KiWi – A Platform for Semantic Social Software

Sebastian Schaffert, Julia Eder, Szaby Grünwald, Thomas Kurz, Mihai Radulescu, Rolf Sint, Stephanie Stroka

Salzburg Research Forschungsgesellschaft
Jakob Haringer Str. 5/II, A-5020 Salzburg, Austria
firstname.lastname@salzburgresearch.at

Abstract. Semantic Wikis have demonstrated the power of combining Wikis with Semantic Web technology. The KiWi system goes beyond Semantic Wikis by providing a flexible and adaptable platform for building different kinds of Social Semantic Software, powered by Semantic Web technology. This article describes the main functionalities and components of the KiWi system with respect to the user interface and to the system architecture. A particular focus is given to what we call "content versatility", i.e. the reuse of the same content in different kinds of social software applications. The article concludes with an overview of different applications we envision can be built on top of KiWi.

1 Motivation: From Semantic Wikis to KiWi

Semantic Wikis have been under investigation in the research community since 2005 (see e.g. [1,2]). Although there are many different systems with many different properties, a common trait between all Semantic Wikis is that they aim to combine ordinary wiki content and technology with Semantic Web technologies, either in order to provide better wikis ("Semantic Web for Wikis") or to ease the creation of Semantic Web data ("Wikis for the Semantic Web") [3]. Many also see Semantic Wikis as the "Semantic Web in a Nutshell", because – like Wikis show similar traits to the Web as a whole – Semantic Wikis share many properties and also problems with the envisioned Semantic Web.

The EU-funded project KiWi ("Knowledge in a Wiki")[1] takes the Semantic Wiki approach to the next level by providing a platform that allows to build many different kinds of Social Semantic Software, based on the conviction that most social software follows the "Wiki Principles". By "Wiki Principles", we mean that the term "Wiki" does not refer to technology alone. It is more a new philosophy of working with Web content influenced by ideas in the OpenSource community. These principles have revolutionised the way we work with content in the (Social) Web:

- **Wikis allow anyone to edit:** The core principle is that there are no access restrictions or strict hierarchies on the content of a wiki. Anyone can easily contribute his or her own knowledge, his or her own ideas, and his or her own content.

[1] http://www.kiwi-project.eu

– **Wikis are easy to use:** Anyone who is sufficiently familiar with the basic functionalities of word processing software (write, delete, save) has all the skills required to edit, correct and expand a wiki.
– **Wiki content is linkable:** By allowing users to create links between words and as such between concepts, wikis also allow for the creation of semantic relations, i.e. of meaning.
– **Wikis support versioning:** Never does information disappear on a wiki. If a page is edited, the previous version is still stored somewhere. This has an important psychological effect as it takes away the wiki writer's block: the fear that something might get lost through editing.
– **Wikis support all media:** Wikis are web-based. So whichever type of content you have, be it text, images, audio, spreadsheets, documents – anything that can be displayed in a web browser can be displayed in a wiki. And even if a file cannot be displayed in the browser itself, it can still be downloaded.

This same philosophy is underlying not only Wikis (technically-speaking), but also a large array of other Social Software applications: e.g., a weblog or social networking platform can be seen as just a different user interface (and different way of using), but is otherwise very similar concerning the underlying principles and also technology.

In the following, we describe how we realised the KiWi vision in the *KiWi System*, a generic Semantic Social Software platform based on the Wiki principles. We begin in Section 2 with introducing the concept of what we call "Content Versatility": content with flexible structures that can be reused in different applications. In Section 3, we then describe KiWi Core Applications: the Wiki, the Dashboard, TagIT, the KiWi Search, and the KiWi Inspector. Section 4 is dedicated to the more technical aspects of the KiWi platform: the system architecture, the data model, the KiWi entity manager, and façading. We conclude this article with an outlook and summary in Section 5.

2 Content Versatility: Same Content, Different Views

As we have already outlined in the introduction, what we call the "Wiki Principles" is actually applicable to many Social Software applications. It is therefore close at hand to build generic platforms for developing different kinds of Social Software. And indeed, there are several products already available that aim to deliver a platform that allows to combine wikis, weblogs, social networking, etc. Such platforms are for example Clearspace Community[2] from Jive Software, Community Server[3] from telligent, and Liferay Social Office[4] from Liferay.

However, all these systems only provide integration on the user interface level and still see wiki articles, blog posts, etc. as separate kinds of content that is visualised in a specific way. This has a number of serious disadvantages. For

[2] http://www.jivesoftware.com/products/clearspace-community
[3] http://communityserver.com/
[4] http://www.liferay.com/web/guest/products/social_office

instance, something that started as a wiki article can never become a blog post, it will never be possible to attach comments to a wiki article if not foreseen in the data model, and new kinds of content (like our TagIT locations, cf. Section 3.3) cannot be added easily without also doing fundamental changes to the system.

KiWi follows a completely different approach which we call "Content Versatility". The underlying principle is that every piece of information is a combination of human-readable content and associated metadata, and that the same piece of information can be presented to the user in many different forms: as a wiki page, as a blog post, as a comment to a blog, as a photo, or even in a bubble in a map-based application. The decision how the information is displayed is taken based on the metadata of the content, and the context of the content and the user (e.g. metadata, user preferences, different applications). Metadata is represented using RDF and thus does not require a-priori schema definitions, so the data model of the system can be extended even at runtime.

In KiWi, we call the smallest piece of information a "content item". A content item is identified by a URI and consists of human-readable content in the form of XHTML and associated metadata in the form of RDF. The KiWi core system provides means to store, update, search, and query content items, and offers automatic versioning of all updates (content and metadata). Whereas core properties of a content item (like the content, the author, and the creation date) are represented in XML and persisted in a relational database, all other properties can be flexibly defined using RDF properties or relations. The URI of a content item is generated in such a way that it is possible to make a KiWi system part of the Linked Open Data cloud [4]. This allows to easily integrate KiWi content with other services on the Semantic Web.

3 KiWi Core Applications

Content Versatility makes it possible to offer completely different views on the content inside the KiWi system without any modifications to the core system or data model. We call such views "KiWi Applications", and they are one kind of extension offered by the KiWi system (others are currently: KiWi Services, KiWi Widgets, KiWi Actions, and Exporters/Importers). In the following, we describe the set of applications that are part of the KiWi System to illustrate how the Wiki Principles and Content Versatility are realised in KiWi. The applications have been selected based on the requirements identified in the two KiWi use cases and accompanying SNML projects; additional applications are conceivable and reasonable. Additional applications will be offered as separate packages in later stages of the project.

KiWi applications share the same context, i.e. when the user browses a content item in the Wiki or Dashboard, she can switch to the TagIT application and view the same content item on a map or to the Inspector and get debugging information on the content item.

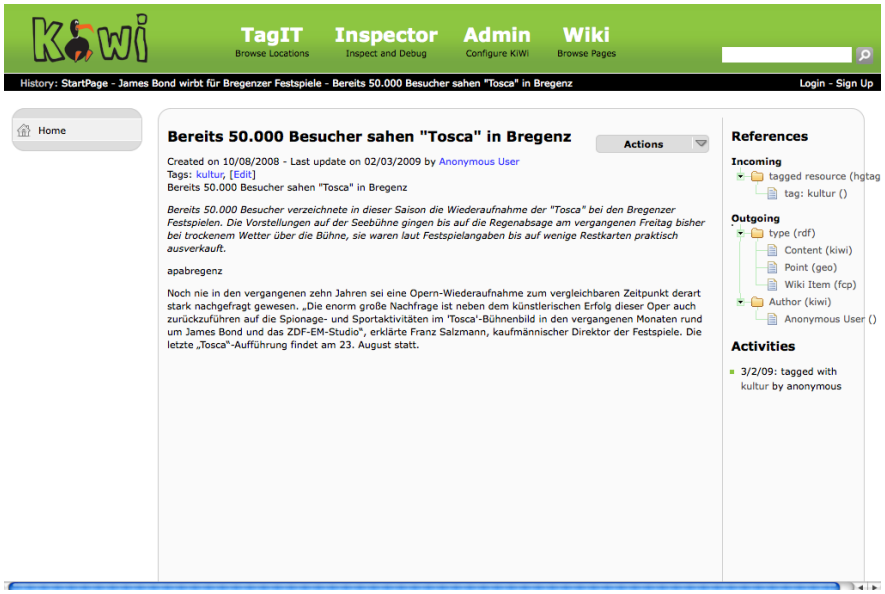

Fig. 1. The KiWi Wiki: A Semantic Wiki built on top of KiWi

3.1 The KiWi Wiki

The primary and most generic interface to the KiWi system is a Semantic Wiki. The layout and functionality (Figure 1) of the KiWi Wiki is inspired by its predecessor IkeWiki[5]: the left column offers navigation functionality, the centre column contains the main (human-readable) wiki content, and the right column contains dynamic widgets that display additional information about the content item based on its metadata (e.g. a map or incoming and outgoing links).

The centre column by default displays the content of the content item. The section below the page title contains maintenance information about the item as well as the list of tags that have been associated with it. By clicking on the "Action" menu in the top right corner, the user can select to edit the content, edit the metadata (i.e., OWL Datatype Properties), edit the relations (i.e., OWL Object Properties) to other content items, edit the list of tags, and access the history of the item's content and metadata as provided by the versioning subsystem. In principle, the KiWi Wiki interface can thus be used in the same way as IkeWiki. Additional actions can be registered using KiWi's extension mechanism (e.g. domain specific actions like "Geolocate" or "Share").

The widgets in the right column are visually part of the content box to emphasise that they contain additional information about the currently displayed item. Currently, the KiWi system offers to display the list of references (based on RDF relations with other items) and a "Stream of Activities" listing the

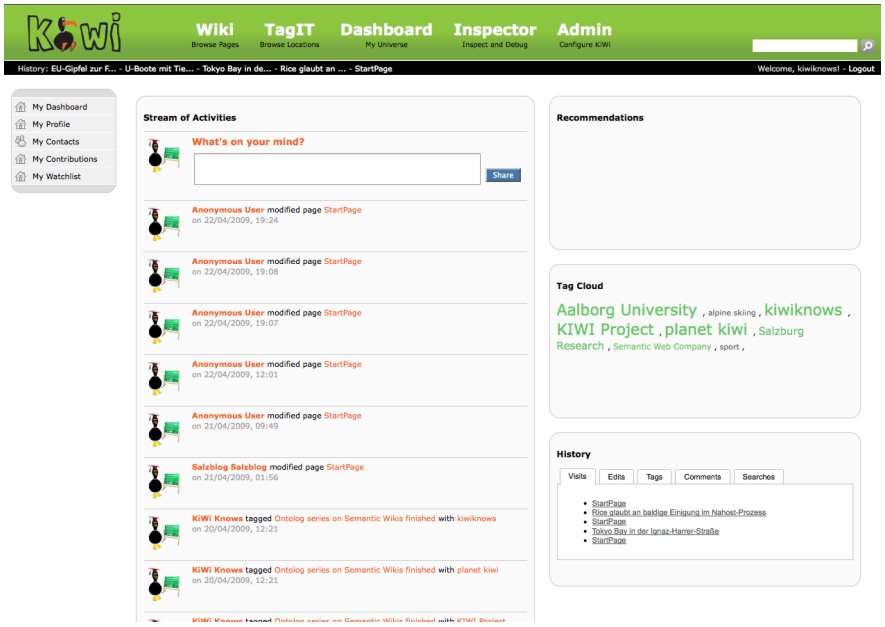

Fig. 2. KiWi Dashboard: personal user page with stream of activities, history, personal tag cloud, social networking, and recommendations

recent activities associated with the content item (e.g. modification, tagging, annotation).

3.2 The Dashboard

The Dashboard is a user's personal(ized) start page in the KiWi system. Figure 2 shows an early stage of its user interface, which is currently still under development. The Dashboard follows the same general layout as the Wiki, i.e. the left column provides generic navigation while the centre and right columns contain the actual content. While the look of the Dashboard is freely customisable by the user, the KiWi core system by default provides the following information:

- the *Stream of Activities* is the most important part of the Dashboard: it contains an aggregated list of activities that happened in the user's "universe", i.e. updates to content items that are either explicitly watched by the user or implicitly added to the user's universe e.g. because they have been edited by the user, because they have been edited or rated "good" by one of the "friends" of the user, or because they have been recommended to the user based on previous activites by the personalisation component of KiWi
- the *Recommendations* widget provides a list of additional content items that might be relevant to the user; different recommendation algorithms are investigated as part of the KiWi project [6]

Fig. 3. TagIT: description of locations / geolocation of content; double circles indicate clusters of points, single circles single points of interest

- the *History* widget lists the content items the user visited or worked on recently to give quick access to the issues the user is currently concerned with
- the *Tags* widget lists the tags used by the user and is a quick and flexible means of structuring and accessing the content items that are relevant to the user; clicking on a tag redirects to the search interface described below

Besides the main view, the Dashboard is also the place where the user manages his own profile. The most important part of the user's profile is the list of friends, which is the primary way to use the social networking functionality of KiWi. The requirements analysis carried out as part of the KiWi use cases showed that social networking is a crucial aspect in a modern knowledge management system like KiWi as it helps define the context the user works in.

3.3 TagIT

TagIT is an application originally developed in a separate project with the goal to create the "Youth Atlas of Salzburg"[5], which has been running as a prototype successfully for over a year and has now been ported to the KiWi platform (Figure 3). In TagIT, users browse through a map (based on Google Maps) where they can "tag" locations with descriptions and provide interesting information

[5] http://tagit.salzburgresearch.at

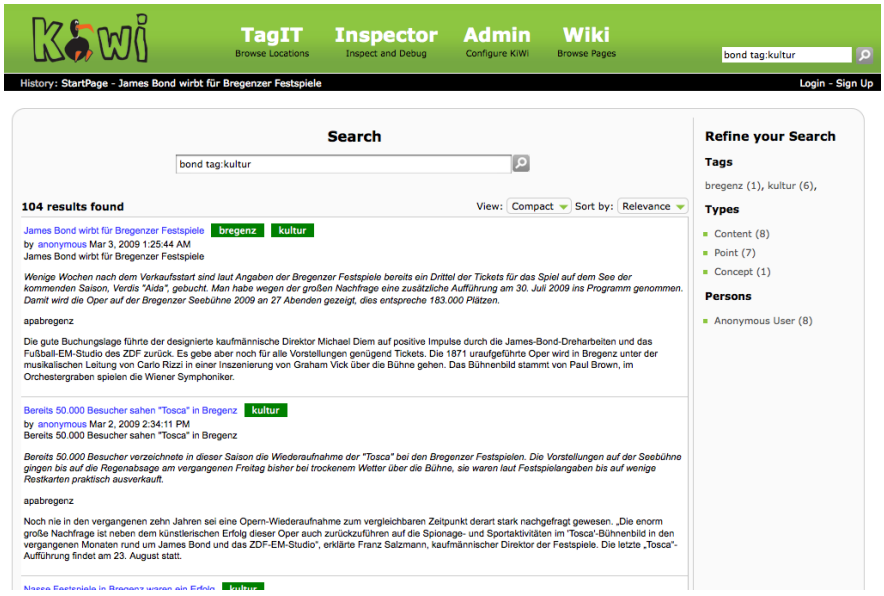

Fig. 4. KiWi Search: generic, facetted search over KiWi content

about them, e.g. cafés, bars, sports parks, hiking tours, etc. Such tags can be associated with categories from a SKOS thesaurus and with additional multimedia data like photos or videos. Other users can then browse or search for interesting locations using the same interface. In addition to the web-based platform, TagIT also offers a mobile client that can run on a GPS-enabled mobile phone. When visiting an interesting location, users can then start the TagIT application, take a picture of the location, add a short description and directly upload the "tag" using UMTS. The tag is automatically geolocated and immediately available for other users.

Although quite different on the user interface and in the way it is used, TagIT actually closely follows the wiki principles: everyone can add and edit tags, the system is easy to use, tags can be linked, tags are versioned, and different kinds of content are supported. On top of KiWi, tags are realised as content items and can thus be displayed in both the TagIT user interface and the previously described KiWi Wiki (in which case a small map widget is displayed in the right column showing the position).

3.4 KiWi Search

In addition to these different ways of presenting content, the KiWi core system also provides a generic search functionality accessible from within all KiWi applications. When a user switches to search and selects a content item, he is redirected back to the previous application afterwards. KiWi currently allows

a combination of full-text search, metadata search (tags, types, persons), and database search (date, title). A more sophisticated search language is currently under development [7].

The KiWi search interface implements a so-called "facetted search" (see Figure 4): the user starts with a keyword search, resulting in a list of content items ordered by relevance or time. In case the user is not satisfied with the results, he then has the option to refine his search using one or more of the facets offered in the right column of the search result box. Currently, the KiWi system offers the facets "tags", "types", and "persons", which we have identified as the core facets needed in any system. For each facet, only the criteria occurring in the currently displayed search results are listed, together with a count of the content items matching the criterion. Selecting one of the criteria narrows down the search.

All search facets are included in the full-text search box; this decision has been made to provide all search functionality in one place without confusing the user and to allow advanced users to directly search using the text field. Also, it makes it much simpler to bookmark searches or include them in a user's personal stream of activities on the Dashboard.

In later stages of the project, it is planned to make the set of widgets customisable to adapt it to different application domains. For example, in a biology scenario, an interesting facet could be different protein structures, in a music scenario it could be different instruments, and in a history scenario it could be different countries.

3.5 KiWi Inspector

The KiWi Inspector is an application developed for advanced users and developers. It provides a more technical insight into the current context. It currently provides the following functionalities:

- *Content Item Inspector*: displays the RDF data associated with the current content item as RDF/XML; the shown RDF data is the same as would be displayed to a Linked Open Data client when accessing the KiWi system
- *Tag Inspector*: displays a list of all tagging actions of the current content item and the RDF data generated by them as RDF/XML
- *User Inspector*: displays the RDF data associated with the currently logged in user as RDF/XML

In addition, the KiWi Inspector will later also provide more details about the revisions (particularly metadata revisions) of the current content item and additional debugging information as needed.

4 The KiWi Platform

The applications described in the previous section are built on top of the KiWi Platform, which provides all the core functionalities needed by most Semantic Social Software applications. In the following, we briefly describe architecture, core data model, and core services offered by the KiWi platform.

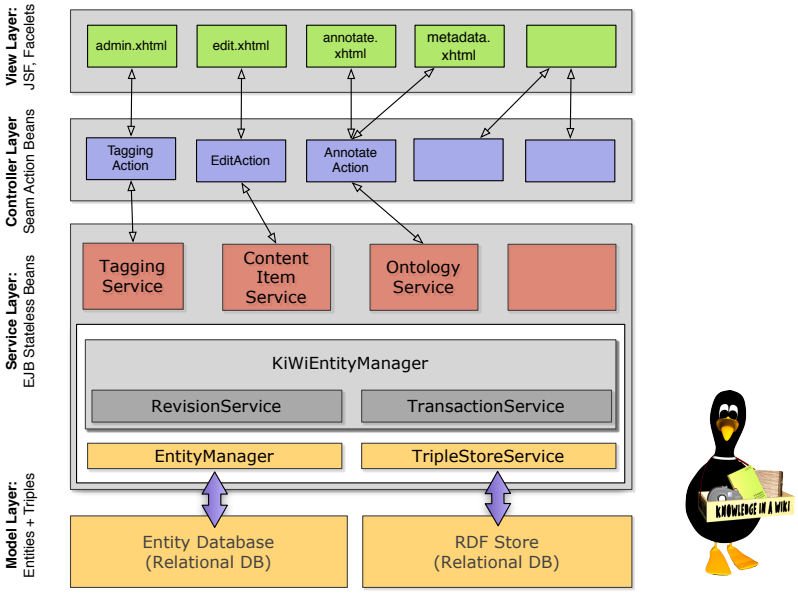

Fig. 5. KiWi Service-Oriented Architecture: Model Layer, Service Layer, Controller Layer, View Layer

4.1 Architecture: Service-Oriented and Component-Based

The KiWi system is implemented on top of JBoss Seam[6] and Java Enterprise Edition (Java EE 5), and thus follows a component- and service-oriented architecture. Figure 5 depicts the overall structure of the KiWi system:

The *model layer* comprises the KiWi data model (see below) and is represented in a relational database, a triple store, and a full-text index. Entities are persisted using the Hibernate framework[7], which maps Java objects to relational tables. The KiWi triple store is a custom implementation also based on the relational database, because existing triple store implementations provide insufficient support for features like versioning and additional metadata about triples that are needed by KiWi. The full-text index is implemented using Hibernate Search and currently allows to search over title, textual content, tags, authors, and RDF literals.

The *service layer* provides services to other components in the KiWi system. Of central importance is the KiWi Entity Manager, which provides unified access to content items and RDF metadata (see below). Further core services are the revision service – taking care of versioning, and the transaction service,

[6] http://www.seamframework.org
[7] http://www.hibernate.org

allowing to manage all updates to KiWi content in reliable transactions. Both services are heavily used internally by the KiWi Entity Manager and usually not used by further components. Besides these core services, the service layer may contain additional services that offer certain common functionalities. For example, the KiWi system currently offers an "ontology service" that provides convenient access to the triple store using higher-level concepts like "classes" and "properties", and a "content item service" that allows to easily access all functionalities associated with content items (creating, loading, updating).

The *controller layer* consists of action components that implement a specific functionality in the KiWi system. For example, the Semantic Wiki application contains a "view action", a "edit action", a "annotation action", and a "history action", and the TagIT application contains a "explorer action" and a "tagging action". Action components are usually very close to some functionality offered in the user interface, and they make use of service components to access the content in the KiWi system.

The *view layer* is implemented using Java Server Faces (JSF), which are used to generate the HTML presentation of the KiWi user interface and the user interaction with the system. JSF pages are linked with action components in the controller layer. Also part of the view layer are web services offered by KiWi. Currently, there are web services for accessing the triple store and SKOS thesauruses, and there is a "linked open data" service offering the content of the KiWi system to linked open data clients.

4.2 Data Model: ContentItems, Tags, and Triples

KiWi's core data model makes use of few concepts, namely Content Items, Tags, and Triples. Additional functionality is added by "KiWi Façades" (Section 4.3).

Content Items. The *content item* is the core concept of the KiWi system. It represents a "unit of information" in KiWi, e.g. a page about a certain topic, a user profile, etc. When a user accesses the KiWi system, he is always interacting with exactly one (primary) content item, the context content item. The context content item can be viewed, modified, and annotated by the user. Though changes might also affect other content items, the context content item is always the primary content item.

Each content item has both, a machine readable symbolic representation and a human readable textual or multimedia representation.

- content items are all different kinds of content and data items that are stored in the KiWi system, i.e. (wiki) pages, multimedia, users, roles, rule definitions, layout definitions, widgets, and possibly more. The KiWi system is not restricted a priori to specific content formats.
- URIs or blank nodes serve as machine-readable symbolic representations of resources, to be used in extended triples in the triple store. The URI is used to embed a resource in its context and provide machine-readable meaning, e.g. by annotation with formal annotations, reasoning, etc.

– the textual / media content related to a resource is meant for human consumption. The content resembles a wiki page, is easy to edit, supports linking, and is versioned. In the current design each wiki article in a specific language is represented as an own content item describing a single concept. The assumption for this design is that the content about a topic and its authors may differ from language to language. The definition of connections between content items with equivalent content but different languages can be accomplished with metadata relations.

These assumptions distinguish the KiWi data model from other wikis and content management systems. Most important, it treats all kinds of resources equally, leading to a very clean and simple model where every resource has both, a machine-readable and a human-readable description. A consequence of the direct relationship between content items and RDF resources is that every RDF resource, even those representing widgets, layouts, users, or even rules, can also be described in human readable form.

Textual content is represented internally as (structured) XML documents that can be queried and transformed to other representations like HTML or XSL-FO (for PDF and other print formats) using standard XML query languages (XQuery, XSLT) or using the rule-based reasoning language developed in KiWi. The XML format used for page representations resembles a subset of HTML, taking into account only core structuring elements.

Tags. Tagging is one of two ways of annotating content items in the KiWi system. In KiWi, tags serve many different purposes, for example associating content items with certain topics or grouping content items in "knowledge spaces". There are two kinds of tags: explicit tags are explicitly added to a content item by a user; implicit tags are created by the system, e.g. based on automatic mechanisms like information extraction from text or reasoning on existing tags.

Conceptually, an explicit tag is a 3-ary relation between two content items (the tagged content item and the tagging content item) and a user (the tagging user or tagger). An implicit tag is a binary relation between two content items, a tagged and a tagging one. Tagging content items are identified using one or more labels that are available for annotating content items. In case of ambiguous tag labels (i.e. the same tag label for different content items), the KiWi system asks the user to choose the appropriate content item. If the user enters a new label that is not yet used elsewhere, it is displayed like a wiki-link to a non-existing page; when the user clicks on it, he is given the choice to either associate the label with an existing content item or to create a new content item explaining this tag label. Internally, a tag is furthermore given maintenance information like creation time and date and a URI for uniquely identifying a tag.

For example, the content item that describes "Mickey Mouse" could be tagged with the label "Mouse", thereby associating it with the content item describing "Mouse" (the animal). The tagged content item would be "Mickey Mouse", the tagging content item would be "Mouse", and the tag label used for tagging would be "Mouse", which is a tag label of the content item "Mouse".

Inside the system, a tag is mapped to an RDF structure that can be used for deriving additional RDF metadata by means of reasoning. Also, tags can be "lifted" to taxonomy or ontology concepts by advanced users, e.g. by using the "meaning of a tag" (MOAT)[8] or "social semantic cloud of tags" (SCOT)[9] ontologies. In this case, more information about the meaning and context of a tag becomes available, e.g. for reasoning or querying.

Tags can be used by the KiWi system for many different purposes. For example, tags can help with searching by offering a facetted search interface or by offering tag clouds. Furthermore, it is possible to derive user preferences from the tags she has used or to identify users with similar interests via clustering. Similarly, tags can also be used for grouping related content items, e.g. for defining group work spaces or for clustering thematically related items. Beyond that, the way how tags are used is left to the application developers and users that implement a specific instance of the KiWi system.

Extended Triples. Machine-readable metadata is represented using what we call extended triples. Extended triples contain additional maintenance information that is used internally by the KiWi system for various tasks like versioning, transactions, associating a triple with a certain workspace, user, or group, or for reason maintenance (i.e. storing why a certain triple has been asserted). In principle, an extended triple can thus be seen as a "triple with attributes".

Note that these attributes could also be represented as a RDF subgraph using triples and reification. However, such a representation has several disadvantages compared to the extended triples proposed for KiWi:

- it requires reification, meaning that the original triple, which assumingly provides the most interesting information, is broken up into parts that have to be reassembled, and
- it mixes up several levels of abstraction, which is inconvenient not only for machines and reasoning, but also for the user
- it makes it difficult to filter out the information that is used for internal purposes and this not supposed to be exchanged with external systems

The implementation of extended triples is straightforward and fits easily with already existing tools and standards without disguising the original meaning. Triple attributes containing maintenance information are only represented programmatically inside the system, avoiding problematic situations. To the outside world, extended triples look like ordinary triples and can be exported into the usual Semantic Web formats like RDF.

In the current implementation, extended triples are represented in special tables in the relational database, and queries to the triple store are executed in Hibernate's object oriented query language HQL. We chose not to use one of the existing triple store implementations because they are restricted to simple triples

[8] http://moat-project.org/
[9] http://scot-project.org/scot/

without the possibility to add additional metadata, they have only basic transaction support [8], and they offer poor scalability if one wants to use reasoning. Building KiWi on top of these systems has proven to be extremely difficult and has been abandoned in favour of the more flexible database solution. Mapping SPARQL queries to Hibernate is currently under development.

4.3 KiWi Entity Manager and KiWi Façades: Content Versatility in Java EE

The KiWi system offers a number of core services that are needed in many situations. Of particular importance is the KiWi entity manager (named in analogy with the Java EE entity manager, as it provides the technological background for realising content versatility. In the following, we briefly introduce its functionality and then discuss a technique we call KiWi Façades.

KiWi Entity Manager. The KiWi entity manager is a central service providing unified access to data stored in the relational database, in the triple store, and in the fulltext index. It provides functionalities for storing, querying and searching entities (content items, triples, tags) in different languages:

- *Storing* of entities is handled through the methods `persist()` (for new entities) and `merge()` (for updated entities). Both methods take care of forking the data associated with a Java entity appropriately into relational database, triple store, and fulltext index.
- *Querying* of entities is handled by the method `createQuery()`, which takes as argument a query string in either HQL (Hibernate's object oriented query language) or SPARQL and returns a Query object that can be used in the same way as the ordinary Java EE Query object for retrieving results.
- *Searching* of entities is handled by the method `search()`, taking as argument a label-keyword search string and then delegates – depending on the label – to either the fulltext index, the relational database, or the triple store.

All KiWi Entity Manager methods support façading, described below. Additionally, all updates performed through the KiWi entity manager are automatically wrapped in appropriate transactions that support transaction isolation and commit/rollback functionality. Also, KiWi entity manager transactions are automatically versioned using KiWi's revision service and can be reverted individually. The KiWi transaction system is discussed in [8].

KiWi Façades. A particularly salient aspect of the KiWi system is its façading mechanism. Façading can be seen as a way of providing different application-dependent Java views on content items. They are thus a way of realising content versatility in Java in a way natural to developers. A KiWi Façade is specified as a Java interface, annotated with Java 5 annotations that map Java methods to RDF properties (see Figure 6). When calling an annotated method, an appropriate query to the triple store is issued instead of accessing the Java field.

```
@KiWiFacade
@RDFType( Constants.NS_GEO+"Point" )
public interface PointOfInterestFacade extends ContentItemI {

    @RDF(Constants.NS_GEO+"long")
    public double getLongitude();

    public void setLongitude(double longitude);

    @RDF(Constants.NS_GEO+"lat")
    public double getLatitude();

    public void setLatitude(double latitude);
}
```

Fig. 6. A Java interface annotated as KiWi Façade; the methods get/setLongitude() map to the RDF property `geo:long`, whereas get/setLatitude() map to `geo:lat`. A content item façaded with this interface is automatically assigned the `geo:Point` type.

For a Java developer working with the system, a façaded content item behaves exactly like an ordinary content item with the additional methods specified in the façade interface, and can be used in any context a content item can be used, e.g. as backing bean for user interface components. This functionality is realised using Java's dynamic proxy mechanism (implemented in the KiWi invocation handler). All methods in the KiWi entity manager may optionally take or return a KiWi Façade instead of a content item if they are passed a façade interface as additional argument.

5 Outlook and Conclusion

KiWi is an ongoing research project of which we have only demonstrated the first results. Many more salient aspects are still to come. Particularly, KiWi will be extended with "enabling technologies" in the following areas:

- **Reasoning and Reason Maintenance:** in this area, the KiWi system will be extended with a custom, rule-based querying and reasoning component where advanced users will be able to add custom inference rules to the knowledge base; reasoning will be based on content as well as metadata [9]. In addition, there will be a reason maintenance component that allows users to get explanations why a certain information has been inferred; reason maintenance is also beneficial for update performance and might even allow for different users having different rule sets.
- **Information Extraction:** the KiWi system will furthermore provide a component that semi-automatically extracts metadata from the content that is

stored in the knowledge base. Information Extraction will be performed in interaction with the user to minimise the number of errors.
- **Personalisation:** based on the metadata for a content item, a user model, and the rule-based reasoning, KiWi will also offer a personalisation component that allows to further customise the presentation of a content item. The personalisation component will further demonstrate the flexibility of the Content Versatility approach taken by KiWi.

TagIT is now further developed as part of the Salzburg NewMediaLab project "Future Content Platforms" together with our partner *Salzburger Nachrichten*. The goal is to integrate in the same interface not only wiki content and TagIT locations but also news articles, blog posts, and small advertisements from our partner. For the purpose of the demonstration, we have already imported 20.000 online news articles, but the system is designed to scale to hundreds of thousands.

Acknowledgement. This research has been partly funded by Salzburg New Media Lab and by the the European Commission within the 7th Framework Programme project KiWi - Knowledge in a Wiki (No. 211932). The latest KiWi source code is available as Open Source at the KiWi website http://www.kiwi-project.eu. The demonstration system is published from time to time at http://showcase.kiwi-project.eu.

References

1. Völkel, M., Schaffert, S., eds.: 1st Workshop "From Wiki to Semantics" (SemWiki'06) – colocated with ESWC'06, Budva, Montenegro (2006)
2. Lange, C., Schaffert, S., Skaf-Molli, H., Völkel, M., eds.: 3rd Workshop "From Wiki to Semantics" (SemWiki'08) – colocated with ESWC'08, Tenerife, Spain (2008)
3. Schaffert, S.: Semantic Social Software: Semantically Enabled Social Software or Socially Enabled Semantic Web? In: Semantics 2006, Vienna, Austria (2006)
4. Bizer, C., Heath, T., Ayers, D., Raimond, Y.: Interlinking Open Data on the Web. In: 4th European Semantic Web Conference (ESWC2007) – Posters Track, Innsbruck, Austria (2007)
5. Schaffert, S., Westenthaler, R., Gruber, A.: IkeWiki: A User-Friendly Semantic Wiki. In: Proceedings of the 3rd European Semantic Web Conference (ESWC06) – Demonstrations Track, Budva, Montenegro (2006)
6. Durao, F., Dolog, P.: Analysis of tag-based recommendation performance for a semantic wiki. In: Fourth Workshop on Semantic Wikis (SemWiki2009) in conjunction with the 6th Annual European Semantic Web Conference (ESWC2009), Heraklion, Greece, CEUR.org (2009)
7. Weiand, K., Furche, T., Bry, F.: Quo vadis, web queries. In: Proceedings of International Workshop on Semantic Web Technologies, Belgrade, Serbia (29th–30th September 2008). (2008)
8. Stroka, S.: Transaction management in federated, heterogeneous database systems for semantic social software applications. In: submitted to 20th Int. Conference on Database and Expert Systems Applications (DEXA'09), Linz, Austria (2009)
9. Bry, F., Kotowski, J.: Towards reasoning and explanations for social tagging. In: Proceedings of 3rd International Workshop on Explanation-aware Computing, Patras, Greece (21st–22nd July 2008). (2008)