Towards End-User Programming with Wikis

Craig Anslow and Dirk Riehle
SAP Research, SAP Labs LLC
3475 Deer Creek Road
Palo Alto, CA, 94304, USA

craig@mcs.vuw.ac.nz, dirk@riehle.org

ABSTRACT

When business software fails to provide the desired functionality, users typically turn to spreadsheets to perform simple but general computational tasks. However, spreadsheets enforce a view of the world that consists mostly of tables and numbers rather than the domain concepts users have in mind. We are using wikis as a platform for empowering end-users to perform computational tasks of their choice. This paper discusses how core properties of wikis can support end-user programming. We illustrate our approach using wiki prototype software for working with business objects as made available by SAP's business application suite.

Categories and Subject Descriptors

D.2.2 Design Tools and Techniques, D.2.6 Programming Environments, D.3 Programming Languages, H.5.2 User Interfaces, H.5.4 Hypertext/Hypermedia, I.7 Document and Text Processing.

General Terms

Design, Human Factors.

Keywords

Wikis, Application Wikis, End-User Programming, User Innovation.

1. INTRODUCTION

Successful business software is typically flexible and powerful. However, it can never fully anticipate the needs of all possible users. End-user programming empowers business software users to perform computational tasks that are not supported by the business applications at hand. There is a need to explore how end-user programming can help business software.

The SAP business application platform provides a comprehensive backbone on which to build software for one-off computational tasks. To enable end-user programming, end-users need tools and documentation to make it easier to understand the application platform, including its core data structures, the business objects. End-user skills vary; thus they require tools that aid understanding the platform using different approaches. It should be easy to explore business objects and their functionality.

Prior work on empowering end-users using wikis suggests that a long tail for yet-to-be-written end-user applications exists [3]. Traditional applications are not tailored to the needs of individual end-users. The long tail consists of one-off applications fulfilling the need of only a small number of users. Using traditional means, this long tail of software is hard to write, brittle once deployed, and expensive to distribute.

Wikis are a collection of web pages that can be edited by anyone, at any time, from anywhere [4]. Until recently, wikis contained mostly prose. We are investigating wiki engines as an application platform for end-user programming. Wikis empower end-users by allowing them to share information, collaborate, and publish information more easily than other online collaboration software. To this we are adding functionality for working with structured data, that is, SAP's business objects.

Our prototype helps end-user programmers explore the business objects of the SAP application platform and perform one-off computational tasks. The prototype documents the business objects in wiki pages instead of a traditional text document. It also allows a user to write a business query in a wiki page; when the page is rendered the query is executed and instance data is displayed. Finally, it allows a user to build a query by exploring the model of the business objects, selecting the attributes they want in the query, and then executing the query to return the objects. This paper discusses our assumptions about end-user programming and describes our prototype.

2. END-USER PROGRAMMING

Our approach and the motivation to use wikis as an end-user programming platform are based on the following assumptions about end-user programming:

1. *Incremental and incomplete programming.* End-users don't "think through" their programs. They put down parts of the computational tasks to be performed and expect that these tasks can be performed, even if the program is not complete by traditional means (nor will it ever be). We view end-users as incremental tinkerers.
2. *No clear distinction between instance and model.* End-users don't or can't program in a traditional sense. They describe what they want the software to do by performing the computations themselves and by laying out how the results are supposed to look like. End-users can't distinguish (well) between a model and its instances [5].
3. *Execution through failures.* End-users expect their incomplete and incremental programs to provide useful results, even if these programs are full of potential and real failures during execution. Thus, the underlying platform must not

only be robust to such errors but recover from failure and keep performing the end-user tasks [7].

Spreadsheet programs like Microsoft's Excel have properties that fulfill these properties:

1. End-users can program in Excel and expect the spreadsheet to perform the computations, independently of whether the spreadsheet is finished or not.
2. Excel does not make clear a distinction between model of computation and instance (execution) of computation. Both are available in one connected view.
3. Excel does not crash because computations cannot fail, instead cell values keep being computed. Some cells may provide nonsensical values but it is left to the end-user to judge the output of the cell value.

Spreadsheets are based on a tables and numbers paradigm. Hence, to empower end-users to work with domain concepts and have representations different from those found in spreadsheets, we decided to use wikis as a platform for end-user programming.

Wiki engines fulfill the end-user programming assumptions:

1. A wiki page is always in a consistent state, i.e. "the markup always parses," even if the results of a page may not make much sense. There is no intermediate or final stage of a wiki page, it is always in development.
2. The wiki page is the computation, and the current distinction between edit and view mode, or "design-time" and "run-time" is rather accidental. In fact, the latest breed of wiki WYSIWIG editors attempts to integrate these two views.
3. Wiki pages don't crash. Rather the wiki engine parses what it can interpret and if it doesn't make sense, the computation falls back to text display.

In contrast to spreadsheets, wikis are not constrained to a specific computational paradigm or a specific visual display. Wikis are not by nature domain-specific. Hence, we believe wikis are a platform better suited to end-user programming than spreadsheets.

Many wiki engines today support simple computational tasks in an ad-hoc fashion. MediaWiki offers parameterized templates [6] and TWiki offers macros [8]. Some wiki engines have been specifically set up to support general computational tasks, for example JotSpot [2] and XWiki [9].

In general, to evolve current wiki engines into general platforms for end-user programming, wiki markup needs to be extended to allow for the expression of programs. The next section describes our first steps towards such a platform.

3. BUSINESS OBJECTS IN WIKIS

We are working on the integration of business objects into wikis. A business object is a data structure that represents an instance of a business domain concept. In object-oriented terms, a business object is an object, and the type of a business object is its class. The business object model includes all of the types of the business objects including their relationships.

This integration of business objects into wikis has three parts: the description of the business objects, the definition of queries that read and write business objects, and the execution and presentation of queries. These components are only a first step on a way to a more comprehensive integration of SAP's business application platform with wikis.

3.1 WIKI DOCUMENTATION

The majority of the documentation of the SAP business object model is available as word processor documents. Using a word processor to document the business objects makes it hard for the end-user programmer to find information about business objects. In contrast, wikis make it easier to browse the object model to find the desired information. In addition, wikis provide useful features for collaborating, source control, and searching.

Our Wiki Docs prototype uses the MediaWiki wiki engine [6] to represent a business object type from the object model as one wiki page. Using a wiki to document the business object model instead of word processor documents makes it easier for end-users to collaborate and understand the model. End-user programmers can easily browse the business object wiki pages and don't need to download each separate business object document. The search feature of wikis allows end-users to search for a business object they are interested in. Maintainers of the business object model description can collaborate easier since wikis can be concurrently edited online and provide the revision history of a wiki page rather than having to use another piece of software for source control. The discussion page allows maintainers to discuss the contents of a page that is associated with a business object. Finally, it is possible to control who can edit a wiki page which can be useful to distinguish between end-users and maintainers.

We have documented the business objects in a wiki which helps show the hierarchal structure of the object model better than a set of word processor documents. We map the namespace structure onto the wiki. We provide three pages for an end-user to browse the navigation structure of the business object model. The first page is a list of all business objects generated automatically from the backend SAP business application. The second page alphabetically groups each business object by the different namespaces. The third page annotates each business object by the business object type category and namespace category.

Figure 2 shows an example business object type in a wiki page. Each business object wiki page provides information on how to use the business object, the structure of the business object, what namespace the business object belongs to, associations of the business object with other business objects, and technical people to contact for further information.

We have another project which uses a recommendation engine to provide recommendations of wiki pages to end-users. Using a recommendation engine in our Wiki Docs prototype could be useful to end-users as the engine will be able to inform them of other business objects end-users have looked at or related business objects while browsing an existing business object wiki page.

3.2 WIKI BUSINESS QUERY

The Wiki Business Query prototype allows users to write business queries from within a wiki. The result of a query is structured data which is then displayed inside a wiki page as one or more business objects.

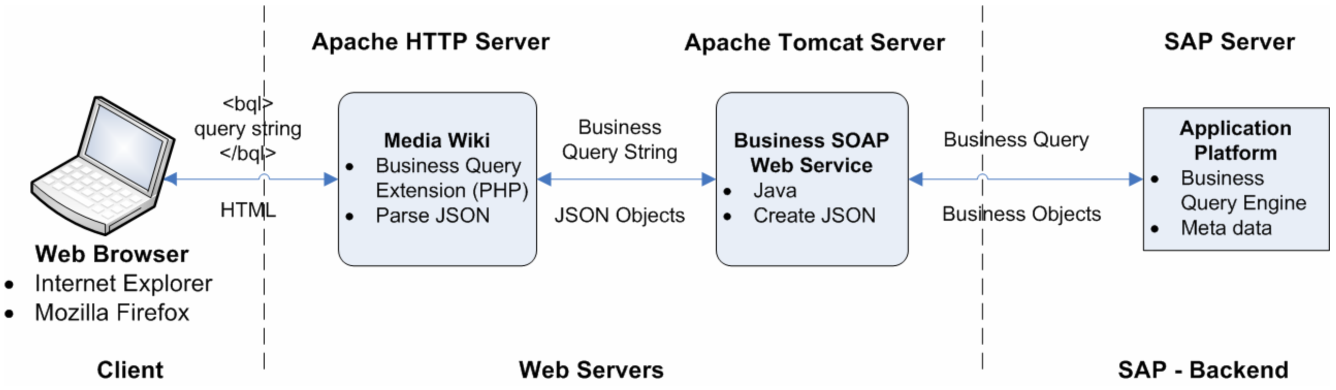
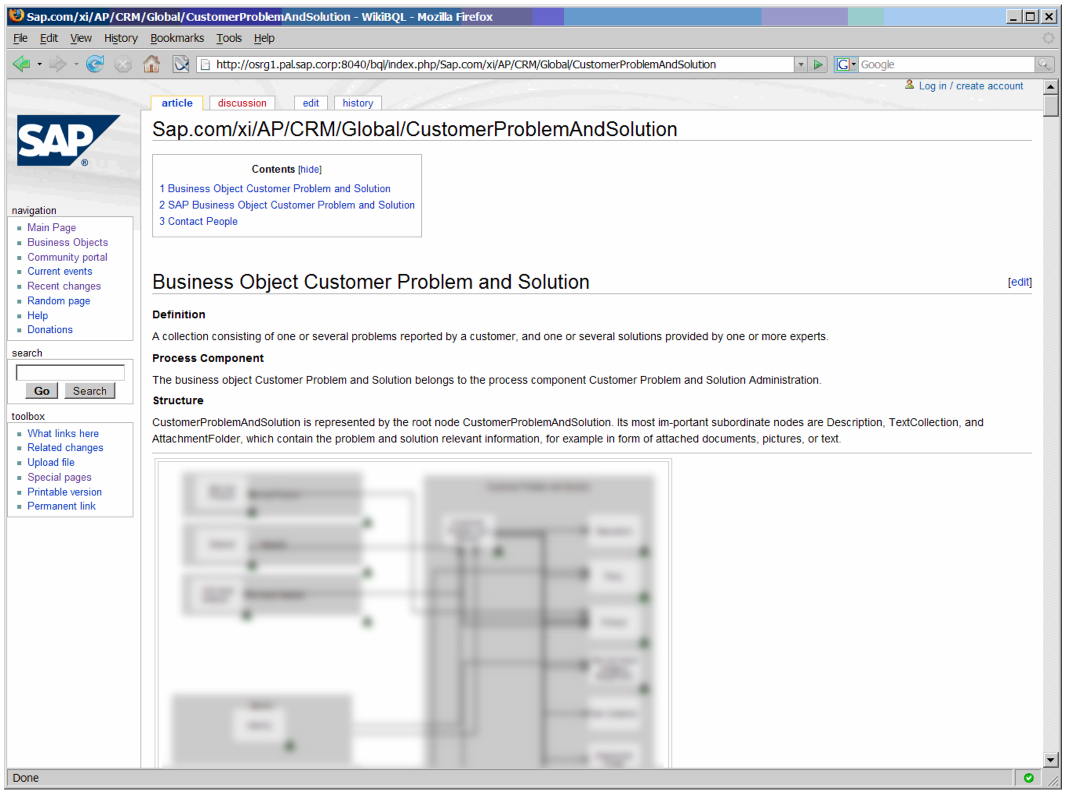


Figure 1. SAP Wiki Business Query Architecture.



Sap.com/xi/AP/CRM/Global/CustomerProblemAndSolution

Contents [hide]

1 Business Object Customer Problem and Solution
2 SAP Business Object Customer Problem and Solution
3 Contact People

Business Object Customer Problem and Solution [edit]

Definition

A collection consisting of one or several problems reported by a customer, and one or several solutions provided by one or more experts.

Process Component

The business object Customer Problem and Solution belongs to the process component Customer Problem and Solution Administration.

Structure

CustomerProblemAndSolution is represented by the root node CustomerProblemAndSolution. Its most im-portant subordinate nodes are Description, TextCollection, and AttachmentFolder, which contain the problem and solution relevant information, for example in form of attached documents, pictures, or text.

Figure 2. Documenting a business object from the SAP business object model in a wiki.

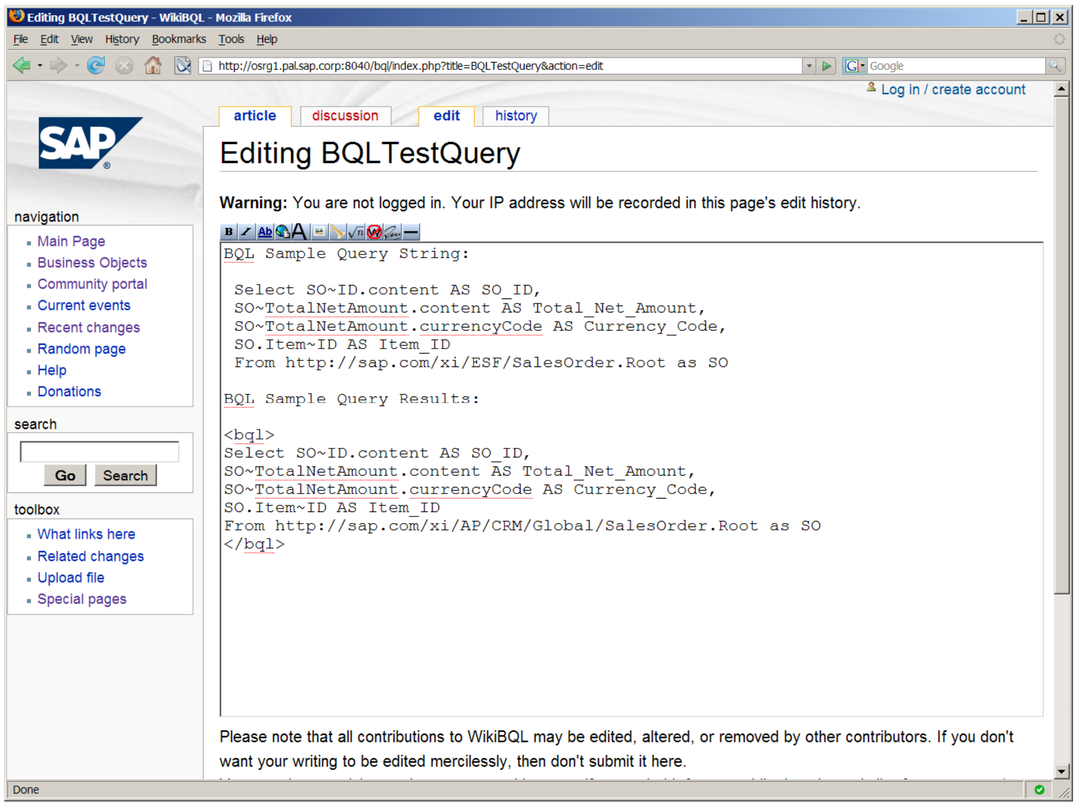
The business queries operate on business objects. A sales order is an example of a business object. A sales order is an order received by a business from a customer and contains a buyer party, seller party, the items for sale, etc.

Figure 1 shows the basic architecture of the Wiki Business Query prototype. Users view the MediaWiki [6] instance in their web browser. Users edit a wiki page through the edit mode and can then write business queries which are encapsulated inside XML-like tags, e.g. <bql> query string </bql>. Metadata queries use <bqlm> tags instead.

When the MediaWiki engine parses the tags the PHP extension is executed. The extension takes the business query string as an argument and calls a Java SOAP web service operating on an Apache Tomcat instance. The web service creates a Java business object query and then calls the execute method on the business object query engine inside of SAP's application platform.

If the business query is valid, a list of Java data objects is returned. If the query fails, an error message is returned. The web service then parses the data objects and creates structured JavaScript Object Notation (JSON) objects. The JSON objects are then returned to the MediaWiki instance which is responsible for parsing and rendering the JSON objects into HTML.

Figure 3 shows a test query which gets all the sales orders and Figure 4 shows the results of the query displayed as a table. We are currently exploring using JavaScript to provide features for sorting and filtering the query results displayed in the table.



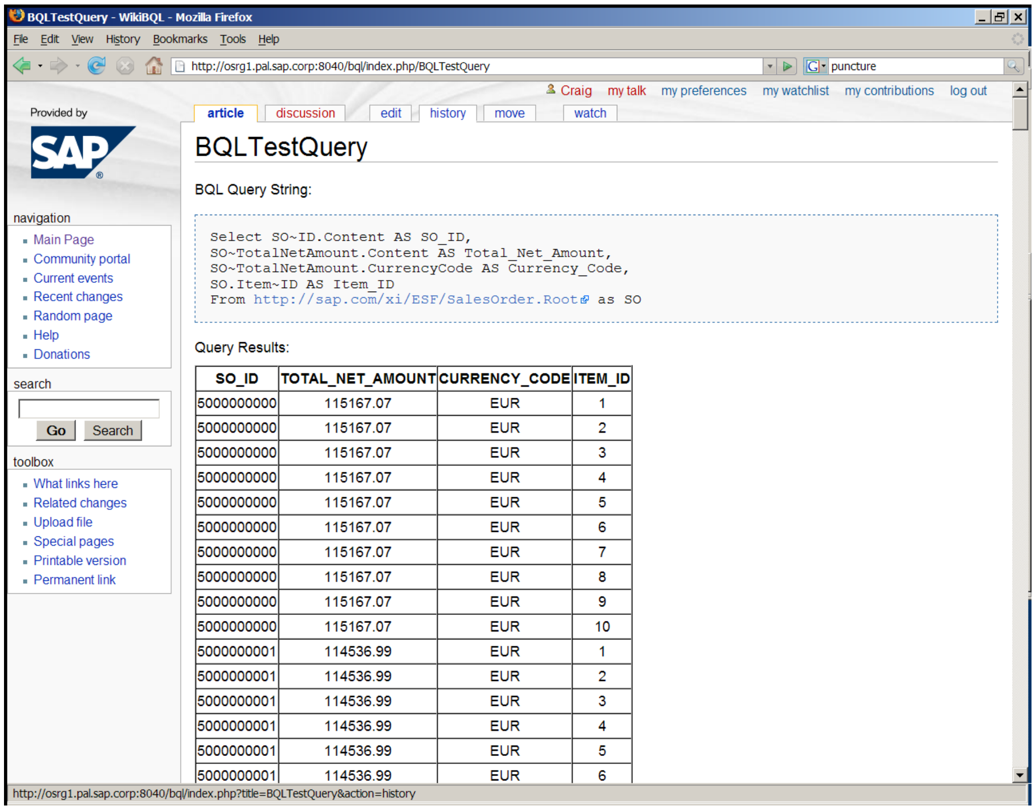
Editing BQLTestQuery

Warning: You are not logged in. Your IP address will be recorded in this page's edit history.

```
BQL Sample Query String:

 Select SO~ID.content AS SO_ID,
 SO~TotalNetAmount.content AS Total_Net_Amount,
 SO~TotalNetAmount.currencyCode AS Currency_Code,
 SO.Item~ID AS Item_ID
 From http://sap.com/xi/ESF/SalesOrder.Root as SO

BQL Sample Query Results:

<bql>
Select SO~ID.content AS SO_ID,
SO~TotalNetAmount.content AS Total_Net_Amount,
SO~TotalNetAmount.currencyCode AS Currency_Code,
SO.Item~ID AS Item_ID
From http://sap.com/xi/AP/CRM/Global/SalesOrder.Root as SO
</bql>
```

Please note that all contributions to WikiBQL may be edited, altered, or removed by other contributors. If you don't want your writing to be edited mercilessly, then don't submit it here.

Figure 3. Creating a business object query inside the wiki edit mode.



BQLTestQuery

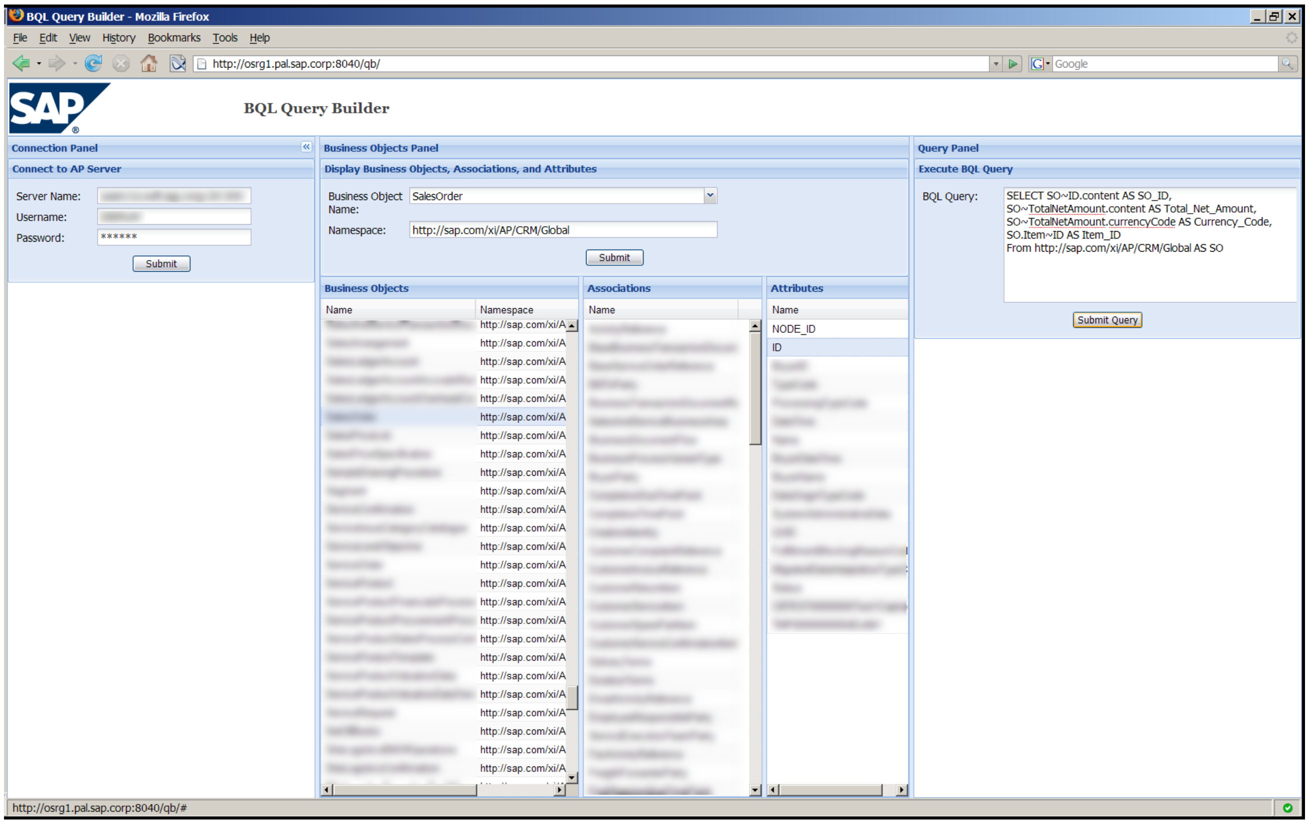
BQL Query String:

```
Select SO~ID.Content AS SO_ID,
SO~TotalNetAmount.Content AS Total_Net_Amount,
SO~TotalNetAmount.CurrencyCode AS Currency_Code,
SO.Item~ID AS Item_ID
From http://sap.com/xi/ESF/SalesOrder.Root as SO
```

Query Results:

SO_ID	TOTAL_NET_AMOUNT	CURRENCY_CODE	ITEM_ID
5000000000	115167.07	EUR	1
5000000000	115167.07	EUR	2
5000000000	115167.07	EUR	3
5000000000	115167.07	EUR	4
5000000000	115167.07	EUR	5
5000000000	115167.07	EUR	6
5000000000	115167.07	EUR	7
5000000000	115167.07	EUR	8
5000000000	115167.07	EUR	9
5000000000	115167.07	EUR	10
5000000001	114536.99	EUR	1
5000000001	114536.99	EUR	2
5000000001	114536.99	EUR	3
5000000001	114536.99	EUR	4
5000000001	114536.99	EUR	5
5000000001	114536.99	EUR	6

Figure 4. Displaying the results of a business query.


BQL Query Builder - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://osrg1.pal.sap.corp:8040/qb/
BQL Query Builder
Connection Panel
Connect to AP Server
Business Objects Panel
Display Business Objects, Associations, and Attributes
Query Panel
Execute BQL Query
Business Objects
Associations
Attributes
Submit
Submit Query
http://osrg1.pal.sap.corp:8040/qb/#

Server Name: [redacted]
Username: [redacted]
Password: ******

Business Object Name: SalesOrder
Namespace: http://sap.com/xi/AP/CRM/Global

Attributes: NODE_ID, ID

BQL Query:

```
SELECT SO~ID.content AS SO_ID,
SO~TotalNetAmount.content AS Total_Net_Amount,
SO~TotalNetAmount.currencyCode AS Currency_Code,
SO.Item~ID AS Item_ID
From http://sap.com/xi/AP/CRM/Global AS SO
```

Figure 5. Query Builder.

3.3 QUERY BUILDER

The Query Builder prototype allows a user to browse the underlying business object model in a top down approach to build a query and then execute that query. Figure 5 shows the Query Builder prototype which contains the following panel components: connection, business objects, and query.

A user first connects to SAP's application platform in the connection panel. Once connected to the platform end-users can search for a business object type and the associated namespace. When the form is submitted the business object is highlighted along with the associations the business object has amongst other business objects and the attributes for the highlighted business object. A user can also browse deeper into the business object hierarchy by selecting an association which will then show the business objects that can be traversed along this path and the new attributes for the selected business object association. When a user selects any of the attributes from a business object this will build up a query in the far right query panel. Once a user has built a query they can submit the query form and get instance data returned which is displayed below the query form.

The Query Builder prototype uses the same web service as that of the Wiki Business Query prototype but the client interface is implemented using the Ext JavaScript 2.0 library [1]. We intend to make a smaller widget version of the Query Builder prototype and integrate it with the edit mode of MediaWiki.

4. CONCLUSION

We have described a prototype based on MediaWiki that allows end-users to work with SAP business objects. The prototype is based on our vision of wikis as a general computational platform for end-user programming. The design of our extensions is based on the three fundamental assumptions that end-user programs are never finished but rather remain incomplete, that end-user programs are a direct reflection of their execution, and that end-user programs should be robust towards computational errors.

5. REFERENCES

[1] Ext JS 2.0 – JavaScript Library. http://extjs.com

[2] JotSpot. http://www.jot.com

[3] Joe Kraus. The long tail of software. Millions of Markets of Dozens. March, 2005. http://bnoopy.typepad.com/bnoopy/2005/03/the_long_tail_o.html

[4] Bo Leuf and Ward Cunningham. The Wiki Way: Quick Collaboration on the Web. Addison Wesley, 2001.

[5] Henry Lieberman. Your Wish Is My Command: Programming by Example. Morgan Kaufmann, 2001.

[6] MediaWiki. http://www.mediawiki.org

[7] Martin Rinard. Acceptability-oriented computing. In the Companion of the ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications (OOPSLA), 2003.

[8] Twiki. http://twiki.org

[9] Xwiki. http://www.xwiki.org