DynamicBrowsePrototype

An experiment by: Richard Jones.

This is an experiment in live documentation of the design of a new system during the development process. The design for the system is already on paper, but will inevitably change throughout the course of the work, which should be limited to just a few days.

The objective is to develop the browse system to be more flexible and to scale to a much greater degree. To this end, the starting point is my initial Browse patch that was placed on the SF patch tracker some months back. The specific scalability problems that are to be addressed are:

- pagination for second level browse pages (e.g. all items by a specific author)
- faster load times for browse pages
- improved SQL and SQL generation
- improved indexing

Configuration

The current browse configuration in the dynamic browse looks like this:

```
webui.browse.index.1 = dateissued:dc.date.issued:date:full
webui.browse.index.2 = author:dc.contributor.*:text:single
webui.browse.index.3 = title:dc.title:title:full
webui.browse.index.4 = subject:dc.subject.*:text:single
webui.browse.index.5 = dateaccessioned:dc.date.accessioned:date:full
```

# and some extra examples ones
# webui.browse.index.6 = type:dc.type:text:single
# webui.browse.index.7 = itemstatus:icadmin.status:text:single

It is therefore of the form:

```
webui.browse.index.<n> = <index name>:<metadata field>:<data type>:
```

The proposed addition is the following:

```
webui.browse.sort-option.<x> = <sort name>:<metadata field>:<data type>
```

This will allow the browse index mechanism to generate tables which contain enough information to sort results without having to instantiate any Item objects at all. This will make browsing faster and enable browses such as “all items by the author XXX” to be paginated successfully.

Therefore, new configuration will look like this (which will have the same effect as the current default):

```
# Set the options for what can be sorted by
webui.browse.sort-option.1 = title:dc.title:text
webui.browse.sort-option.2 = date:dc.date.issued:date
```

Data Model

The next stage is to rewrite the BrowseIndex class to be able to pick up this configuration for each of the indexes mentioned above. It will then create a number of Browse tables (when necessary):

- `index_<n>`-seq: a SEQUENCE for use by the `index_<n>` table
- `index_<n>`: indexed on the "value" column
- `collection_index_<n>`: a VIEW on the `index_<n>` table with collection2item.item_id = `index_<n>`-item_id
- `community_index_<n>`: a VIEW on the `index_<n>` table with community2item.item_id = `index_<n>`-item_id
- `index_<n>-value_index: the INDEX on index<n>` value

**QUESTION:** do we need to create INDEXes on all the other columns in every table? We may need to be open to the possibility.
This is similar to the existing browse and the dynamic browse patch, but the critical difference is in the table structure. index_<n> will be structured thus:

```
item_id: int not null FK  // the item id
value: text               // the text value of the core browse value
sort_value: text          // the normalised text value of the core browse value
sort_<x>                  // x number of columns corresponding to the normalised value of the sort-
options defined above
```

Progress Update: 21-11-2006, 13:40 GMT

The initial code to build the index tables from the configuration has been written (or, more accurately, adapted from the previous browse patch). I have successfully generated 7 indices which make allowances for sorting by title and by date issued. This has touched the following existing files: dspace.cfg, BrowseIndex.java and resulted in the creation of the following files: IndexBrowse.java, SortOption.java. Note that the IndexBrowse.java file is one I had previously written locally to alleviate some performance issues with the default indexer, and therefore contains a lot of the scalability improvements necessary for the indexing side of things. More on that later.

The Indexing Process

The next challenge is to get the data in the database actually indexed into the new tables. To do this we are moving the indexing process from Browse to IndexBrowse, and will be replacing calls to Item with calls to an ultra-lightweight Item like object called BrowseItem, whose only task will be to obtain metadata from the item tables.

The process then is as follows

- obtain all the BrowseItem objects
- obtain all the BrowseIndex objects (which will, in turn, contain the SortOption objects)
- for each BrowseItem object
  - for each BrowseIndex object
    - delete the existing index data for the BrowseItem
    - get the "value" metadata from the BrowseItem
    - for each SortOption object
      - get the "sort" metadata from the BrowseItem
      - for each metadata value (primary index)
        - write a line into the database with the index value and sort option values (normalised)
    - commit the transaction
- for each BrowseIndex object
  - delete all item ids that are in the index table but not the item table

NOTE: a potential problem arises. The "sort" fields need to be singular, while the "value" field can be multiple. That is, an item may have more than one author as the value to browse on, but may not have more than one title as the value to sort on. In the cases where an item has more than one value in the sort metadata, the code will select only the first value that is returned. This is a caveat that people configuring their system will need to be aware of.

Progress Update: 21-11-2006, 14:50 GMT

Against all belief, the indexing code appears to already be working. It as much as 0.1 seconds slower per item in a small database, which may need to be addressed (see TODO below). This code touches the following files: IndexBrowse.java, SortOption.java, BrowseException.java. No new files were necessary. I have included below two screen dumps of my test database, as an example as to what I am currently seeing in the indices:
1. select * from index_2;

| 1 | 1 | Jones, Richard | Jones, Richard | 2006-11-16T17:08:11Z | submit 1 |
| 2 | 2 | Jones, Richard | Jones, Richard | 2006-11-16T17:08:42Z | submit 2 |
| 3 | 3 | Jones, Richard | Jones, Richard | 2006-11-16T17:09:05Z | submit 3 |
| 4 | 4 | Jones, Richard | Jones, Richard | 2006-11-16T17:09:26Z | submit 4 |
| 5 | 5 | Jones, Richard | Jones, Richard | 2006-11-16T17:09:52Z | submit 5 |
| 6 | 6 | Jones, Richard | Jones, Richard | 2006-11-16T17:10:18Z | submit 6 |
| 7 | 7 | Jones, Richard | Jones, Richard | 2006-11-16T17:10:43Z | submit 7 |
| 8 | 8 | Jones, Richard | Jones, Richard | 2006-11-16T17:11:08Z | submit 8 |
| 9 | 9 | Jones, Richard | Jones, Richard | 2006-11-16T17:11:30Z | submit 9 |
| 10 | 10 | Jones, Richard | Jones, Richard | 2006-11-16T17:11:56Z | submit 10 |
| 12 | 12 | Jones, Richard | Jones, Richard | 2006-11-16T17:12:45Z | submit 12 |
| 17 | 17 | Jones, Richard | Jones, Richard | 2006-11-16T17:14:37Z | submit 17 |
| 20 | 20 | Jones, Richard | Jones, Richard | 2006-11-16T17:15:42Z | submit 20 |
| 21 | 21 | Jones, Richard | Jones, Richard | 2006-11-16T17:16:02Z | submit 21 |

(22 rows)

1. select * from index_1;


(22 rows)

TODO: the scale problem is at least in part because the sort values are obtained for each item for each browse index, which is an unnecessary amount of work. Refactoring should sort this out, but it remains as-is for the moment because it slipped easily into existing code

**Browse Servlet and User Interface**

Although we're not quite ready to start putting the UI together, it is now time to specify exactly what we want out of the UI interaction with the Servlet, because this gives us our window into the Browse engine itself. Therefore, I will write a primitive implementation of the BrowseServlet to deal directly with the Browse engine, and to initial just deposit useful debug to the screen.
The following are variables that will need to be passed into the Browse engine in order for appropriate results to be returned:

- **type**: the type of browse being undertaken. This will be used to identify the Browse Index from the config
- **sortBy**: which of the available sort options in config is to be sorted by
- **order**: which way to interpret the sortBy. ASC or DESC
- **value**: a specific value to browse upon. For example "Jones, Richard" to view all items where I am the author (in conjunction with type=author, of course)
- **resultsperpage**: number of results to display on the page at any one time
- **community**: the community we are browsing in
- **collection**: the collection we are browsing in
- **next**: the id of the item to be at the top of the "next" page
- **prev**: the id of the item to be at the top of the "previous" page
- **focus**: the target point in the listing to point the browse. This will be utilised by the paging system
- **year**: the year to use as a focus in date browse
- **month**: the month to use as a focus in date browse
- **startsWith**: the characters to use for a stem search. Will be used with the focus
- **vfocus**: the string to form the focus for single browse contexts

**NOTE**: "next" and "prev" are not clearly defined as to what the best way to obtain them is, and exactly what their relationship to "focus" is. It may be that "next" and "prev" are only used in the Servlet/UI layer to represent the "focus" for the next and previous functionality.

**SQL Queries**

This section is a summary of my first (untested) stabs at the SQL required by the Browse engine. Some of them MAY NOT WORK, or may not be complete yet

Obtain the results for a given value or focus:

```
SELECT * FROM <index>
WHERE sort_value [<|=*] [<value> | __focus__]
    [AND collection_id = __collection__]
    [AND community_id = __community__]
ORDER BY <sortBy> <order>
LIMIT <resultsperpage> + 1
```

So if "focus" is used to tell us which "next" or "prev" we should be looking at, then we may need to be able to dispense with them all together. <index> is updated to refer to the relevant table name (whether it is index_<n>, collection_index_<n> or community_index_<n>), and if the sort terms are correctly prepared then the simple comparators should be enough to ensure that we get everything in the desired order.

In order to output the string "Results A - B of C" the following are required (B = A + <resultsperpage>):

- **A:**

  ```
  SELECT COUNT(*) FROM <index>
  WHERE value [<_] <focus>
  ```

  Here the "focus" must be implicitly defined for every request so that this query always returns, although sometimes will return 0. That means that after the SELECT above we must at least always assign the first result to be the focus if it has not already been defined.

- **C:**

  ```
  SELECT COUNT(*) FROM <index>
  [WHERE value = __value__]
  ```

**Ongoing programmer notes**

- **sortBy** is an int parameter, indicating which sort field to use. sortBy = 0 will therefore be sort by the index value
- **focus** is either: a value pulled from the UI top navigation; a specific item id to browse to. This strikes me that it might need to be divided into two parts
  - I have made an executive decision that focus will refer only to item ids. Everything else must go through "value" or "starts_with"
  - If order = ASC value comparator = >, if order = DESC value comparator = <. How about =? This is when a value is supplied, in which case the comparator is applicable to the sortBy field instead
- **QUESTION**: do we do our comparisons for browse on the "value" or the "sort_value" fields. Since "sort_value" has some sort of normalisation applied to it we must either normalise the request and compare it to that or not normalise the request and compare it to the "value" column

```
SQL Revisited

It has become necessary, on implementation, to modify the first SQL query given above, and add an additional query to satisfy paging with item focusses:

The first query becomes:

```sql
SELECT * FROM <index>
WHERE sort_value [<|=*] [<value> | _focus-value_]  
[AND collection_id = _collection_]  
[AND community_id = _community_]
ORDER BY <sortBy> <order>
LIMIT <resultsperpage + 1>
```

Just a minor change to indicate that the sort_value is not the <focus> as previously indicated, but the value of the <focus> id in the relevant context. This means we must add the following SQL:

```sql
SELECT sort_value FROM <index>
WHERE item_id = <focus>
```

This seems like a slightly dodgy solution, but this is almost exactly how the current browse mechanism does it. I will implement it this way, and revisit later if there are problems.

Further to this, here are 3 SQL case studies:

1) Browse all by title:

```sql
SELECT * FROM index_1
ORDER BY sort_value ASC
LIMIT 21
```

2) Browse page 3 of the author list

```sql
SELECT * FROM index_2
WHERE sort_value > [focus value]
ORDER BY sort_value ASC
LIMIT 22
```

3) Browse page 2 of author Jones, Richard, ordering by title

```sql
SELECT * FROM index_2
WHERE sort_value > [focus value]
ORDER BY sort_value ASC
LIMIT 22
```
SELECT * FROM index_2
WHERE sort_1 > [focus value]
AND sort_value = 'jones, richard'
ORDER BY sort_1 ASC
LIMIT 22

Ongoing programmers notes

- As yet I have not attempted a treatment of the startsWith parameter. It strikes me that this needs to be dealt with in just the same way as the focus, and therefore may need to be merged with the focus parameter right at the start
- I'm also proposing that instead of passing Item objects into the BrowseInfo, that actually we passBrowseItem objects, which are extra lightweight, and have been written for the index process already
- in the parlance of the BrowseInfo (I think):
  - position = B
  - total = C
  - offset = A
- position relies on us having the focus for the top item. If a focus is not supplied to the engine, then we must get the value from the top result from the query

Progress Update: 22-11-2006, 12:10 GMT

This lunch time we reach that first goal of writing a chunk of code: it compiles. That is, I have written and compiled the initial version of the code which I think can take inputs through a URL according to the variables defined above, and perform the appropriate queries on the database defined also above. The next goal, then, is that great achievement - the elimination of any runtime problems. After that we'll know if it actually does what I think it does.

Ongoing programmers notes

- Here's an oddity which rings a bell from the last time I looked at the Browse code. It seems that the result of a SELECT COUNT query, despite being a number can't be retrieved using the TableRow.getIntColumn() method. I'm trying TableRow.getStringColumn and Integer.parseInt, but I have a horrible feeling that won't work either.
  - Nope, as anticipated:

```
Exception:
java.lang.IllegalArgumentException: Value for number is not an integer
    at org.dspace.storage.rdbms.TableRow.getIntColumn(TableRow.java:162)
```

```
Exception:
java.lang.IllegalArgumentException: Value is not an string
    at org.dspace.storage.rdbms.TableRow.getStringColumn(TableRow.java:244)
```

- a spot of experimentation suggests that it can be got hold of as a "long"
- Initial experiments with the browse code are looking positive. At least there are no errors coming out of the SQL and result sets containing actual data are being returned. Of course, we don't know for certain that it is the correct data yet. One problem that we need to address is some sort of entity object to cover the display end of things. As you may know, the current configuration for browse listings looks like this:

```
webui.itemlist.columns = dc.date.issued(date), dc.title, dc.contributor.*
```

In order to make our debug better, it would be handy to have this wrapped up in an object that could do the configuration, rather than buried in the ItemListTag.java which is where it is just now

IMPORTANT NOTE: the previously optional configuration line "webui.itemlist.columns" has now become compulsory

- OK, some good progress. I can now see that the list config is being picked up and dealt with properly, and therefore have worked out the prototype to the display logic (although not actually written any display code beyond BrowseInfo.toString()). There appears to be a problem getting hold of the right metadata through the BrowseItem object, which I am moving onto now
I have got code that "works" in the minimal sense of the word, which appears to give us useful results (still to confirm that they are correct in each context, see below). Adding a toString method to the BrowseInfo, I have dumped browse results to the screen for debugging. Below I have pasted the browse results corresponding to index_2 and index_1 in my dev box, which represent the tables shown further up this page. The work has touched the following files: BrowseInfo.java, DatabaseManager.java (for extra debug), dspace-web.xml, and created the following new files: BrowserServlet.java, BrowserScope.java, ItemListConfig.java.

<table>
<thead>
<tr>
<th>Item ID</th>
<th>dc.date.issued</th>
<th>dc.title.null</th>
<th>dc.contributor.*</th>
</tr>
</thead>
<tbody>
<tr>
<td>22</td>
<td>2006-11-16T17.17.24Z</td>
<td>submit 22</td>
<td>Jones, Richard</td>
</tr>
<tr>
<td>21</td>
<td>2006-11-16T17.16.02Z</td>
<td>submit 21</td>
<td>Jones, Richard</td>
</tr>
<tr>
<td>20</td>
<td>2006-11-16T17.15.42Z</td>
<td>submit 20</td>
<td>Jones, Richard</td>
</tr>
<tr>
<td>19</td>
<td>2006-11-16T17.15.19Z</td>
<td>submit 19</td>
<td>Jones, Richard</td>
</tr>
<tr>
<td>15</td>
<td>2006-11-16T17.13.52Z</td>
<td>submit 15</td>
<td>Jones, Richard</td>
</tr>
<tr>
<td>14</td>
<td>2006-11-16T17.13.31Z</td>
<td>submit 14</td>
<td>Jones, Richard</td>
</tr>
<tr>
<td>13</td>
<td>2006-11-16T17.13.09Z</td>
<td>submit 13</td>
<td>Jones, Richard</td>
</tr>
<tr>
<td>12</td>
<td>2006-11-16T17.12.45Z</td>
<td>submit 12</td>
<td>Jones, Richard</td>
</tr>
<tr>
<td>11</td>
<td>2006-11-16T17.12.19Z</td>
<td>submit 11</td>
<td>Jones, Richard</td>
</tr>
<tr>
<td>10</td>
<td>2006-11-16T17.11.56Z</td>
<td>submit 10</td>
<td>Jones, Richard</td>
</tr>
<tr>
<td>9</td>
<td>2006-11-16T17.11.30Z</td>
<td>submit 9</td>
<td>Jones, Richard</td>
</tr>
<tr>
<td>8</td>
<td>2006-11-16T17.11.08Z</td>
<td>submit 8</td>
<td>Jones, Richard</td>
</tr>
<tr>
<td>7</td>
<td>2006-11-16T17.10.43Z</td>
<td>submit 7</td>
<td>Jones, Richard</td>
</tr>
<tr>
<td>6</td>
<td>2006-11-16T17.10.18Z</td>
<td>submit 6</td>
<td>Jones, Richard</td>
</tr>
<tr>
<td>5</td>
<td>2006-11-16T17.09.52Z</td>
<td>submit 5</td>
<td>Jones, Richard</td>
</tr>
<tr>
<td>4</td>
<td>2006-11-16T17.09.26Z</td>
<td>submit 4</td>
<td>Jones, Richard</td>
</tr>
<tr>
<td>3</td>
<td>2006-11-16T17.09.05Z</td>
<td>submit 3</td>
<td>Jones, Richard</td>
</tr>
<tr>
<td>2</td>
<td>2006-11-16T17.08.42Z</td>
<td>submit 2</td>
<td>Jones, Richard</td>
</tr>
<tr>
<td>1</td>
<td>2006-11-16T17.08.11Z</td>
<td>Submit 1</td>
<td>Jones, Richard</td>
</tr>
</tbody>
</table>
Listing over 3 columns: dc.date.issued, dc.title.null, dc.contributor.*

{{Item ID: 1 :: [dc.date.issued:2006-11-16T17.08.11Z][dc.title.null:Submit 1][dc.contributor.*:Jones, Richard]}}

{{Item ID: 2 :: [dc.date.issued:2006-11-16T17.08.42Z][dc.title.null:submit 2][dc.contributor.*:Jones, Richard]}}

{{Item ID: 3 :: [dc.date.issued:2006-11-16T17.09.05Z][dc.title.null:submit 3][dc.contributor.*:Jones, Richard]}}


{{Item ID: 5 :: [dc.date.issued:2006-11-16T17.09.52Z][dc.title.null:submit 5][dc.contributor.*:Jones, Richard]}}

{{Item ID: 6 :: [dc.date.issued:2006-11-16T17.10.18Z][dc.title.null:submit 6][dc.contributor.*:Jones, Richard]}}

{{Item ID: 7 :: [dc.date.issued:2006-11-16T17.10.43Z][dc.title.null:submit 7][dc.contributor.*:Jones, Richard]}}

{{Item ID: 8 :: [dc.date.issued:2006-11-16T17.11.08Z][dc.title.null:submit 8][dc.contributor.*:Jones, Richard]}}

{{Item ID: 9 :: [dc.date.issued:2006-11-16T17.11.30Z][dc.title.null:submit 9][dc.contributor.*:Jones, Richard]}}

{{Item ID: 10 :: [dc.date.issued:2006-11-16T17.11.56Z][dc.title.null:submit 10][dc.contributor.*:Jones, Richard]}}


{{Item ID: 20 :: [dc.date.issued:2006-11-16T17.15.42Z][dc.title.null:submit 20][dc.contributor.*:Jones, Richard]}}


TODO: This is currently returning one more row that it's supposed to. This is because I've asked the query to get the row after the current page so that we have a focus for the next page browse. I need to strip this from the result set before giving them to the browse info object.

Testing the Browse URL

To get an idea as to whether the Browse is working correctly, we need to just run through some tests with the URL API. At this stage, "month", "year", "starts_with", "community" and "collection" cannot be tested as they have not been built in to the engine yet. The following can be tested, though:

- type = dateissued | author | title | subject
- order = ASC | DESC
- value = [free text]
- focus = [item id]
- rpp = [integer: 1 - X]
- sort_by = [integer: 1 - N]

The full browse URL is of the form:

`browse?type=<type>&amp;order=<order>&amp;value=<value>&amp;focus=<focus>&amp;rpp=<rpp>&amp;sort_by=<sort_by>`
So for example:

```
browse?type=author&order=ASC&value=Jones%2C+Richard&focus=34&rpp=10&sort_by=1
```

The following are URLs that have been tested, and the results:

```
browse?type=dateissued&order=ASC&focus=3&rpp=12&sort_by=0
```

BrowseInfo String Representation: Browsing 2 to 15 of 22|

Listing over 3 columns: dc.date.issued, dc.title.null, dc.contributor.*|

<table>
<thead>
<tr>
<th>Item ID</th>
<th>dc.date.issued</th>
<th>dc.title.null</th>
<th>dc.contributor.*</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>2006-11-16T17.09.05Z</td>
<td>submit 3</td>
<td>Jones, Richard</td>
</tr>
<tr>
<td>4</td>
<td>2006-11-16T17.09.26Z</td>
<td>submit 4</td>
<td>Jones, Richard</td>
</tr>
<tr>
<td>5</td>
<td>2006-11-16T17.09.52Z</td>
<td>submit 5</td>
<td>Jones, Richard</td>
</tr>
<tr>
<td>6</td>
<td>2006-11-16T17.10.18Z</td>
<td>submit 6</td>
<td>Jones, Richard</td>
</tr>
<tr>
<td>7</td>
<td>2006-11-16T17.10.43Z</td>
<td>submit 7</td>
<td>Jones, Richard</td>
</tr>
<tr>
<td>8</td>
<td>2006-11-16T17.11.08Z</td>
<td>submit 8</td>
<td>Jones, Richard</td>
</tr>
<tr>
<td>9</td>
<td>2006-11-16T17.11.30Z</td>
<td>submit 9</td>
<td>Jones, Richard</td>
</tr>
<tr>
<td>10</td>
<td>2006-11-16T17.11.56Z</td>
<td>submit 10</td>
<td>Jones, Richard</td>
</tr>
<tr>
<td>11</td>
<td>2006-11-16T17.12.19Z</td>
<td>submit 11</td>
<td>Jones, Richard</td>
</tr>
<tr>
<td>12</td>
<td>2006-11-16T17.12.45Z</td>
<td>submit 12</td>
<td>Jones, Richard</td>
</tr>
<tr>
<td>13</td>
<td>2006-11-16T17.13.09Z</td>
<td>submit 13</td>
<td>Jones, Richard</td>
</tr>
<tr>
<td>14</td>
<td>2006-11-16T17.13.31Z</td>
<td>submit 14</td>
<td>Jones, Richard</td>
</tr>
<tr>
<td>15</td>
<td>2006-11-16T17.13.52Z</td>
<td>submit 15</td>
<td>Jones, Richard</td>
</tr>
</tbody>
</table>

(A brief pause then ensued while I added some more information to my debug output so that I can quickly test that things are as they should be; below is the same debug, but with the extra data)
Success Metric: type = dateissued, order = ASC (by date issued), focus = 3, results per page = 12 (+1 as documented above), sort by = dateissued (the browse value)

browse?type=dateissued&order=DESC&focus=5&rpp=10&sort_by=1

Failure Analysis: This problem appears to be because the value obtained for the focus item (in this case item id 5) is the actual value in the desired index (dateissued), which in this case is "2006-11-16T17:09:52Z". The desired value for the focus value should, though, be the value for the focus item id in the relevant sort field of the desired index (in this case it should have been "submit 5").
BrowseInfo String Representation: Browsing 4 to 15 of 22 in index: dateissued(data type: date, display type: full) ||
Listing over 3 columns: dc.date.issued, dc.title.null, dc.contributor.* ||
Sorting by: dc.title DESC(option 1) ||
{{Item ID: 5 :: [dc.date.issued:2006-11-16T17.09.52Z][dc.title.null:submit 5][dc.contributor.*:Jones, Richard]}}
{{Item ID: 3 :: [dc.date.issued:2006-11-16T17.09.05Z][dc.title.null:submit 3][dc.contributor.*:Jones, Richard]}}
{{Item ID: 20 :: [dc.date.issued:2006-11-16T17.15.42Z][dc.title.null:submit 20][dc.contributor.*:Jones, Richard]}}
{{Item ID: 2 :: [dc.date.issued:2006-11-16T17.08.42Z][dc.title.null:submit 2][dc.contributor.*:Jones, Richard]}}

Success Metric: type = dateissued, order = DESC (by title), focus = 5, results per page = 10 (+1 as documented above), sort by = title (sort_1)

NOTE: it should only be possible to set the "value" variable from a browse page which is of type "single"

NOTE: it should only be possible to specify the "sort_by" variable from a browse page which is of type "full"

browse?type=author&order=ASC&rpp=25

BrowseInfo String Representation: Browsing 0 to 22 of 22 in index: author(data type: text, display type: single) ||
Failure Analysis: This is wrong in a fairly significant way, in that since there is only one author name "Jones, Richard", there should only be one result. It therefore needs a DISTINCT clause inserted into the SELECT in the condition that the display type is "single" and there is no value specified for the browse. In order to make testing this index possible I will also add a couple of items by other authors to the test set.

Single Browse vs Full Browse

The above testing has yielded a logical flaw in my reasoning. Inserting DISTINCT into the query is not straightforward, as this changes the very nature of what you are querying for. Code so far has assumed that it could obtain an item_id as a focus, but this falls down here. The old browse code dealt with this in a complex way, and the reason that it was impossible to generalise the old code, and thus necessitating the creation of this new code was because how the semantics of these two essentially different ways of making the browse function were blended. Fortunately, the approach I have adopted is sufficiently clear that it will be possible to build in a new mechanism for browsing in this second way that will not be a horrible confusion!

I propose, inside the BrowseEngine, to have an initial catch thus:
if (browseIndex.isSingle() && !scope.hasValue())
{
    browseByValue(scope);
}
else
{
    browseByItem(scope);
}

This will then allow us to keep all the logic separate. Obviously much of the supporting methods will work well in both context and can be reused.

The BrowseInfo object already supports two methods: getItemResults and getStringResults, which gives me a place in the existing code to hook the results of this functionality without too much work at that end. The SQL that we need to achieve is quite simple, and looks like this

    SELECT DISTINCT(value) FROM <index>
    [WHERE sort_value [<|=] <vfocus>]
    ORDER BY sort_value <order>
    LIMIT <rpp> + 1

Here we have introduced one new variable (which has been fed back to the earlier list of UI variables) called "vfocus", and which is the text value of the target focus. For pagination this is obtained by the +1 on the next page in the LIMIT portion of the query. More on this later.

Ongoing programmers notes

- The SQL appears to be coming together quite quickly
- We have a method called getFocusValue which is supposed to return a string value from the item id integer. It would be good if we can generalise this so that it dealt with both "focus" and "vfocus". I've added a vFocus member variable to the BrowserScope object. The next thing to do (probably not until Friday now) is track this back through to the Servlet to ensure that it gets properly populated by the UI
- The URL parameter "vfocus" has been added (and retrofitted to the list above)
- the vfocus parameter has been propagated from the URL to the browse engine, and the engine has been modified so that it should now be able to deal with both browsing by item or by value. Once this compiles, testing info to follow...
- Having forgotten how to write SELECT DISTINCT statements properly, we modify our SQL for value browsing to be:

    SELECT DISTINCT(value), sort_value
    FROM index_<n>
    [WHERE sort_value [<|=] <vfocus>]
    ORDER BY sort_value <order>
    LIMIT <rpp> + 1

- in order to make the BrowseInfo.toString method work with the new value browse, I need to go in and make some changes

Testing the Browse URL (part 2)

Having got through end-to-end on the first single value browse (author), we are ready to throw some more stuff at the browse engine and see how it copes. Let's start simple:

    browse?type=author

BrowseInfo String Representation: Browsing 0 to 1 of 22 in index: author(data type: text, display type: single)
**Failure Analysis:** while this looks correct, we can see that the count value 0 - 1 of 22 is incorrect. This is because of the following factors: the `BrowseEngine.getTotalResults` is not implementing `DISTINCT`, and because of the way that `BrowseInfo.toString` makes a minor error in calculating the range (although the actual range is correct). It is also displaying which columns it lists over, even though it will not do this. These to be fixed before the next test.

```
BrowseInfo String Representation: Browsing 1 to 1 of 1 in index: author(data type: text, display type: single)||
Listing single column: dc.contributor.*||
Sorting by: dc.contributor.* ASC(option 0)||
\( \\{ Value: Jones, Richard\}\)||```

**Success Metric:** type = author, 1 result

To be sure that this is working properly I will add more data to the system and run the same URL again

**Progress Update: 29-11-2006, 15:05 GMT**

While the single browse pages now look within reach, a new problem has emerged which means we cannot continue the above testing immediately. It appears that the submission process which requests the indexing of the item is broken. This is not surprising, since I've not put much work into that area. Nonetheless, I have written an indexer which should be re-usable, and I will therefore divert my attention to this for a short while so that I can then add more items to continue the above testing.

**Ongoing Programmer's Notes**

- The individual item indexing is currently achieved with `Browse.itemAdded`, `Browse.itemChanged` and `Browse.itemRemoved`. We will need to provide reasonable alternatives to these for the new indexer. I expect that `IndexBrowse.itemAdded`, `IndexBrowse.itemChanged`, `IndexBrowse.itemRemoved` will be the way to go. This will mean that we must modify the Item object to call that class instead.
- `Browse` is referenced from `Item.update`, `Item.withdraw` and `Item.delete`

**Progress Update: 30-11-2006, 12:55 GMT**

A new index process for individual items has been added and tested in the most basic way: an item has been added, and it has appeared in the index tables. This means that we can resume our primary path which is to add some more data to the browse tables and continue testing the URL

**Testing the Browse URL (part 3)**

```
browse?type=author
```

```
BrowseInfo String Representation: Browsing 1 to 6 of 6 in index: author(data type: text, display type: single)||
Listing single column: dc.contributor.*||
Sorting by: dc.contributor.* ASC(option 0)||
\( \\{Value: Ardman, Alfred\}\)||
\( \\{Value: Boothroyd, Betty\}\)||
\( \\{Value: Chaplin, Charlie\}\)||
\( \\{Value: Decimal, Dewey\}\)||
\( \\{Value: Eagle, Eddie\}\)||
\( \\{Value: Jones, Richard\}\)||```

**Success Metric:** single value browse, type-author, results=6, sort by name ASC

Now we can go on and push the browse URL a little further.

**Note:** I have gone back and re-run all the tests done above and they appear, without significant analysis, to be correct. I have also re-run the last URL which caused the problems before, and it produces exactly the same results as above, which is what we would expect.
Failure Analysis: **sigh.** Well this has behaved pretty much correctly, in so much as it starts with “Boothroyd, Betty” and works its way in descending order to the end of the list. Unfortunately, the “26 to 27” bit is a little out! This appears to be because the count prior to the current value is not employing a DISTINCT(value) section to the query. Therefore, the range is correct, and if the starting point were correct then everything would have gone as planned.

Success Metric: type=author, value focus = Boothroyd, Betty, sorted by value descending.

Success Metric: type=title, order = undeterminable for this set, as all the authors are the same, focus = 13, range is correct (+1 as above), rpp = 5 (+1). Sorting by author needs to be bourne out by other tests
<table>
<thead>
<tr>
<th>Item ID</th>
<th>dc.date.issued</th>
<th>dc.title.null</th>
<th>dc.contributor.*</th>
</tr>
</thead>
<tbody>
<tr>
<td>27</td>
<td>2006-11-30T12.56.42Z</td>
<td>Submit E</td>
<td>Eagle, Eddie</td>
</tr>
<tr>
<td>26</td>
<td>2006-11-30T12.56.19Z</td>
<td>Submit D</td>
<td>Decimal, Dewey</td>
</tr>
<tr>
<td>25</td>
<td>2006-11-30T12.55.58Z</td>
<td>Submit C</td>
<td>Chaplin, Charlie</td>
</tr>
<tr>
<td>24</td>
<td>2006-11-30T12.55.33Z</td>
<td>Submit B</td>
<td>Boothroyd, Betty</td>
</tr>
<tr>
<td>23</td>
<td>2006-11-30T12.49.29Z</td>
<td>Submit A</td>
<td>Ardman, Alfred</td>
</tr>
<tr>
<td>22</td>
<td>2006-11-16T17.17.242Z</td>
<td>Submit 22</td>
<td>Jones, Richard</td>
</tr>
<tr>
<td>21</td>
<td>2006-11-16T17.16.022Z</td>
<td>Submit 21</td>
<td>Jones, Richard</td>
</tr>
<tr>
<td>20</td>
<td>2006-11-16T17.15.422Z</td>
<td>Submit 20</td>
<td>Jones, Richard</td>
</tr>
<tr>
<td>19</td>
<td>2006-11-16T17.15.192Z</td>
<td>Submit 19</td>
<td>Jones, Richard</td>
</tr>
<tr>
<td>15</td>
<td>2006-11-16T17.13.522Z</td>
<td>Submit 15</td>
<td>Jones, Richard</td>
</tr>
<tr>
<td>14</td>
<td>2006-11-16T17.13.312Z</td>
<td>Submit 14</td>
<td>Jones, Richard</td>
</tr>
<tr>
<td>13</td>
<td>2006-11-16T17.13.092Z</td>
<td>Submit 13</td>
<td>Jones, Richard</td>
</tr>
<tr>
<td>12</td>
<td>2006-11-16T17.12.452Z</td>
<td>Submit 12</td>
<td>Jones, Richard</td>
</tr>
<tr>
<td>11</td>
<td>2006-11-16T17.12.192Z</td>
<td>Submit 11</td>
<td>Jones, Richard</td>
</tr>
<tr>
<td>10</td>
<td>2006-11-16T17.11.562Z</td>
<td>Submit 10</td>
<td>Jones, Richard</td>
</tr>
<tr>
<td>9</td>
<td>2006-11-16T17.11.302Z</td>
<td>Submit 9</td>
<td>Jones, Richard</td>
</tr>
<tr>
<td>8</td>
<td>2006-11-16T17.11.082Z</td>
<td>Submit 8</td>
<td>Jones, Richard</td>
</tr>
<tr>
<td>7</td>
<td>2006-11-16T17.10.432Z</td>
<td>Submit 7</td>
<td>Jones, Richard</td>
</tr>
<tr>
<td>6</td>
<td>2006-11-16T17.10.182Z</td>
<td>Submit 6</td>
<td>Jones, Richard</td>
</tr>
<tr>
<td>5</td>
<td>2006-11-16T17.09.522Z</td>
<td>Submit 5</td>
<td>Jones, Richard</td>
</tr>
<tr>
<td>4</td>
<td>2006-11-16T17.09.262Z</td>
<td>Submit 4</td>
<td>Jones, Richard</td>
</tr>
<tr>
<td>3</td>
<td>2006-11-16T17.09.052Z</td>
<td>Submit 3</td>
<td>Jones, Richard</td>
</tr>
<tr>
<td>2</td>
<td>2006-11-16T17.08.422Z</td>
<td>Submit 2</td>
<td>Jones, Richard</td>
</tr>
<tr>
<td>1</td>
<td>2006-11-16T17.08.112Z</td>
<td>Submit 1</td>
<td>Jones, Richard</td>
</tr>
</tbody>
</table>

Success Metric: type=title, order = date issued descending (option 2 was date issued not author, as I have not (and cannot) set an author sort option (see notes above)). less than 30 results are on the page, and this is all of them

browse?type=subject&amp;order=DESC&amp;rpp=10

BrowseInfo String Representation: Browsing 1 to 2 of 2 in index: subject(data type: text, display type: single)
Listing single column: dc.subject.*
Sorting by: dc.subject.* DESC

Success Metric: type=subject, order = subject descending, all values in results

browse?type=subject&order=ASC&rpp=10&vfocus=fdsa

BrowseInfo String Representation: Browsing 2 to 2 of 2 in index: subject(data type: text, display type: single)

Listing single column: dc.subject.*
Sorting by: dc.subject.* ASC

Success Metric: type=subject, order = subject ascending (even though there is only one result, because we focus on "fdsa" we know that it is sorting the right way), vfocus = fdsa (asdf would be number 1, which is not displayed)

Now we can continue to test the "second level browse" pages, which are Single/Value browses which have value parameters specified. This could be the source of some more bugs

browse?type=author&order=ASC&value=Jones%2C+Richard&rpp=10&sort_by=1

Failure Analysis: this URL returns a blank page. This appears to be caused by a problem with turning the BrowseInfo object into a String

java.lang.ClassCastException: org.dspace.browse.BrowseItem
    at org.dspace.browse.BrowseInfo.valueListingString(BrowseInfo.java:462)
    at org.dspace.browse.BrowseInfo.toString(BrowseInfo.java:336)

This is because there is a test BrowseIndex.isSingle which only remarks on the browse type - it does not consider whether we are at the top or second level of the browse. The BrowseInfo object needs to know whether it is doing top level or second level browsing, as does the BrowseScope object for other uses. I propose the addition of isTopLevel and isSecondLevel to both of these objects, and to have them populated by the BrowseServlet

Ongoing Programmer's Notes

* This means that things which were done previously in the BrowseEngine thus:

        if (browseIndex.isSingle() && scope.hasValue())

can now be done thus:

        if (scope.isSecondLevel())

Testing the Browse URL (part 4)

browse?type=author&order=ASC&value=Jones%2C+Richard&rpp=10&sort_by=1

BrowseInfo String Representation: Browsing 1 to 11 of 22 in index: author(data type: text, display type: single)
Success Metric: somewhat miraculous that there were no significant problems

tyype=author, level=2, sort by = title, ASC, value="Jones, Richard"

Failure Analysis: it appears that this browse has not used the "vfocus" variable in the SQL. This means that the SQL has not been written quite correctly. All other features of the browse have functioned correctly as far as I can tell.

Ongoing Programmer's Notes

- It seems the fix might be as simple as to ensure that the browseByItem method checks both for item id and string value focusses, as at the moment it doesn't.
  - yup, looks like it's done the job ...

Testing the Browse URL (part 5)
Failure Analysis: well, it appears to have selected the correct values, and the reason there aren't 6 (rpp + 1) is because it reached the end of the index for those parameters. They are in the correct order for the correct value, but now we are in the range 23 - 27 of 22, which is interesting. The range is correct, so only the start value is at fault. This looks like an application (or misapplication) of the DISTINCT SQL construct.

Success Metric: good news type=author, order = title descending (sorting by title), value = "Richard, Jones" (and it is known that at this point there are 22 items by that author), vfocus = "submit 13"

Success Metric: type=author, order = unknowable, but should be by date, value = "Eagle, Eddie" (and it is known that there is only 1 item by this author)
TODO: the results don’t yet tell us which value we are browsing on - this will need to be fixed for the real UI

Success Metric: type=subject, order = title descending, by value "asdf" (although you can’t see this - we know there are 22 items with that subject)

Browse URL Testing Summary

A summary of the URL parameters tested above

<table>
<thead>
<tr>
<th></th>
<th>test !! mode !! type !! order !! value !! focus !! vfocus !! rpp !! sort_by</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Full/Item</td>
</tr>
<tr>
<td>2</td>
<td>Full/Item</td>
</tr>
<tr>
<td>3</td>
<td>Single/Value</td>
</tr>
<tr>
<td>4</td>
<td>Single/Value</td>
</tr>
<tr>
<td>5</td>
<td>Full/Item</td>
</tr>
<tr>
<td>6</td>
<td>Full/Item</td>
</tr>
<tr>
<td>7</td>
<td>Single/Value</td>
</tr>
<tr>
<td>8</td>
<td>Single/Value</td>
</tr>
<tr>
<td>9</td>
<td>Single/Value with Value</td>
</tr>
<tr>
<td>10</td>
<td>Single/Value with Value</td>
</tr>
<tr>
<td>11</td>
<td>Single/Value with Value</td>
</tr>
<tr>
<td>12</td>
<td>Single/Value with Value</td>
</tr>
</tbody>
</table>

Progress Update: 30-11-2006, 16:50 GMT
We now have a Browse Engine which is capable of taking the core set of parameters that we might want to browse by, and turning them into meaningful results. We also have a BrowseInfo object which is capable of carrying all the information that will be required by the UI to render these into pages for the user. There are a few major things outstanding:

- The implementation of "starts with"
- The restriction to communities and collections
- Next and Previous page items or values (next can be got with the current code, but previous will require more work)
- the UI

These 4 items will be attacked in approximately that order now ...

Introducing "starts with"

"starts_with" is used through the user interface to indicate what value the search should look for strings starting with to display. For example:

\[
\text{starts_with=Jon}
\]

should match "Jones, Richard". This also works with dates, where

\[
\text{year=2006\&month=01}
\]

should be equivalent to:

\[
\text{starts_with=2006-01}
\]

and therefore will match all things published in the year 2006 in January

Converting "year" and "month" into "starts_with" will be done in the BrowseServlet. The impact that it will have on the SQL is to require us to stop using "= 'some value'" and start using "LIKE 'some value\%'" when a "starts_with" parameter is available. So the SQL will look like this:

```sql
SELECT * FROM <index>
WHERE sort_value <![CDATA[<[value] | _focus-value_] | LIKE <starts_with>% ]]
  [AND collection_id = _collection_] 
  [AND community_id = _community_]
ORDER BY <sortBy> <order>
LIMIT <resultsperpage + 1>
```

Note: this utilises the database's regular expression engine, which will have a performance impact. Therefore, we cannot always use LIKE for convenience, we must only use it when there is a "starts_with" parameter.

Important Note: the logic here has been refuted below - it is no longer necessary to worry about the regular expression features

Ongoing Programmer's Notes

"starts_with" is mutually exclusive of "focus" and "vfocus"

- "starts_with" is intrinsically linked to the "sort_by" field, in the same way that "focus" and "vfocus" are

Testing the Browse URL (part 6)

```url
browse?type=dateissued&order=ASC&rpp=10&sort_by=0&starts_with=2006-11
```

BrowseInfo String Representation: Browsing 1 to 11 of 27 in index: dateissued(data type: date, display type: full)
Listing over 3 columns: dc.date.issued, dc.title.null, dc.contributor.* starting with value: 2006-11||
Sorting by: dc.date.issued ASC(option 0)||
{{Item ID: 1 :: [dc.date.issued:2006-11-16T17.08.11Z][dc.title.null:Submit 1][dc.contributor.*:Jones, Richard]}}
{{Item ID: 2 :: [dc.date.issued:2006-11-16T17.08.42Z][dc.title.null:Submit 2][dc.contributor.*:Jones, Richard]}}
{{Item ID: 3 :: [dc.date.issued:2006-11-16T17.09.05Z][dc.title.null:Submit 3][dc.contributor.*:Jones, Richard]}}
{{Item ID: 5 :: [dc.date.issued:2006-11-16T17.09.52Z][dc.title.null:Submit 5][dc.contributor.*:Jones, Richard]}}
{{Item ID: 6 :: [dc.date.issued:2006-11-16T17.10.18Z][dc.title.null:Submit 6][dc.contributor.*:Jones, Richard]}}
{{Item ID: 7 :: [dc.date.issued:2006-11-16T17.10.43Z][dc.title.null:Submit 7][dc.contributor.*:Jones, Richard]}}
{{Item ID: 8 :: [dc.date.issued:2006-11-16T17.11.08Z][dc.title.null:Submit 8][dc.contributor.*:Jones, Richard]}}
{{Item ID: 9 :: [dc.date.issued:2006-11-16T17.11.30Z][dc.title.null:Submit 9][dc.contributor.*:Jones, Richard]}}
{{Item ID: 10 :: [dc.date.issued:2006-11-16T17.11.56Z][dc.title.null:Submit 10][dc.contributor.*:Jones, Richard]}}

Success Metric: type=dateissued, order = date issued ascending, rpp= 10 (+1), starts with "2006-11"; we know that there are 27 items in the database, and they were all entered in November 2006

browse?type=dateissued&order=DESC&rpp=10&sort_by=0&starts_with=2006-11-30

BrowseInfo String Representation: Browsing 6 to 10 of 27 in index: dateissued(data type: date, display type: full)||

Listing over 3 columns: dc.date.issued, dc.title.null, dc.contributor.* starting with value: 2006-11-30||
Sorting by: dc.date.issued DESC(option 0)||
{{Item ID: 27 :: [dc.date.issued:2006-11-30T12.56.42Z][dc.title.null:Submit E][dc.contributor.*:Eagle, Eddie]}}
{{Item ID: 24 :: [dc.date.issued:2006-11-30T12.55.33Z][dc.title.null:Submit B][dc.contributor.*:Boothroyd, Betty]}}
{{Item ID: 23 :: [dc.date.issued:2006-11-30T12.49.29Z][dc.title.null:Submit A][dc.contributor.*:Ardman, Alfred]}}||

Failure Analysis: the perplexing thing about this is that only the results that actually start with the value are supplied. A moment examining the SQL we wrote shows us that we were over-zealous in our logic. There is no need to invoke the regular expression engine, it is simply enough to supply the "starts_with" parameter in place of the "focus" or "vfocus" in the query, thus:

```
SELECT * FROM <index>
WHERE sort_value [<=] [value] [starts_with_]
[AND collection_id = _collection_]
[AND community_id = _community_]
ORDER BY <sortBy> <order>
LIMIT <resultsperpage + 1>
```

The code update to achieve this now gives us this result:
Failure Analysis: this is not as wrong as it looks. It has been produced as an oddity in the way that string comparisons are dealt with in the database. The query which generated this looks like this:

```
SELECT * FROM index_1 WHERE sort_value <= '2006-11-30' ORDER BY sort_value DESC LIMIT 11
```

which will actually only match everything after 2006-11-30, not including. The original Browse code notes this problem as follows:

```c
/*
 * When the user is browsing with the most recent items first,
 * the browse code algorithm doesn't quite do what some people
 * might expect. For example, if in the index there are entries:
 * and the user has selected "Feb 2000" as the start point for
 * the browse, the browse algorithm will start at the first
 * point in that index *after* "Feb 2000". "Feb 2000" would
 * So, the browse code in this case will start the browse at
 * "15-Jan-2000". This isn't really what users are likely to
 * want: They're more likely to want the browse to start at the
 * first Feb 2000 date, i.e. 15-Feb-2000. A similar scenario
 * occurs when the user enters just a year. Our quick hack to
 * produce this behaviour is to add "-32" to the startsWith
 * variable, when sorting with most recent items first. This
 * means the browse code starts at the topmost item in the index
 * that matches the user's input, rather than the point in the
 * index where the user's input would appear.
 */
```

We will adopt the same approach for the new browse code. This means that we must abandon this particular test (see note below).

Note: there is an implied limit to the functionality of the Browse Engine here. All string comparison problems for dates are overcome by the application of "-32" to the end of the string. This works perfectly well for years without months and months without days (because there are less than 32 in both), but does not work at the days level (that is, if starts_with=2006-11-30, then the query will be on a date before "2006-11-30-32", which does not successfully compare to a date of the form "2006-11-30T17:24Z"
browse?type=author&order=ASC&rpp=10&sort_by=0&starts_with=Jon

BrowseInfo String Representation: Browsing 6 to 6 of 6 in index: author(data type: text, display type: single)

Listing single column: dc.contributor.* starting with value: Jon
Sorting by: dc.contributor.* ASC(option 0)
{ {Value: Jones, Richard} }

Success Metric: type=author, sort by = author ascending (otherwise, Jones, Richard wouldn't be last). Correct range and everything.

browse?type=title&order=DESC&rpp=10&sort_by=0&starts_with=submit+2

BrowseInfo String Representation: Browsing 16 to 26 of 27 in index: title(data type: title, display type: full)

Listing over 3 columns: dc.date.issued,dc.title.null,dc.contributor.* starting with value: submit 2
Sorting by: dc.title DESC(option 0)
{{Item ID: 2 :: [dc.date.issued:2006-11-16T17.08.42Z][dc.title.null:submit 2][dc.contributor.*:Jones, Richard]}}
{{Item ID: 10 :: [dc.date.issued:2006-11-16T17.11.56Z][dc.title.null:submit 10][dc.contributor.*:Jones, Richard]}}

Success Metric: type=title, sort by = title descending, staring with the last submit+2 value when descending

Note: using "starts_with" with an order=DESC is an odd thing to do, and produces the above technically correct, but slightly misleading result. Worth mentioning.

browse?type=subject&order=ASC&rpp=10&sort_by=0&starts_with=a

BrowseInfo String Representation: Browsing 1 to 2 of 2 in index: subject(data type: text, display type: single)

Listing single column: dc.subject.* starting with value: a
Sorting by: dc.subject.* ASC(option 0)
{ {Value: asdf} }
{ {Value: fdsa} }

Success Metric: type=subject, sort by = subject ascending, starting with a
Success Metric: type=author value = "jones, richard", sort by = title, starting with submit 2. This is true but misleading; it would be better if the value was reported as "Jones, Richard" and the starts_with reported as "starting with focus"

Testing Browse URL Summary (part 2)

<table>
<thead>
<tr>
<th>test</th>
<th>mode</th>
<th>type</th>
<th>order</th>
<th>value</th>
<th>focus</th>
<th>vfocus</th>
<th>rpp</th>
<th>sort_by</th>
<th>starts_with</th>
</tr>
</thead>
<tbody>
<tr>
<td>13</td>
<td>Full/Item</td>
<td>dateissued</td>
<td>ASC</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>10</td>
<td>0</td>
<td>2006-11</td>
</tr>
<tr>
<td>14</td>
<td>Single/Value</td>
<td>author</td>
<td>ASC</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>10</td>
<td>0</td>
<td>Jon</td>
</tr>
<tr>
<td>15</td>
<td>Full/Item</td>
<td>title</td>
<td>DESC</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>10</td>
<td>0</td>
<td>submit 2</td>
</tr>
<tr>
<td>16</td>
<td>Single/Value</td>
<td>subject</td>
<td>ASC</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>10</td>
<td>0</td>
<td>a</td>
</tr>
<tr>
<td>17</td>
<td>Value</td>
<td>author</td>
<td>ASC</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>10</td>
<td>1</td>
<td>submit 2</td>
</tr>
</tbody>
</table>

Introducing restriction to Community or Collection

Community and Collection data can be obtained from the URL, when the browse URL is of the form:
Where 123456789/4321 is the handle of the community or collection to be browsed in.

When we are inside a community or collection the browse must be done on one of the views created on the data which lists browse results by collection id. This means that we can construct SQL queries of the form:

```sql
SELECT * FROM [community|collection]_<index>
WHERE [collection_id|community_id] = [<collection>|_community_]
    AND sort_value ['|='] [<value> | _starts_with_]
ORDER BY <sortBy> <order>
LIMIT <rpp> + 1
```

To achieve this we need to place the community or collection object into the BrowseScope to go into the BrowseEngine. The engine can then construct the query in the same way for both item and value browses, for example:

```java
if (scope.isCollection())
{
    table = browseIndex.getTableName(false, true);
}
else if (scope.isCommunity())
{
    ....
}
```

To obtain the table name, and similarly to construct the relevant segment of the WHERE clause. The community or collection then needs to be passed back into the BrowseInfo object so that it can report on the scope of the browse.

### Ongoing Programmer's Notes

- At home time, you had implemented the Servlet end of things for taking the collection or community, and added them to the BrowseScope object, with the relevant accessors. Next, implement in the BrowseEngine.
- Looks like I’ve been excessive with my application of the constraints to community and collection. Probably this is just a missing check in the BrowseEngine, no biggie:
  - The container must be a community or a collection

```java
h4. h4. h4. h4. h2. h1. org.dspace.browse.BrowseException: The container must be a community or a collection
at org.dspace.browse.BrowseInfo.setBrowseContainer(BrowseInfo.java:167)
at org.dspace.browse.BrowseEngine.browseByItem(BrowseEngine.java:242)
at org.dspace.browse.BrowseEngine.browse(BrowseEngine.java:448)
```

- Aside from the fact that the BrowseInfo reports browsing NOT in a community or collection as being in an Invalid Container, the very first primitive tests suggest that the application of the constraint code has not immediately broken anything else. Always a good start!
- The very first tentative test of the constraint code indicates that while the code is working, it appears to have missed the 5 items submitted later on. Since they should be in the same collection, there must be some sort of problem ... investigating.
  - Actually, there appears to be a problem with the constraining process, so that constraining to collection does not take effect
    - The reason for this is that the bit of code which tells which container you are in is clever enough to lift out the community /and/ the collection if you are in a collection. I'm not sure what happens when you are in a stack of communities. Perhaps I need to modify the code slightly to eliminate this danger.
    - This all appears to happen somewhere high up the stack, possibly in the DSpaceServlet (I've decided not to track it any further). Anyway, the logic just needs to figure out that if we are in a collection then it doesn't need to bother with the community.
  - With the above logic implemented, the collection browse functions apparently correctly (tests still to be fully carried out). Unfortunately, it still doesn't appear to work correctly for the community
    - The table community_index_3 is a view on index_3 with additional community information. This table really doesn't contain the values that we want to see in the browse, so the problem is most likely in the indexing process itself (or, more likely still, in the community2item table)
    - The problem is that there is one table called "communities2item" which contains only 22 of the 27 current records, and another called "community2item" which contains all the 27 records, and is a view on the data. The question is, why are there two so similarly named tables, and which one should we really be using, and why do they contain (slightly) different data.
      - The "communities2item" table was used by the previous browse code, and was not being updated by the new indexer.
      - I have decided to stick with the existing view "community2item" instead. With that change to the community_index_<n> views, everything appears to be selected. Next, on to test the new functionality...
It looks as though the Indexer might not be dropping old tables and views. Not quite sure why yet - one problem at a time.

Testing the Browse URL (part 7)

In order to successfully test the constraining code is is necessary to do the following things first:

- Create more top level communities
- Create some second level communities
- Create more collections as the second and third levels
- Add more items to the system ensuring there are results for all collections
- map items into more than one collection.

```
handle/123456789/2/browse?type=dateissued&amp;order=ASC&amp;rpp=10&amp;sort_by=0
```

Failure Analysis: Although the result set appears to be correct, there are not 40 elements in this collection (which is the correct collection). This is evidently a missing statement in the WHERE clause of the count mechanism.

```
Browsing in collection: 1 (123456789/2)
Listing over 3 columns: dc.date.issued, dc.title.null, dc.contributor.* on value: null
Sorting by: dc.date.issued ASC

{{Item ID: 1 :: [dc.date.issued:2006-11-16T17.08.11Z][dc.title.null:Submit 1][dc.contributor.*:Jones, Richard]}}
{{Item ID: 2 :: [dc.date.issued:2006-11-16T17.08.42Z][dc.title.null:submit 2][dc.contributor.*:Jones, Richard]}}
{{Item ID: 3 :: [dc.date.issued:2006-11-16T17.09.05Z][dc.title.null:submit 3][dc.contributor.*:Jones, Richard]}}
{{Item ID: 5 :: [dc.date.issued:2006-11-16T17.09.52Z][dc.title.null:submit 5][dc.contributor.*:Jones, Richard]}}
{{Item ID: 6 :: [dc.date.issued:2006-11-16T17.10.18Z][dc.title.null:submit 6][dc.contributor.*:Jones, Richard]}}
{{Item ID: 7 :: [dc.date.issued:2006-11-16T17.10.43Z][dc.title.null:submit 7][dc.contributor.*:Jones, Richard]}}
{{Item ID: 8 :: [dc.date.issued:2006-11-16T17.11.08Z][dc.title.null:submit 8][dc.contributor.*:Jones, Richard]}}
{{Item ID: 9 :: [dc.date.issued:2006-11-16T17.11.30Z][dc.title.null:submit 9][dc.contributor.*:Jones, Richard]}}
{{Item ID: 10 :: [dc.date.issued:2006-11-16T17.11.56Z][dc.title.null:submit 10][dc.contributor.*:Jones, Richard]}}
```
BrowseInfo String Representation: Browsing 1 to 11 of 27 in index: dateissued (data type: date, display type: full) ||
Browsing in collection: 1 (123456789/2)||
Listing over 3 columns: dc.date.issued,dc.title.null,dc.contributor.* on value: null||
Sorting by: dc.date.issued ASC(option 0)||
{{Item ID: 1 :: [dc.date.issued:2006-11-16T17.08.11Z][dc.title.null:Submit 1][dc.contributor.*:Jones, Richard]}}
{{Item ID: 2 :: [dc.date.issued:2006-11-16T17.08.42Z][dc.title.null:submit 2][dc.contributor.*:Jones, Richard]}}
{{Item ID: 3 :: [dc.date.issued:2006-11-16T17.09.05Z][dc.title.null:submit 3][dc.contributor.*:Jones, Richard]}}
{{Item ID: 5 :: [dc.date.issued:2006-11-16T17.09.52Z][dc.title.null:submit 5][dc.contributor.*:Jones, Richard]}}
{{Item ID: 6 :: [dc.date.issued:2006-11-16T17.10.18Z][dc.title.null:submit 6][dc.contributor.*:Jones, Richard]}}
{{Item ID: 7 :: [dc.date.issued:2006-11-16T17.10.43Z][dc.title.null:submit 7][dc.contributor.*:Jones, Richard]}}
{{Item ID: 8 :: [dc.date.issued:2006-11-16T17.11.08Z][dc.title.null:submit 8][dc.contributor.*:Jones, Richard]}}
{{Item ID: 9 :: [dc.date.issued:2006-11-16T17.11.30Z][dc.title.null:submit 9][dc.contributor.*:Jones, Richard]}}
{{Item ID: 10 :: [dc.date.issued:2006-11-16T17.11.56Z][dc.title.null:submit 10][dc.contributor.*:Jones, Richard]}}
Success Metric: right number of results, range, total, right collection, ordered by date ascending

handle/123456789/32/browse?type=title&amp;order=DESC&amp;rpp=5&amp;sort_by=0

BrowseInfo String Representation: Browsing 1 to 3 of 3 in index: title (data type: title, display type: full) ||
Browsing in collection: 2 (123456789/32)||
Listing over 3 columns: dc.date.issued,dc.title.null,dc.contributor.* on value: null||
Sorting by: dc.title DESC(option 0)||
{{Item ID: 29 :: [dc.date.issued:2006-12-04T11.55.38Z][dc.title.null:Submit G][dc.contributor.*:Garrison, Gertrude]}}
{{Item ID: 28 :: [dc.date.issued:2006-12-04T11.55.08Z][dc.title.null:Submit F][dc.contributor.*:Frankfurt, Freddie]}}
{{Item ID: 1 :: [dc.date.issued:2006-11-16T17.08.11Z][dc.title.null:Submit 1][dc.contributor.*:Jones, Richard]}}||
Success Metric: correct range and result count, sorted by title descending, limited to correct collection

handle/123456789/34/browse?type=author&amp;order=ASC&amp;rpp=5&amp;sort_by=0

BrowseInfo String Representation: Browsing 1 to 3 of 3 in index: author (data type: text, display type: single) ||
Browsing in collection: 3 (123456789/34)||
Listing single column: dc.contributor.* on value: null||
Sorting by: dc.contributor.* ASC(option 0)||
\( \{\{Value: Harrison, Harry\}\}\)
\( \{\{Value: Ianson, Irene\}\}\)
\( \{\{Value: Jones, Richard\}\}\)||
Success Metric: correct range and result count and results, sorted by contributor ascending, limited to correct collection
Success Metric: no results, as expected (no items in this collection have subjects)

Success Metric: correct range and result count, starting from item id 20, sorted by title descending, limited to correct community

NOTE: during testing it became clear that the browse URL not only needs to be validated for structure, but also that sort_by (for one example) is sensitive to the value passed, therefore we need to build in value validation also

Success Metric: correct range and result count, starting from item id 30, sorted by date issued ascending, limited to correct community

Success Metric: correct range and result count, starting from item id 30, sorted by date issued ascending, limited to correct community
Success Metric: correct range and result count, displaying only author "Jones, Richard", focussing on value submit 2 (this doesn't exist in the list, so the result set is correct), sorted by date issued descending, limited to correct community. It is interesting to note that these are both "mapped" items.

BrowsInfo String Representation: Browsing 1 to 2 of 2 in index: subject (data type: text, display type: single) ||

Success Metric: correct range and result set, displaying only items (mapped in) with the subject asdf (although for some reason the BrowsInfo is not reporting on the value), sorted by subject ascending (doesn't really make much sense)

Testing Browse URL Summary (part 3)

<table>
<thead>
<tr>
<th>test !! mode</th>
<th>type</th>
<th>order</th>
<th>value</th>
<th>focus</th>
<th>vfocus</th>
<th>rpp</th>
<th>sort_by</th>
<th>starts_with</th>
</tr>
</thead>
<tbody>
<tr>
<td>18</td>
<td>Full/Item</td>
<td>dateissued</td>
<td>ASC</td>
<td>N/A</td>
<td>-</td>
<td>-</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>/2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>Full/Item</td>
<td>title</td>
<td>DESC</td>
<td>N/A</td>
<td>-</td>
<td>-</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>/32</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>Single/Value</td>
<td>author</td>
<td>ASC</td>
<td>N/A</td>
<td>-</td>
<td>-</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>/34</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>Single/Value</td>
<td>subject</td>
<td>DESC</td>
<td>N/A</td>
<td>-</td>
<td>-</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>/38</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>Full/Item</td>
<td>dateissued</td>
<td>DESC</td>
<td>N/A</td>
<td>20</td>
<td>-</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>/A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>Full/Item</td>
<td>title</td>
<td>ASC</td>
<td>N/A</td>
<td>30</td>
<td>-</td>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>/A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>Value</td>
<td>author</td>
<td>DESC</td>
<td>Jones, Richard</td>
<td>-</td>
<td>submit 2</td>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>/A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>Value</td>
<td>subject</td>
<td>ASC</td>
<td>asdf</td>
<td>-</td>
<td>-</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>/A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Progress Update: 04-12-2006, 13:15 GMT

With the above coding and testing complete that means that our Browse URL API is complete and correct as far as we can tell and test. We now must turn our attention to the next and previous links that the UI will need to render. It also means that we are tantalisingly close to a workable browse system.

There are some other things that we might want to consider after this:

- Browse Caching
- More indexes on the browse tables, depending on performance

Introducing a Next Button

Introducing the next button ought to be easy, as we have already made provisions for it. We will simply strip the last result off the result set and set that as the target of the next button.

Introducing a Previous Button

The previous button is going to be slightly more complicated, but basically we have to do the same query as the main SELECT, but in reverse, so that we can get the value which will be top of the previous page. It ought to be possible to simply flip the comparison operators, and perhaps attach an OFFSET clause to the query, so as to only refer to a single value result.

Ongoing Programmer's Notes

The first pass on the next page stuff looks promising. The only obvious thing it is doing wrong is*always* stripping the last value off the results, even if it isn't supposed to.

- OK, that minor problem is fixed.
- It looks like it will be quite straightforward in concept to get the "previous" value. Nonetheless, the code might benefit from some refactoring before we try it, as there is a lot of reuse which is not taken advantage of yet.
- The refactoring has taken the shape of creating a new class to represent the browse SQL query. This class is populated by the relevant values, and then assembles a query from those values on request (as opposed to the un-refactored version, which assembled the query as it went along). This will be useful for obtaining the "previous" value, because we can simply flip the ordering of the query via the API, and regenerate it. Otherwise it means reassembling the inline built query with only one section changed, which would be the wrong thing to do. About to test the new code, to be sure that we haven't broken anything.
- During testing the refactored code, a problem has turned up with the BrowseEngine.getPosition method. It looks as thought this problem has always been there, and has simply gone unnoticed until this point. The problem is that the SQL which determines the current position of the start of the browse doesn't appear to return the true results for specific value browses. It is performing a SELECT DISTINCT where it shouldn't be. Investigating ...

Obtaining the current position

A flaw has been found in the code that determines the current position of the first item to be displayed. This arises because the code which generates the query does not take into account the value being browsed on, and does not correctly negate the SQL query to obtain the relevant position. For example, the following SQL generates a valid result set:

```sql
SELECT * FROM index_2 WHERE sort_1 <= 'submit 13' AND sort_value = 'jones, richard' ORDER BY sort_1 DESC LIMIT 6;
```

The query to obtain the position of the start pointer generated is as follows:

```sql
SELECT COUNT(DISTINCT(value)) AS number FROM index_2 WHERE sort_1 > 'submit 13'
```

This is incorrect. Instead the query ought to read:

```sql
SELECT COUNT(*) FROM index_2 WHERE sort_1 > 'submit 13' AND sort_value = 'jones, richard';
```

Note here how although the direction of the comparator is correct, no sort_value was specified in the original query, and it also does a SELECT DISTINCT with insufficient cause. Some modification to the browse engine will fix this reasonably quickly.

- getPosition only needs to select distinct when in a value based top-level browse
NOTE: There is a problem (around line 198) where value and focus are being conflated for the UI - this is why there is a problem displaying the value of a browse

- This problem now appears to be fixed. It would certainly benefit from being refactored into the BrowseQuery class, which will make the whole engine a lot easier to look at. Meanwhile, back to testing the refactoring ...

Ongoing programmer notes

- arg. Now there is a problem with "starts_with" not being included in the query. Hopefully this is just an oversight in the refactoring.
  - Yup, this was just a typo that occurred during the refactoring
- There's still a misunderstanding between value and focus in the BrowseInfo object. This needs to be looked at now in case it becomes problematic later.

NOTE: we really need to do something at some point about what happens when there are no results. It's not causing any actual problems, so not yet.

Progress Update: 04-12-2006, 17:30 GMT

The code has been refactored to include a class to manage just the SQL. It is my hope that this class can handle construction of all the SQL required by the browse engine, and therefore will be made into a pluggable class which will allow a similar class for Oracle support to be created at a later date. This brief round of refactoring was to support the facility to turn the SQL query around easily so as we can obtain the top value of the previous page quickly and easily.

Meanwhile, another thought occurs for the todo list:

Proper internationalisation. The browse code doesn't support sorting by non-latin characters, but provided this functionality can be pushed to the database layer (*fingers crossed*), then a plugin class which is loaded and applied as the normaliser (in replacement for the current NormalizeTitle class) would enable a stack of plugins that know how to normalise sorting for multiple languages.

Yet More Programmer Notes

- For the record, I originally thought that I could use LIMIT 1 OFFSET <rpp> - 1, but now that I come to look at it, I realise of course that if there are fewer records on the previous page than a full page (which is allowable), then the query won't work, so instead we just carry on with a straightforward LIMIT <rpp> on the "previous" query.
- Not forgetting, of course, that if there is no focus for the query, then we must be on page 1.
- First (trivial) test of the previous page code has executed successfully. Testing against all previous browse URL tests.
- Although we have run all the browse tests again, we have not posted here the results. For example, this is the sort of thing that we are now seeing reported by the BrowseInfo object:

  ```
  browse?type=author&order=ASC&rpp=10&sort_by=1&starts_with=submit+2&value=Jones%2CRichard
  ```

BrowseInfo String Representation: Browsing 12 to 21 of 22 in index: author (data type: text, display type: single) ||
As you can see, it now presents the next and previous page values, and the ones reported above are correct.

- All previous URL tests have been repeated, and with superficial examination are correct for all parameters

**User Interface**

OK, so now we have enough confidence to invest some time building the UI for this browse code. We have already attacked the logic of the display to generate the strings used for debug above, so it will be a matter of applying that logic to a nicer interface, with a variety of widgets. There is no need to significantly redesign the Browse UI, so the following outlines are almost identical except in some minor ways, to the existing interface. Single/Value Browse:

When we are looking at a single value browse, such as "by author", the page would look as follows:

![Single/Value Browse](image)

Full/Item & Value Browse:

When looking at a list of actual items, such as "by title" or we are looking at a list of items associated with a value browse such as "items by author X", the page would look as follows:

![Full/Item & Value Browse](image)
Still More Programmer's Notes

- To Revisit: there is going to be some confusion over which navigation to use when rendering the full index pages, because the user could be sorting by a value other than the one which is indexed on.
- Although the pages are much the same, I am going to take this opportunity to write most of the UI from scratch again, as it is currently a mess.
- On reflection, placing the Results/Page selector on the top right is going to make FORM layout interesting. Instead, I've decided to place that selector alongside the Sort and Order boxes, and have that as one of the defined functional units of the page.

**NOTE**: we have had to disable browsing with thumbnails turned on, because this is not covered by the functionality supported by the current version of the Browseltem.

**NOTE**: we have also abandoned the highlight row, and I'm just going to hack in the emphasised column. Might not bother putting the highlight row back in.

Additional: I have NOT hacked back in the emphasised column - it turns out to be quite tricky; leaving for later.

- The next and previous links are going to require only a subset of all the actual parameters that can be passed into the browse. They should both be of the form:
Woohoo! We can now see the next and previous links operating against a live browse listing. Time to call it a day.

For the Record: this amounts now to five long days of development work, excluding prior thinking time and initial work on the browse index. This is noted so that I can try to improve my ability to predict how long this sort of work will take!

The next victory has been won: we can now control the browse in sort order, direction and results per page using the latest page widget. Next up are the skip to and date entry stuff, which will be ripped almost wholesale from the old browse code.

BUG: Using the "starts_with" navigation, if there are no items which start with the given value, then you get a "no results" page. This sort of makes sense, and sort of doesn't. It ought to display the last page of the browse, I suppose.

Testing the Browse UI

Now we have what is basically a functional Browse UI for Full/Item browse. Before moving on to make this work for Single/Value browses, we want to make sure that our widgets are working properly.

1. run all our previously tested Full/Item URLs through the system to see if the UI reflects what we expect to see.
2. Navigate our way through the browse using previous and next buttons to ensure it behaves correctly.
3. On various pages from the start to the finish of the listing, modify each of the 3 parameters: sort_by, order, rpp to ensure that it behaves as expected.
4. Try all of the possible starts with options available for string search.
5. Try all of the date options available for jumping to date browse.
6. On pages navigated to through the starts with interface, try using previous, next, and modifying the 3 parameters: sort_by, order, rpp to ensure that it behaves as expected.

Test 1: previous browse URLs

I ran the previous browse URLs for Full/Item and Item/Value browse, with a reasonable degree of success. The following issues arose:

NOTE: The Results/Page parameter needs to be divisible by 5 for the UI to make sense of it. If it doesn't, the results are still correct, but the Results/Page modifier menu doesn't pick it up, because it only counts in 5. I'm not going to fix this - someone else can if they really care, but it is not a problem.

NOTE: Browsing by date issued and sorting by title doesn't make much sense, because the later is Browsing by title. Vice versa is also true, and both of these options are possible. I propose to remove the sort_by option box on pages like this.

BUG: starts_with for 2006-11 order ASC gives everything after 2006-11, NOT inclusive of it. This isn't really correct (certainly not expected behaviour. It ought to start with the beginning of Nov 2006, not Dec 2006 as it does. This was using the query:

browse?type=dateissued&order=ASC&rpp=10&sort_by=0&starts_with=2006-11

This problem is actually related to the way that year and month become starts_with. In order to solve this problem we must subtract 1 from the month number when order is ASC and leave it as-is when order is DESC.

UI ISSUE: the layout of the page header could be better. Wrap quote marks round the collection or community name where appropriate for starters.

BUG: "next" (and probably "previous") links are broken for community and collectio browse (missing /) Test 2: previous and next buttonsh2. On a wide selection of previously tested URLs, I tried out the next and previous buttons to see how they cope.

All tests were successful.

UI ISSUE: for consistency of style, the <value> parameter in the header should be surrounded by quote marks just as the collection or community is. Test 3: sort_by, order and rpp2. BUG: there is a problem with the URL encoding of the "value" field with these 3 parameters' widget. "Jones%2C+Richard" in the original URL becomes "Jones%252C%2BRichard" in the new URL.

This was just due to use of URLEncoder.encode inside a form input value, which was then double encoding the value on submit.

UI ISSUE: Changing the way you browse part way through a listing can have unusual results. For example, if you are in the middle of the result set, and you choose to change any of the three parameters we are testing, it keeps the current top item as the focus, and turns the results around about that focus. If, though, you are on page 1 or the last page, a change of ordering can leave you with just the first/last result on a page on its own, with all other results behind it. Although this is "correct" behaviour, it is totally unintuitive, and weird.

On discussion with the DSpace IRC channel, and also referencing how Amazon does its browse re-ordering, I have decided that this behaviour is too weird, and that any change of these 3 parameters should therefore reset you to page 1 of the browse, with the new parameters applied.

BUG: the form action is also wrong, as above for "next" links, in the constraints of community and collection Test 4: starts_with2. UI ISSUE: once you have entered a date or value to select on, it is not reflected when the page loads (i.e. it does not remain in the search boxes, etc). QUESTION - does that really matter? I'm not going to do anything about it, as this is the same as current DSpace functionality.
UI ISSUE: Choosing your order of search and choosing your focus point (i.e. ascending from 2006-11) are two separate requests. Due to the changes we made above (test 3, UI ISSUE) you have to choose your order first, and then your jump-to point. I'm going to leave this issue to simmer for a while, and see if anyone has any preferences as to a solution.

BUG: a starts_with request that yields no results in a browse listing which has results generates a "no results in index" page. This is technically correct if this were a search, but it isn't, and instead the browse should leap to the nearest point. It might be necessary to write some "special case" code to lift the last <rpp> results out of the database and display them.

- The original browse code deals with this by simply displaying the results of a query it calls Browse.getResultsBeforeFocus. We can therefore achieve the effect we desire by setting the result set obtained in the BrowseEngine.getPrevious method. This means re-coding the engine so that it will keep this result set if we ask it to.
  - Not forgetting of course to:
    - Get the results sorted correctly (since getPrevious obtains them backwards)
    - double the lookback of getPrevious where it is capturing results, so that it can still supply us with a "previous" link
  - well, I have fixed up this bit, so we do at least get results, but now I am looking at Results 37 to 56 of 36, which is probably wrong! There needs to be some sort of adjustment applied to the position parameter in the case that there are no results - this can be applied after the fact, so shouldn't be a hassle
  - OK, this is fixed for Full/Item and Full/Value browse. Applying the fix, to be tested properly later, to the Single/Value browse

Test 5: date jump toh2. Although we really did this as part of the above, some further attention is warranted.

BUG: when the conditions are that year, month and value are supplied for a value browse, it claims to be sorting by date, but then presents the text based browse jump to navigation

- This is the tip of an iceberg. The bug is due to the fact that the following expression does not evaluate to true:

```java
if (sortOrder.isDate() || (browseIndex.isDate() && sortOrder.isDefault()))
```

We have not, according to this, selected a sortOrder that works by date, even though we are jumping to a date browse. The broken bit is in two bits, depending on how you look at it.

- The trivial way this is broken is that the UI should enforce a sort_by parameter on the value browse. If sort_by = 2 is appended to the URL for a value browse, then it evaluates correctly. The error arises because there is no sort_by specified. See the next point
- Deep down, the following SQL is executed:

```sql
SELECT * FROM index_2 WHERE sort_value >= '2006-10' AND sort_value = 'jones, richard' ORDER BY sort_value ASC LIMIT 21
```

As you can see, it is asking two mutually exclusive (in terms of our semantics) things of the sort_value column. This is the symptom of there being no sort_by order specified, and the engine defaulting to the sort_value column

- What is the solution?
  - The quick way of doing this (and which will actually be quite reliable) is to default the sort_by for value browses
  - Fixed

BUG: when starts_with is submitted at the same time as year and month, starts_with loses its value and becomes -1. This is a bug in the servlet, and can be quickly rectified

BUG: the date jump to does not remember your supplied values for sort_by, order or rpp. This just needs some hidden fields added to the form. Incidentally, another check indicates that the form on the text value navigation does the same thing

fixedTest 6: all combinationsh2. My test strategy here was fairly childish - I just clicked around quickly, jumping through the browse in a variety of ways. There were no discernable problems.

Progress Update: 07-12-2006, 17:30 GMT

The browse now looks fully functional at least from the Full/Item and Full/Value browse pages. The next thing to do is the Single/Value browse pages, which should be much simpler now that we have the full listings developed (in fact, it will utilise a lot of the same components). There are still some things to deal with imminently and then in the long run. Something which has been bugging me for a while is that the BrowseItem object does not support handles yet (which is a failing), so the item titles don't yet link through to the items they represent. Once those two things have been dealt with, though, we will just have the following list of things to think about:

- Further testing and checking (and adding of verbose mode) to the indexing process
- Performance testing the browse tables, and determining which additional indices it might need
- Refactoring of the BrowseEngine. There is a lot of code here which could be reused better, and more things need to be pushed into the BrowseQuery class, making the code more manageable.
- Improvements to the layout of the UI. As an engineer, this is not my forte!
- Input validation; I want to build a separate bit of code whose job it is to make sure that the browse is sensible
- TODO review - a review of this document with all the tweaks that it recommends
- Browse Cacheing - this will be important for the scaling process; looking into how the old browse cached and seeing what we can add/improve
Before all that can happen, this code will first be rolled out to our dev box where it can be tested against a much larger data set.

Programmer Notes

- I have written the UI for the single browse and it appears to be working
- The main remaining challenge for the first version of the system is to get the title links to actually link to the item. This currently doesn't work because BrowseItem.getHandle returns null
- BrowseItem.getHandle now returns the handle
- In order to get the first version of this working to a testable standard, the only remaining job is to i18n the UI components

TODO: would we like a "return to browse by author" link on pages where you have gone into the items by a particular author. By extension, we also care about this for any "single" browse page

TODO: Would it be useful to generalise the columns that we browse over so that they can not only be defined for DSpace in config, but be defined per browse type in configuration?

TODO: the arrows in the navigation don't work for telling you which browse you are in. Also, how do we rationalise this with browsing by a specific value. If we are browsing a specific author, which navigation item is arrowed?

TODO: the order that things appear in the menu is in reverse order to how it appears in the config file. This should be made uniform

i18n

The following conventions will be used for setting up the i18n for the browse code.

All Browse Indexes will be looked up as follows:

```java
browse.type.<type> = <value>
```

for example

```java
browse.type.author = Author
```

All page content for the Full/Item and Item/Value browse will be looked up as follows:

```java
browse.full.<content> = <value>
```

for example

```java
browse.full.range = Showing Results \{0\} to \{1\} of \{2\}
```

All page content for Single/Value browse will be looked up as follows:

```java
browse.single.<content> = <value>
```

for example

```java
browse.single.order = Order
```

Further to this, the page title will have the i18n key

```java
browse.page-title
```
And the elements for the standard navigations at the top of each browse page will have they form

```
   browse.nav.<element> = <value>
```

for example

```
   browse.nav.jump-to = Jump to a point in the index:
```

Elements for the no results page will have the form

```
   browse.no-results.<content> = <value>
```

Elements that will appear in the left navigation will have the form

```
   browse.menu.<type> = <value>
```

Progress Update: 11-12-2006, 16:00 GMT

The first version of the Browse code appears to be complete. This afternoon I am rolling it out on one of our production spec dev servers where it will get tested with a large amount of data (75,000 records). I also hope to get some feedback off the community once I am able to provide a patch to the current DSpace CVS. In the mean time, if you are itching for a copy of the code, then you can ask me directly on the dspace-devel list.

Notes on Scalability

Now that this code has been rolled out onto a production grade development server with approximately 75,000 records, we can take some notes on how it performs.

Initial Indexing

The initial indexing is working across 8 separate indexes, with 2 additional sort columns. After approximately 15 minutes we check the rate of indexing to find that we are getting approximately 800 items indexed per minute. This means that if performance does not degrade we will index the full 75,000 records in approximately 90 minutes. After approximately 80 minutes we checked the performance again, and observe a slight degradation in performance, where now we are getting approximately 650 items indexed per minute. This is possibly correlated to the increased memory usage noted below. The final time for indexing 73,195 items came to 94min 17sec, which averages around 13 items per second.

On the first attempt to index this dataset we had the JVM started with a 256Mb maximum on memory. This caused an OutOfMemory exception after indexing 12,000 items. For the second attempt, the JVM heap size was upped to 2Gb (on a machine with 8Gb of RAM), which was successful. It would be interesting to know if the memory usage goes up linearly, and what the source of the usage is - the code attempts to keep caches clear, and actual stored objects to a minimum, so presumably we are either leaking somewhere, or there is a memory usage that has not been adequately considered.

Browsing

For some reason, browsing single value tables takes much longer than browsing full item tables. Probably this is to do with the DISTINCT statement. The good news is that browsing by full items records is sufficiently fast even with 75,000 records as to meet initial scalability requirements. Perhaps there is a better way of storing the data for single value browse that will speed things up. It will also be of benefit to analyse the processes that take place and identify the bottle-necks for further work. (note: perhaps a SELECT DISTINCT view on the single value browse might speed things up?)

Progress Update: 19-01-2007, 16:20 GMT

I haven't done any work on this over the Christmas period, but now I have come back to it the following bug has become apparent:

**BUG:** because of the complexity of the string construction, the PreparedStatement class is not used. There are, therefore, some string escaping problems that need to be resolved. Fortunately, the BrowseQuery class provides a wrapper to the query, so the escaping should be easy to introduce, and protect against both unwanted internal server errors and SQL injection attacks.

Further notes on Scalability

Download the attached file to see the output of the first 27,000 items indexed in my test environment. This operation ran as an indexing job for over 15 hours. The data file consists of 4 columns: the integer number of the index job, the unix time in milliseconds of the start of the operation, the item id, and the time it took to index. Plot this with gnuplot to see what's going on:
plot "browse.uat.dat" u 1:4 w linespoints

Browse.uat.dat