Bitstream order

Background
At Leiden University, The Netherlands, we needed a way to change the order of the bitstreams as they are displayed in an easy way.

By default in version 1.3 of DSpace, when you upload more than 1 bitstream for a single item, the bitstreams are presented in the reversed order they were uploaded.

This implementation lets administrators change the display order of bitstreams independently from their upload order, in an easy way.

How it works
The order of bitstreams is defined by using the Basic RDF implementation. The bitstreams are semantically linked to each other by using "precedes" predicate. The special class BitstreamArranger manages the ordering of bitstreams.

Which DSpace version
The modifications were made to Dspace 1.3.2. This is also the version of DSpace where this functionality was tested extensively.

It should be possible to use this with another version of DSpace, and on first glance it should not be hard at all.

How to install
This installation does not change any functionality, but it adds functionality potentially.
If you want to use it, write code that makes use of the classes described below, or implement the order of bitstreams.

Step 1: Install the Basic+RDF+implementation
See Basic RDF implementation for details on how to do this.

Step 2: Add java class
- Download the file BitstreamArranger.java and place it in the directory src/nl/leidenuniv/dspace/rdf/

Step 3: Change java classes
org/dspace/app/webui/servlet/admin/EditItemServlet.java
- add some new import statements:

```java
import java.util.ArrayList;
import nl.leidenuniv.dspace.rdf.BitstreamArranger;
import nl.leidenuniv.dspace.rdf.RDFpredicate;
import nl.leidenuniv.dspace.rdf.RDFtriple;
```
- add line of code just before "// Sort the list" near line 372

```java
BitstreamArranger arranger = BitstreamArranger.getBitstreamArranger();
```
- add lines of code inside "if (button.equals("submit_delete_bitstream_" ..." block near 463 as the first statements:
RDFpredicate predicate = arranger.precedesPredicate(context);
RDFtriple triple = RDFtriple.find(context, bitstream, predicate, Bitstream.class);
if (triple != null) {
    triple.delete();
}

- add a new "else if" block just before the "item.update()" near 524:

```java
else if (p.equals("position"))
    //    && AuthorizeManager.isAdmin(context))
{
    String[] values = request.getParameterValues(p);
    Bitstream[] newOrder = new Bitstream[values.length];
    for (int i = 0; i < values.length; i++) {
        String v = values[i];
        StringTokenizer st = new StringTokenizer(v, "_"); // should use: v.split("_")
        // Ignore "position"
        st.nextToken();
        // Bundle ID and bitstream ID next
        int bundleID = Integer.parseInt(st.nextToken());
        int bitstreamID = Integer.parseInt(st.nextToken());
        String key = String.valueOf(bundleID) + "_" + bitstreamID;
        if (!newPos.equals("N")) {
            int pos = Integer.parseInt(newPos);
            if (bitstream != null && newOrder[pos] == null
                && !button.equals("submit_delete_bitstream_"+key)) {
                newOrder[pos] = bitstream;
            }
        }
    }
    int count = 0;
    for (int i = 0; i < newOrder.length; i++) {
        if (newOrder[i] != null) count++;
    }
    Bitstream[] bitOrder = new Bitstream[count];
    for (int i = 0; i < newOrder.length; i++) {
        if (newOrder[i] != null) {
            bitOrder[j++] = newOrder[i];
        }
    }
    arranger.orderListOfOriginalBitstreams(context, item, bitOrder);
}
```

- add some new import statements:

```java
import org.dspace.core.Context;
import nl.leidenuniv.dspace.rdf.BitstreamArranger;
```

- replace the following code near line 632:

```java
for (int i = 0; i < bundles.length; i++) {
    Bitstream[] bitstreams = bundles[i].getBitstreams();
    for (int k = 0; k < bitstreams.length; k++)
    {
        // Skip internal types
        if (!bitstreams[k].getFormat().isInternal())
        {
```
with this:

```java
BitstreamArranger arranger = BitstreamArranger.getBitstreamArranger();
Context context = null;
Bitstream[] bitstreams = null;
try {
    context = UIUtil.obtainContext(request);
    bitstreams = arranger.getOrderedListOfOriginalBistreams(context, item);
} catch (SQLException e) {
    // something wrong...
    throw new IOException(e.toString());
} for (int k = 0; k < bitstreams.length; k++) {
```

and do not forget to balance the number of brackets (remove 2 brackets) near line 718

Step 4: edit jsp page

```java
jsp/tools/edit-item-form.jsp
```

Make a local copy of this page if you did not do this already

- add some page imports:

```java
<%@ page import="nl.leidenuniv.dspace.rdf.BitstreamArranger" %>
<%@ page import="nl.leidenuniv.dspace.rdf.RDFtriple" %>
<%@ page import="nl.leidenuniv.dspace.rdf.RDFpredicate" %>
<%@ page import="java.sql.SQLException" %>
<%@ page import="org.dspace.app.webui.util.UIUtil" %>
<%@ page import="org.dspace.core.Context" %>
<%@ page import="org.dspace.content.DSpaceObject" %>
```

- near 320 add 2 columns

```java
<th id="t10" class="oddRowOddCol">&nbsp;</th>
<th id="t11" class="oddRowOddCol"><strong><fmt:message key="jsp.tools.edit-item-form.elem6"/></strong></th>
```

- replace "Bitstream[] bitstreams = bundles[i].getBitstreams();" near line 335 with the following code:
Bitstream[] bitstreams = new Bitstream[0];
boolean isOriginal = bundles[i].getName().equals("ORIGINAL");
int posKnownFrom = 0;
int posKnownTill = -1;
if (isOriginal) {
    BitstreamArranger arranger = BitstreamArranger.getBitstreamArranger();
    try {
        Context context = UIUtil.obtainContext(request);
        bitstreams = arranger.getOrderedListOfOriginalBistreams(context,item);
        RDFpredicate predicate = arranger.initialPredicate(context);
        if (bitstreams.length > 0) {
            RDFtriple triple = RDFtriple.find(context,item,predicate,Bitstream.class);
            if (triple != null) {
                DSpaceObject first = triple.getObject();
                for (int p=0; p<bitstreams.length; p++) {
                    if (bitstreams[p].equals(first)) {
                        posKnownFrom = p;
                        posKnownTill = p;
                    }
                }
                if (posKnownFrom >= 0) {
                    predicate = arranger.precedesPredicate(context);
                    for (int p=posKnownFrom; p<bitstreams.length-1; p++) {
                        triple = RDFtriple.find(context,bitstreams[p],predicate,bitstreams[p+1]);
                        if (triple == null) {
                            break;
                        }
                        posKnownTill++;
                    }
                }
            }
        }
    } catch (SQLException e) {
        // do nothing, maybe originals will be retrieved by getBitstreams below
    }
}
if (bitstreams.length == 0) {
    bitstreams = bundles[i].getBitstreams();
}

• replace "<% if (bundles[i].getName().equals("ORIGINAL")) {" near line 344 with the following code:

```
<td headers="t10" class="<%= row %>RowEvenCol">
    <a target="_blank" href="<%= request.getContextPath() %>/retrieve/<%= bitstreams[j].getID() %>"/>

</td>
```

• near line 353 replace "<td headers="t11"> </td>" with:

```
<td headers="t11" class="<%= row %>RowOddCol">

</td>
<td headers="t11" class="<%= row %>RowEvenCol">

</td>
```

• near line 372 replace:
with this:

```html
<input type="submit" name="submit_delete_bitstream_<%= key %>" value="<fmt:message key="jsp.tools.general.remove"/>
```
Step 5: build / install / restart

Do the stuff you normally do when deploying a new version of DSpace.

Future work

Besides implementing new functionality using this basic RDF implementation, there are two obvious omissions:

- a way to use a subject or an object that is not a dspace object
- related to this: use a plain text object
- an implementation of RDF containers like Bag, Seq and Alt. This maybe addressed in the future.

Contact me

If you need help, or have any comments, or you just want to inform me that you (are going to) use this, please contact me at schaik (at) library.leidenuniv.nl