Introducing Pergamos: A Fedora-based DL System Utilizing Digital Object Prototypes

George Pyrounakis[1,2], Kostas Saidis[1], Mara Nikolaidou[2],Vassilios Karakoidas[2]
forky@libadm.uoa.gr,saiko@di.uoa.gr,mara@di.uoa.gr,bkarak@aueb.gr

[1] Department of Informatics and Telecommunications
[2] Libraries Computer Center
University of Athens
University Campus, Athens, 157 84, Greece

Abstract. This demonstration provides a "hands on" experience to the "internals" of Pergamos, the University of Athens DL System. Pergamos provides uniform high level DL services, such as collection management, web based cataloguing, browsing, batch ingestion and automatic content conversions that adapt to the underlying digital object type-specific specialities through the use of *Digital Object Prototypes* (DOPs). The demonstration points out the ability of DOPs to effectively model the heterogeneous and complex material of Pergamos. Special focus is given on the inexpensiveness of adding new collections and digital object types, highlighting how DOPs eliminate the need for custom implementation.

1 Introduction

Pergamos is the Digital Library System we developed for handling the heterogeneous and complex material of the University of Athens, originating from numerous sources, including the Senate Archive, the Theatrical Collection, the Folklore Collection and the Papyri Collection, to name a few. Pergamos is a web-based Digital Library implemented in Java that builds upon Fedora repository [2].

Pergamos provides a powerful digital object manipulation mechanism based on *Digital Object Prototypes* (DOPs) [1]. DOPs focus on the uniform resolution of digital object typing issues in an automated manner, releasing cataloguers, developers and collection designers, from dealing with the underlying typing complexity manually. All digital object typing information is expressed in terms of DOPs. The latter capture and express digital object typing requirements in a fine-grained manner, while they deploy a uniform "type conformance" implementation that makes all digital objects conform to their corresponding DOP specifications automatically. This way, the definition of new collections and respective digital object types is performed in a straightforward fashion, requiring no custom implementation or code development.

DOPs provide a detailed specification of: (a) the metadata sets used by the digital object type at hand (b) the digital content formats supported by this type, (c) the relationships in which instances of this type are allowed to participate and (d) the behaviors that all instances of this type should expose. DOPs

are defined in terms of XML. The `DO Dictionary` depicted in Figure 1, loads the DOP XML definitions during DL startup. It then translates the DOP supplied definitions into Java artifacts that are exposed to higher level application logic through the DOPs API. All digital objects are associated with DOPs. At runtime, the `DO Dictionary` loads stored digital objects from the underlying repository and generates their corresponding digital object instances that automatically conform to the object's DOP. The details of the underlying repository remain hidden as all application logic's functionality is directed through digital object instances. The ability to expose "typefull" instances to the services of the application logic allows us to generate single, uniform service implementations which are capable to operate upon any DOP-defined type of material.

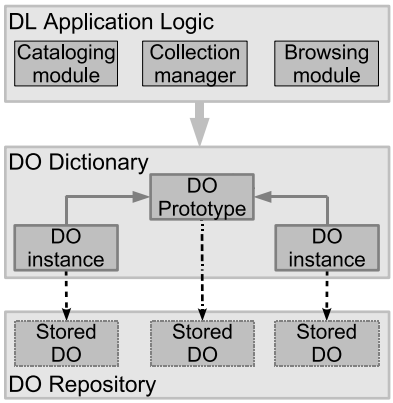

Fig. 1. Pergamos 3-tier architecture incorporating the "type enforcement" layer of DO Dictionary [1]

2 DOP-based Pergamos Features

The demonstration consists of a "mixed" viewpoint approach on Pergamos web based DL services, elaborating on end-user's and cataloguer's, designer's and developer's perspectives. The demonstration pinpoints how the use of DOPs allows us to deal with important DL development issues in a uniform yet adaptive manner. We particularly emphasize on the following Pergamos features.

Collection Management & Complex Objects

For uniformity reasons, we treat collections as digital objects. DOPs support aggregation relationships – the objects of one type are allowed to "contain" objects of another type. A collection object is allowed to contain other collection objects, generating a collection hierarchy. The root of the hierarchy is the Digital Library itself, a "super collection" object containing all other collections.

DOPs are defined in the context of a specific collection. For example, Folklore Collection consists of instances of the `Notebook`, `Chapter` and `Page` DOPs. DOPs are supplied with fully qualified identifiers such as `folklore.Page` and `folklore.Notebook`, allowing us to support user-defined types of objects with collection-pertinent scopes. Thus, although the collection of the Senate Archive's Session Proceedings consists of objects belonging to the `Folder`, `Session` and `Page` DOPs, the latter is distinguished from the `folklore.Page` through having the `senate.Page` fully qualified identifier.

The addition of a new collection refers to specifying the individual DOPs that model each different type of material this new collection supports. Additionally, the DL designer is able to add sub-collection objects in the collection at hand, specifying each sub-collection's supported DOPs recursively. This way, a collection is made up of the digital object instances belonging to the DOPs the collection supports. An instance is either "added" to the collection explicitly or implicitly, through belonging to a DOP of one of the collection's sub-collections. DOPs also support complex objects in a same manner. Senate Archive's `Sessions` are modelled as complex objects that are allowed to contain `Page` objects.

Metadata Handling & Cataloguing Capabilities

DOPs specify the metadata used for each different digital object type in a fine-grained manner. Each digital object type may contain one or more metadata sets for descriptive or administrative purposes. Each metadata set specification in a DOP contains one or more metadata element definitions. For each metadata element, a DOP provides: its identifier and multi-lingual labels and descriptions along with additional element characteristics that assist in the proper treatment of its values at runtime.

The behavioral characteristics we support are:

- `isMandatory`, that directs instances to forbid null values for the element,
- `isRepeatable`, that directs instances to render the element values in a list,
- `defaultValue`, that directs instances to supply this value to the element if the cataloger has not explicitly provided another value
- `validation`, that executes the user-supplied validation plugin for enforcing desired constraints on the element's value.

DOPs also support the definition of mappings among elements of different metadata sets. For example, the archival nature of the Senate Archive's Session Proceedings is modelled as follows. We use the `dc` and `ead` metadata sets for `Folder` and `Session` objects. `dc` refers to a qualification of the DC elements, while `ead` follows the principles of EAD without encoding the Finding Aid in its entirety. Our ability to define and handle metadata sets and their respective elements in a type-specific manner enables us to generate a uniform implementation of the web-based Cataloguing service that effectively copes with all Pergamos material. The Cataloguing service can generate detailed metadata element representations for all types of objects in a unified way by exploiting the specifications residing in the object's DOP.

Automatic Content Conversions & Batch Ingestion

DOPs provide a detailed definition of the file formats supported by each different object type. For example, the `senate.Page` DOP specifies that its instances should consist of a high quality TIFF file held for preservation purposes, a lower quality JPEG file used for web display and a thumbnail JPEG image used for browsing.

We use digital content specifications of DOPs to automate content conversions. Each file specification in a DOP is defined as `primary` or `derivative`. `primary` file format specifications provide conversion information that is used by the respective instances to automatically convert the `primary` file format to its corresponding DOP-defined `derivatives`. For example, the `senate.Page` contains conversion specifications that allows its instances to automatically generate the JPEG images from the high quality TIFF image, whenever the latter is either ingested or replaced by the user.

Moreover, we use DOPs to generate effective batch content ingestion for diverse types of objects. We model `senate.Sessions` as containers of `senate.Page` objects. The `senate.Session` DOP provides a `container` file format specification that allows `Session` instances to automatically create `senate.Page` objects from a suitable user-supplied zip file. The batch ingestion process is invoked when the user uploads a zip archive to a `senate.Session` instance. If the archive contains files that belong to the `senate.Page primary` format (TIFF), the `senate.Session` instance automatically creates new `Page` objects for each TIFF file. Then it saves each TIFF file to its corresponding `Page` object, triggering the `senate.Page`'s automatic conversions described above.

Browsing and Searching

The hierarchical structure of digital material generated by the use of DOPs is reflected in Pergamos web-based browsing facility. Although the Browsing service resides in a uniform implementation, objects belonging to different types are displayed according to their corresponding type's requirements. Browsing service fetches the `browseView` behavior on each instance and the latter interprets the call in a DOP-defined manner automatically.

Pergamos search capabilities reflect the ones provided by FEDORA. However, the use of DOPs allows us to provide additional search functionality to our end users, allowing them to limit search results on selected collections, sub-collections or types. Moreover, we use FEDORA's built-in DC-based searching to support cross-collection searches, yet we are able to provide enriched metadata to our end users by exploiting the mappings capabilities of DOPs.

References

1. K. Saidis, G. Pyrounakis, and M. Nikolaidou. On the effective manipulation of digital objects: A prototype-based instantiation approach. In *Proceedings of the 9th European Conference on Digital Libraries (ECDL 2005)*, pages 26–37, 2005.
2. T. Staples, R. Wayland, and S. Payette. The fedora project: An open-source digital object repository management system. *D-Lib Magazine*, 9(4), April 2003.