Introducing the JAX-RS API

Phuc H. Duong, *M.Sc.*

phuc@newai.vn

Textbook

- Bogunuva Mohanram Balachandar. RESTful Java Web Services - Third Edition (2017). Packt Publishing Limited.
  - Chapter 3: Introducing the JAX-RS API

Sample Code

- Available at https://go.newai.vn/2TOgMFP

Update Log

- 11/01/2020: release first version of this slide

Objective

- An overview of the JAX-RS annotations
- Understanding data binding in JAX-RS
- Building your first RESTful web service with JAX-RS
- Client APIs for accessing RESTful web services

Outline

1. An overview of JAX-RS
2. JAX-RS annotations
3. Understanding data binding rules in JAX-RS
4. Building your first RESTful web service with JAX-RS
5. Client APIs for accessing RESTful web services
6. Summary

An overview of JAX-RS

An overview of JAX-RS

- Some of the popular JAX-RS implementations:
  - **Jersey RESTful web service framework** ⭐
    - This framework is an open source framework for developing RESTful web services in Java
    - It serves as a JAX-RS reference implementation. You can learn more about this project at https://jersey.github.io

An overview of JAX-RS

- Some of the popular JAX-RS implementations:
  - **Apache CXF**
    - This framework is an open source web services framework
    - CXF supports both JAX-WS and JAX-RS web services
    - To learn more about CXF, refer to http://cxf.apache.org

An overview of JAX-RS

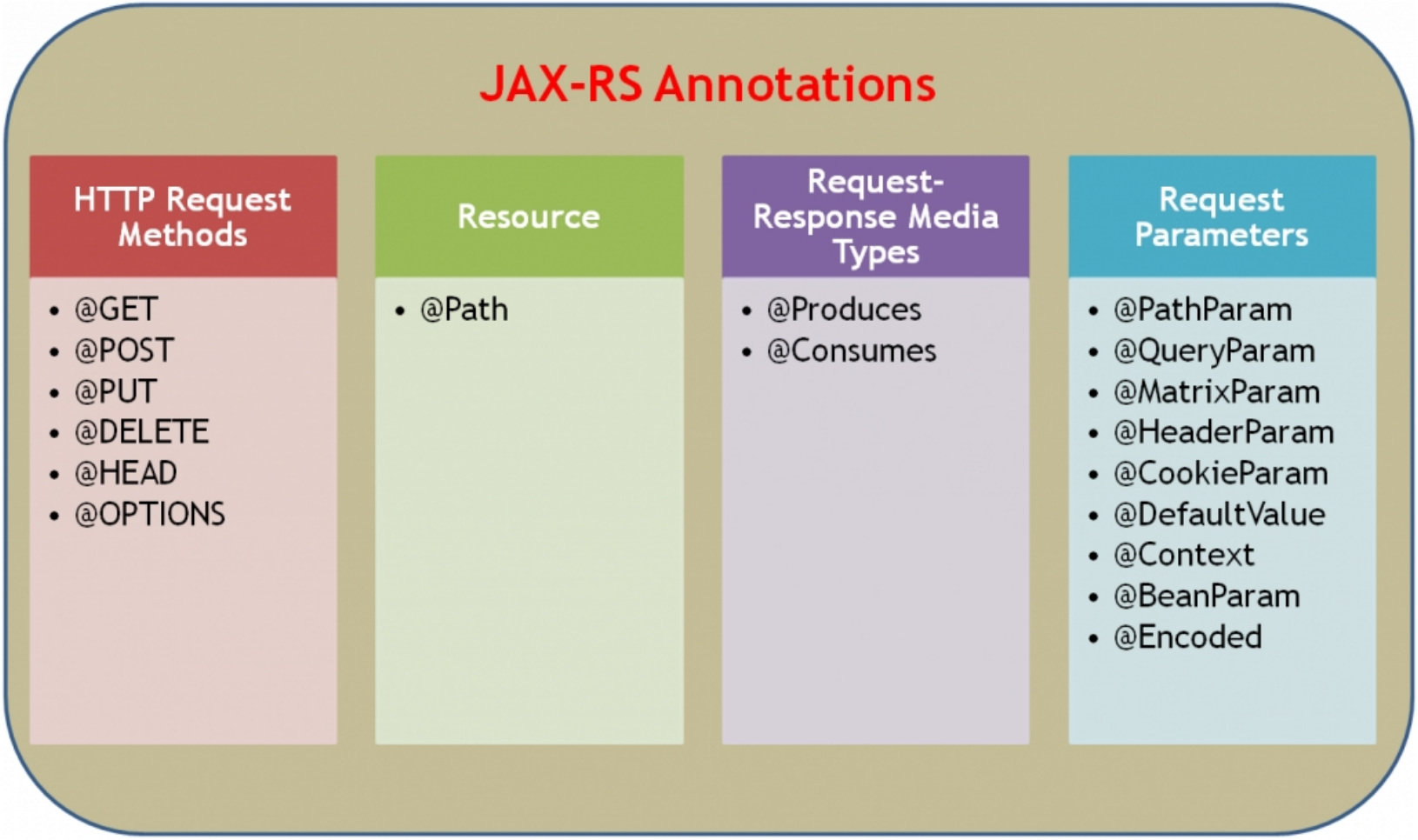
- Some of the popular JAX-RS implementations:
  - **RESTEasy**
    - This framework is an open source project from JBoss, which provides various modules to help you build a RESTful web service
    - To learn more about RESTEasy, refer to http://resteasy.jboss.org

An overview of JAX-RS

- Some of the popular JAX-RS implementations:
  - **Restlet**
    - This framework is a lightweight open source RESTful web service framework
    - It has good support for building both scalable RESTful web service APIs and lightweight REST clients (which suits mobile platforms well)
    - You can learn more about Restlet at http://restlet.com

JAX-RS annotations

JAX-RS annotations



Specifying the dependency

- To use JAX-RS APIs in your project, you need to add the **javax.ws.rs-api** JAR file to the class path
- If the consuming project uses ***Maven*** for building the source, the dependency entry in the Project Object Model (***POM***) file should be defined as follows:

```
<dependency>
    <groupId>javax.ws.rs</groupId>
    <artifactId>javax.ws.rs-api</artifactId>
    <version>2.0.1</version><!-- set the right version -->
    <scope>provided</scope><!-- compile time dependency -->
</dependency>
```

https://mvnrepository.com/artifact/javax.ws.rs/javax.ws.rs-api

Using JAX-RS annotations to build RESTful web services

- We will discuss 5 types of annotations:
  - defining a RESTful resource
  - specifying request-response media types
  - processing HTTP request methods
  - accessing request parameters
  - inheritance

Annotations for defining a RESTful resource

- REST resources are the fundamental elements of any RESTful web service
- A REST resource can be defined as an object that is of a specific type with the associated data and is optionally associated with other resources
- It also exposes a set of standard operations corresponding to the HTTP method types using the `@HttpMethod` annotation

Annotations for defining a RESTful resource

Resource classes are **Java POJO** classes, which use JAX-RS annotations to implement a web service

Each resource class must have at least one method annotated with `@Path` or a request method designator (`@HttpMethod`)

Only public methods of a resource class can be exposed as resource methods

@Path

- The `@javax.ws.rs.Path` annotation indicates the URI path to which a resource class or a class method must respond
- The value that you specify for the `@Path` annotation is relative to the URI of the server where the REST resource is hosted
- This annotation can be applied at both the class and method levels

@Path

- Specifying the `@Path` annotation on a resource class

```
import javax.ws.rs.GET;
import javax.ws.rs.Path;
import javax.ws.rs.Produces;
@Path("departments")
public class DepartmentService {
    @GET
    @Path("count")
    @Produces("text/plain")
    public Integer getTotalDepartments() {
        return findTotalRecordCount();
    }
  //Rest of the code goes here
}
```

To invoke method, you will use the URI as follows:

http://host:port/<context-root>/departments/count

@Path

- Specifying variables in the URI path template

```
import javax.ws.rs.Path;
import javax.ws.rs.DELETE;

@Path("departments")
public class DepartmentService {

  @DELETE
  @Path("{id}")
  public void removeDepartment(@PathParam("id")
  short id) {
    removeDepartmentEntity(id);
  }
  //Other methods removed for brevity
}
```

To invoke method, you will use the URI as follows:

http://host:port/<context-root>/departments/10

@Path

- Restricting values for path variables with regular expressions

```
@DELETE
@Path("{name: [a-zA-Z][a-zA-Z_0-9]}")
public void removeDepartmentByName(@PathParam("name")
  String deptName) {
    //Method implementation goes here
}
```

To invoke method, you will use the URI as follows:

http://host:port/<context-root>/departments/name1

Annotations for specifying request-response media types

- The content types are represented using standard internet media types
- JAX-RS allows to specify which internet media types of representation a resource can produce or consume by using
  - `@javax.ws.rs.Produces`
  - `@javax.ws.rs.Consumes`

@Produces

- **@javax.ws.rs.Produces** annotation is used for defining the internet media type(s) that a REST resource class method can return to the client
- A REST API can produce:
  - `application/atom+xml`
  - `application/json`
  - `application/octet-stream`
  - `application/svg+xml`
  - `application/xhtml+xml`
  - `application/xml text/html`
  - `text/plain`
  - `text/xml`

@Consumes

- @javax.ws.rs.Consumes annotation defines the internet media type(s) that the resource class methods can accept
- A REST API can consume:
  - application/atom+xml
  - application/octet-stream
  - application/xhtml+xml
  - text/plain
  - multipart/form-data
  - application/json
  - application/svg+xml
  - application/xml text/html
  - text/xml
  - application/x-www-form-urlencoded

Annotations for processing HTTP request methods

- For processing HTTP request methods
- RESTful web services communicate over HTTP with the standard HTTP verbs, such as `GET`, `PUT`, `POST`, `DELETE`, `HEAD`, and `OPTIONS`

@Get

- A RESTful system uses the HTTP GET method type for retrieving the resources referenced in the URI path
- The `@javax.ws.rs.GET` annotation designates the method of a resource class to respond to the HTTP GET requests.

@Put

- The HTTP PUT method is used for updating or creating the resource pointed by the URI
- The `@javax.ws.rs.PUT` annotation designates the method of a resource class to respond to the HTTP PUT requests
- The PUT request generally has a message body carrying the payload, such as the JSON object, XML structure, plain text, HTML content, or binary stream
- When a request reaches a server, the framework intercepts the request and directs it to the appropriate method that matches the URI path and the HTTP method type

@Post

- The HTTP POST method posts data to the server
- Typically, this method type is used for creating a resource
- The `@javax.ws.rs.POST` annotation designates the method of a resource class to respond to the HTTP POST requests

@Delete

- The HTTP DELETE method deletes the resource pointed by the URI
- The `@javax.ws.rs.DELETE` annotation designates the method of a resource class to respond to the HTTP DELETE requests

@Head

- The `@javax.ws.rs.HEAD` annotation designates a method to respond to the HTTP HEAD requests
- The HEAD method is the same as the GET request, but it only transfers the status line along with the header section (without the response body) to the client
- This method is useful for retrieving the metadata present in the response headers

@Options

- The `@javax.ws.rs.OPTIONS` annotation designates a method to respond to the HTTP OPTIONS requests
- This method is useful for obtaining a list of HTTP methods allowed for a resource.

Annotations for accessing request parameters

- JAX-RS offers annotations to pull some information out of a request
- You can use this offering to extract the following parameters from a request:
  - a query
  - URI path
  - form
  - cookie
  - header
  - matrix

@PathParam

- A URI path template has a URI part pointing to the resource
- It can also take the path variables embedded in the syntax; this facility is used by the clients to pass parameters to the REST APIs, as appropriate
- The `@javax.ws.rs.PathParam` annotation injects (or binds) the value of the matching path parameter present in the URI path template into
  - a class field,
  - a resource class bean property (the getter method for accessing the attribute),
  - or a method parameter

@QueryParam

- The `@javax.ws.rs.QueryParam` annotation injects the value(s) of a HTTP query parameter into a class field, a resource class bean property (the getter method for accessing the attribute), or a method parameter

@MatrixParam

- Matrix parameters are another way of defining parameters in the URI path template
- The matrix parameters take the form of name-value pairs in the URI path, where each pair is preceded by a semicolon (`;`)
  - For instance, the URI path that uses a matrix parameter to list all departments in Bangalore city: `/departments;city=Bangalore`
- The `@javax.ws.rs.MatrixParam` annotation injects the matrix parameter value into a class field, a resource class bean property (the getter method for accessing the attribute), or a method parameter

@HeaderParam

- The HTTP header fields provide the necessary information about the request and response contents in HTTP
  - For example, the header field, Content-Length: 348, for an HTTP request says that the size of the request body content is 348 octets (8-bit bytes)
- The `@javax.ws.rs.HeaderParam` annotation injects the header values present in the request into a class field, a resource class bean property (the getter method for accessing the attribute), or a method parameter.

@CookieParam

- The `@javax.ws.rs.CookieParam` annotation injects the matching cookie parameters present in the HTTP headers into a class field, a resource class bean property (the getter method for accessing the attribute), or a method parameter

@FormParam

- The `@javax.ws.rs.FormParam` annotation injects the matching HTML form parameters present in the request body into a class field, a resource class bean property (the getter method for accessing the attribute), or a method parameter
- The request body carrying the form elements must have the content type specified as *application/x-www-form-urlencoded*

@DefaultValue

- The `@javax.ws.rs.DefaultValue` annotation specifies a default value for the request parameters accessed using one of the following annotations: *PathParam*, *QueryParam*, *MatrixParam*, *CookieParam*, *FormParam*, or *HeaderParam*
- The default value is used if no matching parameter value is found for the variables annotated using one of the preceding annotations

@Context

- The JAX-RS runtime offers different context objects, which can be used for accessing information associated with the resource class, operating environment, and so on
- There are various context objects that hold information associated with the URI path, request, HTTP header, security, and so on
- Some of these context objects also provide the utility methods for dealing with the request and response content

@BeanParam

- The `@javax.ws.rs.BeanParam` annotation allows you to inject all the matching request parameters into a single bean object

@Encoded

- the JAX-RS runtime decodes all the request parameters before injecting the extracted values into the target variables
- Using `@javax.ws.rs.Encoded` to disable the automatic decoding of the parameter values

Annotation inheritance

- JAX-RS annotations defined in a class or interface method are inherited by the corresponding subclass or implementation class method, provided that the method and its parameters do not define their own annotations

Returning additional metadata with responses

- JAX-RS allows to return additional metadata via the `javax.ws.rs.core.Response` class, which wraps the entity and any additional metadata, such as the HTTP headers, HTTP cookie, and status code

Understanding data binding rules in JAX-RS

Understanding data binding rules in JAX-RS

- Read the textbook [1]
  - Chapter 3
  - Section: Understanding data binding rules in JAX-RS (Pg. 105)

Building your first RESTful web service with JAX-RS

Setting up the environment

- This example uses the following software and tools:
  - Java SE Development Kit 8 or newer
  - NetBeans IDE 8.2 (with Java EE bundle) or newer
  - Glassfish Server 4.1 or newer
  - Maven 3.2.3 or newer
  - Oracle Database Express Edition 11g Release 2 or newer with HR sample database schema
  - Oracle Database JDBC Driver (ojdbc7.jar or newer)

Building your first RESTful web service with JAX-RS

- Read the textbook [1]
  - Chapter 3
  - Section: Building your first RESTful web service with JAX-RS (Pg. 110)

Client APIs for accessing RESTful web services

Client APIs for accessing RESTful web services

- Read the textbook [1]
  - Chapter 3
  - Section: Client APIs for accessing RESTful web services (Pg. 121)

Summary

Summary

- With the use of annotations, the JAX-RS API provides a simple development model for RESTful web service programming
- In this chapter, we covered the frequently used features of the JAX-RS framework and developed a non-trivial RESTful web service to get a feel of the offerings
- In the next chapter, we will discuss the advanced features of JAX-RS, such as asynchronous REST APIs, advanced topics on filters and interceptors, validations and error handling, custom message body providers, and custom media types. Take a deep breath and be prepared for a deep dive

END OF CHAPTER