Undefined 0 (2013) 1–0
IOS Press

The Information Workbench – A Platform for Linked Data Applications

Anna Gossen [a], Peter Haase [a], Christian Hütter [a], Michael Meier [a], Andriy Nikolov [a], Christoph Pinkel [a], Michael Schmidt [a], Andreas Schwarte [a] and Johannes Trame [a]
[a] *fluid Operations AG, Walldorf, Germany*
E-mail: firstname.lastname@fluidops.com

Abstract. In this system paper, we describe the Information Workbench, a platform for developing Linked Data applications. The platform features a highly customizable user interface to present the data to the users and realize different interaction mechanisms. UI development is based on Semantic Wiki technologies, enriched with a large set of widgets for data access, navigation, exploration, visualization, data authoring, analytics, as well as data mashups with external data sources. Widgets can be easily embedded into Semantic Wiki pages in a fully declarative way using a simple wiki-based syntax. In this way, the structure and behavior of the user interface can be easily customized to create domain- and application-specific solutions with little effort. Our paper describes the technical architecture of the Information Workbench platform, particularly focusing on the tight coupling between the domain-specific semantic data model and the elements of the user interface, which in turn facilitates UI customization. Furthermore, we describe examples of applying the Information Workbench platform to build applications with semantic interfaces for real-world use case scenarios in different domains, including data center management and semantic content publishing.

Keywords: Linked Data, Visualization, User Interfaces

1. Introduction

In recent years, a large amount of Linked Open Data (LOD) has been published on the Web, comprising over 200 data sets from domains such as media, geography, publication, government, life sciences, as well as cross-domain data [1]. Growing in size and domain coverage, this data becomes more and more interesting for building innovative applications that integrate heterogeneous data from different sources, with the goal to overcome the limitations of traditional data management systems. Apart from a new era of Web applications that exploit the large LOD corpus, this development offers opportunities to build novel applications for the enterprise by bringing together company-internal data with external data, thereby augmenting and contextualizing internal knowledge bases [6].

The development of specific applications that benefit from Linked Data often remains a time-consuming and costly task. First, at the data integration and management side, developers are faced with a variety of new data formats and query languages (such as RDF, OWL, and SPARQL). They also struggle with heterogeneity at the data level (facing Linked Data available via HTTP lookups, RDF dumps, and SPARQL endpoints), which requires various new database systems and tools to store, process, and access this data. Second, once the relevant data has been identified and integrated into the system, Linked Data applications require new data interaction paradigms to deal with the specific challenges – and opportunities – of the underlying data formats, such as schema flexibility and data semantics. In particular, leveraging the benefits of Linked Open Data requires the dynamic discovery of available data sources, seamless integration of Linked Data from multiple sources, provenance and information quality assessment. Third, end-user interfaces that implement generic visualization, exploration, and interaction paradigms for Linked Data without overwhelming the user with technical details of represen-

0000-0000/13/$00.00 © 2013 – IOS Press and the authors. All rights reserved

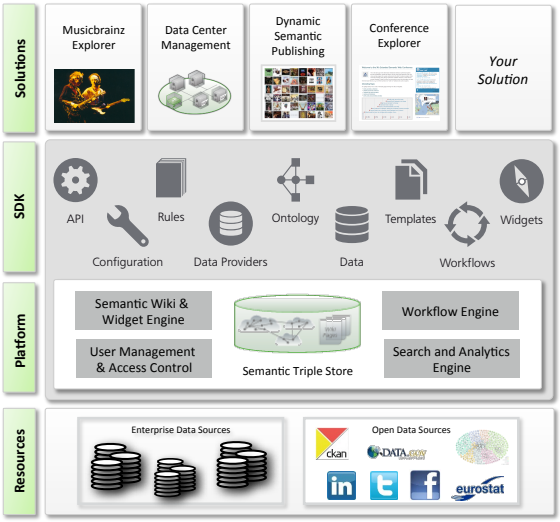


Fig. 1. Information Workbench Architecture

tation formats are important aspects when building domain-specific applications [4].

Pursuing the goal to lower the entry barrier into the world of Linked Data and to leverage its benefits, the Information Workbench[1] is an open platform for Linked Data applications. It accelerates the development process by abstracting from the technical details behind application deployment, data management, and UI customization. This paper presents the key concepts of the Information Workbench, focusing on the following novel aspects of the platform:

- Concepts for UI development based on a semantic data corpus
- Tight coupling between a domain-specific ontology and elements of the user interface
- Building semantic applications for real-world use cases in different domains

2. The Information Workbench

The Information Workbench aims at supporting the entire Linked Data application building process – from data integration to rich UIs that visualize information and allow user interaction. Figure 1 shows the high-level architecture of the Information Workbench.

Starting with the **Resources** layer, the platform offers built-in capabilities for integrating and converting data into semantic formats. Depending on the actual use case, this process may involve enterprise data sources or Open Data available on the Web. One way to integrate such data is via so-called *data providers* - configurable plug-in components, which periodically gather and refresh information from the source systems, convert it into RDF, and materialize the output in a central triple store following a data warehouse approach. The platform ships with standard providers for integrating RDF data from dumps or SPARQL endpoints, and also offers generic providers for common other formats and data sources (including RDBMS, XML and Web Services). Another option for importing data into the platform are predefined UIs and APIs. Data coming from the same source (e.g., gathered by a specific provider or entered in a single user edit session) are stored in a dedicated named graph in the central repository, which is annotated with relevant provenance metadata. This helps to address several data maintenance challenges: particularly, keeping the data up-to-date (new data gathered by a provider replace its old data) and realizing access control.

As an alternative to the data warehouse approach, the platform supports virtualized integration of local and public Linked Data sources using a federation layer provided by the FedX federated SPARQL query engine [9]. The advantage of this approach is that data does not need to be materialized in a central store – thus allowing on-demand access to *up to date* data as well as scaleout. The FedX federation layer does not rely on a pre-computed local metadata index describing federation member repositories, but is able to retrieve necessary information at query time, which guarantees complete and up-to-date query results. The costs of this approach in comparison with the data warehouse one include the time delays caused by network latency and federated query planning. By means of virtual mappings[2], federation techniques can applied to also integrate relational data sources [3].

On top of the (physically or virtually) integrated resource layer, the core **Platform** provides out-of-the-box modules and functionalities covering generic needs of Linked Data applications. Central in the Information Workbench architecture is the *Semantic Wiki & Widget Engine*, which builds upon Semantic Wiki [8] technologies: In the platform each resource in the data graph is automatically associated with a Semantic Wiki page, which enables visualization of and in-

[1] http://www.fluidops.com/information-workbench/

[2] http://www.w3.org/TR/r2rml/

teraction with the resource. Built-in mechanisms make it possible to access, extract and display information from the underlying data graph (e.g., by means of SPARQL queries) as well as to write resource-related data back to the store. Particularly the widgets – small UI components for visualization, exploration and interaction that can be declaratively embedded – are a key element for composing domain specific UIs and applications. Standard Semantic Wiki principles thus enable out-of-the-box functionality for managing and interlinking structured and unstructured content from multiple sources. To provide a homogenous view over resources of the same type, *template wiki pages* can be defined and associated with ontological classes. Such a template will then be applied to all resources belonging to the associated class. In this way, the structure of the underlying RDF data defines the composition of the user interface.

Complementing the *Semantic Wiki & Widget Engine*, the platform further provides customizable modules for *User Management & Access Control*, a *Search & Analytics Engine* and a *Workflow Engine*. While the latter enables declarative and data-driven collaborative workflows, the search engine supports the user in constructing expressive search queries, ranging from keyword search to complex graph-pattern based exploration. With these modules the core platform out-of-the-box provides the key technologies required for implementing most standard use-cases of enterprise-ready applications.

On top of the core platform, the Information Workbench provides support for developing, maintaining, and deploying specific applications. A comprehensive **SDK** (Solution Development Kit) allows developers to build customized applications targeting specific domains or use cases. As shown in Figure 1, the SDK comprises various components serving as customization and extension points in different dimensions. First, there are the data related components, including extensible *data providers* for data integration as well as general *data management facilities* for import, export and querying. These facilities allow solutions to integrate RDF data into the application's triple store, as well as to manage and modify *ontologies*. Second, the SDK provides components to build and customize the user interface of the application. On top of the Semantic Wiki & Widget Engine mentioned before, *templates* can be used to define the UI in a declarative manner. Here, developers are particularly supported by an extensive set of common purpose *widgets* for visualization, exploration and authoring. In case some special functionality is required in the target application, it is possible to implement custom widgets using the platform's AJAX framework. Finally, the SDK offers well-defined *APIs* and an extensive system *configuration* to automate processes and access the core functionality of the platform. By means of these APIs it is easily possible to define *rules* (for reacting on events or changes in the data), as well as collaborative *workflows*.

The solution development process is supported by the concept of packaged and reusable **Solution** artifacts that can be deployed by the Information Workbench through a RESTful API or by placing the zipped file into a dedicated deployment directory on disk. The deployment process then automatically installs the solution components on top of the core platform, converting it into a domain-specific and end user oriented application. This concept can be seamlessly used to install a combination of multiple solutions, assuming that the artifacts do not contain contradicting elements.

A solution may consist of the required system configuration (e.g., triple store configuration, ACL setup, or label and preferred language configuration), *data files*, *ontologies*, and *semantic wiki pages and templates*, which may contain widgets allowing the application end users to interact with the underlying data corpus. Advanced solutions may even ship own widgets and data providers and thus extend the platform with custom functionality, implemented on top of the SDK APIs through Java or Groovy code.

The *data files* are typically provided as serialized RDF file(s) (e.g., as NTriples) and used to bootstrap the system's repository. To organize the data and to make knowledge available as structured information, it is recommended to also provide an *ontology* describing the concepts and classes of the application scenario as well as the relationships between them. In the Information Workbench this information is for instance used to improve the user experience, e.g., to support the user with suitable auto-suggestions in data input forms.

From a developer's perspective, the solution mechanism is backed by a variety of platform features and best practices. Beyond mechanisms for versioning and upgrading solutions, there are best practices and guidelines for collaborative development, maintenance, and staging during the development process. The platform for instance offers a dedicated UI facility for exporting wiki pages from a running system, as well as an automated backup process for restoring old solution versions on demand.

Fig. 2. Sample widgets available in the Information Workbench.

As a running example for a solution built on top of the Information Workbench, we present a publicly available explorer for the MusicBrainz[3] database accessible at `http://musicbrainz.fluidops.net`. The system allows for browsing through the available database entities such as music artists, releases, or tracks. A user can start by searching for a band, say U2, using one of the pre-configured search widgets on the start page. On the band's page, the system visualizes various pieces of information related to the band specified by means of widget definitions. All U2 releases are, for instance, shown in a a timeline widget, which is constructed from the result of a simple SPARQL query. The user can then choose a specific U2 release and look, e.g., for information about some track on the release.

Technically, the underlying solution artifact consists of the Linked Data version of the MusicBrainz dataset[4] in combination with the Music Ontology[5] (provided by the system as RDF dumps), some system configuration settings (e.g. the login schema and access rights), plus a set of templates and widgets for defining the user interface.

3. Self-Service User Interface

The user interface of the Information Workbench comes as an extensible AJAX-based Web application that supports mashups on both data and UI level. This makes it possible to interlink heterogeneous data from multiple sources using different visualization and exploration techniques in a homogeneous *Living UI*. Knowing what, when, and where to show information is controlled by the data and realized through a flexible configuration mechanism.

The living UI can be composed in two different ways: First, widgets can be embedded into wiki pages providing 'inline' interaction components that enrich the unstructured content. Using wiki-based syntax this can be done in a fully declarative way. Second, the platform offers so-called *views*, basically tabbed UIs linked and configured for particular resources and classes. Consider, for example, a map widget that shows up automatically whenever geolocations are modeled in the underlying data. By default, there is a tabular view showing the underlying RDF triples associated to the current resource, a graph view visualizing the neighborhood of the resource, and a Pivot view for visual exploration.

Out of the box, the Information Workbench provides a rich set of UI components – the widgets – supporting different interaction paradigms with the data (see Figure 2). In the following we present the most relevant widgets targeting different dimensions.

Navigation and exploration. The user can explore underlying data in several ways. Standard table and graph views over resource properties realize Linked Data browsing functionality and allow users to navigate through the data graph by following references to linked semantic entities. To find relevant data in both the RDF triple store and the associated wiki pages keyword search is the natural paradigm. Here, customizable search widgets can be used to pre-configure parameterized SPARQL query patterns, allowing users to access structured data without specifying SPARQL queries (compare the configuration a) and the rendered result b) in Figure 3). Finally, SPARQL queries provide an expressive way to retrieve results for specific information needs in a structured form.

Data visualization. In the Information Workbench SPARQL queries serve as the basis for accessing structured RDF data from the triple store. Provided as part of the widget configuration, these define the data which are then presented to the user in the selected widget's form (e.g., table, bar chart, stock chart, map, timeline, etc.). These widgets rely on common libraries and APIs such as amCharts[6] and Google Maps API[7] to

[3] The system was developed as a Linked Data training system for the project EUCLID, as a continuation of the work in the UK JISC-funded project LinkedBrainz.

[4] `http://www.musicbrainz.org/`

[5] `http://musicontology.com/`

[6] `http://www.amcharts.com/`

[7] `https://developers.google.com/maps/`

generate the corresponding UI controls which are then inserted into the corresponding wiki pages.

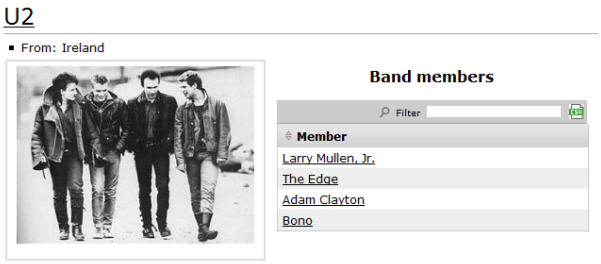
The second very important UI concept are *wiki templates* which describe how instances of ontology concepts are presented. A wiki template includes textual content as well as parameterized widget configurations, in which queries can use the actual instance as a parameter. These wiki templates are automatically instantiated for a particular instance of a class. In this way, in accordance with the visual analytics paradigm [7] ("intial analysis and overview first, details on demand"), the conceptual views of a class can provide a data overview and initial analysis results, while the instantiated templates provide details for the instances on demand.

The wiki template concept allows the user interface to be customized and to develop domain-specific solutions: customized templates are packaged as a part of the solution bundle and extend or override the default views. In the following we illustrate this concept for our running MusicBrainz example. The solution provides a wiki template for the type `mo:MusicArtist` that specifies how musicians are displayed. The template contains the name of the artist, his country of origin, an image, as well as a list of band members:

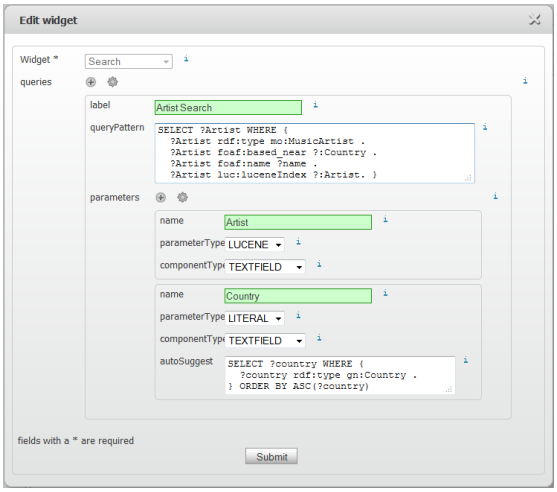
```
= {{#show: {{this}}| foaf:name}} =
* From: {{#show: {{this}}| foaf:based_near}}

{{#show: {{this}}| foaf:img | valueResolver=IMAGE}}

=== Band members ===
{{#widget: TableResult |
query = 'SELECT ?Member WHERE { ?? mo:member ?Member }' }}
```

We use the `#show` parser function defined in the Semantic MediaWiki to display the values for `foaf:name`, `foaf:based_near`, and `foaf:img` of the current resource. Also, the widgets are accessible as parser functions where configuration parameters (such as a SPARQL query) can be specified as named arguments. To access the URI of the current resource within parser functions, the magic word `{{this}}` can be used, while in SPARQL queries the syntactic placeholder `??` is applicable. When the user navigates to a particular music artist, say the band *U2*, the wiki page is instantiated automatically from the template and presents the desired information:

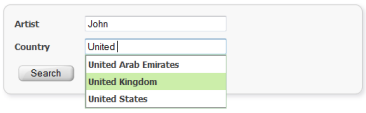
Following the self-service paradigm, the Information Workbench provides *wizards* that support the configuration of widgets, allowing non-expert users to build UIs without the need to specify wiki syntax. Consider for instance the configuration of the before-



mentioned search widget in Figure 3. This widget defines a parameterized query pattern that allows the user to search for music artists in a given country. To improve the user-experience an `autoSuggest` query can be provided which automatically populates the list of countries.



a) Wizard based configuration of a search widget



b) Rendered UI element

Fig. 3. Example of a search widget configuration: the configuration defines a search interface for music artists, offering the artist name and the country to be used as parameters.

Mashups with external services. In addition to presenting data contained in the RDF repository, special *mashup* widgets can be used to incorporate related information from external sources: e.g., to embed related videos from Youtube, to show up-to-date news items, or to present Twitter feeds. A mashup widget receives its input by selecting the relevant property of a data instance (e.g., the name of a music artist to search on

Youtube) and generates the HTML elements that display the corresponding external content.

Collaborative data editing. Finally, the platform provides customizable data input forms for collaborative editing with the triple editor. This widget provides a rich interface for curating, authoring, or adding information. It uses information from the schema ontology to select relevant editable properties of an instance and offers appropriate form elements for them (e.g., text inputs, drop-down boxes, multi-line text areas, or date-time pickers). The triple editor provides many configuration options to customize the form in the desired way. As an example, autosuggestion queries for properties can be provided to assist users in modifying data with suitable suggestions.

In this way, the Information Workbench provides a rich UI toolbox which can be customized for specific use case needs with a minimal effort.

4. The Information Workbench-based Applications

The Information Workbench platform serves as a basis for several productized application solutions covering diverse domain areas. The main benefits of building a domain specific solution with the Information Workbench are twofold. The developers can focus on domain-specific functionalities and custom configurations instead of building the application from scratch, while the application-specific data and ontology define the structure for the initial interface. The end users gain the opportunity to interact with the data from the start and tailor the UI to their specific needs at any time. In this section, we will provide an overview of some real-life solutions based on the platform and show the added value in each of them. Live examples of the Information Workbench are accessible at `http://www.fluidops.com/demonstrators/`.

4.1. Data Center Management

The eCloudManager [5] is a software solution built on top of the Information Workbench platform focusing on the task of enterprise cloud environments management. It provides users with the possibility to manage heterogeneous data center resources in an integrated way. These resources comprise the various layers of the IT stack: the *Hardware layer* (physical storage, network, and computing infrastructure), the *Virtualization layer* (virtual clusters, live migration, and other management capabilities), and the *Application layer* (applications and landscapes running on of virtualized resources). Information concerning all these resources is gathered by data providers, which make use of specific low-level APIs of each resource type, and expressed in RDF by means of the fluidOps Enterprise Cloud Ontology[8]. Using semantic technologies it is automatically integrated with other business resources, including, for instance, data about customers or system users from the Active Directory.

The user interface of the eCloudManager serves two main purposes: (1) presenting combined information collected from different sources concerning managed resources and (2) allowing the user to perform various control tasks (e.g., cloning a virtual machine from a template). Organization of the user interface is centered around the concepts of the eCloudManager ontology and helps to achieve both these goals intuitively. Semantic wiki pages corresponding to ontological classes contain dashboards providing overview information about all entities of a specific class: for instance, the wiki page associated to the concept *Host* contains a table showing all virtual and physical hosts in the data center, a pie chart showing the distribution of different VM types, and a menu with links to related concepts (e.g., servers and virtual machines). The wiki page template for particular instances of class *Host* combines detailed information about the specific object (e.g. a list of data stores linked to this physical host or a timeline chart showing the utilization of the host over time). In addition, it provides a control interface allowing the user to perform certain operations for the instance (e.g., start/stop/delete a virtual machine or a whole virtual landscape). The templates are composed of separate wiki modules encapsulating atomic UI subunits, thus allowing common interface elements to be reused across different pages.

While the eCloudManager itself represents a domain-specific application built on top of the Information Workbench platform, it needs to be easily adjustable with respect to additional customer requirements, i.e., to display additional related data or hide non-relevant functionality parts. The UI based on semantic wiki templates makes this possible with minimal effort: wiki pages and templates can be simply customized by the users at runtime. Using the concepts for application development sketched in the previous section, main-

[8] `http://www.fluidops.com/ontologies/ecloud`

taining these customizations becomes straightforward: modified wiki pages and templates are exported and encapsulated into a customer-specific solution bundle, which at the installation time extends or overrides the default resources.

4.2. Semantic Authoring and Publishing

In the context of Dynamic Semantic Authoring and Publishing, the Information Workbench supports content producers and content consumers to author, curate and publish ontology and instance data following an editorial workflow. For example, for the 2012 Olympics we have developed a solution to support the editorial process of creating, editing, and publishing data around sports competitions and athletes[9]. The solution is built around the BBC Sports Ontology[10], covering concepts such as athletes, teams, events, competitions, and disciplines. For each of these concepts, wiki page templates have been composed that visualize available information and also allow for easy curation, editing and interlinking using intuitive data forms. The template for the concept athlete for instance shows an athlete's profile with the help of visualization widgets and provides a form to edit all or selected data.

The semantic authoring and publishing process is supported by an editorial workflow. By means of different user roles, the editorial workflow allows to model multi-level internal approval processes, such that, e.g., data journalists may consume the data, subeditors can see and edit data, and data architects can decide if and when the data are published on the website. An important part of this solution thus is the definition of the wiki page templates and the editorial workflow with the respective access control mechanisms that realize logical views for users depending on their assigned role.

In the editorial process, end users have different roles with specified permission levels assigned (e.g., data journalists who consume the data, subeditors who can see and edit data, and data architects who decide if and when the data is published on the website). Using a review and approval process, data are only published after a responsible data architect performs the respective action.

4.3. Conference Explorer

The Conference Explorer solution is a one-stop-shop for conference visitors, providing information about conference events, presenters, publications, and discussions. It integrates publicly available conference information, metadata about publications from DBLP, as well as data from social networks such as LinkedIn, Twitter, and Facebook. The solution is built around existing ontologies for the conference domain (Semantic Web Conference Ontology[11] and Semantic Web for Research Communities Ontology[12]) as well as other standard ontologies (FOAF, Dublin Core, and ICAL).

Many conference series (including ISWC, ESWC, WWW, and Semtech) already publish their metadata according to these ontologies. With this solution, creating a Linked Data frontend for a new conference simply requires to point the data provider to the respective endpoint (or RDF dump) of the conference metadata, as well as to set up providers that augment the conference data with social network data (e.g., from the public profiles of the presenters or the event itself).

Fig. 4. A conference page of the Conference Explorer solution

The conference explorer UI comes with a set of wiki pages and templates that allow for intuitive browsing through the data. As illustrated in Figure 4, it embeds a rich set of different widgets, including event timelines, tag clouds depicting the conference topics, graph and pivot views for visual exploration, and mashups with external data sources. In addition, it provides detailed statistical information about the conference on special dashboards, showing, for instance, the demographics of presenters.

[9] http://www.fluidops.com/solutions/semantic-authoring-publishing/

[10] http://www.bbc.co.uk/ontologies/sport/

[11] http://data.semanticweb.org/ns/swc/ontology

[12] http://ontoware.org/swrc/

5. Lessons Learned

Having presented some example scenarios in the previous section, we now want to summarize some technical insights and our experiences. When building Linked Data applications, it is important to know about technical limitations. One limitation is the reliability and serviceability of public SPARQL endpoints. We empirically found that most of today's endpoints have deficiencies in one or more of the following dimensions [2]: conformance to the exchange protocol, service uptime, query execution performance, completeness of result sets, and latency. To overcome these limitations, we recommend using SPARQL endpoints in controlled environments (e.g., in a local intranet) that can guarantee reliability for interactive applications with a strong coupling between UI and database.

A second technical limitation becomes relevant when dealing with large amounts of highly dynamic data. In these scenarios the warehousing approach does not scale, and alternative options such as virtual integration with federation technologies are still in their infancy. Here it is a challenge for the research community to advance technologies over the next years.

With the Information Workbench we offer a platform that enables advanced end users (such as data scientists) without deep programming skills to create custom applications. E.g., in the *dynamic semantic publishing* use case the Information Workbench is used by data journalists and data architects as a semantic backend for authoring and publishing data in an editorial workflow. The *data center management scenario* is tailored towards enterprise clouds environment management and as such is primarily used by system administrators.

In summary, the benefits for using the Information Workbench as a platform for building Linked Data applications are:

- The comprehensive SDK (components, tools, development process)
- Very fast setup and guided application building (which does not necessarily require programming skills)
- Immediate view on any RDF data through default template pages
- Customization and extensibility at various points

6. Conclusions and Outlook

We have presented the Information Workbench as a platform for Linked Data applications. It is available as a Community Edition under an Open Source license as well as an Enterprise Edition with a commercial license. The platform provides a generic frontend for visualizing and interacting with Linked Data, which can be further customized and extended for domain specific applications through an SDK. On the UI side, the customization is done in a completely declarative way, resorting to a rich pool of widgets and creating template pages in wiki-syntax, which are associated with elements of domain ontologies. In our experience, this significantly simplifies and speeds up the application development. Following a wiki-based approach, even the end user is enabled to customize the UI, to create personal dashboards, etc.

So far we – as well as numerous partners and customers – have developed Linked Data application in a variety of domains, including media, sports, events, data center management, pharma, and others. Wherever possible, these applications are based on established and standardized ontologies. As of today, we have templates available for dozens of ontologies with hundreds of concepts. The solutions developed with the SDK are highly reusable artifacts that can be easily combined and extended for new applications. The ultimate goal is that components for ontology-based Linked Data applications can be shared and reused just as the ontologies and the data. Towards this goal, we plan to provide an open repository of reusable solutions as a Linked Data App Store.

Further, in a number of projects we are extending the platform in various other directions. For example, in the project Optique[13] the Information Workbench is used as the basis for developing a platform for scalable end user access to Big Data, integrating techniques for Ontology-based Data Access (OBDA) [3] over highly distributed and heterogeneous relational data sources. One of the focus areas are user interface techniques that enable the end user to pose on-demand, ad hoc analytical queries. In the project Durchblick[14], we are extending the platform towards Augmented Reality applications and supporting new end-user devices such as Google Glass and similar Head Mounted Displays. In the scenario of the conference explorer, we will thus be able to proactively deliver information based on the current context of the conference visitors via Google Glass.

[13] http://www.optique-project.eu/

[14] http://www.fluidops.com/projects/durchblick/

References

[1] C. Bizer, T. Heath, and T. Berners-Lee. Linked Data – The Story So Far. *Int. J. Semantic Web Inf. Syst.*, 5(3):1–22, 2009.

[2] C. Buil-Aranda, A. Hogan, J. Umbrich, and P.-Y. Vandenbussche. SPARQL web-querying infrastructure: Ready for action? In *International Semantic Web Conference – ISWC 2013*, 2013.

[3] D. Calvanese, M. Giese, P. Haase, I. Horrocks, T. Hubauer, Y. Ioannidis, E. Jiménez-Ruiz, E. Kharlamov, H. Kllapi, J. Klüwer, M. Koubarakis, S. Lamparter, R. Möller, C. Neuenstadt, T. Nordtveit, O. Özcep, M. . Rodriguez Muro, M. Roshchin, M. Ruzzi, F. Savo, M. Schmidt, A. Soylu, A. Waaler, and D. Zheleznyakov. The Optique Project: Towards OBDA Systems for Industry. In *OWL Experiences and Directions Workshop (OWLED)*, 2013.

[4] A.-S. Dadzie and M. Rowe. Approaches to visualising Linked Data: A survey. *Semantic Web Journal*, 2:89–124, 2011.

[5] P. Haase, T. T. Mathäß, M. Schmidt, A. Eberhart, and U. Walther. Semantic technologies for enterprise cloud management. In *International Semantic Web Conference – ISWC 2010*, 2010.

[6] M. Hausenblas. Exploiting linked data to build web applications. *IEEE Internet Computing*, 13(4), 2009.

[7] D. A. Keim, F. Mansmann, J. Schneidewind, J. Thomas, and H. Ziegler. Visual analytics: Scope and challenges. In S. J. Simoff, M. H. Böhlen, and A. Mazeika, editors, *Visual Data Mining*, volume 4404 of *Lecture Notes in Computer Science*, pages 76–90. Springer, 2008.

[8] M. Krötzsch, D. Vrandecic, and M. Völkel. Semantic mediawiki. In *The Semantic Web - ISWC 2006*, pages 935–942, 2006.

[9] A. Schwarte, P. Haase, K. Hose, R. Schenkel, and M. Schmidt. Fedx: Optimization techniques for federated query processing on linked data. In *International Semantic Web Conference – ISWC 2011*, pages 601–616, 2011.