NextMonet Smart Browse Enhancements and Artist Search Functional Specifications

Jack Hodges
Andrew Stevko
January 19, 2001

Abstract

This document serves to define the basic functional and user requirements for enhancing the SmartBrowse search utility. This project will consist of modifications to the following components: Custom Search page, Thumbnail page, Working Page, and the SmartBrowse application layer. In addition, an enhancement will be introduced to allow searching for artists rather than art objects.

The requirements for the major tools will be specified by the combination of the functional use cases for the catalog model, the user interface (UI) elements, and the storyboard of pages that will implement the use cases pertinent to the catalog display and functionality.

Objective

This project has two objectives, to make browsing for art: 1) effective, and 2) painless. By effective, we mean that browsing via search criterion needs to return highly relevant art objects, and by painless, we mean to reduce the number of operations a visitor must apply to find a desirable object.

Overall Requirements

The project objective can be met by considering the following nine functional requirements:

- Provide the SmartBrowse component with the ability to re-order the art objects immediately beyond the art object currently in focus. This differs from the current behavior of re-ordering art objects that were never displayed. This should make the benefits of SmartBrowse more identifiable.
- Provide the SmartBrowse component with the ability to start a new search with every explicit ranking, in addition to the cumulative mechanism currently used. This will provide users who wish to browse an immediate response, while allowing users who want a focused search to locate items they like over a longer browse session.
- Modify the weighting on user's keyword selection so that those objects with selected keywords are primary to other weights. That is, the selected category will be ordered first in the sorted list.
- Change the Keyword constraints ordering within the Custom Search page to better reflect the relationships between them.
- Persist the filters on the custom search page.
- Provide an option whereby registered users can save their search criterion as well as their page settings across sessions.
- Make the response to rating an art object the display of the thumbnail page.
- Introduce new metrics for rating an art object subject, style, color, medium, and all
- Introduce new SmartBrowse constraining criteria to allow for "Search for Artists". This criterion will be specified on the Custom Search page and will most impact the SmartBrowse component.

Components Impacted

Three pages and one internal component are directly affected by this project.

1) **Custom Search Page:** This interface *currently* allows the user to select filters to limit the result set along with keywords to weight the result set by. This page will be modified by: (a) ordering the filters presented, (b) persisting the filters between searches in the same way that keywords are persisted, (c) re-evaluation of the layout of controls, and (d) the introduction of the Search for Artists constraint. The affected webpage component is:

   `"jhtml/search/detailedSearch.jhtml"`

2) **Thumbnail Browsing Page:** This should be nearly identical in logic and overall design to the current Thumbnail Browsing page that focuses on art objects. The difference is that, once clicked, the redirection will be to the artists' portfolio rather than to the working page. The closest match in terms of existing pages is the suggestions.jhtml page. The affected webpage component is:

   `"jhtml/search/searchResults.jhtml"`

3) **Working Page:** This interface *currently* allows the user to rate an art object in 5 discrete/subjective values, and then the keywords associated with this art object are modified to account for the rating, and all art objects in the collection are reordered by the new ranking. This page will be modified to allow the user to rate an art object along five metrics or to return to browsing. Each of the five metrics, which are: (a) subject, (b) style, (c) color, (d) medium, and (e) all (i.e, all keywords in the selected artwork, as it is at present), are forms of positive rating. The 'back' button is a form of negative rating. The user will be provided with an option to use a cumulative session or an immediate response (on the custom search page, but maybe eventually in their profile; the default will be set experimentally). When a judgement is made, the user is immediately returned to browsing thumbnails. The cumulative session will continue to behave as it does now. The immediate session will create a new session with the attribute sorting of the current filters and keywords. The user will be able to toggle between these browse modes at will. The affected webpage component is:

   `"jhtml/art/artImage.jhtml"`

4) **SmartBrowse:** The information specified within the Custom Search page is used by the SmartBrowse component to produce a sorted collection of the art objects that satisfy the filters and rank the keywords. This component will undergo the following major and minor modifications:

   - Change sorting boundaries to affect objects immediately after the object in focus.

   - Allow for additional criteria that will produce the Search by Artists behavior desired. The sort will remain the same, but after the sort, another sort will be performed to restrict the art objects to one per artist.

- Introduce an API to allow for ranking by the 5 metrics enumerated above.

SmartBrowse Criteria

The following criteria are to be used within the SmartBrowse component to produce a subset of art objects in an order that most appeals to the user.

Constraints

A constraint is an *explicit* filter. If one of the constraints below is selected by the user, the number of images that are returned is reduced to those which satisfy the constraint. Thus, if the catalog begins with 10,000 pieces of art and the height constraint is set to 10 inches or less, then browsing will be possible only on pieces of art that are 10 inches high, or less.

- Height range specified in Inches
- Width range specified in Inches
- Price range specified in Inches
- Medium (one or more selections)
- Subject (one or more selections)
- Color (one or more selections)
- Is available for sale
- One object per artist (a.k.a. Search by Artist)

Weighting

A weighting is an *implicit* filter, and is used to order the display of the images that remain after the constraints are applied. A *keyword* can be used to positively affect the ordering of the set. Thus, if the constraints produce a subset of 1,000 art pieces, then the combination of constraint/keyword values are used to rank the 1,000 pieces and produce an ordering that will help the visitor find art of interest. Note that both constraints and keywords will have values that affect the ordering, even though constraints are used to restrict the size of the set.

- Keywords
- Subject
- Style
- Color
- Medium

The following actions influence the weight designated to the art objects within the subset.

Custom Search page

- User enters either or both Heights (constraint)
- User enters either or both Widths (constraint)
- User enters either or both Price Ranges (constraint)
- User selects one or more Mediums (constraint)
- User selects one or more Subjects (constraint)
- User selects one or more Colors (constraint)
- User selects “Don’t show me sold works” (constraint)

- User selects "Search by Artist" (constraint)
- User selects one or more Keywords (weight by keyword)

Thumbnail Browse page

- User selects a piece (weight using this object's Keywords)

Working Page

- Enlarge piece (weight using this object's Keywords)
- Rate the Subject (weight using this object's Subject)
- Rate the Style (weight using this object's Style)
- Rate the Color (weight using this object's Color)
- Rate the Medium (weight using this object's Medium)

Functional Components - Background

In this section the functional components of this project will be detailed a little deeper. In particular, the existing functional relationships will be articulated. The next section will outline the implementation strategy for each required functionality.

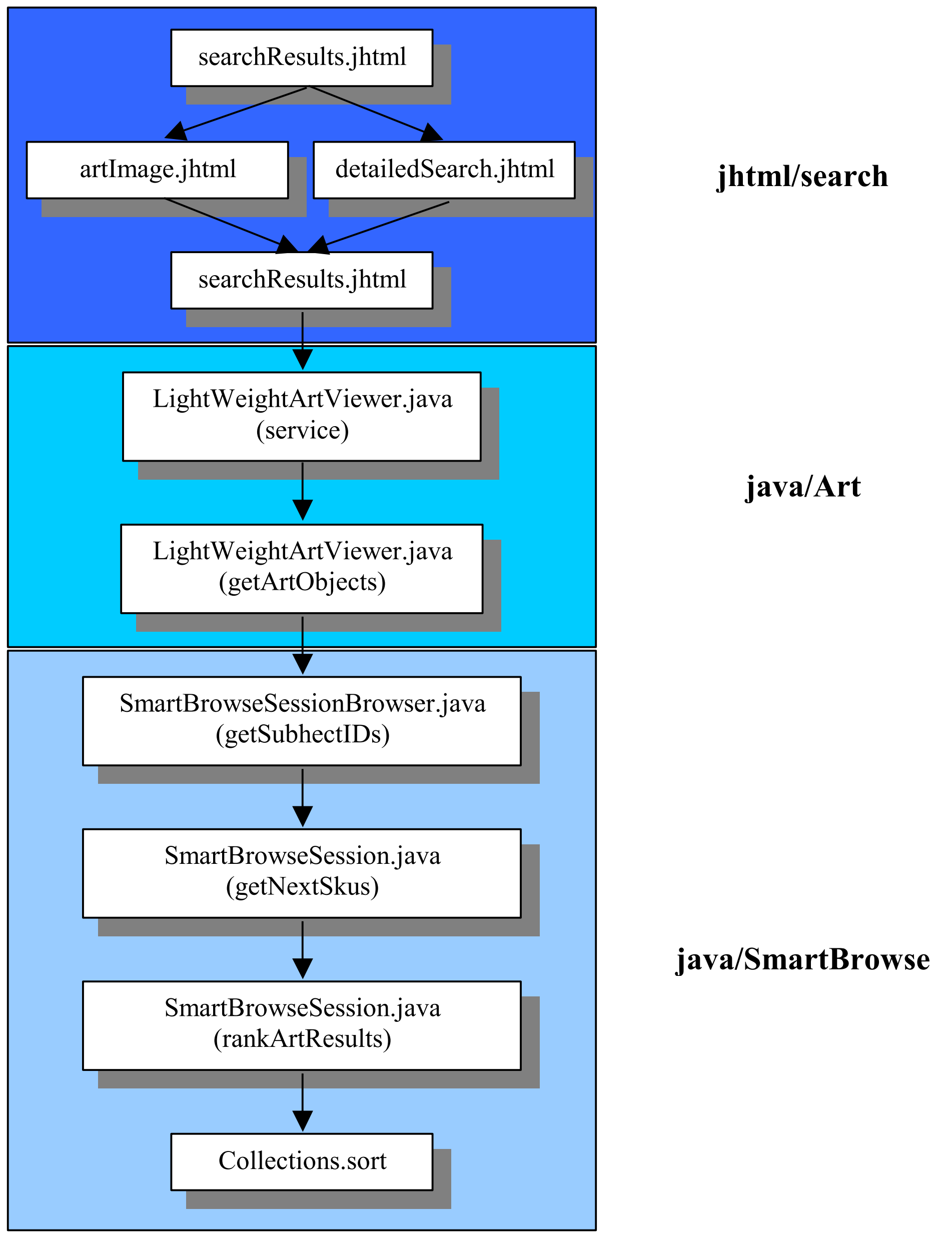
Immediate Reordering

This functionality means that when an explicit judgement (i.e., a rating) is made by a user on the working page (artImage.jhtml), the reordering that is displayed in searchResults.jhtml is based on the entire collection and not just the unranked images. In the present method, those images displayed on the thumbnail page (searchResults.jhtml) are not part of the ordering considered by SmartBrowse. In the proposed method, the entire set of images will be considered. The affected components are:

- `ArtImage.jhtml`
- `SearchResults.jhtml`
- `SmartBrowseFormHandler.java`
- `LightWeightArtViewer.java`
- `SmartBrowseSessionBrowser.java`
- `SmartBrowseSession.java`

In this case, the reordering affects the starting image of the session, so after the Collections.sort call, the searchResults.jhtml page displays using the startImage = 1 instead of startImage = size + 1, where size is the number of images on the current thumbnail page.

The overall flow is depicted in the figure below:


searchResults.jhtml
artImage.jhtml
detailedSearch.jhtml
searchResults.jhtml
LightWeightArtViewer.java (service)
LightWeightArtViewer.java (getArtObjects)
SmartBrowseSessionBrowser.java (getSubhectIDs)
SmartBrowseSession.java (getNextSkus)
SmartBrowseSession.java (rankArtResults)
Collections.sort
jhtml/search
java/Art
java/SmartBrowse

The function of each of these components, particularly with respect to the proposed project, is described below:

searchResults.jhtml

This is the target webpage for searches, the so-called thumbnail page. When this page is invoked, ultimately the current SmartBrowse session is resorted. The images ranging from startIndex to Max(startIndex + 7, endIndex – startIndex) are then displayed. Each of these images is in the new sorting, with the selected image being the first in the sort.

bean:LightWeightArtViewer.DisplayStartIndex
bean:LightWeightArtViewer.artCollection.length
(LightWeightArtViewer) param:displayableItems

param in the LWAV object above is a request paramter that is retrieved from the LWAV servlet when the service method is called, and represents the artObjects collection as a sorted array of lightweight art objects.

LightWeightArtViewer.java

Two methods are called in this servlet class. The first one is called right out of the webpage invocation, the service method. The service method tries to get the startIndex from the request, and it tries to set the displayableItems in the request to the result of calling the getArtObjects method. It then returns control to the webpage.

The getArtObjects method gets an array of art object Ids from the ArtObjects relational view. It then users that relational view to get another relational view of art objects by their sku numbers. It builds an array of art objects from that relational view. It does a similar thing with the art collections. It then returns the reordered set of the collection, the ids, or a new collections, in that preference order.

SmartBrowseSessionBrowser.java

The getSubsetIDs method is required by the LightWeightBrowsable interface, and is implemented here to

SmartBrowseSession.java

The getSubsetIDs method is called in the getNextSkus method defined in SmartBrowseSession.java. This method's real claim to fame is that it calls the rankArtResults method, but only if an explicit judgement has been made on the working page.

rankArtResults does a number of things. First it loops through the selected keywords and creates an accumulator for each one. Then it sets the judgement value to the appropriate value and adds that value to the appropriate keyword accumulator (or creates one if not found). Then it goes through all the artObjects again and checks to see if they exceed the correlation threshold (+ or -). If so, it adds the correlation threshold to the object's ranking and sets the art object's rank to be that ranking. Finally it calls Collections.sort on the ranked list.

Collections.sort

This is a method in the Collections class, which is in java.util.Collections. The sort method takes a single parameter, an ArrayList, and sorts by its natural elements.

Immediate Search

This functionality means that when an explicit judgement (i.e., a rating) is made (a rating) by the user on the working page (artImage.jhtml), there is no accumulation of values prior to the collection sorting. The keywords associated with the currently selected image, and whatever category of search that is selected, are the only determiners in the ordering of objects in searchResults.jhtml. The most profound difference in this approach is that a new SmartBrowseSession is created, so there is no accumulation to be performed. This action works in conjunction with the immediate reordering function. Currently, when a user makes an explicit judgement, the Collections.sort method is called, but control isn't shifted away from the working page. In the proposed model, if the user makes any judgement control is immediately returned to the thumb nail page. A 'back' judgement returns to the initial thumbnail page, and a 'show me more' judgement forwards to a new thumbnail page.

```
<input type    = "select"
       bean    = "SmartBrowseFormHandler.rating"
       value   = "50"
       onClick = "rateIt.submit()">

<input type  = "hidden"
       bean  = "SmartBrowseFormHandler.sku"
       value = "param:sku">

<input type  = "hidden"
       name  = "sku"
       value = "param:sku">

<input type  = "hidden"
       name  = "submitRanking"
       bean  = "SmartBrowseFormHandler.explicitRanking"
       value = "submit">
```

Category Search

This functionality means that a user may select a category of "show me more like this" rather than the current mechanism which equally weights all keywords associated with the artwork that is rated. The categories will be general: (1) color, (2) style, (3) subject, (4) medium, and the current mechanism (all). In order to implement this functionality, new methods must be developed in SmartBrowseSession.java since the current mechanism in artImage.jhtml sets the rating and that rating is applied to all keywords for the art object equally. Under the new functionality, the selected category gets a huge bump and the remaining keywords get equal but smaller bumps, and all other keywords are either reinitialized or left the same.

Filter Fix on Custom Search Page

This functional change amounts to looking at the tables associated with the attributes that are displayed in the detailedSearch.jhtml page and removing ones that aren't appropriate and reordering the ones that are. There are three groups of interest on the Custom Search page (detailedSearch.jhtml): Medium, Subject, and Color. Each of them use the following fragment of pseudocode:

```
Select bean = SmartBrowseFormHandler.medium
  droplet bean = /nextmonet/search/AttributeValues
    param name = attributeID value = 1007
```

```
oparam name = output
  droplet foreach
    param name = array value = param:attributeValues
    param name = sortProperties value = +name
    display-stuff
```

The only difference is that the value (1007 above) is 1003 for subject and 1002 for color. The `AttributeValues` call points to `getAttributeValues` method in the `AttributeValues` class, which invokes the `AttributeValueRView` Relational View, and the `ByTypeID` subview. This Relational View is found in the `"r2/db/rviews/art.rvw"` file

The table that this Relational View points to is ATTRIBUTE_VALUES, and has three fields: name, ID, and typeID.

Implementing this fix will require the removal of some of the attribute values from ATTRIBUTE_VALUES, and constructing a new subview that returns the items in the order they are found in, which means that we will probably have to reorder their typeIDs. What will the effect on other collections be? Are these attributes used in any capacity except for search?

Persistent Custom Search Filters

This functionality simply requires that we reload the SmartBrowse parameters into the Custom Search page (detailedSearch.jhtml) in the same manner that we reload the SmartBrowse keywords at present. The following code fragment should serve as a guide:

```
Select bean = SmartBrowseFormHandler.color size = 4 multiple
  droplet switch
    param name = value value =
bean.SmartBrowseFormHandler.ColorEmpty
    oparam name = true
      droplet bean = /nextmonet/search/AttributeValues
        param name = attributeID value = 1002
        oparam name = output
          OPTION VALUE = "" Any /OPTION
          droplet bean = ForEach
            param name = array value = param:attributeValues
            param name = sortProperties value = +name
    oparam name = false
      droplet bean = RelationalViewDroplet
        param name = rviewname value = ArtObjectColors
        param name = subviewName value = byID
        param name = subviewParam0 value =
bean.SmartBrowseFormHandler.color
         oparam name = output
           droplet name = ForEach
             param name = array value = param:result
             param name = sortProperties value = +name
```

The concept here is that if there is a color attribute set from a previous search, and stored in the SmartBrowseFormHandler, then we will retrieve it and display it. Otherwise we will do what we are doing now. This requires three new methods, to check if medium, subject, and color values have been set, and three Relational View subviews to retrieve them into the form.

Cross-Session Search Criterion Persistence

This functionality means that the current persistent banner information, and the custom search criterion, will be selectable by the user as persistent in their profile. If so, then the ProfileHandler will load these persisted values from the user table in the database when the user logs in. This will require the addition of medium, subject, color, and keywords attributes to the ProfileHandler, along with the six URLs that are used to maintain smart banners. To do this, the "`r2/xml/NextMonet-profile-template.xml`" file must be modified to account for the new attributes, along with the `Users` table in the database.

Display Thumbnail After Rating

This functionality means that immediately after selecting a "more like this" button on the working page (artImage.jhtml), the result will be displayed on the searchResults.jhtml page. Currently the ranking system does the following:

SmartBrowseFormHandler.rating value = #	
SmartBrowseFormHandler.explicitRanking	Sets value for keyword
SmartBrowseFormHandler.handleExplicitRanking	
SmartBrowseSession.explicitJudgement	

Notice that this doesn't actually result in a new sorting, let alone forwarding anywhere. Thus the proposed functionality must perform the ranking, and must also perform the sort and forward to searchResults.jhtml. The sort must put the current image at the front of the list.

Search for Artists

This functionality adds onto the existing functionality, in that the result of a sort will be the same, but that another pass is made through the sorted list prior to display, and in this pass the following pseudocode will be implemented:

```
Loop through the artObjects list
  perform a lookup on the art object's artistID

  if the artistID hasn't been seen
    locate the representative art object ID for this artist
    add the art object ID to a temporary list
    add the artistID to another temporary list
    continue

Return the temporary art list as the new sorting
```

This operation is O(N), but at least it is being performed on the reduced set of art objects.

Functional Components – Implementation Strategy

In this section the functional components of this project will be detailed a little deeper. In particular, the existing functional relationships will be articulated. The next section will outline the implementation strategy for each required functionality will be discussed. and associated with functional flows in the existing architecture. From here it is expected an engineer will be able to implement the functionality.

Immediate Reordering

This functionality means that when an explicit judgement (i.e., a rating) is made by a user on the working page (artImage.jhtml), the reordering that is displayed in searchResults.jhtml is based on the entire collection and not just the unranked images. In the present method, those images displayed on the thumbnail page (searchResults.jhtml) are not part of the ordering considered by SmartBrowse. In the proposed method, the entire set of images will be considered. The affected components are:

- `ArtImage.jhtml`
- `SearchResults.jhtml`
- `SmartBrowseFormHandler.java`
- `LightWeightArtViewer.java`
- `SmartBrowseSessionBrowser.java`
- `SmartBrowseSession.java`

In this case, the reordering affects the starting image of the session, so after the Collections.sort call, the searchResults.jhtml page displays using the startImage = 1 instead of startImage = size + 1, where size is the number of images on the current thumbnail page.

Immediate Search

This functionality means that when an explicit judgement (i.e., a rating) is made (a rating) by the user on the working page (artImage.jhtml), there is no accumulation of values prior to the collection sorting. The keywords associated with the currently selected image, and whatever category of search that is selected, are the only determiners in the ordering of objects in searchResults.jhtml. The most profound difference in this approach is that a new SmartBrowseSession is created, so there is no accumulation to be performed. This action works in conjunction with the immediate reordering function. Currently, when a user makes an explicit judgement, the Collections.sort method is called, but control isn't shifted away from the working page. In the proposed model, if the user makes any judgement control is immediately returned to the thumb nail page. A 'back' judgement returns to the initial thumbnail page, and a 'show me more' judgement forwards to a new thumbnail page.

```
<input type    = "select"
       bean    = "SmartBrowseFormHandler.rating"
       value   = "50"
       onClick = "rateIt.submit()">
```

```
<input type  = "hidden"
       bean  = "SmartBrowseFormHandler.sku"
       value = "param:sku">

<input type  = "hidden"
       name  = "sku"
       value = "param:sku">

<input type  = "hidden"
       name  = "submitRanking"
       bean  = "SmartBrowseFormHandler.explicitRanking"
       value = "submit">
```

Category Search

This functionality means that a user may select a category of “show me more like this” rather than the current mechanism which equally weights all keywords associated with the artwork that is rated. The categories will be general: (1) color, (2) style, (3) subject, (4) medium, and the current mechanism (all). In order to implement this functionality, new methods must be developed in SmartBrowseSession.java since the current mechanism in artImage.jhtml sets the rating and that rating is applied to all keywords for the art object equally. Under the new functionality, the selected category gets a huge bump and the remaining keywords get equal but smaller bumps, and all other keywords are either reinitialized or left the same.

Filter Fix on Custom Search Page

This functional change amounts to looking at the tables associated with the attributes that are displayed in the detailedSearch.jhtml page and removing ones that aren’t appropriate and reordering the ones that are. There are three groups of interest on the Custom Search page (detailedSearch.jhtml): Medium, Subject, and Color. Each of them use the following fragment of pseudocode:

```
Select bean = SmartBrowseFormHandler.medium
  droplet bean = /nextmonet/search/AttributeValues
    param name = attributeID value = 1007
    oparam name = output
      droplet foreach
        param name = array value = param:attributeValues
        param name = sortProperties value = +name
        display-stuff
```

The only difference is that the value (1007 above) is 1003 for subject and 1002 for color. The `AttributeValues` call points to `getAttributeValues` method in the `AttributeValues` class, which invokes the `AttributeValueRView` Relational View, and the `ByTypeID` subview. This Relational View is found in the “`r2/db/rviews/art.rvw`” file

The table that this Relational View points to is ATTRIBUTE_VALUES, and has three fields: name, ID, and typeID.

Implementing this fix will require the removal of some of the attribute values from ATTRIBUTE_VALUES, and constructing a new subview that returns the items in the order they are found in, which means that we will probably have to reorder their typeIDs. What will the effect on other collections be? Are these attributes used in any capacity except for search?

Persistent Custom Search Filters

This functionality simply requires that we reload the SmartBrowse parameters into the Custom Search page (detailedSearch.jhtml) in the same manner that we reload the SmartBrowse keywords at present. The following code fragment should serve as a guide:

```
Select bean = SmartBrowseFormHandler.color size = 4 multiple
  droplet switch
    param name = value value =
bean.SmartBrowseFormHandler.ColorEmpty
    oparam name = true
      droplet bean = /nextmonet/search/AttributeValues
        param name = attributeID value = 1002
        oparam name = output
          OPTION VALUE = "" Any /OPTION
          droplet bean = ForEach
            param name = array value = param:attributeValues
            param name = sortProperties value = +name
    oparam name = false
      droplet bean = RelationalViewDroplet
        param name = rviewname value = ArtObjectColors
        param name = subviewName value = byID
        param name = subviewParam0 value =
bean.SmartBrowseFormHandler.color
         oparam name = output
           droplet name = ForEach
             param name = array value = param:result
             param name = sortProperties value = +name
```

The concept here is that if there is a color attribute set from a previous search, and stored in the SmartBrowseFormHandler, then we will retrieve it and display it. Otherwise we will do what we are doing now. This requires three new methods, to check if medium, subject, and color values have been set, and three Relational View subviews to retrieve them into the form.

Cross-Session Search Criterion Persistence

This functionality means that the current persistent banner information, and the custom search criterion, will be selectable by the user as persistent in their profile. If so, then the ProfileHandler will load these persisted values from the user table in the database when the user logs in. This will require the addition of medium, subject, color, and keywords attributes to the ProfileHandler, along with the six URLs that are used to maintain smart banners. To do this, the "`r2/xml/NextMonet-profile-template.xml`" file must be modified to account for the new attributes, along with the `Users` table in the database.

Display Thumbnail After Rating

This functionality means that immediately after selecting a "more like this" button on the working page (artImage.jhtml), the result will be displayed on the searchResults.jhtml page. Currently the ranking system does the following:

SmartBrowseFormHandler.rating value = #	
SmartBrowseFormHandler.explicitRanking	Sets value for keyword
SmartBrowseFormHandler.handleExplicitRanking	
SmartBrowseSession.explicitJudgement	

Notice that this doesn't actually result in a new sorting, let alone forwarding anywhere. Thus the proposed functionality must perform the ranking, and must also perform the sort and forward to searchResults.jhtml. The sort must put the current image at the front of the list.

Search for Artists

This functionality adds onto the existing functionality, in that the result of a sort will be the same, but that another pass is made through the sorted list prior to display, and in this pass the following pseudocode will be implemented:

```
Loop through the artObjects list
  perform a lookup on the art object's artistID

  if the artistID hasn't been seen
    locate the representative art object ID for this artist
    add the art object ID to a temporary list
    add the artistID to another temporary list
    continue

Return the temporary art list as the new sorting
```

This operation is O(N), but at least it is being performed on the reduced set of art objects.

Example

This section is to give a *single* example of each of the parts of the specification. The parts include:

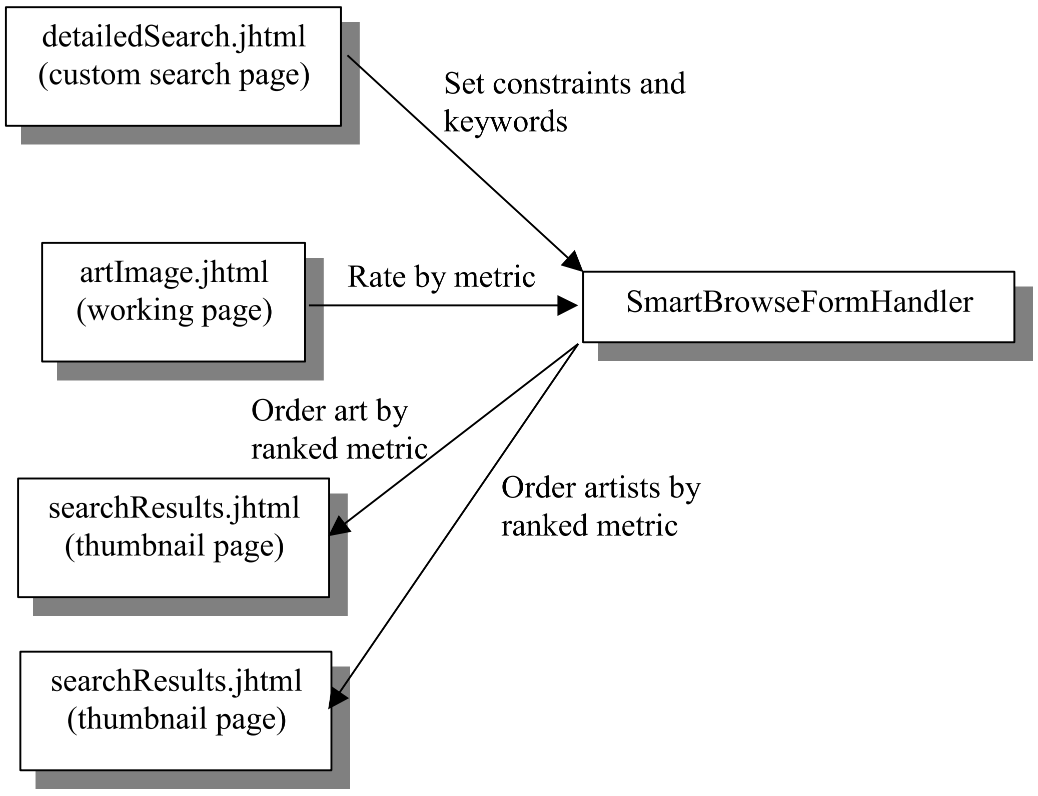
- The use cases
- User interface elements
- Page flows (storyboards)

The Detailed Use Cases

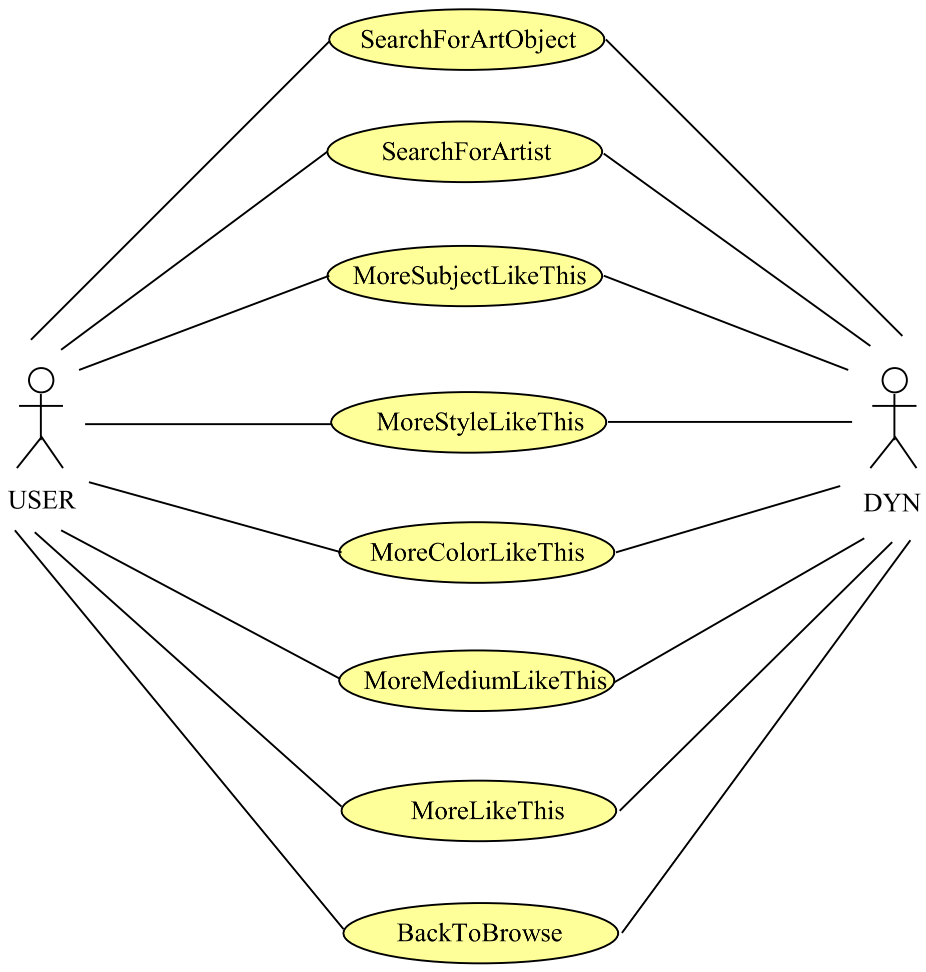
These talk about the use cases for each tool that is specified and the primary and major secondary scenarios for each use case. First the actors for the project are defined, and then the use cases.

Actors

- **USER:** Viewing the catalog intent on finding a relevant piece of art.
- **DYN:** The Dynamo Application Server
- **DB:** The Oracle Database Server
- **IS:** The True Spectra Image Server


detailedSearch.jhtml (custom search page)
Set constraints and keywords
artImage.jhtml (working page)
Rate by metric
SmartBrowseFormHandler
Order art by ranked metric
Order artists by ranked metric
searchResults.jhtml (thumbnail page)
searchResults.jhtml (thumbnail page)

User-Driven Use Cases



USER SearchForArtObject into DYN

Precondition: Selections are available on the 'custom search page'

Flow of Events:

Primary Scenario

1) USER selects constraints and keywords to search on from the Custom Search page

2) USER selects the Search For Art Object from the Custom Search page pull down

3) DYN displays the first page of art objects which satisfy the USERs constraints

Postcondition: Art objects matching the search constraints and keywords are displayed in the browser page.

USER SearchForArtists into DYN

Precondition: Selections are available on the 'custom search page'

Flow of Events:

Primary Scenario

1) USER selects constraints and keywords to search on from the Custom Search page
2) USER selects the Search For Artists from the Custom Search page pulldown
3) DYN displays a page of artists links, by their signature image and their name, which satisfy the USERs constraints and keywords

Postcondition: Artists matching the search constraints and keywords are displayed in the artist page.

USER MoreSubjectLikeThis into DYN

Precondition: An artwork is viewable on the 'working page' section of the catalog working page.

Flow of Events:

Primary Scenario

1) USER selects 'More Subject Like This' button from catalog 'working page' segment
2) DYN uses SmartBrowse to locate and display a browse page with similar works on it starting with the current art object.

Postcondition: Similar artworks are displayed in browse format, ordered most prominently by subject.

USER MoreStyleLikeThis into DYN

Precondition: An artwork is viewable on the 'working page' section of the catalog working page.

Flow of Events:

Primary Scenario

1) USER selects 'More Style Like This' button from catalog 'working page' segment
2) DYN uses SmartBrowse to locate and display a browse page with similar works on it starting with the current art object and weighted toward style.

Postcondition: Similar artworks are displayed in browse format, ordered most prominently by style.

USER MoreColorLikeThis into DYN

Precondition: An artwork is viewable on the ‘working page’ section of the catalog working page.

Flow of Events:

Primary Scenario

1) USER selects ‘More Color Like This’ button from catalog ‘working page’ segment

2) DYN uses SmartBrowse to locate and display a browse page with similar works on it starting with the current art object and weighted toward color.

Postcondition: Similar artworks are displayed in browse format, ordered most prominently by color.

USER MoreMediumLikeThis into DYN

Precondition: An artwork is viewable on the ‘working page’ section of the catalog working page.

Flow of Events:

Primary Scenario

1) USER selects ‘More Medium Like This’ button from catalog ‘working page’ segment

2) DYN uses SmartBrowse to locate and display a browse page with similar works on it starting with the current art object and weighted toward medium.

Postcondition: Similar artworks are displayed in browse format, ordered most prominently by medium.

USER MoreLikeThis into DYN

Precondition: An artwork is viewable on the ‘working page’ section of the catalog working page.

Flow of Events:

Primary Scenario

1) USER selects ‘More Like This’ button from catalog ‘working page’ segment

2) DYN uses SmartBrowse to locate and display a browse page with similar works on it starting with the current art object and using a general ordering metric.

Postcondition: Similar artworks are displayed in browse format, ordered most prominently by NextMonet's weighting selection.

USER ReturnToBrowse into DYN

Precondition: An artwork is viewable on the 'working page' section of the catalog working page.

Flow of Events:

Primary Scenario

1) USER selects 'Return To Browse' button from catalog 'working page' segment

2) DYN returns to the last browse page with same ordering as previous.

Postcondition: The previous browse page is displayed.

User Interface Elements

The UI Elements involved in this enhancement are those on the Working Page and the Custom Search Page. There are three UI components directly affected by this enhancement:

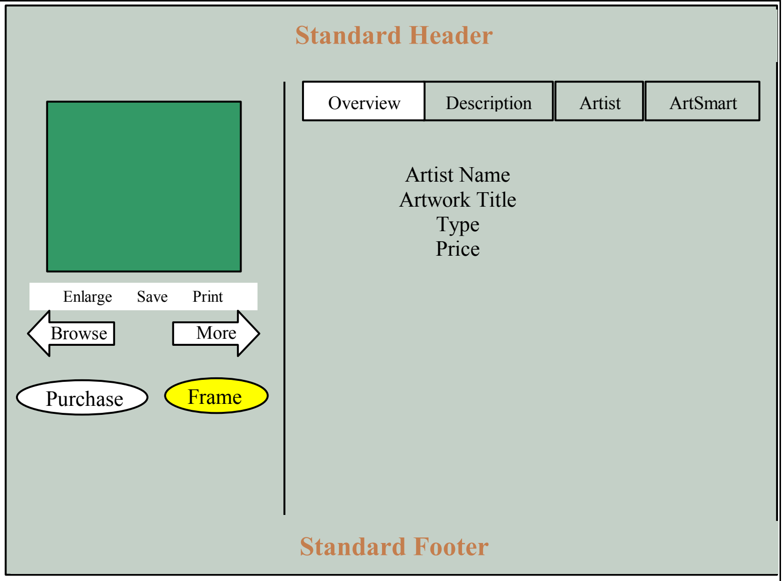
1) Working page rating

2) Working page selection

3) Custom search page search type selection

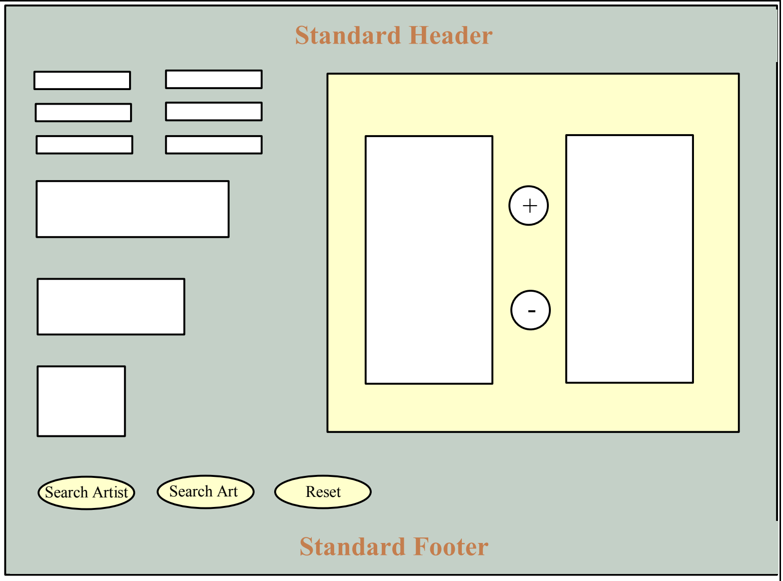
Working Page Rating

Description	Working page
Functionality	Allows user to rate an artwork for browsing preference from page
Applies To	Browse pages
Key Screen Elements	'return to browse' button, 'more like this' button, 'more like this subject' button, 'more like this style' button, 'more like this medium' button, 'more like this color' button.
General Appearance (illustrative)	

	 Standard Header Overview Description Artist ArtSmart Artist Name Artwork Title Type Price Enlarge Save Print Browse More Purchase Frame Standard Footer
Pages	jhtml/art/artImage.jhtml
Implementation Priority	
Implementation Difficulty	
Implementation Notes	Affects SmartBrowseFormHandler (explicitRanking)

Custom Search Page Selection

Description	Custom search page
Functionality	Allows user to select search results as artwork or artist from pulldown
Applies To	Browse pages
Key Screen Elements	New element on page.
General Appearance (illustrative)	

	 Standard Header + - Search Artist Search Art Reset Standard Footer
Pages	jhtml/search/detailedSearch.jhtml
Implementation Priority	
Implementation Difficulty	
Implementation Notes	SmartBrowseFormHandler.submitFilterCriteria, refineSearch = false, successURL = /search/searchResults.jhtml