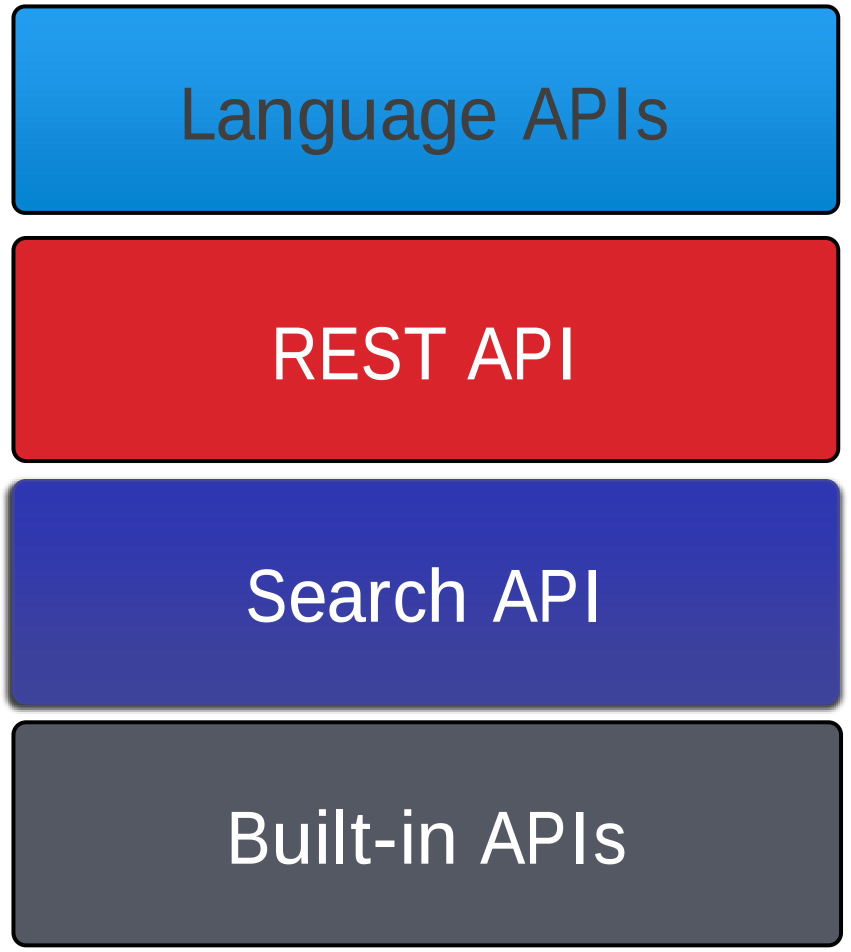
Unit 3: Using the REST API

MarkLogic

Learning Objectives

- Describe the MarkLogic API stack and the REST API.
- Setup a MarkLogic REST API instance.
- Perform CRUD operations using the REST API and cURL.
- Perform a command line query using the REST API and cURL.

MarkLogic API Stack



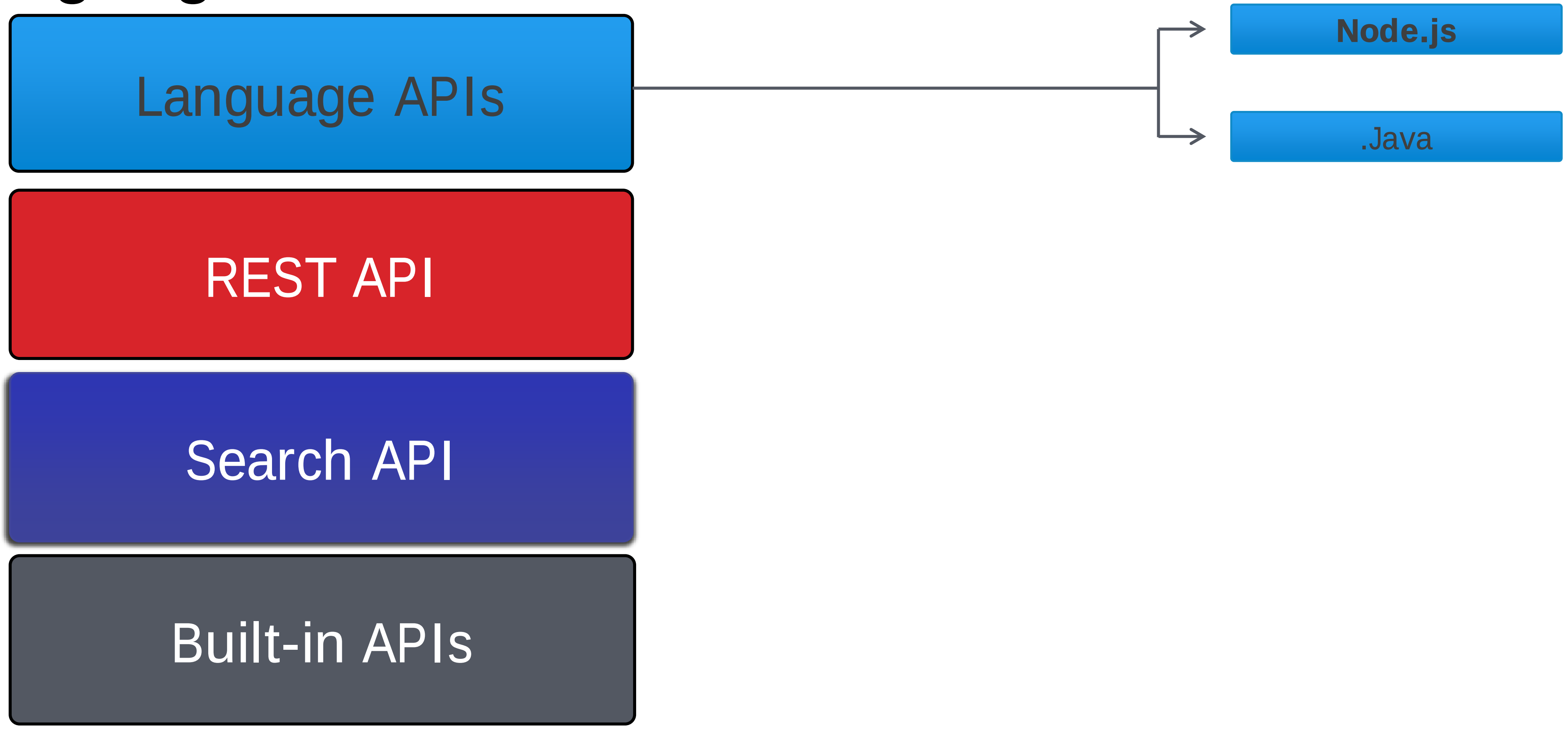
- Language specific libraries
- Example: Node.js API

- High level, task specific interface
- Abstracts complexity of lower APIs
- Language independent

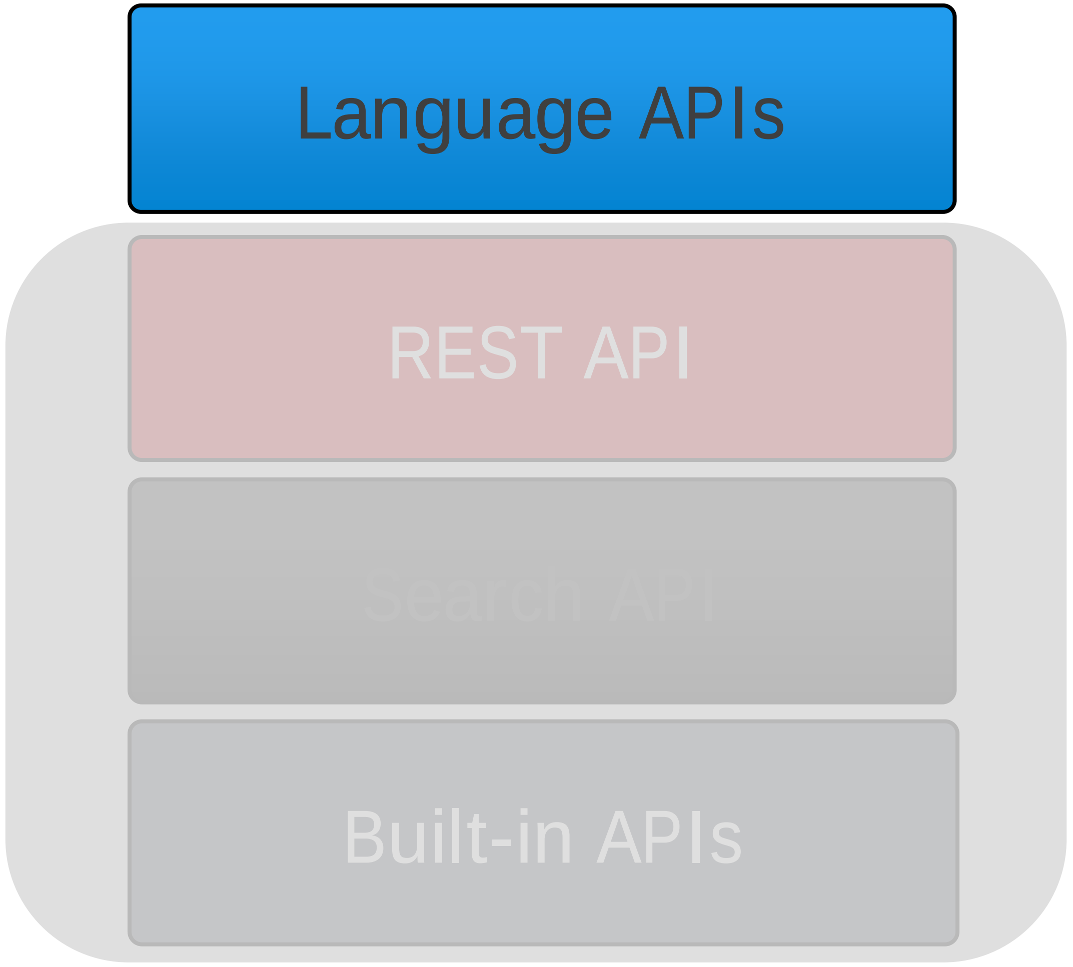
- Higher level functions for building robust search applications

- C++ functions exposed as XQuery
- Example: cts:search( )

Language APIs



The Node.js Developer



- Good news!!
- The Node.js API enables you to build apps without being an expert on the underlying layers.
- …but because the Node.js API uses the REST API under the hood, we are going to take a few minutes to understand how to setup and use the REST API.

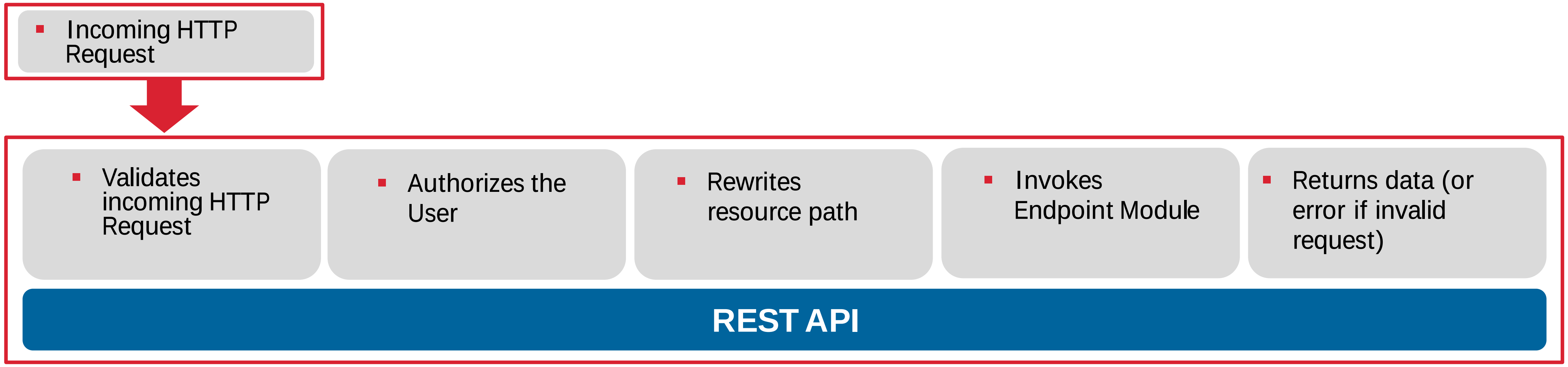
REST Overview

REST: What, Why and When

- REST
  - **RE**presentational **S**tate **T**ransfer
- Client REST API
  - A set of services for CRUD and query of documents and metadata in a MarkLogic database.
- Why?
  - Take advantage of MarkLogic as your database with a programming language agnostic interface.

REST API Overview

- REST API
  - A set of functions and a MarkLogic REST vocabulary



REST API Overview

- REST Client API: Build Applications, CRUD, Search
- REST Management API: Perform Administrative Tasks
- REST Packaging API: Configuration Management

Task	HTTP Method
• CREATE • UPDATE	• POST • PUT
• READ	• GET
• DELETE	• DELETE

REST API Services Examples

- Many more services exist...see the documentation for the inclusive list

Service Name	Description
`/v1/rest-apis`	• POST • Create an instance of the MarkLogic REST API, including an HTTP app server, required modules DB, and optionally a content DB. • GET • Retrieve a list of REST API instances, including config details.
`/v1/documents`	• PUT • **Insert or update** document contents and/or metadata, at a caller-supplied **document URI**. • GET • **Retrieve** document content and/or metadata from the database. • DELETE • **Remove** a document, or reset document metadata.
`/v1/search`	• GET • **Search** the database using a **string and/or structured query**. • DELETE • **Remove** documents in a collection or directory, or clear the DB.

REST API Extensibility

- Out of the box, the REST API provides:
  - An HTTP interface for core search functionality
  - Document CRUD (Create, Read, Update, Delete)
  - JSON interface
  - Transactions
  - Key – Value (JSON), Element – Value (XML) Interface
  - Administration
  - Configuration Management
- The REST API is extendable:
  - MarkLogic product experts can create new services to provide additional functionality

Creating a REST Instance

- myconfig.xml:

```
<rest-api xmlns="http://marklogic.com/rest-api">
  <name>top-songs-appserver</name>
  <group>Default</group>
  <database>top-songs-content</database>
  <modules-database>top-songs-modules</modules-database>
  <port>7010</port>
</rest-api>
```

- Invoking the REST API with cURL:

```
curl --anyauth --user admin:admin -X POST \
-d@"./myconfig.xml" -i -H "Content-type:application/xml“ \
http://localhost:8002/v1/rest-apis
```

- So what’s going on inside this cURL statement?

MarkLogic®

Creating a REST Instance

- myconfig.xml:

```
<rest-api xmlns="http://marklogic.com/rest-api">
  <name>top-songs-appserver</name>
  <group>Default</group>
  <database>top-songs-content</database>
  <modules-database>top-songs-modules</modules-database>
  <port>7010</port>
</rest-api>
```

- Invoking the REST API with cURL:

```
curl --anyauth --user admin:admin -X POST \
-d@"./myconfig.xml" -i -H "Content-type:application/xml“ \
http://localhost:8002/v1/rest-apis
```

- --anyauth
  - “Figure out the authentication method by yourself; use the most secure”
  - This is done by first sending a request and checking the response-headers, thus possibly inducing an extra network round-trip.
  - Eliminate this extra round trip by specifying **--digest** instead.

MarkLogic

Creating a REST Instance

- myconfig.xml:

```
<rest-api xmlns="http://marklogic.com/rest-api">
  <name>top-songs-appserver</name>
  <group>Default</group>
  <database>top-songs-content</database>
  <modules-database>top-songs-modules</modules-database>
  <port>7010</port>
</rest-api>
```

- Invoking the REST API with cURL:

```
curl --anyauth --user admin:admin -X POST \
-d@"./myconfig.xml" -i -H "Content-type:application/xml“ \
http://localhost:8002/v1/rest-apis
```

- --user
  - Specifies that authentication will be provided in the form of USERNAME:PASSWORD
  - admin:admin is the admin user that we created after installing the product.
  - Any user with the **rest-writer** role can perform this action

Creating a REST Instance

- myconfig.xml:

```
<rest-api xmlns="http://marklogic.com/rest-api">
  <name>top-songs-appserver</name>
  <group>Default</group>
  <database>top-songs-content</database>
  <modules-database>top-songs-modules</modules-database>
  <port>7010</port>
</rest-api>
```

- Invoking the REST API with cURL:

```
curl --anyauth --user admin:admin -X POST \
-d@"./myconfig.xml" -i -H "Content-type:application/xml“ \
http://localhost:8002/v1/rest-apis
```

- -X
  - Specifies a custom request method. If not provided, the **default is GET**.
  - For our action we are specifying **POST**
  - \ is simply a line break for formatting this text on the slide. You should enter this all on one command line and eliminate the \

MarkLogic

Creating a REST Instance

- myconfig.xml:

```
<rest-api xmlns="http://marklogic.com/rest-api">
  <name>top-songs-appserver</name>
  <group>Default</group>
  <database>top-songs-content</database>
  <modules-database>top-songs-modules</modules-database>
  <port>7010</port>
</rest-api>
```

- Invoking the REST API with cURL:

```
curl --anyauth --user admin:admin -X POST \
-d@"./myconfig.xml" -i -H "Content-type:application/xml“ \
http://localhost:8002/v1/rest-apis
```

- -d
  - Sends the specified data in a POST request to the HTTP server, in the same way that a browser does when a user has filled in an HTML form and presses the submit button.
  - @ is a reference to a file containing configuration info.

Creating a REST Instance

- myconfig.xml:

```
<rest-api xmlns="http://marklogic.com/rest-api">
  <name>top-songs-appserver</name>
  <group>Default</group>
  <database>top-songs-content</database>
  <modules-database>top-songs-modules</modules-database>
  <port>7010</port>
</rest-api>
```

- Invoking the REST API with cURL:

```
curl --anyauth --user admin:admin -X POST \
-d@"./myconfig.xml" -i -H "Content-type:application/xml“ \
http://localhost:8002/v1/rest-apis
```

- -i
  - Include the HTTP-header in the output. The HTTP-header includes things like server-name, date of the document, HTTP-version and more.
- -H
  - Extra header we are going to send indicating content-type as XML

MarkLogic

Creating a REST Instance

- myconfig.xml:

```
<rest-api xmlns="http://marklogic.com/rest-api">
  <name>top-songs-appserver</name>
  <group>Default</group>
  <database>top-songs-content</database>
  <modules-database>top-songs-modules</modules-database>
  <port>7010</port>
</rest-api>
```

- Invoking the REST API with cURL:

```
curl --anyauth --user admin:admin -X POST \
-d@"./myconfig.xml" -i -H "Content-type:application/xml“ \
http://localhost:8002/v1/rest-apis
```

- The REST service (endpoint) that we are invoking to accomplish the task.

MarkLogic

Creating a REST Instance - JSON

- myconfig.json:

```
{
  "rest-api":
  {
    "name": "top-songs-appserver",
        "group": "Default",
    "database": "top-songs-content",
        "modules-database": "top-songs-modules",
    "port": "7010"
  }
}
```

- Invoking the REST API with cURL:

```
curl --anyauth --user admin:admin -X POST \
-d@"./myconfig.json" -i -H "Content-type:application/json“ \
http://localhost:8002/v1/rest-apis
```

MarkLogic

More REST API Examples…

- Deleting a REST instance:

```
curl -X DELETE --anyauth --user admin:admin "http://localhost:8002/v1/rest-
apis/top-songs-appserver?include=content&include=modules"
```

- Loading an XML document + collections:

```
curl --anyauth --user admin:admin -X PUT -T ./song1.xml \
"http://localhost:7010/v1/documents?uri=/songs/song1.xml&format=xml&collection=m
usic&collection=classic rock"
```

- Loading a JSON document + collections + metadata:

```
curl --anyauth --user admin:admin -X PUT -T ./song2.json \
"http://localhost:7010/v1/documents?uri=/songs/song2.json&format=json&collection
=music&collection=classic rock&prop:album=Full Moon Fever&prop:misc=some
additional metadata"
```

More REST API Examples…

- Reading a document:

```
curl --anyauth --user admin:admin -X GET \
"http://localhost:7010/v1/documents?uri=/myDocumentURI"
```

- Searching for a document:

```
curl --anyauth --user admin:admin -X GET \ "http://localhost:7010/v1/search?q=my
search query"
```

- Deleting a document:

```
curl --anyauth --user admin:admin -X DELETE \
"http://localhost:7010/v1/documents?uri=/myDocumentURI"
```

Demo: Samplestack Gradle Deployment

Labs: Unit 3

Exercise 1 – Exercise 3: Creating and Deleting REST Instances
Exercise 4 – Exercise 5: Loading Documents and Metadata
Exercise 6 – Exercise 7: Document Reads and Searches
Exercise 8: Document Deletes
DIY: Create REST Instances
Appendix: Samplestack Gradle Deployment Roadmap

Unit Review Question 1:

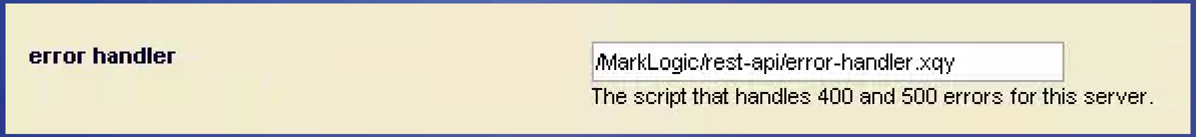
A REST instance represents which type of MarkLogic application server:

1. REST
2. HTTP
3. XDBC
4. ODBC

Unit Review Question 1:

A REST instance represents which type of MarkLogic application server:

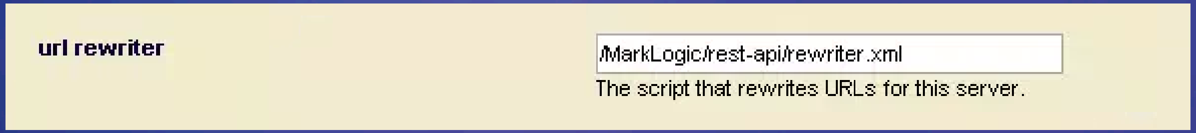
1. REST
2. **HTTP, with a bit of specific configuration:**
3. XDBC
4. ODBC



error handler

/MarkLogic/rest-api/error-handler.xqy

The script that handles 400 and 500 errors for this server.



url rewriter

/MarkLogic/rest-api/rewriter.xml

The script that rewrites URLs for this server.

Unit Review Question 2:

Assume you execute the following as a GET request:

```
"http://localhost:7010/v1/search?q=cat OR dog"
```

What database will this request run against?

Unit Review Question 2:

Assume you execute the following GET request:

```
"http://localhost:7010/v1/search?q=cat OR dog"
```

What database will this request run against?

Answer:
Whatever database is defined for the REST instance on 7010.

Unit Review Question 3:

Which REST API would you use if you wished to script the deployment of a cluster:

1. Management API
2. Client API
3. Packaging API

Unit Review Question 3:

Which REST API would you use if you wished to script the deployment of a cluster:

1. **Management API**
2. Client API
3. Packaging API