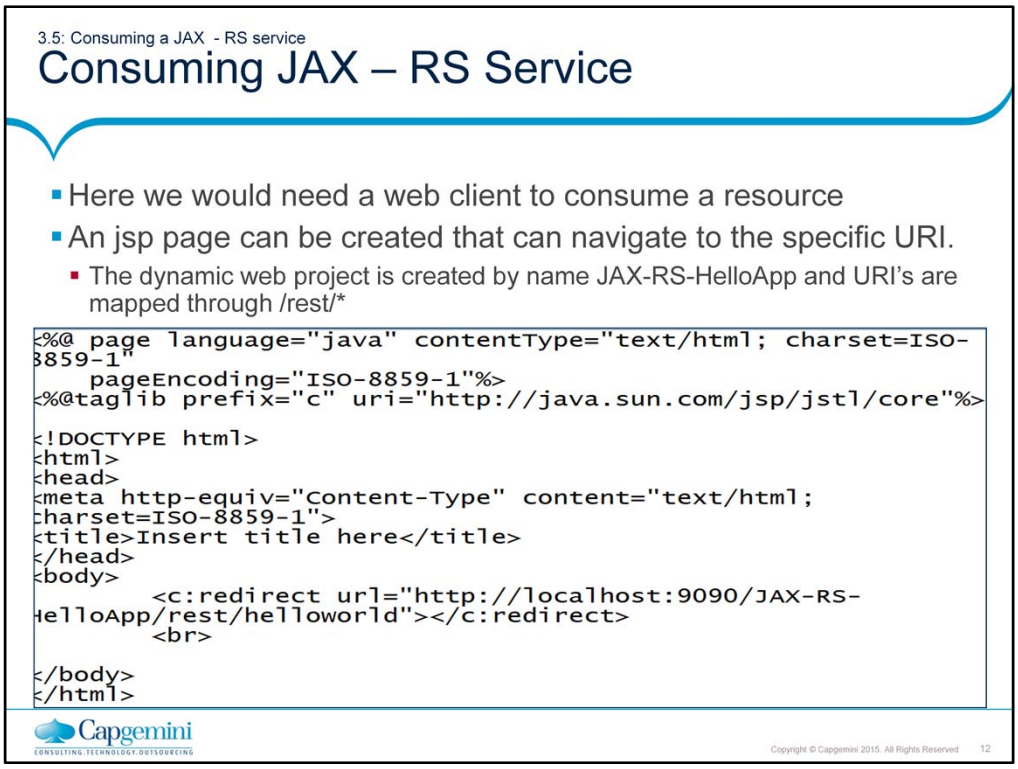
3.4: Creating a JAX - RS service

Creating JAX – RS Service

URI Path Template	URI After Substitution
http://example.com/{name1}/{name2}/	http://example.com/james/gatz/
http://example.com/{question}/{question}/{question}	http://example.com/why/why/why/
http://example.com/maps/{location}	http://example.com/maps/MainStreet
http://example.com/{name3}/home/	http://example.com/james/home/

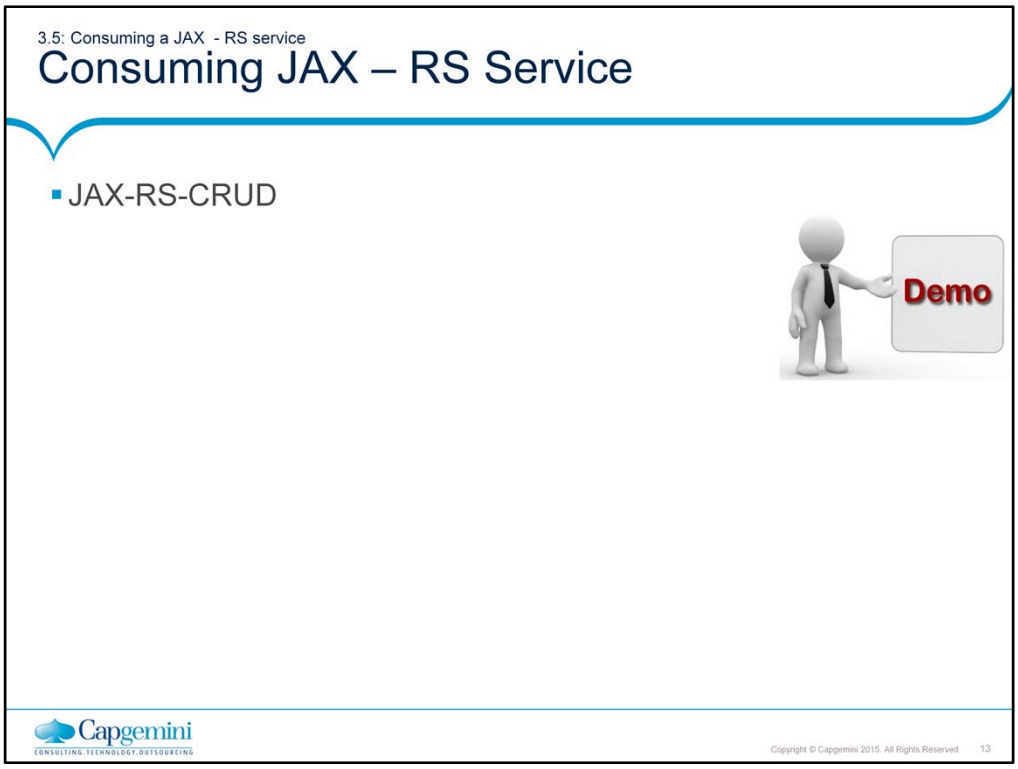
Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 11

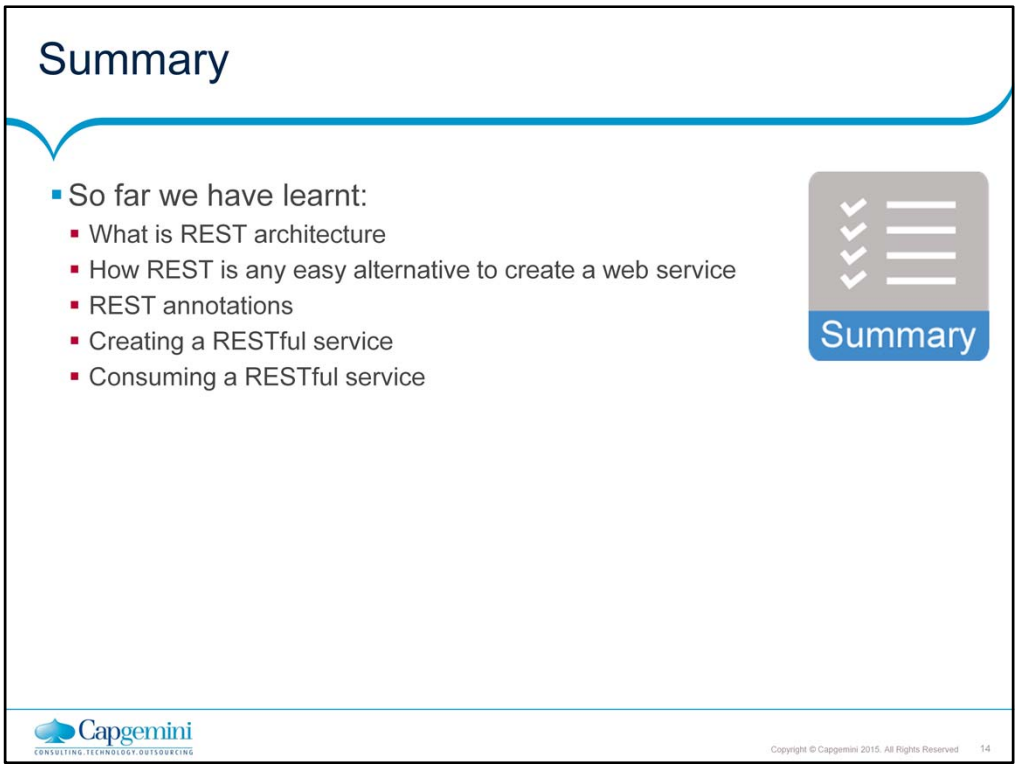


When above jsp is executed, REST Service will look for Root resource class with @Path mapping as "/helloworld" and on finding the mapping will execute that corresponding method and return result in MIME type specified.

(Refer slide number 8)



Add the notes here.



Add the notes here.

Working with JAX-WS

Lab

- Lab 2

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 15

Add the notes here.

Review Question

- Question 1: Which of the following are true?
  - Option1 : Resource classes are POJOs that have at least one method annotated with @Path
  - Option 2: Resource methods are methods of a resource class annotated with a request method designator such as @GET, @PUT, @POST, or @DELETE
  - Option 3: @FormParam binds query parameters value to a Java method

- Question 2: The @Path annotation's value is a relative URI path indicating where the Java class will be hosted?
  - True
  - False

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 16

Add the notes here.

Review Question

- Question 3: _____ specifies a media type a resource can generate.
  - @PUT
  - @POST
  - @Produces
  - @Consumes

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 17

Add the notes here.