**Building Cloud Services with the Java Spring Framework**

1. **How is metadata about an HTTP request transmitted?**

URL Encoded Body Parameters

Query Parameters

Path Variables

Headers

JSON Body

Metadata about an HTTP request is typically transmitted in the form of headers. HTTP headers are pieces of additional information that are sent along with an HTTP request or response. They provide important details about the request, such as the content type, the length of the content, the type of encoding used, and any cookies that are being sent.

Headers are transmitted as key-value pairs, with each header providing specific information about the request. Some commonly used headers include:

* Content-Type: specifies the format of the data being sent in the request body (e.g. application/json, text/html)
* Content-Length: specifies the length of the data being sent in the request body
* Accept: specifies the types of data that the client is willing to accept in response
* Authorization: provides authentication information for the request
* User-Agent: identifies the client making the request (e.g. the web browser or mobile app)

These headers (and many others) are included in the HTTP request to provide additional context and information to the server, allowing it to process the request appropriately.

1. **Which of the following are ways of sending data to a server in an HTTP request?**

JSON Encoded Body

URL Encoded Body Parameters

the Path

Query Parameters

There are several ways to send data to a server in an HTTP request, including:

1. Query parameters: Query parameters are appended to the end of the URL with a "?" separator and are used to send small amounts of data, such as filtering or sorting parameters.
2. URL-encoded body parameters: URL-encoded parameters are included in the body of the request and are formatted as key-value pairs separated by "&" characters. They are typically used to send small amounts of data, such as form data.
3. JSON-encoded body: JSON-encoded data is included in the body of the request and is used to send structured data in a compact format. JSON is a lightweight and widely supported format, and is often used for APIs.
4. XML-encoded body: XML-encoded data is included in the body of the request and is used to send structured data in a format that is human-readable and widely supported. XML is often used for APIs and web services.
5. Multipart form-data: Multipart form-data is used to send binary data or large amounts of text data, such as files or images, and is formatted as a series of key-value pairs.

The choice of data format will depend on the specific requirements of the application, as well as the capabilities of the client and server.

1. **Which is not an appropriate use of an HTTP header?**

* To communicate the type of client sending the request
* To specify the path for a request
* To communicate the type of server responding to a request
* To communicate the primary payload of an Http request

HTTP headers are used to transmit additional information between the client and server in an HTTP request or response. The appropriate use of an HTTP header depends on the specific header in question, as well as the context of the request or response. However, there are certain types of information that should not be transmitted via HTTP headers, including:

1. Sensitive information: HTTP headers can be viewed by anyone who has access to the network traffic between the client and server, and should not be used to transmit sensitive information, such as passwords or credit card numbers.
2. Large amounts of data: While some headers can contain data, they are not intended for transmitting large amounts of data, such as images or videos. Large data should be sent in the request or response body.
3. Content that should be cached: Some headers can indicate that content should not be cached, but others, such as the "Cache-Control" header, can be used to control caching behavior. Content that should be cached should not use headers that indicate otherwise.
4. Malicious content: HTTP headers should not be used to transmit malicious content, such as scripts or other code that can be used to exploit vulnerabilities in the client or server.

In general, HTTP headers should be used for transmitting metadata and other non-sensitive information that is relevant to the request or response.

1. **What are mime types used for?**

to define a schema for the allowed request headers

to describe the type of data being sent to the client

to define a schema for the allowed response headers

to describe the type of data being sent to the server

to specify the return type of a function

MIME types (also known as media types or content types) are used to identify the type of data that is being sent in an HTTP request or response. The acronym "MIME" stands for Multipurpose Internet Mail Extensions.

MIME types are used for several purposes, including:

1. Specifying the format of data being sent: For example, a server may specify the MIME type of an image file being sent as "image/jpeg" or "image/png" to indicate the format of the data being sent.
2. Helping clients understand how to handle the data: By specifying the MIME type of the data being sent, the server can help the client understand how to handle the data. For example, a web browser may use the MIME type to determine how to display an image or video on a webpage.
3. Facilitating server-side processing: In some cases, the server may need to process the data being sent and the MIME type can help the server understand how to handle the data. For example, a server may use the MIME type of an uploaded file to determine how to process it.

MIME types are standardized by the Internet Assigned Numbers Authority (IANA) and new types can be registered by submitting a request to IANA.

1. **Which of the following are components of a communication protocol?**

URLs

Semantics

Timing

Syntax

Request Parameters

A communication protocol typically includes the following components:

1. Syntax: The syntax defines the format of the data being transmitted, including the rules for encoding and decoding data. This includes the data structures, message formats, and other rules for encoding and decoding messages.
2. Semantics: The semantics define the meaning of the data being transmitted, including how it should be interpreted and what actions should be taken based on the data. This includes definitions of the different data types, how data is used and interpreted, and other contextual information.
3. Timing: The timing defines the rules for when data can be transmitted and how quickly it should be processed. This includes definitions of timeouts, response times, and other time-related rules.
4. Error handling: The error handling component defines how errors are handled during communication, including how errors are detected, how they are reported, and how they should be handled by the sender and receiver.
5. Security: The security component defines how the communication is secured, including authentication, encryption, and other security measures.

Together, these components help ensure that communication between different systems is reliable, efficient, and secure. Communication protocols are used in a wide range of applications, including networking, web services, and IoT devices.

1. **What are cookies used for?**

to store data needed by the server in the request headers

to store data needed by the server on the client

to store data needed by the client on the server

to store data needed by the server in the load balancer

Cookies are small text files that are stored on a user's device by a web server. They are used to store information about the user's activity on a website and can be accessed by both the server and the client-side scripts running on the website.

Cookies can be used for a variety of purposes, including:

1. Session management: Cookies can be used to manage user sessions on a website, allowing the server to identify the user as they move between different pages or perform actions on the site. This can help the website provide a more personalized experience for the user.
2. Personalization: Cookies can be used to store user preferences and other information about the user, such as their language preference or their favorite products on an e-commerce website. This information can then be used to personalize the user's experience on the website.
3. Tracking: Cookies can be used to track user activity on a website, such as the pages they visit or the items they add to a shopping cart. This information can then be used for analytics or advertising purposes.
4. Authentication: Cookies can be used to authenticate users on a website, allowing them to access restricted content or perform actions that require authorization.
5. Advertising: Cookies can be used to track user activity across different websites, allowing advertisers to serve targeted ads to users based on their browsing history.

Cookies can be both first-party (set by the website the user is visiting) and third-party (set by other websites that the user visits through ads or social media). While cookies can be useful for providing a better user experience, they also raise concerns about user privacy and data security.

1. **What type of protocol is HTTP?**

Request / Response

Peer to Peer

Secure

Onion-routed

HTTP stands for Hypertext Transfer Protocol, and it is an application layer protocol that is used for transmitting data over the World Wide Web. It is the foundation of data communication on the web, and is used by web browsers, servers, and other applications to exchange information.

HTTP is a stateless protocol, which means that each request and response is independent of previous requests and responses. This design makes it simple and efficient for transmitting data, but it also means that it cannot maintain state between transactions, such as remembering user preferences or login credentials.

There are two versions of HTTP currently in use: HTTP/1.1 and HTTP/2. Both versions use the same basic structure for requests and responses, but HTTP/2 is designed to be faster and more efficient, with features such as multiplexing, header compression, and server push.

1. **Which of the following differentiate query parameters from data sent in the body of a request?**

* larger volumes of data are typically sent in the body
* query parameters cannot be included in a URL
* query parameters can't include addressing parameters
* body parameters can be encoded in multiple formats

Query parameters and data sent in the body of a request are two different ways to send data in an HTTP request, and they have some key differences:

1. Location: Query parameters are part of the URL of the request, while data sent in the body of a request is included in the body of the request. Query parameters are typically appended to the end of the URL with a "?" separator, while data in the body is included after the HTTP headers.
2. Size: Query parameters are limited in size and can only accommodate a small amount of data, while data sent in the body can be much larger. This is because the maximum size of a URL is limited by the browser, while the size of the request body is limited by the server.
3. Encoding: Query parameters are typically URL-encoded, while data sent in the body can be encoded in different ways, such as JSON, XML, or form-encoded.
4. Usage: Query parameters are often used for filtering or sorting data, while data sent in the body is typically used for more complex operations, such as creating or updating resources.

In general, query parameters are more appropriate for simple requests that require only a small amount of data, while data in the body is more appropriate for more complex requests that require a larger amount of data or more complex operations.

# **Cloud Services Quiz**

### **What is the purpose of request routing?**

To determine which client should receive the response

To determine which function should generate the response for a given request

To determine which mime type the request should be interpreted as

To determine which header should be used to authenticate the user

Request routing is the process of directing incoming requests to the appropriate server, application, or service based on the request's destination address, protocol, or other characteristics. The purpose of request routing is to optimize the performance, availability, and security of distributed systems by ensuring that requests are handled by the most appropriate resource.

Here are some of the key purposes of request routing:

1. Load balancing: Request routing can be used to distribute incoming requests across multiple servers or instances, allowing the system to handle larger volumes of traffic and preventing any one server from becoming overwhelmed.
2. Traffic management: Request routing can be used to manage traffic flows, prioritizing certain types of traffic or users, throttling traffic to prevent overload, or routing traffic to specific geographic regions or servers.
3. Failover and high availability: Request routing can be used to ensure that requests are automatically routed to a backup server or instance in the event of a failure or outage, minimizing downtime and ensuring that the system remains available to users.
4. Security and access control: Request routing can be used to enforce security policies, such as requiring authentication or authorization before allowing access to certain resources or services.

Overall, the purpose of request routing is to optimize the performance, availability, and security of distributed systems by ensuring that requests are handled by the most appropriate resource. By routing requests intelligently, organizations can improve the efficiency and reliability of their systems, while also providing better experiences for their users.

### **Which of the following are true of client request data?**

The data will be in the correct format or the HTTP server would reject it

The data could lead to an injection attack

The data should be carefully sanitized and validated

The data will be in the correct format due to spring annotations

### **How is HTTP used in cloud services?**

HTTP can be used as the communication protocol for talking to cloud services

HTTP provides the data format used inside of services

HTTP cookies provide the security for cloud applications

HTTP headers provide routing information for cloud commands

### **What is the relationship between a Spring controller and the dispatcher servlet?**

The dispatcher servlet routes HTTP requests to one or more controllers to produce a response

Controllers provide the dispatcher servlet with access to a database

The dispatcher servlet guarantees the security of controllers

Controllers inherit from the dispatcher servlet

### **Which of the following are true?**

The PathVariable annotation is used to indicate that a method parameter should be bound to a component of the path the HTTP request was sent to

The RequestBody annotation is used to indicate that a method parameter should be bound to the body of an HTTP request

The RequestParam annotation is used to indicate that a method parameter should be bound to a specific parameter from the HTTP request

# **Cloud Services Quiz 2**

### Which of the following are true of dependency injection?

It is a form of cyber-attack stemming from improper sanitization of client data

None of the above

It requires users of a dependency injection library to write code to pass dependencies to consumers of the dependencies

It is best used when there is exactly one implementation of each interface

### What is horizontal scaling?

Increasing the number of hosts supporting a cloud application

Increasing the width parameter in an HTTP request

Increasing the CPU power (e.g., a better CPU) of the hosts supporting a cloud application

Adding load balancers to a cloud application

### Which of the following would make horizontal scaling more difficult?

Having state used by all functions on a host that service a given request

Having state that persists across requests from a client

Having cookies that store state data needed by the server

Having a load balancer distributed requests in a "sticky" fashion

### What is the purpose of the ResponseBody annotation?

To indicate that the return value from a method should be transformed into the HTTP response body

To indicate which view associated with a controller should produce the HTTP response body

To indicate which controller method should be invoked to produce the HTTP response body

To indicate which member variable should be used to create the HTTP response body

### **Which of the following are examples of IaaS?**

Providing the ability to provision virtual machines to a cloud service

Providing the ability to provision database tables to a cloud service

Providing the ability to provision messaging queues to a cloud service

Providing the ability to provision network addresses to a cloud service