Root Resource class stands for? (select two) Select one or more: a. The service class, which loads the actual service to the container b. The service class, which can receive requests c. The service class, which can give a response d. The service class, which binds the actual service to the container

ChatGPT

Based on my knowledge cutoff of 2021, none of the options accurately describe the Root Resource class in the context of web services or RESTful APIs. The Root Resource class is a concept in JAX-RS (Java API for RESTful Web Services) that represents the entry point for handling incoming HTTP requests to a web service. It is not necessarily a "service class" in the traditional sense, but rather a Java class that defines the HTTP methods (e.g. GET, POST, PUT, DELETE) and corresponding URI paths that the web service can handle.

Therefore, the correct answers would be:

b. The Root Resource class can receive requests c. The Root Resource class can give a response

When creating Restful web services with Java in the web.xml file there is an element url-pattern having some content (e.g. /rest/\*). This has a meaning? (select one) Select one: a. This is the first part of the service's URI after 'server.com:PORT/' in the URL b. The URL of the service must end with that content (/rest) in any case c. The service can be found from the web by just using this as an URL

ChatGPT

The element **url-pattern** in the **web.xml** file is used to map a web resource to a URL pattern. It specifies the URL pattern(s) that will be matched against incoming requests to determine which servlet or JSP (JavaServer Pages) should handle the request.

The value of the **url-pattern** element specifies the pattern to match, and it can include one or more wildcards, which are represented by the **\*** character. The wildcard character **\*** matches any sequence of characters, including the empty sequence.

Therefore, the correct answer would be:

a. The **url-pattern** element specifies the first part of the service's URI after **server.com:PORT/** in the URL, and any path segments that match the specified pattern will be routed to the corresponding servlet or JSP. For example, if the **url-pattern** is set to **/rest/\***, any request with a path that starts with **/rest/** will be handled by the servlet or JSP mapped to that URL pattern.

When having an external exception handler, which implements ExceptionMapper, the ExceptionMapper must always be parametrisized with Exception? (select one) Select one: a. Yes b. No

ChatGPT

The **ExceptionMapper** interface in JAX-RS (Java API for RESTful Web Services) is a generic interface that allows you to map an exception to a response. The generic type parameter of **ExceptionMapper** specifies the type of exception that the implementation will handle.

Therefore, the correct answer would be:

a. Yes, the **ExceptionMapper** interface must always be parameterized with the specific type of exception that it is

If in the service method there is not @Produces annotation, the method can send? (select one) Select one: a. any type of data b. plain text c. method must have @Produces annotation

ChatGPT

In JAX-RS (Java API for RESTful Web Services), the **@Produces** annotation is used to specify the MIME media types of the response that a resource method will produce. If a resource method does not have a **@Produces** annotation, it means that the media type of the response is not explicitly specified.

Therefore, the correct answer would be:

a. If a resource method does not have a **@Produces** annotation, it can still produce a response, but the type of data that it can produce depends on the default media type configured for the application. If no default media type is specified, the method may be able to produce any type of data, but the client may not know how to handle it. If a default media type is specified, the method can produce data that matches the default media type.

When creating Restful web services with Java, in the web.xml file there must be elements? (select five) Select one or more: a. servlet-mapping b. persistence-unit c. url-pattern d. servlet e. init-param f. servlet-name twice

ChatGPT

When creating a RESTful web service with Java, the **web.xml** file is used to configure the servlet container and map URLs to servlets. The required elements in the **web.xml** file depend on the specific application and its dependencies, but the most common elements that are typically included are:

a. **servlet** - declares a servlet class and maps it to a URL c. **url-pattern** - specifies the URL pattern to which a servlet should respond d. **servlet-mapping** - maps a servlet to a URL pattern e. **init-param** - provides initialization parameters for a servlet or filter

Therefore, the correct answers would be:

a. **servlet-mapping** c. **url-pattern** d. **servlet** e. **init-param**

Select one or more: a. The class must have an annotation @Provider b. The class must have annotation @Exception c. The class must implement the method init d. The class must implement ExceptionMapper interface parametrisized with a proper type e. The class must implement the method toResponse

ChatGPT

In JAX-RS (Java API for RESTful Web Services), an external exception handler is implemented as a class that implements the **ExceptionMapper** interface. This interface defines a method that maps an exception to a response, and the class that implements it must meet certain requirements.

Therefore, the correct answers would be:

a. The class must have an annotation **@Provider**. This annotation indicates that the class is a provider of services, such as message body readers and writers, context resolvers, and exception mappers.

d. The class must implement the **ExceptionMapper** interface parameterized with the specific type of exception that it is intended to handle. For example, if the class is intended to handle **NotFoundException** exceptions, it should implement **ExceptionMapper<NotFoundException>**.

e. The class must implement the **toResponse** method defined in the **ExceptionMapper** interface. This method maps an exception to a response that will be sent back to the client.

When having an html form with two inputs of type text and method parameter post, which can be the correct signature of a method in a Restful web service? (select one) Select one: a. public String saveLaptop(@QueryParam("brand") String brand, @QueryParam("price") float price) b. public String saveLaptop(@PathParam("brand") String brand, @PathParam("price") float price) c. public String saveLaptop(@FormParam("brand") String brand, @FormParam("price") float price)

ChatGPT

In JAX-RS (Java API for RESTful Web Services), when using an HTML form to submit data to a RESTful web service, you can use the **@FormParam** annotation to bind each form parameter to a method parameter.

Therefore, the correct answer would be:

c. The correct signature of the method would be:

javaCopy code

public String saveLaptop(@FormParam("brand") String brand, @FormParam("price") float price) { // method body }

In this case, the **@FormParam** annotation is used to bind the **brand** and **price** form parameters to the **brand** and **price** method parameters, respectively. The **@FormParam** annotation can only be used with HTTP POST requests that have a **Content-Type** of **application/x-www-form-urlencoded**.

base on the following question select one option : You can start creating a new project by selecting Coloured heaxon or File->New? (select one) Select one: a.Yes b.No

ChatGPT

I'm sorry, but I cannot determine a clear option based on the given question. The question appears to be incomplete or unclear. Please provide more context or clarify the question so I can assist you better.

When having a service method with annotation @Path("/getlaptop/{price}/{brand}"), which is the correct signature to the method? (select two) Select one or more: a. public String getLaptop(@PathParam("price") String price, @PathParam("brand") String brand) b. public String getLaptop(@QueryParam("price") float price, @QueryParam("brand") String brand) c. public String getLaptop(@PathParam("price") float price, @PathParam("brand") String brand)

ChatGPT

The correct signatures for the given service method with the **@Path** annotation are options A and C:

a. **public String getLaptop(@PathParam("price") String price, @PathParam("brand") String brand)**

c. **public String getLaptop(@PathParam("price") float price, @PathParam("brand") String brand)**

Option A uses **@PathParam** to inject the values of the **price** and **brand** path parameters into the method parameters as **String** values.

Option C also uses **@PathParam**, but it injects the values as **float** for **price** and **String** for **brand**.

Option B is not correct because it uses **@QueryParam** instead of **@PathParam** to inject the values, and it specifies **float** as the data type for **price** while the service method signature in the **@Path** annotation uses **String**.

When creating a new project as maven project, you have to set? (select two) Select one or more: a. Java package b. Group ID c. Artifact ID

ChatGPT

When creating a new Maven project, you have to set Group ID and Artifact ID. So, the correct options are B and C:

b. Group ID: It is a unique identifier for your project, typically in reverse domain name notation, that helps to identify your project uniquely among all projects.

c. Artifact ID: It is the name of the jar file that will be generated by Maven when the project is built. It should be unique within the Group ID.

Option A is not mandatory to set during the creation of a new Maven project, but it can be added later.

Root Resource class stands for? (select two) Select one or more: a. The service class, which loads the actual service to the container b. The service class, which can receive requests c. The service class, which can give a response d. The service class, which binds the actual service to the container

ChatGPT

The Root Resource class in Java refers to the service class that can receive requests and give a response. So, the correct options are B and C:

b. The Root Resource class is the service class that can receive requests.

c. The Root Resource class is also the service class that can give a response.

Option A and D are not correct:

a. The Root Resource class does not necessarily load the actual service to the container. It represents the service itself.

d. The Root Resource class does not necessarily bind the actual service to the container. That is typically done through other mechanisms such as annotations or configuration files.

When creating Restful web services with Java in the web.xml file there is an element url-pattern having some content (e.g. /rest/\*). This has a meaning? (select one) Select one: a. This is the first part of the service's URI after 'server.com:PORT/' in the URL b. The URL of the service must end with that content (/rest) in any case c. The service can be found from the web by just using this as an URL

ChatGPT

The **url-pattern** element in the **web.xml** file for a Java web application with RESTful web services is used to specify the pattern that maps to the root resource class.

So, the correct option is:

a. This is the first part of the service's URI after 'server.com:PORT/' in the URL.

The **url-pattern** element does not have to end with a specific content, and it is not an URL that can be used to find the service directly. It is a URL pattern used by the servlet container to map incoming HTTP requests to the appropriate servlet or resource.

When having an external exception handler, which implements ExceptionMapper, the ExceptionMapper must always be parametrisized with Exception? (select one) Select one: a. Yes b. No

ChatGPT

No, an external exception handler in Java that implements **ExceptionMapper** does not always have to be parameterized with **Exception**.

The **ExceptionMapper** interface in Java is a generic interface that can be parameterized with any type of exception that needs to be handled. This means that the implementation of the **ExceptionMapper** can be tailored to handle a specific type of exception or a hierarchy of exceptions.

For example, if you have a custom exception **MyAppException**, you can create an implementation of **ExceptionMapper<MyAppException>** to handle that specific exception type.

So, the correct option is:

b. No, the **ExceptionMapper** interface is generic and can be parameterized with any type of exception that needs to be handled.

If in the service method there is not @Produces annotation, the method can send? (select one) Select one: a. any type of data b. plain text c. method must have @Produces annotation

ChatGPT

If there is no **@Produces** annotation in the service method of a RESTful web service in Java, the service can still send a response. However, the type of data that can be sent is limited to the default media types supported by the framework or application server.

So, the correct option is:

b. The service can still send plain text.

If there is no **@Produces** annotation, the default media type is assumed to be **text/plain**. This means that the service can send plain text as a response.

However, if the service needs to send other types of data such as JSON, XML, or HTML, it is recommended to use the `

When creating Restful web services with Java, in the web.xml file there must be elements? (select five) Select one or more: a. servlet-mapping b. persistence-unit c. url-pattern d. servlet e. init-param f. servlet-name twice

ChatGPT

When creating RESTful web services with Java, in the **web.xml** file there must be the following elements:

a. **servlet-mapping**: This element maps the URL pattern to the servlet that will handle the requests.

c. **url-pattern**: This element specifies the URL pattern that maps to the root resource class.

d. **servlet**: This element defines the servlet that will handle the requests.

e. **init-param**: This element can be used to specify initialization parameters for the servlet or filter.

Select one or more: a. The class must have an annotation @Provider b. The class must have annotation @Exception c. The class must implement the method init d. The class must implement ExceptionMapper interface parametrisized with a proper type e. The class must implement the method toResponse

ChatGPT

When creating an external exception handler in Java using **ExceptionMapper**, the class must have the following elements:

a. The class must have an annotation **@Provider**.

d. The class must implement the **ExceptionMapper** interface, parameterized with the type of exception that it handles.

e. The class must implement the **toResponse** method, which converts the exception to a response that will be sent back to the client.

So, the correct options are:

a. The class must have an annotation **@Provider**.

d. The class must implement **ExceptionMapper** interface parameterized with a proper type.

e. The class must implement the method **toResponse**.

When having an html form with two inputs of type text and method parameter post, which can be the correct signature of a method in a Restful web service? (select one) Select one: a. public String saveLaptop(@QueryParam("brand") String brand, @QueryParam("price") float price) b. public String saveLaptop(@PathParam("brand") String brand, @PathParam("price") float price) c. public String saveLaptop(@FormParam("brand") String brand, @FormParam("price") float price)

ChatGPT

When having an HTML form with two inputs of type text and method parameter post, the correct signature of a method in a RESTful web service depends on the way the form data is being submitted.

If the form data is submitted as **application/x-www-form-urlencoded**, then the correct signature would be:

c. **public String saveLaptop(@FormParam("brand") String brand, @FormParam("price") float price)**

The **@FormParam** annotation is used to inject the values of the form fields into the method parameters.

If the form data is submitted as JSON or XML, then the correct signature would depend on the structure of the data being submitted. In that case, you would typically use the **@Consumes** annotation to specify the media type of the data being consumed by the method.

If in the service method there is not @Produces annotation, the method can send? (select one) Select one: a. any type of data b. plain text c. method must have @Produces annotation

ChatGPT

If there is no **@Produces** annotation in the service method, the method can still send data, but the type of data that can be sent depends on the implementation of the JAX-RS runtime.

Typically, if the JAX-RS runtime cannot determine a specific media type to produce based on the **@Produces** annotation, it will default to producing **text/plain** data. So, in this case, the method can send plain text data.

Therefore, the correct answer is: b. plain text.

When having an external exception handler, which implements ExceptionMapper, the ExceptionMapper must always be parametrisized with Exception? (select one) Select one: a. Yes b. No

ChatGPT

a. Yes, the ExceptionMapper must always be parameterized with **Exception** since it needs to handle any type of exception that may be thrown within the application. By using **Exception** as the parameter type, the ExceptionMapper can handle any subtype of Exception, allowing for flexibility in handling different types of exceptions.