## 0. Introduction

### 1. Jakarta Web Services - Integrating applications over the internet

### 2. What you should know

## 1. Web Services - What and Why

### 1. What is a web service

### 2. Why are web services needed

### 3. APIs, web services, and endpoints

### Chapter Quiz

Question 1 of 3

A web service is a piece of software that enables applications to share data \_\_\_\_\_.

between applications irrespective of the technologies they are based on and on what platforms they operate

Correct

A web service is a piece of software that enables applications to share data between them without having to worry about what technologies they are based on and on what platforms they operate.

between two web based applications

Incorrect

between web based applications & a mobile applications

Incorrect

between web applications that are based on the same technology

Incorrect

Question 2 of 3

Which of these statements are true about web services, APIs and Endpoints?

A. With APIs, Machine-to-Machine interactions are possible.

B. All web services are public APIs.

C. An endpoint is where an API or a web service has its resource located.

A & C

Incorrect

Web services enable an application running on one machine to talk to an application running on another machine over a network to call each other's functionality, or in other words machine-to-machine interaction. Though Web services are APIs not all APIs are web services. Therefore, machine-to-machine interactions are not possible with all APIs.

C

Correct

Web services enable an application running on one machine to talk to an application running on another machine over a network to call each other's functionality, or in other words machine-to-machine interaction. A web service is an API because it exposes an application’s data and/or functionality. However, an API is not essentially a web service because there can be APIs that can work offline, not needing a network to function. Web services expose their functionality to be used only by specific partners. So they are not public. In terms of a web service an endpoint is typically a uniform resource locator (URL) that provides the location of a resource on the server.

A, B, C

Incorrect

A & B

Incorrect

Web services enable an application running on one machine to talk to an application running on another machine over a network to call each other's functionality, or in other words machine-to-machine interaction. Though Web services are APIs not all APIs are web services. Therefore, machine-to-machine interactions are not possible with all APIs. Web services expose their functionality to be used only by specific partners. So they are not public.

Question 3 of 3

When would a web service be an ideal scenario out of these? A. a developer needs to calculate daily exchange rates for many currencies.

B. a bank needs to interconnect their web based banking application with a mobile banking app.

C. a large organization wants to migrate some of the modules of their ERP system to the cloud and have the other modules hosted in-house

A,B,C

Correct

Web services help improve the productivity of developers by making it possible to reuse existing functionality. They also enable applications to be integrated irrespective of technologies and platforms. Web services let enterprises take advantage of the cloud by allowing seamless integration of applications hosted on the cloud and on-premises.

A and B

Incorrect

A

Incorrect

C

Incorrect

## 2. Web Services in Jakarta EE

### 1. REST and SOAP

### 2. XML/SOAP-based vs. RESTful web services

### 3. RESTful and XML/SOAP-based web service use cases

### 4. History of web services - From Java EE to Jakarta EE

### 5. What's new in Jakarta EE for web services

### 6. The cloud-native focus in Jakarta web services

### Chapter Quiz

Question 1 of 6

Though RESTful web services are more popularly used nowadays, which of these is a situation where an XML/SOAP based web service would be a better choice?

to perform chained tasks like banking transactions

Correct

SOAP APIs are stateful which means the server stores information about the client and uses that information over a series of requests or operations. This makes it ideal to be used wherever transactions need to be stateful.

to work with limited network resources

Incorrect

RESTful web services are better when working with limited bandwidth and resources as they can save resources and bandwidth by being stateless and able to cache data.

to be called from Mobile applications

Incorrect

RESTful web services are more suitable to be called fromMobile applications as REST is lightweight, efficient, stateless and therefore cacheable.

to build public APIs

Incorrect

RESTful web services are more suitable for public APIs as they do not need much emphasis on security.

Question 2 of 6

Which of these features were introduced with the Jakarta RESTful web services specification along with Jakarta EE 10 release?

A. Folding the Metadata specification in to the main web services specification

B. The Java SE Bootstrap API

C. Support for multipart media type

A

Incorrect

A,B,C

Incorrect

B,C

Correct

The support for multipart media type and the new Java SE Bootstrap API are the two major new features introduced with Jakarta EE 10 RESTful Web Services specification. Folding the Metadata specification in to the main web services specification was introduced in Jakarta XML/SOAP based web services in Jakarta EE 10.

A,B

Incorrect

Question 3 of 6

With which release of the Java Enterprise Edition Jakarta XML Web Services and Jakarta RESTfulWeb Services were first released?

Jakarta EE 9.1

Incorrect

Jakarta EE 9

Incorrect

Jakarta EE 8

Correct

Jakarta EE 8 was the very first release of the Java Enterprise Edition. Therefore, all specifications were released with the name "Jakarta".

Jakarta EE 10

Incorrect

Question 4 of 6

The \_\_\_\_\_ of Jakarta EE 10 consists of the RESTful web services specification and a few other specifications that facilitates the development of lightweight Microservice based applications suited for the cloud.

Platform Profile

Incorrect

Full Profile

Incorrect

Web Profile

Incorrect

Core Profile

Correct

The Jakarta EE Core Profile only contains a smaller set of specifications, specifically targeted at smaller runtimes which is ideal for Microservices.6

Question 5 of 6

A request made to a RESTful web service is more lightweight than a request to an XML/SOAP based web service. Why is that?

because REST can work with many languages

Incorrect

because the request contains what's required to be performed in the URI itself

Correct

A request to a RESTful web service is more lightweight because the service consumer specifies what is required in the Request URI itself.

because REST works only with XML

Incorrect

REST can work with languages other than XML. However, that isn't a reason for it to be lightweight. Rather, it makes REST more flexible to work with.

because REST works only with the HTTP protocol

Incorrect

REST working only with HTTP is not a reason for a RESTful web service request to be lightweight.

Question 6 of 6

Which of these statements are true about SOAP and REST?

A. Both can be used to perform CRUD operations.

B. Both are protocols.

C. SOAP is a protocol while REST in an architectural style.

D. REST is a protocol while SOAP is an architectural style.

A,B

Incorrect

Both can perform CRUD operations in web services but both are not protocols.

A

Incorrect

C is also true as SOAP is a protocol and REST is an architectural style.

A,C

Correct

Both REST and SOAP are used in web services to perform CRUD operations. Only SOAP is a protocol but REST is not. It is an architectural style.

A,D

Incorrect

## 3. Set the Stage to Build a REST API

### 1. CRUD and HTTP methods

### 2. Create and configure a basic web application

### 3. Configure Jakarta RESTful Web Services

### 4. Add a Resource class

### Chapter Quiz

Question 1 of 4

Which code snippet shows the correct syntax for configuring a path for a resource class?

@Path(path="/booking")

public class BookingResource {

Incorrect

@Path(url="/booking")

public class BookingResource {

Incorrect

@Path("/booking")

public class BookingResource {

Correct

The @Path path annotation has a single element named 'value' which takes a String type value. This value serves as the URI path that the resource class or its methods will serve requests for. Even if the value element is not specified explicitly, the String value passed, is implicitly assigned to it.

@Path(uri="/booking")

public class BookingResource {

Incorrect

Question 2 of 4

The HTTP \_\_\_\_\_ method is not idempotent .

DELETE

Incorrect

POST

Correct

If you make multiple identical GET, PUT or DELETE requests, it produces the same result without causing unintended side effects on the server, which means these methods are idempotent. If you make multiple identical POST requests, it can produce different results or create multiple instances of the same submitted data, which means POST is not idempotent.

PUT

Incorrect

GET

Incorrect

Question 3 of 4

A runtime environment or a Jakarta EE Compatible Implementation should be available for building Jakarta RESTful Web Services. Which is the runtime environment/compatible implementation that is used for the Tour Booking App?

RESTEasy

Incorrect

RESTEasy is a framework for developing RESTful Web Services that implements the Jakarta RESTful Web Services specification and it is shipped with application servers like JBoss WildFly. It cannot be thought as a compatible implementation.

Payara Server

Correct

The compatible implementation that's used here is the Payara Server. That is the runtime environment for the application.

Eclipse GlassFish Server

Incorrect

Eclipse Jersey

Incorrect

Eclipse Jersey is an open source framework for developing RESTful Web Services that implements the Jakarta RESTful Web Services specification and it is shipped with application servers like Payara and GlassFish. It cannot be thought as a compatible implementation.

Question 4 of 4

In the Travel App, what was configured using the below code?

@ApplicationPath("/api")

public class TourBookingAppConfiguration extends Application {

}

resource path

Incorrect

root of the resource URL

Correct

This class is the Jakarta REST configuration class. It extends the jakarta.ws.rs.core.Application class and is annotated with the jakarta.ws.rs.ApplicationPath annotation. The string “/api”, passed into the ApplicationPath annotation is the root of the resource URL. This path is the root path relative to which all REST resources created by the application will be accessed.

root resource package

Incorrect

root source package

Incorrect

## 4. Build the REST API for the Read Action

### 1. Implement resource method for reading - HTTP GET

### 2. Use path parameters

### 3. Use query parameters

### 4. Explore response value types - Media types

### 5. Media types in action

### 6. ResponseBuilder and methods

### 7. Control the response via ResponseBuilder

### 8. Challenge - Travel app

### 9. Solution - Travel app

### Chapter Quiz

Question 1 of 7

A resource method that reads data from a data source should be annotated with \_\_\_\_\_.

@PUT

Incorrect

@POST

Incorrect

@GET

Correct

The @GET annotation indicates that a resource method performs a HTTP GET request to READ data from a data source.

@READ

Incorrect

Question 2 of 7

How should the endpoint for the below resource method that accepts a query parameter be accessed?

@GET

@Path("test")

getSomeMethod(@QueryParam("fname") String firstName) {}

/test&fname=Sam

Incorrect

/test?firstName=Sam

Incorrect

/test?fname=Sam

Correct

The name of the query parameter i.e. which is specified using the @QueryParam annotation should be appended to the path, prepended with a question mark(?). Then the value for the query parameter should follow.

/test/fname=Sam

Incorrect

Question 3 of 7

How should path parameters be configured using the @Path annotation for the following resource method, assuming that the endpoint path is app?

getSomeMethod(@PathParam("first") String param1) {}

@Path("app/{param1}")

Incorrect

@Path("app/first")

Incorrect

@Path("app/{first}")

Correct

The name of the path parameter i.e. which is specified using the @PathParam annotation should be given within curly braces in the @Path annotation.

@Path("app/param1")

Incorrect

Question 4 of 7

When this resource method of the Travel App is called in an endpoint, the Content-Type here yields a value of application/json.

@GET

@Path("flights")

public List<Flight> getAllFlights() {

return travelService.findAll();

}

How is it possible without the presence of the @Produces annotation?

The default content type of all resource methods are application/json.

Incorrect

The Java return type List is mapped by the MessageBodyWriter to the response entity in JSON format.

Correct

In the absence of the @Produces annotation in a resource method, Jakarta REST looks at the return Java type of the resource method and maps it using the MessageBodyWriter to the response entity. In this case, the return Java Type is a list of Flight objects and therefore it's mapped to JSON format. This is why you see the Content-Type here as application/json.

The @Produces annotation is implicitly available for the resource method with a media type of APPLICATION\_JSON.

Incorrect

The Java return type of List is read and mapped by the MessageBodyReader in to JSON.

Incorrect

Question 5 of 7

A resource method declares media types that are acceptable for it to write to the response using the \_\_\_\_\_ annotation.

@Consumes

Incorrect

@ContentType

Incorrect

@MediaType

Incorrect

@Produces

Correct

Using the @Produces annotation it is possible to explicitly declare the media types that are acceptable for a resource method to receive and return, or in other words write to the response. Using this annotation, it's possible to declare & restrict the supported response media type for a resource method.

Question 6 of 7

In order to remove the @Produces annotation from this resource method of the Travel App, which method of ResponseBuilder should be cllaed within the method body?

@GET

@Path("flights/{to}")

@Produces(MediaType.APPLICATION\_JSON)

public List<Flight> getFlightsByDestination(@PathParam("to") String destination) {

return travelService.findFlightsByDestination(destination);

}

Response.ResponseBuilder entity(Object entity)

Incorrect

Response.ResponseBuilder type(MediaType type) method

Correct

The type() method of the ResponseBuilder class that takes in a media type can be called within the method body to set the content type of the response without annotating the resource method with the @Produces annotation.

Response.ResponseBuilder header(String name, Object value)

Incorrect

Response.ResponseBuilder allow(Set<String> methods)

Incorrect

Question 7 of 7

Which of these statements are true about the ResponseBuilder?

A. ResponseBuilder is always required to build the response in a resource method.

B. ResponseBuilder is useful to customize and control building of the response.

C. ResponseBuilder instance cannot be directly created.

B

Incorrect

A,C

Incorrect

A,B,C

Incorrect

B,C

Correct

he ResponseBuilder is not required always but it's helpful when it comes to customizing and controlling how the Response object needs to be built. The Jakarta RESTful Web Services API restricts the direct creation of the ResponseBuilder and only allows the Response class to do so because a ResponseBuilder is only meaningful within the context of a Response.

## 5. Build the REST API for Create, Update, and Delete Actions

### 1. Use HTTP POST for the CREATE action

### 2. Use HTTP PUT for the UPDATE action

### 3. Use HTTP DELETE for the DELETE action

### 4. Challenge - Travel app

### 5. Solution - Travel app

### Chapter Quiz

Question 1 of 3

In the Travel App, there's a resource method that's used to delete an existing Flight. The id of the Flight to be deleted can be passed to it using a path parameter or a query parameter.

What's the correct way to use the path parameter and the query parameter in the endpoint URI respectively, assuming an id value of 50?

..../flight/delete?id=50

..../flight/delete/50

Incorrect

..../flight/delete/50

..../flight/delete/id=50

Incorrect

..../flight/delete/id=50

..../flight/delete?id=50

Incorrect

..../flight/delete/50

..../flight/delete?id=50

Correct

@DELETE @Path("flight/delete/{id}") public Response deleteFlight(@PathParam("id") long id) The above corresponds to an endpoint URI of: ..../flight/delete/50 @DELETE @Path("flight/delete") public Response deleteFlight(@QueryParam("id") long id) The above corresponds to an endpoint URI of: ..../flight/delete?id=50

Question 2 of 3

In this resource method from the Travel App, what should be the correct annotation that should be placed in the blank to indicate that this method corresponds to a HTTP PUT request?

\_\_\_\_\_\_\_\_\_\_\_

@Path("flight/update")

@Consumes(MediaType.APPLICATION\_JSON)

public Response updateFlight(Flight flight) {

travelService.updateFlight(flight);

return Response.ok().build();

}

@Put

Incorrect

@POST

Incorrect

@UPDATE

Incorrect

@PUT

Correct

In order to indicate that a resource method corresponds to a HTTP PUT request that performs an UPDATE operation, the method must be annotated with a @PUT annotation.

Question 3 of 3

In a resource method that performs a create action, the \_\_\_\_\_ annotation is used to declare the type of content that's passed in the request body.

@Produces

Incorrect

@Path

Incorrect

@POST

Incorrect

@Consumes

Correct

When calling an endpoint corresponding to a resource method that performs a CREATE action, the data/the entity to be created is passed in the request body. It should be specified in the resource method using the @Consumes annotation.

## 6. Fine-Tune Your Jakarta RESTful Web Services

### 1. Validate your REST endpoints

### 2. Perform single field validations in REST

### 3. Perform custom validations in REST

### 4. Handle exceptions in REST endpoints

### 5. Implement security in REST endpoints

### Chapter Quiz

Question 1 of 5

Given this resource method that takes in a mandatory query parameter, what's the correct syntax to validate if the query parameter isn't empty?

@GET

@Path("airline")

@Produces(MediaType.APPLICATION\_JSON)

public List<Flight> getFlightsByAirline(@QueryParam("al") String airline) {

return travelService.findFlightsByAirline(airline);

}

@GET

@Path(""airline"")

@Produces(MediaType.APPLICATION\_JSON)

public List<Flight> getFlightsByAirline(@Valid @QueryParam(""al"") @NotEmpty String airline) {

return travelService.findFlightsByAirline(airline);

}

Incorrect

@GET

@Path("airline")

@Produces(MediaType.APPLICATION\_JSON)

public List<Flight> getFlightsByAirline(@QueryParam("al") @NotEmpty String airline) {

return travelService.findFlightsByAirline(airline);

}

Correct

The correct syntax to use when validating method parameters using built-in validations is to place the constraint annotation after the @QueryParam annotation, before the parameter variable.

@GET

@Path(""airline"")

@Produces(MediaType.APPLICATION\_JSON)

public List<Flight> getFlightsByAirline(@NotEmpty @QueryParam(""al"") String airline) {

return travelService.findFlightsByAirline(airline);

}

Incorrect

@GET

@Path(""airline"")

@Produces(MediaType.APPLICATION\_JSON)

public List<Flight> getFlightsByAirline(@QueryParam(""al"") String airline @NotEmpty) {

return travelService.findFlightsByAirline(airline);

}

Incorrect

Question 2 of 5

Which of the below statements are true about validating REST endpoints?

A. Jakarta Bean Validation which is the default validation mechanism in Jakarta EE platform is used for validating REST endpoints.

B. Jakarta Bean Validation only provides built-in validation rules.

C. Custom constraint annotations can be used to perform custom validations.

D. Constraints can be placed on only fields of a class.

A,C,D

A,C

Correct

The Jakarta Bean Validation specification is the default validation mechanism in the Jakarta EE platform. It consists of built-in, out of the box constraints or validation rules that can be used conveniently on single fields of a class, method parameters, methods as well as constructors. It also allows you to create complex custom validation rules for more specific validation requirements. Built-in as well as custom validations can be used for validating REST endpoints by applying them on class fields, method parameters and constructors.

A,B,C

A,B

Question 3 of 5

Which of these statements are true about REST Exception Handling? A. Jakarta RESTful Web Services Container does Exception handling.

B. All runtime exceptions are mapped to the HTTP Response.

C. An Exception Mapper maps an exception to an HTTP error response.

D. In all scenarios, the Web container handles an exception with the help of the jakarta.ws.rs.WebApplicationException class

A,B,C,D

Incorrect

only B,C

Incorrect

only A,B,C

Correct

All are true except for D because the Web container will handle the exception with the help of the jakarta.ws.rs.WebApplicationException class only if no Exception Mapper is found.

only A,C

Incorrect

Question 4 of 5

Given a custom validation annotation named ValidDestination, written to validate the String field destination of the Flight entity, what should be the correct values given in the blanks of the ConstraintValidator implementation class given here?

public class ValidDestinationCheck implements ConstraintValidator<\_\_\_\_\_\_(1)\_\_\_\_\_\_\_\_, \_\_\_\_\_\_(2)\_\_\_\_\_\_\_>{

}

1. String

2. ValidDestination

Incorrect

1. ValidDestination

2. Flight

Incorrect

1. ValidDestination

2. Object

Incorrect

1. ValidDestination

2. String

Correct

To the ConstraintValidator interface implementation, the name of the annotation class (which is ValidDestination in this case) and the type to which this constraint is applied (which in this case is String, because the constraint is applied to the destination String field of the Flight entity class) should be passed.

Question 5 of 5

What's implied by the following security constraint?

<security-constraint>

<web-resource-collection>

<web-resource-name>Flight endpoints</web-resource-name>

<url-pattern>/api/travel/flight/\*</url-pattern>

</web-resource-collection>

<auth-constraint>

<role-name>AIRLINE</role-name>

</auth-constraint>

</security-constraint>

All users with a role other than the role AIRLINE are not allowed to access all URLs with the given

pattern.

Incorrect

All users with a role of AIRLINE are not allowed to access all URLs with the given pattern.

Incorrect

REST endpoints with the url pattern /api/travel/flight/\* is accessible to all users with an AIRLINE role.

Correct

The security constraint restricts access to all endpoints with a url pattern of /api/travel/flight/\* to users with a role of AIRLINE. This means that endpoints with this url pattern can only be accessed by users with a role name of AIRLINE.

REST endpoints with the url pattern /api/travel/flight/\* is accessible to all users with a role other than AIRLINE.

Incorrect

## 7. RESTful Applications in the Java SE Environment

### 1. The Java SE bootstrap API

### 2. Create and configure the Java SE app

### 3. Build the RESTful application

### 4. Run the application and test the endpoint

### Chapter Quiz

Question 1 of 4

The SeBootstrap.start() method returns a CompletionStage. What's the exact type of the returned CompletionStage?

SeBootstrap.Configuration

Incorrect

SeBootstrap.Instance

Correct

The start() method returns a CompletionStage, which is of type SeBootstrap.Instance. It is a handle for the application instance that’s started. This instance can be used to specify how the application should be shut down.

AppConfiguration

Incorrect

SeBootstrap

Incorrect

Question 2 of 4

Which of these statements are true about configuring a RESTful application based on the Java SE Bootstrap API? A. the Jakarta RESTful Web Services API dependency should be put in the pom.xml

B. an HTTP embedded server such as Jersey or Resteasy should be configured to run the application

C. having a beans.xml file in src/main/resources/META-INF is optional

only A,B

Correct

The Jakarta RESTful Web Services API dependency should be put in the pom.xml so that the API classes are available for the application.An HTTP embedded server such as Jersey or Resteasy should be configured to run the application. Having a CDI beans.xml file is optional in a Jakarta EE environment. But it is a must when using SeBootstrap API in a Java SE environment. It is used to specify how the application should discover CDI beans.

only B,C

Incorrect

only A, C

Incorrect

A,B,C

Incorrect

Question 3 of 4

A RESTful application based on the Java SE Bootstrap API needs \_\_\_\_\_to deploy itself.

a Jakarta EE Server

Incorrect

an embedded HTTP server

Correct

When a web application needs to be published in a Java SE environment, it is a common practice to use an embedded HTTP server.

a Java Runtime Environment

Incorrect

a Java Virtual Machine

Incorrect

Question 4 of 4

What's the mandatory parameter that should be passed to the start() method of the jakarta.ws.rs.SeBootstrap class?

the REST Application entry class

Correct

The start() method of the SeBootstrap class accepts a REST Application entry class and an optional SeBootstrap.Configuration type instance that can consist of host, port and protocol details for the embedded web server

SeBootstrap.Configuration type instance

Incorrect

SeBootstrap.Instance

Incorrect

a CompletionStage object

Incorrect

## 8. Conclusion

### 1. Next steps