# **Customizing the behaviors of POJORepository interfaces**

By default, POJORepository interface implements ‘own’ URIs (Class Name/ID). The documents generated does not have a root element and the structure will have the class names included. All these behaviors can be changed by over-riding some implementations. Additionally, customization can be done to add metadata, properties and other collections to the documents.

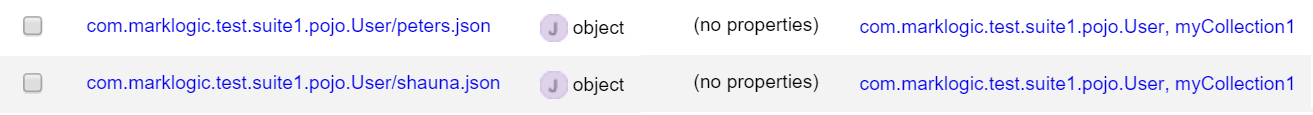


Figure 1: Default URI Generation



Figure 2: Default Document Structure

doPojoRepositoryOperationsTest in POJORepositoryOperationsTest4.java is a Junit method that has the default behavior implemented.

The customization can be done by implementing PojoRepository and and PojoPageImpl. A Factory can be used to get instances of POJORepository. Refer DAOFactory.java, CustomPojoRepositoryImpl.java and CustomPojoPageImpl.java for the implementations.

doCustomePojoRepositoryOperationsTest in CustomPOJORepositoryOperationsTest4 is the Junit method that has the customized behavior implemented.

Note: The Junit tests are extracted from a bigger Junit package. Changes have been made to make it run independently. Changes to host, dbname, username and password will be required in the code to create the DB client. DatabaseClientFactory.newClient..



Figure 3: Customized URI



Figure 4: Document Structure without Class Names

Partial Document Extractions.

Many times, the application requires only part of the document. The extracted part can be included in the search result using query options.

For example, consider a employee document as below.



The below implementation extracts Addresses only as returns as part of search result.

String searchOptions = "<search:search xmlns:search=\"http://marklogic.com/appservices/search\">"

+ "<cts:element-word-query xmlns:cts=\"http://marklogic.com/cts\">"

+ "<cts:element>lName</cts:element>"

+ "<cts:text xml:lang=\"en\">Kennedy</cts:text>"

+ "</cts:element-word-query>"

+ "<search:options>"

+ "<search:extract-document-data>"

+ "<search:extract-path>/employee/Addresses</search:extract-path>"

+ "</search:extract-document-data>"

+ "<search:additional-query>"

+ "<cts:collection-query xmlns:cts=\"http://marklogic.com/cts\">"

+ "<cts:uri>"+ COLLECTION\_NAME + "</cts:uri>"

+ " </cts:collection-query>"

+ "</search:additional-query>"

+ "</search:options>"

+ "</search:search>";

StringHandle queryHandle = new StringHandle(searchOptions).withFormat(Format.XML);

/\*\*\*\* VALUES RETRIEVAL \*\*\*\*/

// create a manager for searching

QueryManager queryMgr = client.newQueryManager();

// create a values definition

QueryDefinition query = queryMgr.newRawCombinedQueryDefinition(queryHandle);

// retrieve the results

SearchHandle results = queryMgr.search(query, new SearchHandle());

MatchDocumentSummary[] matches = results.getMatchResults();

for (MatchDocumentSummary match : matches) {

ExtractedResult extracts = match.getExtracted();

for (ExtractedItem extract: extracts) {

genTestUtils.logComments(" extracted content: " +

extract.getAs(String.class), LOGLEVEL);

}

}

Output:

extracted content: {"Addresses":{"Address":[{"addrType":"R","addrLine1":"678 BirchWood","city":"Chicago","state":"IL","postalCode":"60094"},{"addrType":"M","addrLine1":"987 W. Monroe","city":"Chicago","state":"IL","postalCode":"60066"}]}}

Similarly to get only addrTypes, the search options can be

String searchOptions = "<search:search xmlns:search=\"http://marklogic.com/appservices/search\">"

+ "<cts:element-word-query xmlns:cts=\"http://marklogic.com/cts\">"

+ "<cts:element>lName</cts:element>"

+ "<cts:text xml:lang=\"en\">Kennedy</cts:text>"

+ "</cts:element-word-query>"

+ "<search:options>"

+ "<search:extract-document-data>"

+ "<search:extract-path>/employee/Addresses/Address/addrType</search:extract-path>"

+ "</search:extract-document-data>"

+ "<search:additional-query>"

+ "<cts:collection-query xmlns:cts=\"http://marklogic.com/cts\">"

+ "<cts:uri>"+ COLLECTION\_NAME + "</cts:uri>"

+ " </cts:collection-query>"

+ "</search:additional-query>"

+ "</search:options>"

+ "</search:search>";

Output:

extracted content: {"addrType":"R"}

extracted content: {"addrType":"M"}

The same approach can be used to extract the whole document also.

String searchOptions = "<search:search xmlns:search=\"http://marklogic.com/appservices/search\">"

+ "<cts:element-word-query xmlns:cts=\"http://marklogic.com/cts\">"

+ "<cts:element>lName</cts:element>"

+ "<cts:text xml:lang=\"en\">Kennedy</cts:text>"

+ "</cts:element-word-query>"

+ "<search:options>"

+ "<search:extract-document-data>"

+ "<search:extract-path>/employee</search:extract-path>"

+ "</search:extract-document-data>"

+ "<search:additional-query>"

+ "<cts:collection-query xmlns:cts=\"http://marklogic.com/cts\">"

+ "<cts:uri>"+ COLLECTION\_NAME + "</cts:uri>"

+ " </cts:collection-query>"

+ "</search:additional-query>"

+ "</search:options>"

+ "</search:search>";

Output:

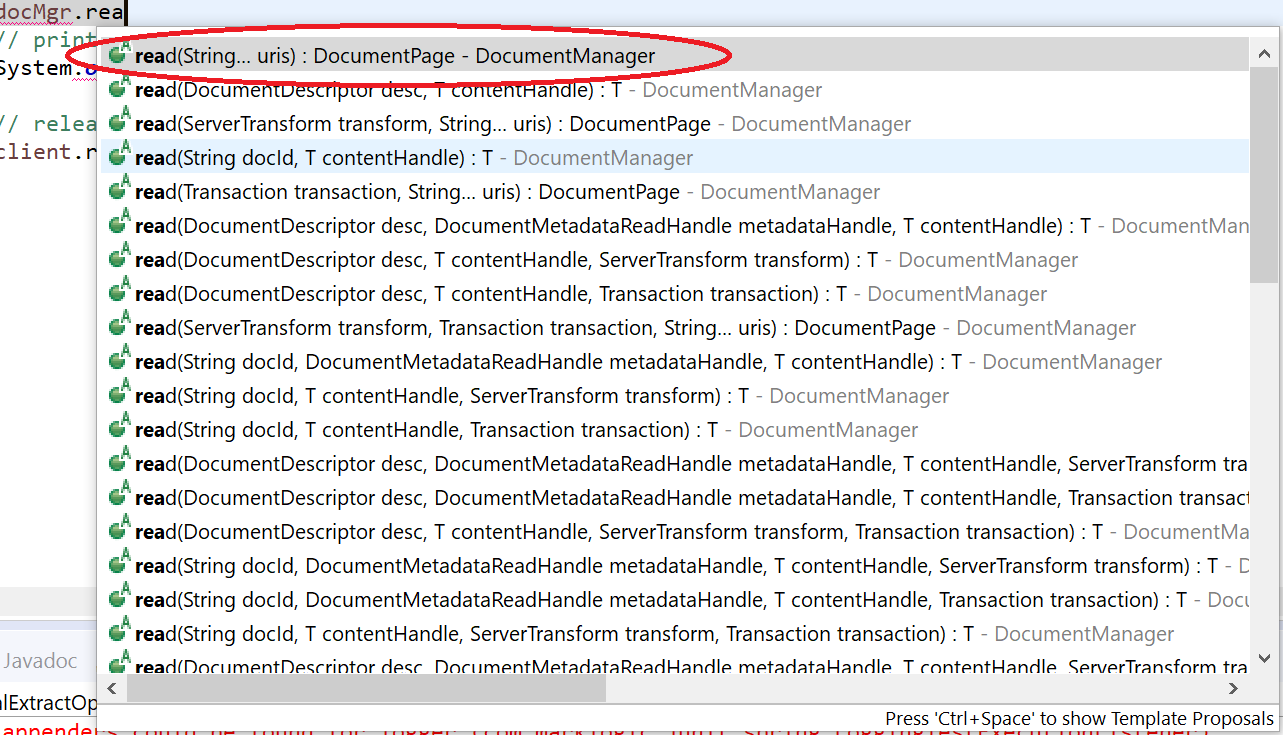
extracted content: {"employee":{"employeeId":"456","fName":"John","lName":"Kennedy","Addresses":{"Address":[{"addrType":"R","addrLine1":"678 BirchWood","city":"Chicago","state":"IL","postalCode":"60094"},{"addrType":"M","addrLine1":"987 W. Monroe","city":"Chicago","state":"IL","postalCode":"60066"}]}}}

As it is obvious, partial extraction of documents is intended to return selected portions of each matching document instead of whole big document. If entire document is extracted, there could be performance implications depending on the size of documents and the number of documents matched.

# Mult-Document Reads.

**Approach 1**

If there are multiple documents to be read in a single call, DocumentManager.read(String… ) can be used to return a DocumentPage which can be iterated for each document.



[**Approach**](https://docs.marklogic.com/javadoc/client/com/marklogic/client/datamovement/QueryBatcher.html) **2**

For very high throughput requirements, QueryBatcher is also a good candidate. Generally applied for high volume batch transactions. Please refer the below documentation to see the applicability in the use case.

<https://docs.marklogic.com/javadoc/client/com/marklogic/client/datamovement/QueryBatcher.html>

================