TK Integration Test Assignment

Task

Create a protected asynchronous proxy for the TK Extraction service using Java (or any other JVM language). In the following this service will be called the 'Integration service'.

The TK Extraction service

The TK Extraction service can be used to extract information from CVs and enrich them with e.g. synonyms. It takes as input authentication tokens and a binary file, and it returns a XML document containing the extracted and enriched information. The API documentation is attached as appendix below.

Your integration service should expose the following endpoint URL's

● /submit (POST) Accept a binary file for processing and return a processid

● /retrieve/<PROCESSID> (GET) Return processing status or processing result

The integration service is asynchronous. It allows a client to submit a cv processing job, wait for the result, and retrieve the result. Both endpoints should be protected with basic authentication.

The client submits the CV and directly receives a processid by POST-ing to /submit The integration service asynchronously forwards the received payload to the TK Extraction service, and temporarily stores the result, until it is retrieved by the client later.

After the client has submitted the CV it polls the integration service for the processing result (/retrieve/<PROCESSID>). If processing is complete, the integration service returns the processing result received from the TK Extraction service to the client, otherwise it returns status 'PROGRESS'.

Notes

● All data can be stored in memory, you should use no extra tools like queues or databases that require separate installation, but embedded queues or databases are OK.

● You can use any open source libraries you want. Please include the dependencies in the zip file (see below) or specify them in the build file

● Add JUnit tests, where appropriate

● Don’t forget to think about error handling, but do not implement fine grained error handling for the Extraction service errors as described in the service documentation in the appendix

● The source code should contain Javadoc to document functions (when not trivial). Algorithm comments or design decisions can be added as code comments

● If you use any AI tools, explain how you use them and why

Submission

Please submit a zip file that includes
● A short document that explains design choices and basic algorithms used (max 150 words)
● A listing of curl (or comparable) commands, that shows how a client can use the service
● Source code with comments
● A build script (ant, maven, or any other build tool)
● A .jar or .war file that can be run either from command line, or in a servlet container

Appendices

● TK Extraction service documentation (Excerpt)
● Test CV

Appendix: TK Extraction service documentation (Excerpt)

There are two ways of calling the Extract service:

1. As an HTTP POST multipart/form-data request, or

2. As a SOAP 1.2 web interface.

The HTTP POST method is described in the following sections.

HTTP Form POST

The CV to be processed must be passed along with login information and any other data to the Sourcebox Extract service as an HTTP POST multipart request with content type multipart/form-data at the following URL:

The request must contain the following form parameters:

Parameter	Description	Parameter MIME type
uploaded_file	the binary file data of the document to be processed	application/octet- stream
account	the account name of the user	text/plain
username	the username of the user	text/plain
password	the password of the user	text/plain

On success, the request returns the templated result XML. Section Error Handling below explains Sourcebox behaviour in case of error.

An example of how to use the service with curl:

```
curl https://staging.textkernel.nl/sourcebox/extract.do \

--form account=exampleaccount \

--form username=exampleuser \

--form password=examplepassword \

--form uploaded_file=@/home/user/petra.doc
```

Error Handling

By default, in case of error, the Extract service returns an HTML error page with HTTP status 200. Information about the error and error description is included in the title section of the HTML, in the meta tags error-code and error-desc.

Alternatively, it is possible to enable HTTP protocol compliance, by adding useHttpErrorCodes=true to the POST URL. It is also possible to return error information as a JSON string by adding parameter useJsonErrorMsg=true to the POST URL. Note that useJsonErrorMsg=true implies useHttpErrorCodes=true.

So to enable both json error messages and HTTP status compliance, POST to

.../sourcebox/extract.do?useJsonErrorMsg=true

To only enable HTTP status compliance, POST to

.../sourcebox/extract.do?useHttpErrorCodes=true

Sourcebox Error Codes

(For reference, do not implement any specific handling in this task)

General Errors

Error Code	Description
DEFAULT_EXCEPTION	An unexpected exception occurred.
UNSUPPORTED_OPERATION_ERROR	The requested method is not supported by Sourcebox.

Error Code	Description
INVALID_CREDENTIALS_ERROR	The login credentials are invalid.
LIMIT_EXCEEDED	The allotted amount of usable processing units for the account has been exceed.
NO_ACCESS	The resource is not visible to the user with the given credentials.
SESSION_INVALID	The session related to the users cookie has already been invalidated.
TRXML_LOCKED	The document cannot be accessed because it is currently edited by another user/process.