NBIA Feature Dynamic Search Description

|  |  |
| --- | --- |
| Last Revised: | DEC 22, 2009 |
| Produced By: | NBIA Team |
| Version: |  |

Document Approvals

The list contains the name and contact information for the core project team and any key stakeholders who have an interest in the success of the project. An “S” identifies persons responsible for approval from the stakeholder groups. Sign off of the document would be required when a decision is made not to take action for defined gaps.

|  |  |  |
| --- | --- | --- |
| S | Name | Role |
| S | Robert Shirley | NCICBIIT |
| S | Peter Yan | COTR/PM |
|  | Eric Kascic | Technical Lead |
|  |  |  |

Revision History

When you make a change to a document, you must add an entry to this Revision History table and you must manually type the Last Revised Date on the front cover.

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Version | Description | Revised by |
| 12/22/2010 | 1 | Draft Document for CTP feature | Jim Zhou |
|  |  |  |  |
|  |  |  |  |

Copyrights and Trademarks

© Copyright 2010 by CBIIT, caBIGTM. All rights reserved.

Table of Contents

[1. Introduction 1](#_Toc279500559)

[2. Summary of Module 1](#_Toc279500560)

[3. Architectural Diagram 1](#_Toc279500561)

[4. Example 1 1](#_Toc279500562)

[5. Example 2 1](#_Toc279500563)

# Introduction

The purpose of this implementation is to provide a dynamic search function to the NBIA application. This function allows users to form a set of search criteria that can be used to search NBIA database. Dynamic search criteria are based on NBIA grid service API. After users provides these search criteria, NBIA application will general SQL/HQL statement to perform the search, and present search results to the user, like NBIA old search result. Users can click on the patient ID to get into details of the search result, such as viewing studies, series, and image.

# Summary of Feature

1. Dynamic Search Items

Current design of dynamic search criteria is list as following,

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | Item label | Item Name | Type |
| Patient Group | Patient |  |  |  |
|  |  | Patient ID | patientId | String |
|  |  | Patient name | patientName | String |
|  |  | Patient gender | patientSex | Enumeration / String (M, F, U,O, NULL) |
|  |  | Ethnic group | ethnicGroup | String |
|  |  | Patient birth day | patientBirthDate | Date |
|  | Trial\_data\_provenance |  |  |  |
|  |  | Site Name | dpSiteName | String |
|  |  | Project | project | Enumeration / String |
| Study Group | Study |  |  |  |
|  |  | Study Instance UID | studyInstanceUID | String |
|  |  | Study Date | studyDate | Date |
|  |  | *Study Time* | *studyTime* | *String* |
|  |  | Study Description | studyDesc | String |
|  |  | Admitting Diagnoses Description | admittingDiagnosesDesc | String |
|  |  | Admitting Diagnoses Code Sequence | admittingDiagnosesCodeSeq | String |
|  |  | Study ID | studyId | String |
|  |  | Trial Time Point ID | timePointId | String |
|  |  | Trial Time Point Description | timePointDesc | String |
|  |  | Patient Age | patientAge | String |
|  |  | Patient Height | patientSize | Double |
|  |  | Patient Weight | patientWeight | Double |
|  |  | Occupation | occupation | String |
|  |  | Additional Patient History | additionalPatientHistory | String |
| Series group | Series |  |  |  |
|  |  | Modality | modality | Enumeration/String (value from DB) |
|  |  | Series Instance UID | seriesInstanceUID | String |
|  |  | Series Laterality | laterality | Enumeration /String (B, L, R, U) |
|  |  | Series Date | seriesDate | Date |
|  |  | Protocol Name | protocolName | String |
|  |  | Series Description | seriesDesc | String |
|  |  | Anatomical Site | bodyPartExamined | Enumeration/String (value from DB) |
|  |  | Series Number | seriesNumber | Integer |
|  |  | Synchronization Frame Of Reference UID | syncFrameOfRefUID | String |
|  |  | Frame of Reference UID | frameOfReferenceUID | String |
|  | *General Equipment* |  |  |  |
|  |  | *Manufacturer* |  |  |
|  |  | *Instition Name* |  |  |
|  |  | *Instition Address* |  |  |
|  |  | *Manufacturer Model Name* |  |  |
|  |  | *software Versions* |  |  |
|  |  | *Station Name* |  |  |
|  |  | *Device Serial Number* |  |  |
| Image Group | General Image |  |  |  |
|  |  | Instance Number | instanceNumber | Integer |
|  |  | Content Date | contentDate | Date |
|  |  | *Content Time* | *contentTime* | *String* |
|  |  | Image Type | imageType | String |
|  |  | Acquisition Date | acquisitionDate | Date |
|  |  | *Acquisition Time* | *acquisitionTime* | *String* |
|  |  | Acquisition Number | acquisitionNumber | Integer |
|  |  | Lossy Image Compression | lossyImageCompression | String |
|  |  | Pixel Spacing | pixelSpacing | Double |
|  |  | Image Orientation Patient | imageOrientationPatient | String? |
|  |  | Image Position Patient | imagePositionPatient | String? |
|  |  | Slice Thickness | sliceThickness | Double |
|  |  | Slice Location | sliceLocation | Double |
|  |  | Contrast Bolus Agent | contrastBolusAgent | String |
|  |  | Contrast Bolus Route | contrastBolusRoute | Enumeration/String (IV, Oral, Oral&IV)? |
|  |  | SOP Class UID | SOPClassUID | String |
|  |  | SOP Instance UID | SOPInstanceUID | String |
|  |  | Patient Position | patientPosition | Enumeration/String(FFDL, FFDR, FFP, FFS, HFDL, HFDR, HFP, HFS, ERECT(?)) |
|  |  | Source To Detector Distance | sourceToDetectorDistance | Double |
|  |  | Source Subject Distance | sourceSubjectDistance | Double |
|  |  | Focal Spot Size | focalSpotSize | Double |
|  |  | Storage Media File Set UID | storageMediaFileSetUID | String |
|  |  | Acquisition Date Time | acquisitionDatetime | String |
|  |  | Image Comments | imageComments | String |
|  |  | Image Laterality | imageLaterality | Enumeration/String(B,L,R,U) |
|  |  | Acquisition Matrix | acquisitionMatrix | Double |
| ? |  | DX Data Collection Diameter???? | dxDataCollectionDiameter | Double |
|  |  | Columns | columns | Integer |
|  |  | Rows | rows | Integer |
|  | CT\_Image |  |  |  |
|  |  | KVP | KVP | Double |
|  |  | Scan Option | scanOptions | String |
|  |  | Data Collection Diameter | dataCollectionDiameter | Double |
|  |  | Reconstruction Diameter | reconstructionDiameter | Double |
|  |  | Gantry Detector Tilt | gantryDetectorTilt | Double |
|  |  | Exposure Time | exposureTime | Integer |
|  |  | X Ray Tube Current | XRayTubeCurrent | Integer |
|  |  | Exposure | exposure | Integer |
|  |  | Exposure In Micro As | exposureInMicroAs | Integer |
|  |  | Convolution Kernel | convolutionKernel | Enumeration/String(value from DB) |
|  |  | Revolution Time | revolutionTime | Integer |
|  |  | Single Collimation Width | singleCollimationWidth | Integer |
|  |  | Total Collimation Width | totalCollimationWidth | Integer |
|  |  | Table Speed | tableSpeed | Integer |
|  |  | Table Feed Per Rotation | tableFeedPerRotation | Integer |
|  |  | CT Pitch Factor | CTPitchFactor | Integer |
|  |  | Anatomic Region Sequence | anatomicRegionSeq | String |
|  |  |  |  |  |

1. SQL/HQL generation

SQL/HQL generation is main implementation for the dynamic search function. It contains 1) Search criteria validation 2) A module to generate SQL/HQL statement 3) Process search result in order to get all necessary information for the search result page.

1. Search Results

NBIA needs to convert dynamic search results into current search results, so current search result page can be reused.

# Architectural Diagram

## Workflow

|  |
| --- |
|  |

**Figure 1: Dynamic Search Page loading**

Figure 1 indicates how NBIA application loads search criteria on dynamic search page. When the user clicks on the dynamic search link, NBIA application service will parse XML file which contains all dynamic search criteria fields, and load them on dynamic search page. So the user can build up his/her search criteria.

|  |
| --- |
|  |

**Figure 2: Performing search**

Figure 2 indicates how searching has been performed. When the user clicks submit button, Dynamic Search action class gets all search criteria, and pass them into SQL Builder module to generate a SQL/HQL statement. Then builder will call backend code to execute SQL/HQL to perform search. After search results return back, NBIA application passes them to Search Result Convertor to convert data into current searching result format. And forward response back to search result page.

## Component s

1. XML parser for initializing dynamic search criteria

All search criteria should be constant variables for this design. Due to this reason, XML files will be used to store all search criteria. In order to enhance performance, all search criteria initialization will happen when the NBIA application lunches. Steps to initialize search criteria,

1. Start NBIA application
2. Initialize criteria bean, this bean will be in session scope.
3. use static block to invoke XML parser to initialize search criteria
4. Other beans can refer to criteria bean to fetch all items.
5. Dynamic Search Criteria

Dynamic search criteria object list hold all user-specified search criteria. By doing this, NBIA application can easily display search criteria, and SQL builder also can easily process them. This object contains field name, search value, files group, and operand.

1. Query builder

Query builder is main module for dynamic search. It will process user-specified search criteria and generate a SQL/HQL statement. This module needs following objects to support SQL/HQL generation,

1. DynamicSearchCriteria
2. RelationshipObject
3. OperandObject

Ideally, DynamicSearchCriteria Object holds the user-specified criteria. Query builder needs this list of DynamicSeachCriteria object to build a query. RelationshipObject holds predefined foreign key information. Based on user-specified criteria, NBIA Query Builder will analyze them, and add corresponding relationship into SQL/HQL statement. For instance, if the user-specified fields come from two field group, it needs to figure it out what is the join key (foreign key). After having this join key, we can easily to create where clause for query statement. OperandObject holds an operand among where clause statement. During query generation, operand will be added in the statement correctly.

Steps to generate SQL/HQL statement,

1. Process search criteria
2. Analyze search criteria and fetch all foreign keys for multiple joined tables.
3. Generate SQL/HQL statement with user-specified operand

At last, Query Builder will return this statement to caller.

1. Dynamic Search Action Bean

This bean will be responsible to collect all user-specified search criteria, operand, and information of selected tables from GUI. It also passes these parameters to Query Builder which will generate corresponded SQL/HQL statement. Then Action bean will invoke QueryExecutor to execute this statement to get results back. If it is necessary, Action bean will also invoke searchResult Convertor to transfer results into current search result format for loading current search result page.

1. Query Executor

This class will invoke HQL/SQL execution with previous generated statement. Also it will process search results.

1. Operator

When Query Builder builds the statement, this class holds user selected operator, like equal, like, greater than, less than, etc. These operators are needed to build a query statement.

1. Logical Connector

The same as the operator, from GUI, users need to choose what is the relationship between each where clause. This object will hold the user selection, and use it in Query Builder class.

1. Data Field

After NBIA application parses the XML file which contains all search criteria fields, Data Field object will hold these values in session. This class has public method for developers to fetch different elements from this object, such as all data sources, all fields in a data source, etc

1. Relational Operator

When the user selects fields from different date sources, Relational Operator needs to figure out what are foreign keys to these data sources. For instance, if the user selects one item from image table, and one item from patient table, Relational Operator will provide foreign keys among patient, study, series, and image tables.

1. Search Result

This class is place hold for search results.

1. Search Result Convertor

If we want to use current search result page, dynamic search result might need a convertor to convert dynamic search result to current search result format.

## Class Diagram



**Figure 3: Dynamic Search Class Diagram**

In this diagram, it indicates all class objects will be used in Dynamic Search process. DatasourceItem Object is used for loading dynamic search page.

## Sequence Diagram



# Change Example 1

## Set Eclipse Environment

* Check out NBIA source from trunk by using Tortoise tool
* Change directory to SOURCE\_CODE\_DIR/build
* Modify install.properties for database connection (user id, password, etc)
* Type “ant build:all” to build project (This must be happened before importing source code into Eclipse)
* Open Eclipse to import all modules as existing project
* Fix build path issue in the Eclipse, such as NBIA\_BASE set up
* After this, you can utilize the Eclipse to change your code.

## Recompile NBIA / Run Test Case

After modifying dynamic search source code, you need to perform the following steps to regenerate NBIA application and test it.

* From DOS prompt, run ant build:nbia-dao (or build:all)
* Run ant -Dunit.testing=true build:nbia-dao for unit test and style check
* Run ant deploy:local:install
* Above process will generate a directory C:\apps by default
* Change directory to C:\apps\ncia\jboss-4.0.5.GA\bin, start JBoss with run.bat command.
* Open a browser, type <http://localhost:45210/ncia> to start application
* Test your fixes.

# Change Example 2

No further example.