Lightweight Data Integration using the WebComposition Data Grid Service

Ralph Sommermeier[1], Andreas Heil[2], Martin Gaedke[1]

[1]Chemnitz University of Technology, Faculty of Computer Science, Distributed and Self-organizing Computer Systems Group, 09107 Chemnitz, Germany
[2]Microsoft Research Cambridge, CB3 0FB Cambridge, United Kingdom
[1]{firstname.lastname}@cs.tu-chemnitz.de, [2]v-aheil@microsoft.com

Abstract. With the advent of Web 2.0, the user becomes a producer creating lots of data by consuming the functionality of the respective Web applications. Even though more and more valuable data is created, it is difficult to reuse it due to lack of structure. In this paper we discuss easing data integration by using the WebComposition Data Grid Service (WebComposition/DGS). Our approach separates technology and information space concepts in a flexible and extendable component model, which yields simplicity for the end user. The model facilitates this by creating, managing and embedding data in different formats and representations to their used applications. Furthermore, machine-readable metadata is implicitly supported and used to link the internal data and external data sources together.

Keywords: WebComposition, Data Grid Service (DGS), Resource Description Framework (RDF), Metadata, Representational State Transfer (REST), Simple Object Access Protocol (SOAP), Service-oriented architecture (SOA)

1 Introduction

A growing number of different data types arise within the scope of Web 2.0 applications, which yield a lot of interesting information. This information becomes even more interesting if multiple data sources are linked together. The power of linked data highlights more intriguing information [1], [2]. The current problem lies in the re-usability of this data. To address this issue, it is required to implement at least one interface for each data source. Obviously, this implementation is time consuming and costly thereby making the linking of different data sources hard to realize. The WebComposition/DGS approach addresses this issue by simplifying writing and reading data as produced or consumed by a Web 2.0 application [3], [4]. This approach enforces concepts of meaningful URIs [5], [6] when creating information spaces by allowing all data to be implicitly addressed by URIs. Beyond that, the WebComposition/DGS natively supports Resource Description Framework (RDF) statements related to these data objects so that they can be annotated with metadata described in a machine-readable format.

In section 2 we examine the state of the art influencing our research. Section 3 shows our approach divided into three subsections. These describe the supported protocols, data formats and data referencing mechanism in the information space concept. In section 4 we discuss our experience with the implementation of components around the WebComposition/DGS to gradually compose and integrate data. Finally, section 5 summarizes our work with a view on future research activities.

2 State of the Art

Many Web 2.0 applications are valuable data silos that mostly provide the corresponding data in very simple formats. This data can usually be accessed for reading by transfer or transport protocols. Often Web 2.0 applications even provide ways for adding and updating data. However, the data is mostly bound within the Web 2.0 application and linking the data in the Web is in most cases very difficult as the data is often not systematically addressable by any URI. In fact, the data produced by its users and held by the Web 2.0 application is its sole asset, distinguishing it from other business rivals. As such, major engineering challenges address the question of how to access data in such silos by using the "best" *protocol* for reading and writing it in a systematic way. In addition, simplicity in "working" with the data, i.e. the *data formats*, and its corresponding metadata is another challenge to be addressed in the context of the Web 2.0 domain and in the context of systematically integrating data.

Protocols. Protocols, within the context of Web 2.0, are usually built on the Hypertext Transfer Protocol (HTTP). They define a set of rules which allow different components of a Web application to communicate with each other. To integrate each other's data, each of the participating components must be capable of understanding the particular protocol and is, as such, limited to the protocols it supports.

The Atom Syndication Format (ATOM) [7], [8] defines a format based on the Extensible Markup Language (XML), using HTTP for publishing and editing data of related resources. ATOM is a well adopted format for aggregating data mainly used by weblogs and wikis providing data through feeds. However, the capability of the ATOM format for writing, modifying and deleting data is barely used. While the format itself is extendable, there is no support for serving multiple representations of a resource. Consequently, the potential consumers integrating data using the ATOM format are forced to support the particular representation.

The Google Data APIs [9] provide simple protocols for reading and writing data on the Web, based on the Really Simple Syndication (RSS) and ATOM formats. The four basic functions *Create*, *Read*, *Update* and *Delete* (CRUD) for working with data [10] are fully supported through an interface using HTTP. Metadata is provided in the form of additional feeds containing referential information using, for example, the Google Base schema. However, this strategy is limited in terms of the fixed semantic information provided by the metadata feeds. This is an issue the Semantic Web [11] aims to solve: data integration and interoperability.

Less data centric protocols include the Simple Object Access Protocol (SOAP) and XML Remote Procedure Calls (RPC). These protocols are not limited to the use of HTTP and can be applied on top of different transport or transfer protocols. They do

not focus on the resource as the primary unit and are often used on a procedural oriented data exchange. By calling business logic SOAP and RPC provide high flexibility in terms of data exchange, however, they are often used in ignorance how the underlying protocols (for example HTTP) work.

While HTTP as a protocol has many advantages, it becomes evident that a solution to the data silos challenge requires not supporting one sole protocol, but as many different protocols as possible.

Data Formats. Plenty of data exists on the Web – data, which could be shared or linked together. However, data in the context of Web 2.0 is mostly under the sole control of the particular Web application and stored in application-specific formats. Limited access to this data or even just subsets of this data is provided only through a small number of restrictive protocols including those described previously. Examples are, the common classical Web 2.0 services for sharing movies (www.youtube.com), pictures (www.flickr.com), private information (www.myspace.com) or private experience and knowledge (www.ciao.com).

Weblogs provide chronologically ordered data, consisting of entries and different views (e.g. by topic) on the data. Wikis provide data with extensive change histories and references to data items that even do not yet exist. Both solutions store their data in platform and vendor specific formats, barely able to exchange. Limited access to the data is often provided using the ATOM or RSS format only. Nevertheless, for writing and modifying the data, the standardized capabilities of these protocols are ignored. Instead, dedicated programming interfaces are offered to access identical functionality. First attempts to establish standardized formats to interchange data between different platforms exist [12] but are not yet recognized by a wider community. The community of DBpedia [13], for example, currently extracts data from various sources, changing this unstructured data into a machine-readable format according to linked data principles [14].

It becomes evident that the simple formats and restrictive mechanisms of these different approaches need to be supported by a solution for integrating as well as annotating this data. As such, a mix of simple data structures and more sophisticated, dedicated data structures, that facilitate reuse and annotation, is needed. We believe that this is a mix of application-specific data structures, usually based on XML, and RDF for annotating the application-specific data.

3 The WebComposition/DGS Information Space

The WebComposition/DGS addresses the proposed approach by adopting a local concept of the global information space. This information space enforces the co-existence of data and corresponding metadata using the abstract components of *containers*, *information stores* and *information items*. Each addressable by a distinct URI to integrate all provided data and metadata within the global information space.

Each WebComposition/DGS is accessible via a dedicated URI that identifies the service as a resource containing so called information stores. These information stores are accessible through nested URIs of the superjacent WebComposition/DGS container. Each information store in turn contains information items which can be addressed by a nested URI and extended by a user-defined path segment. The proposed solution provides the automatic creation of URIs in the information space every time a new information store or item is added. Fig. 1 depicts the composition of these URIs in the information space.

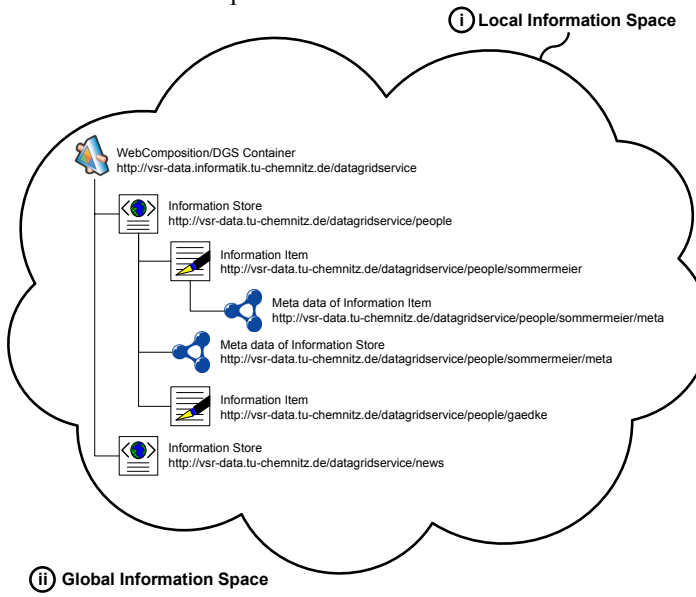

Fig. 1. Information space concepts within the WebComposition/DGS.

For each information store and item a corresponding store for metadata is maintained, which is referred to by extending the information space's URI with the additional path segment */meta*. This metadata describes the information store with all the relevant information semantically describing the data itself. The clear separation of data and metadata allows Web applications to easily access the data using simple protocols and mechanisms. Furthermore, the store for metadata stories is understood as a resource itself and can be addressed by its URI. Accordingly, data or metadata can be requested separately or be combined in terms of linked data [15].

Information Space. Each component within the local information space is addressed by its unique URI. Each sub-path of any particular URI, combined with the given authority, denotes a distinct URI identifying a unique resource within the path hierarchy thereby creating Semantic URIs [16]. Any resource within this information space is identified by its URI and the local information space in Fig. 1 (i) is spanned by one WebComposition/DGS container incorporated into the global information space of the World Wide Web in Fig. 1 (ii).

Container. The WebComposition/DGS service instance provides basic functionality to store, manipulate and easily query resources. The service is understood as a container comprising the functionality to be applied to the enclosed resource.

Information Store. The information store is a logical concept containing a set of related resources. Depending on the applied technology, the information store could be understood as a list, XML file, database or similar. Different implementations of information stores can be hosted within a single container at the same time. It is important to point out that the underlying technology and its evolution are transparent to the consumer of the service and do not affect the data integration.

Information Item. Information items represent the actual resources stored in an information store. Information items could be described in XML, a row in a relational database table, a file or an element out of a list. On a logical level, information items could even contain further information stores.

All components introduced so far outline the fundamentals of an easy data integration process. Besides the standard representation of data using XML, unstructured data, even binary data, is supported in the actual implementation. Enforcing a strict policy of how URIs are generated within the WebComposition/DGS results in any stored information, as well as its corresponding metadata, to be addressed by a dedicated URI. The possibility to access any data and metadata without exception is the fundamental concept that allows us to perform a standardized data integration lifecycle within the WebComposition/DGS.

4 Data Integration Lifecycle

One outstanding engineering challenge to overcome is to simplify handling data used by different types of common protocols in the context of Web 2.0. A data referencing mechanism is required for automatically creating data URIs for each information store, or item, which support the principles of linked data.

Data Referring. As information items are not necessarily bound by any entity (e.g., in form of a file), it is not possible to refer them natively by any URI. Therefore, the WebComposition/DGS provides the capability to create user-defined URIs by applying URI templates [17]. The URI of the information item's superordinate information store is extended with a path segment, which maps to any key that uniquely identifies the item within its native representation. This could be a primary key within a database, a line in a text file or a certain tag within an XML file. Similar to the information store's implementation this mechanism is transparent to the consumer of the service, which solely makes usage of the corresponding URI to integrate the corresponding data.

When using XML as a data format, we can make explicit use of XPath queries to retrieve a particular information item. The XPath query is mapped to the corresponding URI template and saved as metadata for the information store. Fig. 2 depicts the representation of a resource representing a person as XML using a Telnet

session to visualize the integration of the HTTP protocol and different content types. Fig. 2 (1) shows the execution of a HTTP *GET* to an information store accepting the content type *text/xml* request and the resulting XML data.

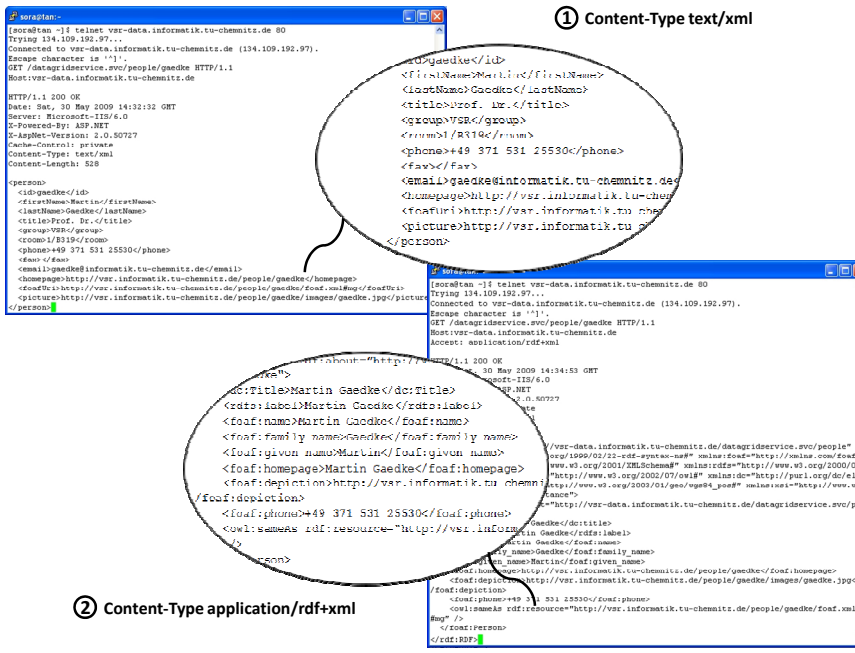

Fig. 2. Data referring using content negotiation.

Fig. 2 (2) shows the execution of a HTTP *GET* request to the same information store. However, here the difference is the accept header of *application/rdf+xml*. The resulting response provides a friend of a friend (FOAF) [18] resource (which is a RDF graph) and contains machine-readable data to be used in terms of linked data. The same result can be retrieved by adding */meta* to the original URI of the request without specifying the accept header. This allows the human user to retrieve the same representation of the data using a convenient mechanism. This mechanism, however, is not restricted to those two content types. Additional components can be implemented and specified to handle further representations of a resource. This characteristic is a fundamental capability to serve as many different data integrators as not all of them are capable of dealing with a single data format (cf. section 2).

Data Integration. The Telnet example above shows the technical realization of the data referencing mechanism using simple HTTP requests. The creation of information stores and information items, however, is not very convenient using these technologies. Data integration is more than simply providing structures of arbitrary data on the Web. Combining data from different sources as well as providing the user with a unified view of this data is an essential part of the data integration lifecycle.

For human use there is still a need for more user-friendly clients. The Telnet way is reasonable for demonstration but not for practical use. A dedicated component within

the WebComposition approach that addresses this issue is the WebComposition/Data Grid Service List Manager (DGSLM). This component, to be used in any Web browser allows creating, modifying or deleting data in the WebComposition/DGS information space. Using this component, we have the possibility to manage our information space via Web browsers without the need for programming. The component is build upon the unified interface of HTTP to read, write, manipulate and delete data. It uses data structures of information stores to dynamically create Web forms based on the metadata of that particular information store. Fig. 3 (a) illustrates the DGSLM displaying a list of information items from an information store.

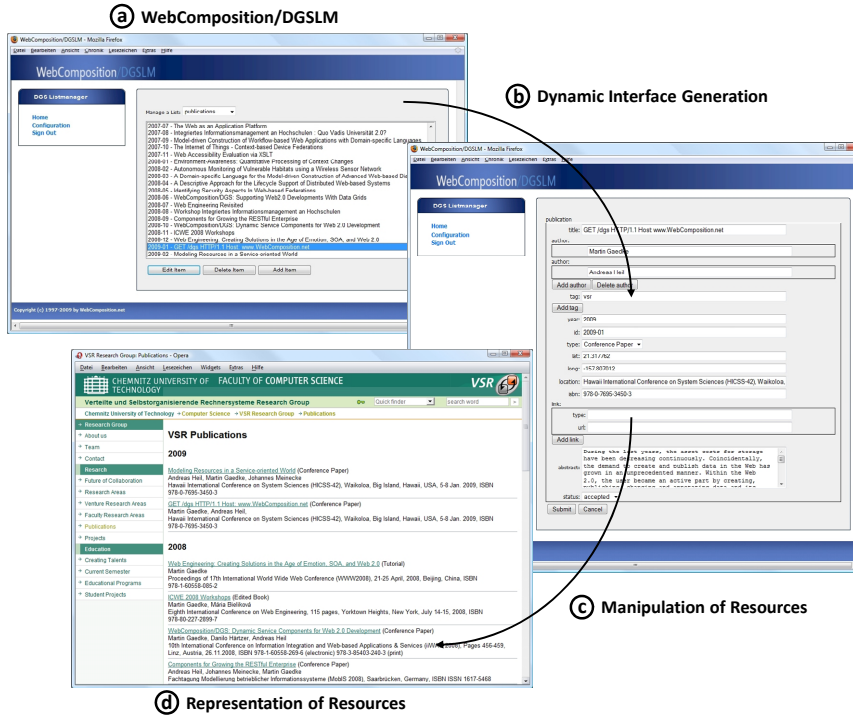

Fig. 3. Dynamic data integration lifecycle.

The corresponding responsibility of the WebComposition/DGSLM, is to provide the data representation independent from the underlying data structures. Data stored in databases, flat XML or binary files can be handled and accessed using a single, simple interface. By requesting a particular information item, a corresponding form is dynamically created Fig. 3 (b) that allows creating or manipulating new or existing information items.

To support the CRUD methods (cf. section 2) of the WebComposition/DGS, the WebComposition/DGSLM offers the complete functionality of HTTP to read, write manipulate and delete data. On behalf of the user the application creates corresponding HTTP request as defined by the endpoint, while the content of the request is dynamically allocated and sent to the WebComposition/DGS. Fig. 3 (c)

depicts representation of the previously created data. Extensible Stylesheet Language (XSL) transformations, also stored within the WebComposition/DGS information store are used to represent the data, in this example at the Website of the Chemnitz University of Technology.

The DGSLM provides the ability of easily managing information stores. It is not limited to a certain information store though. Hence, the WebComposition/DGSLM overcomes the typical difficulty of accessing multiple heterogeneous data sources from within a single Web application.

5. Conclusion and Future Work

For more than nine months, the WebComposition/DGS service is used in a production environment, using real, externally visible data of the research group Distributed and Self-organizing Computer Systems at the Chemnitz University of Technology, Germany. During this time, the data model was gradually exposed and extended with new resources, new representations and new components, according to the emerging needs of the group, and demonstrated the DGS's ability to deal with different representations of the data model for maximum reusability. Some parts of it were transformed from originally unstructured data, mainly managed with Wiki software before. This prior form of managing the data proved to be too hard for integrating and consuming outside of the Wiki itself. Therefore, a WebComposition/DGS implementation was used to successively expose data in accordance to Web standards and the REST principles. Over time, the data of publications, courses, projects, student projects and members of the research group were included. The data typically contains several hundred entries, describing historical and recent data. In addition, the data model was extended several times to accommodate for new information needs, to introduce links between different resources and to add new representations. The approach of encapsulating technologies in components appears to be an important factor for end user support. With the developed system, new data can be created ad-hoc. It is automatically editable in Web forms, without any scripting or code deployment. It can be integrated into Web pages without the need to know the involved internal components, transformations, protocols and formats. Whereas in our current system XML schemas and XSL transformations need to be specified when creating new data, the architecture allows adding more user-friendly components that automate this process in the future. The applied components also favor the reusability of the resources by automating the process of generating content representations according to the content negotiation. On the Website, the data could be integrated at multiple locations for realizing different views on it, e.g. on personal homepages, on project pages and on central group pages. Furthermore, the study illustrates the system's potential for bottom-up data growth [19]. As demonstrated, components were gradually added to the WebComposition/DGS, while the service was in productive use, i.e. integrated into the group's Website.

Future work includes the development of a publish/subscribe mechanism for every information store to support event driven linked data concepts. Another interesting open issue is the transparent handling of URIs as references to other data entries or machine-readable information on the Web with corresponding user interfaces.

An online example of the WebComposition/DGS and its corresponding downloadable components can be found at http://www.webcomposition.net/dgs. The source code is available via http://www.codeplex.com/webcompostition.

References

1. Linked Data, http://www.w3.org/DesignIssues/LinkedData.html
2. How to Publish Linked Data on the Web, http://www4.wiwiss.fu-berlin.de/bizer/pub/LinkedDataTutorial/
3. Heil, A. and Gaedke, M.: WebComposition/DGS: Supporting Web2.0 Developments with Data Grids. In IEEE International Conference on Web Services (ICWS), pp. 212-215. IEEE Computer Society, Los Alamitos (2008)
4. What Is Web 2.0, http://www.oreillynet.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html
5. Cool URIs don't change, http://www.w3.org/Provider/Style/URI
6. Cool URIs for the Semantic Web, http://www.w3.org/TR/2007/WD-cooluris-20071217/
7. The Atom Publishing Protocol – Requests for Comments: 5023, http://www.ietf.org/rfc/rfc5023.txt
8. The Atom Syndication Format – Requests for Comments: 4287, http://www.ietf.org/rfc/rfc4287.txt
9. Google Data APIs, http://code.google.com/intl/de/apis/gdata/
10. Kilov, H.: From Semantic to Object-oriented Data Modeling. In Proceedings of the First International Conference on Systems Integration, pp. 385-393. IEEE, Piscataway (1990)
11. W3C Semantic Web Activity, http://www.w3.org/2001/sw/
12. Völkel, M. and Oren, E.: Towards a Wiki Interchange Format (WIF) - Opening Semantic Wiki Content and Metadata. In First Workshop on Semantic Wikis (2006)
13. DBpedia, http://www.dbpedia.org/
14. W3C SWEO Linking Open Data Community Project http://esw.w3.org/topic/SweoIG/TaskForces/CommunityProjects/LinkingOpenData
15. Linked Data, http://www.w3.org/DesignIssues/LinkedData.html
16. Sahoo, S.S., et al.: Knowledge Modeling and its Application in Life Sciences: A Tale of two Ontologies. In 15th International Conference on World Wide Web, pp. 317-326. ACM, New York (2005)
17. URI Template, http://tools.ietf.org/id/draft-gregorio-uritemplate-03.txt
18. FOAF Vocabulary Specification 0.91, http://xmlns.com/foaf/spec/
19. Heil, A., Meinecke, J., and Gaedke, M.: Components for Growing the RESTful Enterprise. In Fachtagung Modellierung betrieblicher Informationssysteme, pp. 273-283. Springer, Bonn (2008)