A. Heil, M. Gaedke WebComposition/DGS: Supporting Web2.0 Developments With Data Grids IEEE International Conference on Web Services (ICWS 2008), S. 212 - 215, Beijing, China, 23.-26. September 2008, ISBN 978-0-7695-3313-1

WebComposition/DGS: Supporting Web2.0 Developments with Data Grids

Andreas Heil[†,*] and Martin Gaedke[†]

[†]Chemnitz University of Technology
Faculty of Computer Science
Chemnitz, Germany
{firstname.lastname}@cs.tu-chemnitz.de

[*]Microsoft Research
European Science Initiative
Cambridge, United Kingdom
v-aheil@microsoft.com

Abstract

The need to create, store, maintain, and share data as being part of Web2.0 labelled solutions requires the application of different skills and tools sets. The growing complexity of these new kinds of applications leads to cost and time-intensive development life-cycle because many data design and management decisions are directly influenced by distribution and semantic aspects. In this paper we propose the WebComposition Data Grid Service (WebComposition/DGS) as a supporting service to address these problems. The service acts as a highly integrative core component for data-centric applications, taking traditional SOA-based business scenarios as well as REST-architectural style applications in the Web2.0 context into account.

1. Introduction

A new wave of Web-based technologies allows users to create rich user experiences on the Web by building rich internet applications. Functionality and features known from conventional desktop applications are brought to the user’s Web client. Traditionally, these applications access data stored on a dedicated database that is in general responsible for hosting data sets for several different applications. This long and well-established approach provides common functionality such as presenting dynamic content generated from the information stored in the database. For small projects however, this means additional complexity in hosting and maintaining the database.

An increasing demand for not only creating and presenting but also linking, maintaining, annotating, and especially exchanging and sharing this data and information over the Web leads to a rising complexity in the engineering process for Web applications. New concepts must deal with these new “today’s standard” requirements.

We introduce the first component of the fourth generation of the WebComposition approach, considering the most valuable aspects in the various areas of Service Oriented Architectures (SOA), Representation State Transfer (REST)-style architectures [1], within the context of Web2.0 and the established technologies out of the field of the Semantic Web. The WebComposition approach was originally introduced during the WWW6 conference in 1997 as an object-oriented approach for Web Engineering [2]. Today, the WebComposition approach abstracts the development and evolution of Web-based solutions by composing Web components. The WebComposition approach itself only describes the development and evolution process, the concept of composing and reusing Web components. It does not address the concrete implementation technology.

2. Addressing the Content Perspective

2.1 Web Engineering

Major problems for the Web engineering process arise through the usage of data driven components. Subsequently, we will address these major issues and point out our design solution on how to address these problems using the WebComposition/DGS approach. The first problem we experience is the fact that the various solutions address two different engineering approaches; one based on business scenarios, the other within the Web2.0 context. There is no clear distinction as to which traditional engineering approaches applies to any of these scenarios. As depicted in Figure 2 we can distinguish between the two major bases. Business scenario based solutions often require integration within an existing SOA environment, while a REST-architectural style is typically applied within the Web2.0 context focusing on simplicity.

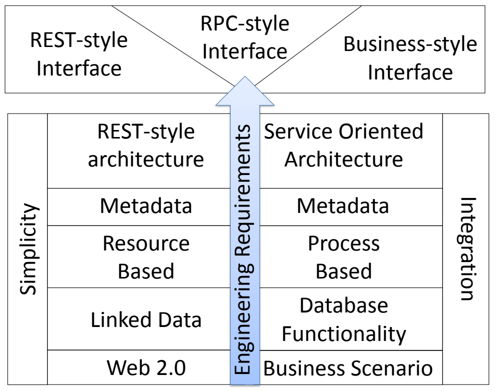

Figure 1. Different engineering approaches with the need for flexible but simple interfaces.

The proposed solution must provide the capabilities of application-centric solutions but must not be a stand-alone approach that makes integration difficult. Also the data management functionality should be similar to the ones provided by database driven approaches however, without the restrictions mentioned above. One important pattern to apply is the representation of data that we can explore as we follow link by link by fetching data. Most data available nowadays on the Web is not accessible in terms of linked data [3], yet.

2.2 System Design

The WebComposition/DGS acts as Web component for the content perspective within the WebComposition approach. It provides a data and metadata-centric component for building Web-based applications designed to meet the requirements for building today's Web applications. The WebComposition/DGS consists of several system components (cf. Figure 3) that can be easily extended and exchanged. Modifications on these components are transparent for the existing Web application using the particular WebComposition/DGS. The various core concepts of the WebComposition/DGS are discussed in the following sections.

2.3 Resources

Within the WebComposition/DGS, the notion of *resources* is a central idea. Any resource is understood as a representation of a concept that can be manipulated through a distinct set of actions. Each concept within a WebComposition/DGS instance is understood as a resource, including the service instance itself. Within the WebComposition/DGS concept we identify the following classes of resources:

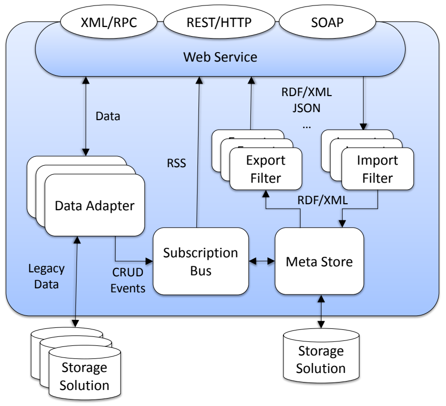

Figure 2. Main WebComposition/DGS components.

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

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, a CVS-file, a database or similar. Different implementations of information stores can be hosted within a single container at the same time.

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

Information space: Each resource is addressed by its unique HTTP-URI and provides metadata as the foundations for linked data. As a result, each sub-path of the particular URI, combined with the given authority, denotes again a distinct URI identifying a unique resource within the path hierarchy.

2.4 Manipulating Resources

Traditionally, the operations "create", "read", "update" and "delete" (also known as CRUD) are used to manipulate persistent data. If mapped to a REST-style like architecture, these operations are corresponding to the HTTP methods POST, GET, PUT and DELETE. Following [4], the methods are defined within the WebComposition/DGS as follows.

POST: The POST method is used to create a new resource on the service using the request URI. Within the WebComposition/DGS the post method is strictly used to create new resources in the form of information stores. The URI used for the request must not exist yet.

GET: The GET method is used to retrieve whatever resource is identified by the request URI. The GET method must be accomplished by the service without any side-effect on the resource.

PUT: In contrast to the POST method, the PUT method is used to modify information store resources. The enclosed information is added to the resource the request URI identifies or existing items are updated. If the request URI point to an information store, the enclosed item is added to the specified information store, otherwise the change is applied to the item, the request URI points to.

DELETE: The DELETE method requests that the resource identified by the request URI is removed as such. The DELETE operation could be applied to items as well as information stores as a whole.

A core feature of the WebComposition/DGS is its triple interface: the REST-style driven HTTP-based interface introduced before, an RPC-style driven interface operating on a straightforward RPC interface offering CRUD capability but also a complex business-style driven interface supporting complex SOAP-messaging, e.g. applied in document-driven business scenarios based on WS-Transfer and WS-Addressing.

2.5 CRUD Events – Publish-Subscribe for Linked Data

Adopting the terminology of the WS-Notification specification [5, 6], we provide an insight into the occurrence of events within the information space processed by the subscription bus (cf. Figure 3). Creation, modification, deletion or a simple request of a resource could raise *notifications*. The container or information store dispatching such a notification appears as a *notification producer*. The *consumer* receives the notification and could react accordingly. Consumers provide filters (not to be confused with WebComposition/DGS import and export filters) to express the particular interest in events. Filters are processed to decide whether to send a notification, or not, to the consumer by the notification for events: *create*, *read*, *update* and *delete* (CRUD). This information is called a *subscription* and can be seen as triple (producer, consumer, filter). Each container provides an individual information store for these subscriptions. The notification producer is identified by its URI, the producer by its respective call-back URI. Filters are stored as metadata and thus identified by their URI in the information space. Each subscription states a new resource and is addressed by its URI as well within the information space. Following the paradigm of linked data, this concept allows to step through this information. A history of CRUD events is finally provided as RSS feed through the subscription bus.

3. WebComposition Data Grid Service

3.1 Extensible Core Architecture

The three core interfaces originate the WebComposition/DGS endpoints depending on its client application and form a unified architectural model. These interfaces provide the core functionality to store, process, and query data and its respective metadata. This extensible core architecture provides two major components proposed to be extended: The *data adapter* provides the key interface to the data storage. The default adapter provided by the WebComposition/DGS is an XML-based data adapter that manages information stores and related meta-information as RDF data based in the file system. Each data adapter can provide a set of according *filters* for presenting and receiving metadata. Both the data adapter and the corresponding export filters to be used for a WebComposition/DGS instance are configured in a corresponding configuration file.

3.2 Metadata

The initial metadata created for a new information store is based on the Dublin Core Meta Initiative [7] and includes all fifteen elements endorsed by the related standards[1]. These common metadata attributes provide the fundamentals for the metadata model. This standardised information is supplemented by additional WebComposition/DGS specific metadata, based on our previous work and includes: Information about the latest modification of the resource, the current approval status of the resource, a custom index if required, information about unit costs and types as well as an interaction transaction index (ITX) meta-attribute to indicate the resource's current processing status. Updating resources appears to be one of the most crucial aspects to be considered carefully in REST-style architectures. It is not obvious to a user if a particular resource was updated after he requested a resource and modified it locally. Updating the resource might lead to a lost update in the resource. The ITX therefore provides a stateless mechanism to track updates on resources. When a resource is updated a new ITX is calculated and stored within the metadata, submitted along with the response of a GET request for

[1] ISO Standard 15836-2003 of February 2003
NISO Standard Z39.85-2007 of May 2007
IETF RFC 5013 of August 2007

A. Heil, M. Gaedke WebComposition/DGS: Supporting Web2.0 Developments With Data Grids IEEE International Conference on Web Services (ICWS 2008), S. 212 - 215, Beijing, China, 23.-26. September 2008, ISBN 978-0-7695-3313-1

resource and must be returned with the corresponding PUT request when the resource is updated on the server. If the ITX for the particular resource is different, the update is refused. Otherwise the update transaction is processed and the ITX is updated.

All settings for a certain information store (including data adapters, input and output filters) can be configured. Therefore, multiple data adapters can be mapped to different content types allowing a single information store to deal with heterogeneous data at the same time.

4. Lessons Learned

A first version of the WebComposition/DGS has already been fully implemented based on Microsoft's .NET technology. The existing version has been tested on Web servers on different Windows platforms (XP, Vista, Server 2003 and Server 2008). The test applications formed a heterogeneous environment where Web-based applications and also desktop clients have been used to successfully accessing the different endpoints of the implementation.

Beside the default XML-based data adapter, a prototypical implementation of an HTML data adapter was developed. This adapter allows creation and modifying of resources in the form of HTML Web pages. In contrast to a WebDAV based solution the WebComposition/DGS approach enables the user to benefit from the business-style SOAP interface to create and modify resources, even accessing endpoint where no direct HTTP connection is given.

The meta store was initially designed to be a highly flexible component. In a preliminary version users have been able to create and store meta-information in any desired way (including RDF). However, it became obvious that that particular metadata was often only meaningful to the creator of the information and due to its heterogeneity it was neither widely reusable nor suitable to follow the concept of linked data. The solution to restrict the internal representation of metadata based on the RDF standard provides more potential in processing and linking of the meta-information. The usage of additional filters for importing and exporting the data in user specific ways allows for the processing of the data into any desired format.

5. Conclusion and Further Work

The WebComposition/DGS is based on the experience of our previous research in applying the WebComposition approach and provides a highly flexible, yet easy to integrate solution for building data-centric Web applications. The WebComposition/DGS especially considers the need to address metadata and linked data, and provides an implicit mechanism that reduces the effort of dealing with these kinds of information.

The first version of the WebComposition/DGS is currently applied in various research projects and within one commercial project using it as a core component. Connecting it to idFS [8] will provide a sophisticated WS-* conform security concept. Subsequently, we extend the security concepts especially to the REST-style driven interface bearing in mind to keep the solution as simple as possible.

A Web-based demonstration application and corresponding documentation can be found at http://www.WebComposition.net/dgs/.

6. References

[1] R. Fielding, "Architectural Styles and the Design of Network-based Software Architectures", PhD Thesis, University of California, Irvine.

[2] H.-W. Gellersen, R. Wicke, and M. Gaedke, "WebComposition: An Object-Oriented Support System for the Web Engineering Lifecycle", in *6th International World Wide Web Conference*, Santa Clara, CA, USA, 1997, pp. 1429-1437.

[3] T. Berners-Lee, "Linked Data" - Website (2006): http://www.w3.org/DesignIssues/LinkedData.html (02-20-2008).

[4] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, P. Leach, L. Masinter, and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1" - Request for Comments: 2616 (1999): http://tools.ietf.org/html/rfc2616 (02-18-2008).

[5] D. Chappell and L. Liu, "Web Services Brokered Notification 1.3 (WS-BrokeredNotification)", 2006.

[6] S. Graham, D. Hull, and B. Murray, "Web Services Base Notification 1.3 (WS-BaseNotification)", 2006.

[7] L. Andresen, "Dublin Core Metadata Element Set, Version 1.1: Reference Description" - DCMI Recommendation (2004): http://dublincore.org/documents/dces/ (02-18-2008).

[8] M. Gaedke, J. Meinecke, and M. Nussbaumer, "A Modeling Approach to Federated Identity and Access Management", in *14th International World Wide Web Conference (WWW'05)*, Chiba, Japan, 2005, pp. 1156-1157.