Programming Challenge

LogMeIn

Task Description

- Implement a Java Web Application that meets the specification of the "Document Storage REST Web Service" below.
- It should contain a single Servlet implemented purely in Java. Do not use application frameworks such as Spring, Struts, JAX-RS, or Jersey. Do not use Java libraries such as Apache Commons. Your code should extend javax.servlet.http.HttpServlet.
- Limit the scope to the specification. The only error cases to be aware of are those outlined in the specification.
- The web application should be packaged as a WAR that runs in a Java EE Web Container such as Tomcat or Jetty.
- Provide both source code and WAR to the interviewer as soon as you are done.
- Documents don't need to be persisted across server shutdown.
- Feel free to use online references and resources. Add citations when lifting blocks of code.

Document Storage REST Web Service

Specification

The Document Storage Service is a simple RESTful web service that allows clients to create, update, query, and delete documents. A document can be anything - text, image, pdf, etc.

A document can be created by sending a POST request with document contents to /storage/documents. The document is simply the HTTP request payload. All content types are supported. The content of the POST response is a unique alphanumeric document ID with a length of 20 characters. The HTTP response has a 201 Created status code.

A document can be queried by sending a GET request to /storage/documents/{docId}, where {docId} is the document ID issued during creation. The content of the GET response is the document exactly as it was created or last updated. On success, a 200 OK response is sent. A 404 Not Found HTTP response is returned if the document ID is invalid.

A document can be updated by sending a PUT request with document contents to /storage/documents/{docId}, where {docId} is the document ID issued during creation. The document is simply the HTTP request payload. On success, a 204 No Content response is sent. A 404 Not Found HTTP response is returned if the document ID is invalid.

A document can be deleted by sending a DELETE request with no content to /storage/documents/{docId}, where {docId} is the document ID issued during creation. On success, a 204 No Content HTTP response is sent. A 404 Not Found HTTP response is returned if the document ID is invalid.

Summary

Create - POST /storage/documents
Query - GET /storage/documents/{docId}
Update - PUT /storage/documents/{docId}
Delete - DELETE /storage/documents/{docId}

Examples

Create

Request:
```
POST /storage/documents
Content-Length: 11

hello world
```

Response:
```
201 Created
Content-Type: text/plain; charset=us-ascii
Content-Length: 20

ONWZ4UUVV8S31JCB662P
```

Query

Request:
```
GET /storage/documents/ONWZ4UUVV8S31JCB662P
```

Response:
```
200 OK
Content-Length: 11

hello world
```

Update

Request:
```
PUT /storage/documents/ONWZ4UUVV8S31JCB662P
Content-Length: 13

goodbye world
```

Response:
```
204 No Content
```

Delete

Request:
```
DELETE /storage/documents/ONWZ4UUVV8S31JCB662P
```

Response:
```
204 No Content
```