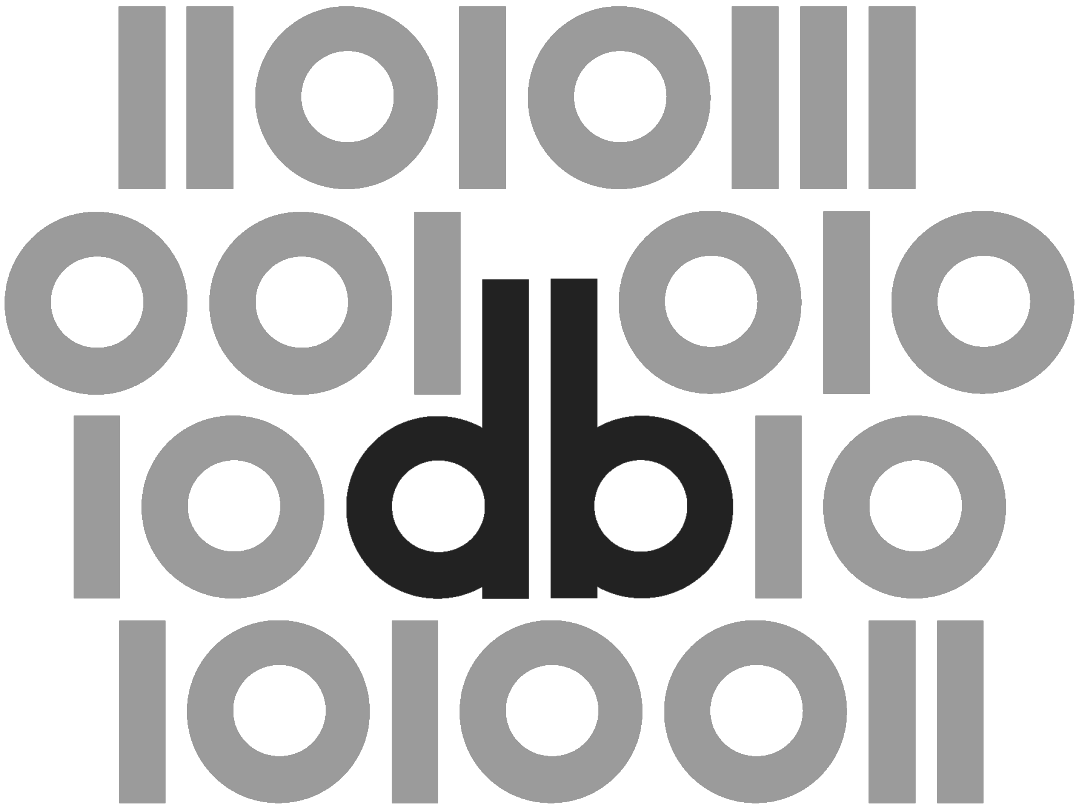
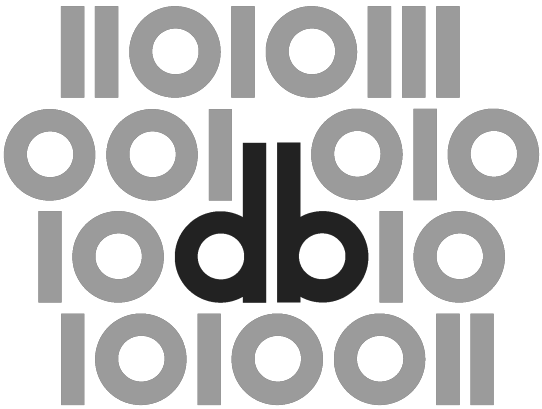
3 (2020) <DIGITÁLIS BÖLCSÉSZET>



3 (2020) </DIGITÁLIS BÖLCSÉSZET>

Digitális Bölcsészet
2020., harmadik szám

<DIGITÁLIS BÖLCSÉSZET>



3 (2020)

Felelős szerkesztő:
Maróthy Szilvia

Szerkesztőség:
Kokas Károly, Parádi Andrea

Rovatvezetők:
Tanulmányok: Kiss Margit
Műhely: Péter Róbert
Kritika: Almási Zsolt
Labor: Maróthy Szilvia

Tanácsadó testület:
Bartók István, Fazekas István, Golden Dániel, Horváth Iván, Palkó Gábor, Pap Balázs, Sass Bálint, Seláf Levente

Korábbi munkatársaink:
Bartók Zsófia Ágnes (szerkesztő, rovatvezető), Fodor János (szerkesztő), †Labádi Gergely (szerkesztő, rovatvezető), †Orlovszky Géza (tanácsadó testület)

ISSN 2630-9696
DOI 10.31400/dh-hun.2020.3
Kiadja a Bakonyi Géza Alapítvány és az ELTE BTK Régi Magyar Irodalom Tanszéke (1088 Budapest, Múzeum krt. 4/A).
Felelős kiadó az ELTE BTK Régi Magyar Irodalom Tanszék vezetője.

Megjelenik az Open Journal Systems (OJS) v. 3. platformon, melynek működtetését az ELTE Egyetemi Könyvtár- és Levéltár biztosítja.

Munkánkat a Nemzeti Kulturális Alap támogatja.

Ez a mű a Creative Commons *Nevezd meg! – Ne add el! – Így add tovább! 2.5 Magyarország Licenc* (`http://creativecommons.org/licenses/by-nc-sa/2.5/hu/`) feltételeinek megfelelően felhasználható.

Honlap: `http://ojs.elte.hu/digitalisbolcseszet`
Email cím: dbfolyoirat@gmail.com

Olvasószerkesztő: Bucsics Katalin
Tördelés: Hegedüs Béla
Grafika: Hegyi Gábor

<LABOR>

Tamás Kiss

Monguz/Qulto International

Eötvös Loránd University Centre for Digital Humanities

tkiss@monguz.hu

Zoltán Kanász-Nagy

Monguz/Qulto International

zkanasznagy@monguz.hu

Record Page Media Player in *DSpace* 7

In recent years, public collections and academic research communities have shown a growing interest in open-source repository solutions that are easy to install, customize, and operate. *DSpace*, a repository software developed by the not-for-profit DuraSpace organization since 2002, and its current version 7 proves to be an adequate solution to these requirements regarding storing and repositing digital objects. However, *DSpace* 7 is currently available only in a beta version and lacks certain features and functionalities, the development of which – due to the urgency of the momentary needs that our team intended to meet – could not wait until their planned implementation by the DuraSpace community. A media viewer and media player were such missing functionalities. Therefore, the authors of this article and their colleagues (web designer Annamária Tápai-Kovács and software developer Dániel Sipos) have undertaken to develop these functionalities and offer the resultant blocks of code to DuraSpace to be implemented in the code base of *DSpace* 7. In this paper we shall discuss the brief history of our collaboration with DuraSpace and share code snippets from our development.

Keywords:
DSpace 7, software development, media viewer, media player, repository, Angular 10

Community Development of *DSpace* 7

DSpace is an open-source repository software that has gained great popularity in recent years, primarily, among public education and public collection institutions. It owes its popularity mainly to its relatively easy usage, configurability and operability. Many projects have been implemented thanks to the *DSpace* repository system, and

drawing on the experience of these projects, several – commercial and freely available – add-ons have now become added value to the core software.

DSpace's version 7 continues to be developed on the basis of the ideas and needs of its developer and user community,[1] but it is also a new milestone in the life of the software. While *DSpace* 7 inherits useful features of its previous versions, partially diverting from the traditions of previous versions, it receives a completely new, web-based user interface – written in the *Angular* 10 framework.[2] In addition, several new features beyond front-end development (e.g. REST API and configurable entities) have been added to the list in which DuraSpace outlined the foundations of the new version and the long-term goals of its development.

Tim Donohue, the chief technology officer of *DSpace*'s development, originally planned the first release, as the result of the work that began in 2018, to be completed in the summer of 2020. However, due to unforeseen resource shortages, the developers have changed the date of completion of the current beta version to the second quarter of 2021. Conclusively, users will have to wait a few more months until the release.[3]

However, this does not mean that the software cannot be used in its current state: most of the final features are already available in it, and anyone who wishes to use *DSpace* 7 to satisfy their own or their customers' repository-related needs before the first release can try the official demo[4] or install the latest version of the software.

Our Contribution to the Development of *DSpace* 7

Our team started to follow the life of the new *DSpace* version in the first half of 2020, at the time of the publication of the first alpha version,[5] which was free to be tested by the general public. At first, our developers set out to learn the potentials of the new version as testers, and then they tried to give as much feedback as possible to the software's developers. Later, we started to help with the completion of the first version with smaller improvements, as a result of which Monguz/Qulto International, our wider professional „family" – now a member of the DuraSpace developer community – contributed to the development of *DSpace* by fixing various bugs, such as ones affecting displaying search results and record pages.

[1] At an organizational level, the development of *DSpace* is coordinated by DuraSpace, a not-for-profit community, which finances its operation not from the sale of the software but from the financial contributions of its community members. The most formidable member institutions supporting the community financially as well, are listed on the following website, accessed at: 2021.04.27, `https://duraspace.org/dspace/community/members/`.

[2] A detailed description of the back-end and front-end technologies used in the development available at the following link, accessed at: 2021.04.27, `https://wiki.lyrasis.org/display/DSDOC7x/Installing+DSpace`.

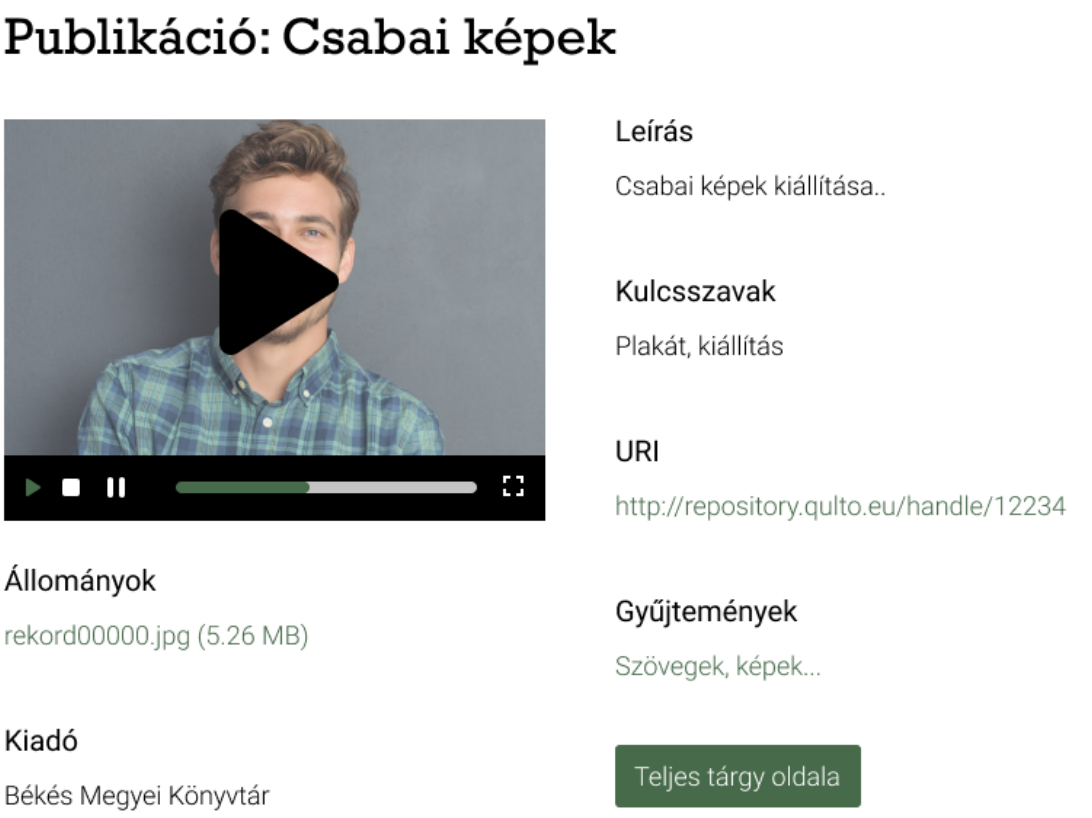
[3] Tim Donohue et al., „On the Road to *DSpace* 7: 2018 Edition," 2018, `http://repositorio.concytec.gob.pe/bitstream/20.500.12390/46/1/2018_Donohue_Dspace7.pdf`.

[4] The demo version is accessible through the following link, accessed at: 2021.04.27, `https://dspace7-demo.atmire.com/home`.

[5] Members of the DuraSpace community communicate with each other not only at larger-scale *DSpace* Steering Group meetings and developer-level technical status discussions, but also through their community chat channel on Slack, to which anyone is free to sign up.

 DOI 10.31400/dh-hun.2020.3.1475

DuraSpace's development roadmap scheduled displaying and playing image, audio, and video content directly on the record page through a web interface only to a later phase of development (late 2021 at the earliest). Consequently, our team undertook the development of functionalities that would enable viewing and playing record-related media, thus furthering the user-friendly operation of the software. Our developers' new implementations will be part of the system in the next (beta 5) version. Those interested in the chronology of this development, which greatly facilitates the proper use of the software, as well as in the details of its implementation, are advised to consult the following Github pages: `https://github.com/DSpace/dspace-angular/issues/885`, `https://github.com/DSpace/dspace-angular/pull/888`.



1. **Figure**. Video player (record page excerpt)

Development Methodology with Media Player Code Snippets

The development methodology we used for displaying image, audio and video content on the record page and the resultant code snippets are as follows:

1. Preparing record page media display: analyzing the descriptive records of all of the bitstreams associated with the *DSpace* record, we create an object of the type *mediaViewerItem,* that is, we map the bitstream resource descriptor to *mediaViewerItem*:

```
src\app\+item-page\media-viewer\media-viewer.component.ts

/**
* This method loads all the Bitstreams and Thumbnails and converts
* them to media item
*/

ngOnInit(): void {
 this.mediaList$ = new BehaviorSubject([]);
 this.isLoading = true;
 this.loadRemoteData('ORIGINAL').subscribe((bitstreamsRD) => {
  if (bitstreamsRD.payload.page.length === 0) {
   this.isLoading = false;
   this.mediaList$.next([]);
  } else {
   this.loadRemoteData('THUMBNAIL').subscribe((thumbnailsRD) => {
   for (
    let index = 0;
    index < bitstreamsRD.payload.page.length;
    index++
   ) {
    bitstreamsRD.payload.page[index].format
     .pipe(getFirstSucceededRemoteDataPayload())
     .subscribe((format) => {
      const current = this.mediaList$.getValue();
      const mediaItem = this.createMediaViewerItem(
       bitstreamsRD.payload.page[index],
       format,
       thumbnailsRD.payload && thumbnailsRD.payload.page[index]
      );
      this.mediaList$.next([...current, mediaItem]);
     });
    }
    this.isLoading = false;
   });
  }
 });
}
```

2. We use the *Ngx-gallery* module modified by Kolkov to display images.[6] For the gallery to work, one needs to define three different image versions for each image file: a small and a medium-sized thumbnail, and a large image – for the latter, we use the originally uploaded image file. The code block below

[6] Andrey Kolkov, „Ngx-gallery," 2020, accessed at: 2021.04.27, `https://www.npmjs.com/package/@kolkov/ngx-gallery`.

 DOI 10.31400/dh-hun.2020.3.1475

demonstrates that if a thumbnail was uploaded together with the original image file, it will be used by the media viewer. However, the media viewer will use a thumbnail generated by *DSpace,* if one is available. In the absence of any thumbnail, we provide a general-purpose thumbnail *(replacement_image.svg).*

```
src\app\+item-page\media-viewer\media-viewer-image\
media-viewer-image.component.ts

/**
* This method convert an array of MediaViewerItem into
* NgxGalleryImage array
* @param medias input NgxGalleryImage array
*/
convertToGalleryImage(medias: MediaViewerItem[]): NgxGalleryImage[] {
 const mappadImages = [];
 for (const image of medias) {
  if (image.format === 'image') {
   mappadImages.push({
    small: image.thumbnail
     ? image.thumbnail
     : './assets/images/replacement_image.svg',
    medium: image.thumbnail
     ? image.thumbnail
     : './assets/images/replacement_image.svg',
    big: image.bitstream._links.content.href,
   });
  }
}
return mappadImages;
}
```

3. Then we display the resulting image list with the aforementioned Kolkov *Ngx-gallery* module:

```
src\app\+item-page\media-viewer\media-viewer-image\
media-viewer-image.component.html

<div [class.change-gallery]="isAuthenticated$ | async">
 <ngx-gallery
  class="ngx-gallery"
  [options]="galleryOptions"
  [images]="galleryImages"
 ></ngx-gallery>
</div>
```

4. When displaying audio or video content, the default HTML5 video player is used, which we supplemented with some extra features, such as a functionality

that skips to the next or previous audio or video file and another one displaying the relevant playlist. The following code snippet contains the initialization of the audiovisual component at the module level:

```
src\app\+item-page\media-viewer\media-viewer-video\
media-viewer-video.component.ts

ngOnInit() {
 this.isCollapsed = false;
 this.filteredMedias = this.medias.filter(
  (media) => media.format === 'audio' || media.format === 'video'
 );
}
```

5. The HTML initialization of the component displaying audio or video files mentioned in the previous section is contained in the following code snippet:

```
src\app\+item-page\media-viewer\media-viewer-video\
media-viewer-video.component.html

<video
 #media
 [src]="filteredMedias[currentIndex].bitstream._links.content.href"
 id="singleVideo"
 [poster]="
  filteredMedias[currentIndex].thumbnail ||
  replacements[filteredMedias[currentIndex].format]
"
 preload="none"
 controls
></video>
<div class="buttons" *ngIf="filteredMedias?.length > 1">
 <button
  class="btn btn-primary previous"
  [disabled]="currentIndex === 0"
  (click)="prevMedia()"
 >
  {{ "media-viewer.previous" | translate }}
 </button>

 <button
  class="btn btn-primary next"
  [disabled]="currentIndex === filteredMedias.length - 1"
  (click)="nextMedia()"
 >
  {{ "media-viewer.next" | translate }}
```

 DOI 10.31400/dh-hun.2020.3.1475

```
 </button>
 <div ngbDropdown class="d-inline-block">
  <button
   class="btn btn-outline-primary playlist"
   id="dropdownBasic1"
   ngbDropdownToggle
  >
   {{ "media-viewer.playlist" | translate }}
  </button>
  <div ngbDropdownMenu aria-labelledby="dropdownBasic1">
   <button
    ngbDropdownItem
    *ngFor="let item of filteredMedias; let i = index"
    class="list-element"
    (click)="selectedMedia(i)"
   >
    {{ item.bitstream.name }}
   </button>
  </div>
 </div>
</div>
```

6. Our developments related to the media viewer need to be enabled in the *DSpace* 7 frontend project (i.e. DSpace-Angular project) in order to become activated. Our developments can be enabled separately for image and audiovisual contents in the settings of media viewer in the *environment.common.ts* file.

Enabling the code from the *environment.common.ts* file:

```
mediaViewer: {
 image: false | true,
 video: false | true
}
```

In order to enable the media viewer one needs to set the relevant value(s) to *true*. In case one intends to fully or partially deactivate the media viewer, they need to set the relevant value(s) as *false*. As the developers of *DSpace* see serious added value in the media viewer, the module is likely to be available as activated by default in future versions of the software.