Smarty-objects for heterogeneous data sources in situation-oriented databases

Gusarenko A. S.
postgraduate, junior researcher.
FSEE HPE Ufa State Aviation Technical University
450000, Ufa, st. Karl Marks, h. 12
phone +79174290226,
e-mail: artymugusarenko@gmail.com

Mironov V. V.
Doctor of Technical Sciences, prof.
dep. ACS, FSEE HPE Ufa State Aviation Technical University
450000, Ufa, st. Karl Marks, h. 12
phone +79174290226,
e-mail: diplomastudents@mail.ru

Submitted: August 28, 2013; Accepted: September 28, 2013;
Published: December 9, 2013. UDC 004.65
Subject MSC mathematical classification: 90C35, 90C39
Subject ACM computer classification: I.6.5

Abstract

There are currently actively developing NoSQL type of databases and as a part of it document-oriented databases. XML-database solves specific problems in the processing and storage of data in XML and JSON as documents. This article focuses on NoSQL databases, and specifically SODB (Situationally-oriented databases), and development tools for its processing data using Smarty-objects. The article proposes a model for automatic generation content from the JSON-sources and automatically templates data using Smarty template engine. SODB has no special tools for processing JSON-documents, it is necessary to develop tools for processing such data sources. This article operates gained popularity toolkits server templating Smarty, and JSON-sources. At the core of the database is a dynamic model, which is processed by the interpreter of dynamic models. We consider two options for handling JSON-data sources first of these is the Smarty-objects, the second Smarty-arrays. As a result, at the conceptual layer are encouraged to develop tools to automatically create objects and arrays Smarty for loading and processing of its data. For SODB formed concept of Smarty-objects, which proposes a dynamic model to equip a special model of Smarty-objects during its interpretation tied to states of the model objects are automatically created, supplemented the data is processed and the resulting generated content. Discussed filling and filtering data in Smarty-objects and its automatic creation. An example of using objects Smarty that is considered as an example of the source data and the user interface, as well as the dynamic model of the underlying example of a Web application. At the final stage of the work shows a diagram for the templating of the finished data from Smarty-object. Finally, in article discuss the contribution of the work in the direction of developing SODB. For this article introduces new Smarty-elements for dynamic model. These elements are processed by the interpreter automatically, without manual programming of routine operations.

Keywords: NoSQL, interpretation, XML databases, JSON, dynamic model, Smarty.
1 Introduction

By developing the concept of situationally-oriented databases (SODB) proposed in (Mironov & Gusarenko, 2012a; Mironov & Gusarenko, 2012b), it should be noted that the basis of NoSQL databases (Benzaken et al., 2013; Takahashi et al., 2013; Kobayashi et al., 2011; O'Connor, Merriman & Nelson, 2010) are different types of sources (Strauch & Kriha, n.d.; Mironov & Gusarenko, 2012b; Mironov & Gusarenko, 2012a; Gusarenko, 2013; Mironov & Gusarenko, 2013a; Mironov & Gusarenko, 2013b). The current situation is such that the databases have in their structure JSON-type sources, it is an array or data object is not only used for data transferring between server and client web applications, and serves as the basis of databases. This format is an optional method of storing data on the server, it can be used to describe objects and complex hierarchical data structures, but SODB in your toolbox, do not have a tangible approach to connecting and processing sources of this type, as well as algorithms, the specific language and the built-in display methods of content to a user.

If the XML database to display content used XSLT-templating technology, the question of sources for JSON representation of the content will remain open through Smarty template engine gained popularity. Compiling PHP Smarty templates has basic and advanced features for displaying data structures such as arrays and objects, creation of templates has a structure similar to the PHP-program, as well as in the JSON-data description similar to the description on the JavaScript-program.

In (Mironov & Gusarenko, 2012; Mironov & Gusarenko, 2013a) considered various aspects of the use of XML source and display dynamic content using XSLT, but not considered issues related to JSON and Smarty as a platform SODB. This led to the fact that now there are no special types of support for SODB to work with this functionality (Mironov & Gusarenko, 2012). This article considers questions of automatic content generation for associated JSON-objects and JSON-arrays with the states of the dynamic model SODB and mapping specification via Smarty model Web application.

2 Smarty-objects and Smarty-arrays for handling JSON-sources

This article is the development of work on the creation of tools, language tools SODB (Mironov & Gusarenko, 2013a), which were developed by means of specs of dynamic DOM-objects. In this case, there is a problem automatically create Smarty-objects and filling them with content of JSON-sources. So as to create and handle new sources requires significant efforts, especially in an environment where JSON-source has a complex hierarchical structure, and SODB hasn’t implemented methods and approaches to their treatment. Required to include in the model special elements that specify a new type of source - JSON documents Fig. 1,2, in the process of interpretation, decoding and loading into Smarty-objects/arrays. To use the resulting object / array in the state of the model is also required to provide elements of tpl-templates. To catch the result output due to compile time are also required attributes indicating the resource type template with the path and name of the template. According to suggested approach in the model to conditions Smarty-elements are bound. Elements Smarty-objects are created when the state to which they are attached, it becomes the current one. Known ap-
approach in comparison with the proposed consists in the specification of the template and programming of functions for processing JSON-sources in the program or subprogram. Manual method doesn’t allow to use the model for improving the abstraction layer by using typical operations of compound extraction and filtering data in processing JSON-sources and outputting the result using tpl-templates. In these circumstances, it is logical to equip by these functions SODB model:

- Elements in states of the dynamic model includes child Smarty-elements, child elements, the sources of which indicate loadable JSON-data and child elements receivers – stored as a JSON-result;
- In the course of interpreting the dynamic model by means of the interpreter is an automatic generation Smarty-objects for the current states of the model and load JSON-data with the specified transformation.

**Smarty-objects.** Smarty object Fig. 3 are generated during interpreting the model and have a description of the programming language, filled with JSON-data, and then used in the tpl templates. Smarty object is the foundation for forming the resulting object based on information from the JSON-source. The resulting document Fig. 1 is

![Diagram 1](image1)

**Fig 1.** The formation of Smarty-object based on information from the JSON-data sources

![Diagram 2](image2)

**Fig 2.** The diagram dynamic model of specification JSON-documents
based on the extraction and filtering of the source document in relation to the resulting source - the source document is a child.

**Smarty-arrays.** The second feature, which has Smarty - the specification of associative arrays. That is why the resulting document represents the contents of confluence of two or more associative arrays, JSON data. In this case, defined data is extracted, filtered in according to the terms specified in the model. Using JSON-arrays and objects is provided by linguistic means (Mironov & Gusarenko, 2013a) and are justified, as there are realized by means of encoding / decoding of JSON-objects and arrays.

Model diagram in Fig. 3 provides for the use of a state with dynamic DOM-objects (He & Zhai, 2013; Han, et al., 2014; Pokorný, 2013; Kaur & Rani, 2013) and Smarty-objects. Dynamic DOM-objects provide for automatic data loading XML-origin, and facilitate the task specifications for processing such data, but don’t solve the problem of using JSON-content sources in situations of application. Therefore, a tool built into interpreter of the dynamic model (Mironov & Gusarenko, 2012b) implements the functionality of automatic loading and handling in the states of the dynamic model JSON-content along with the approach implemented in (Mironov & Gusarenko, 2013a) for processing XML-data sources.

### 3.1. Concept of Smarty-objects

**Smarty elements.** To, the interpretation automatically created Smarty-objects, which are loaded JSON-data associated with the states of the model should be provided in the model Smarty and JSON-elements. Elements JSON (Kobayashi, et al., 2011) describe the source of the data received for processing. For the treatment of conditions specified in the data provided for in the model Smarty-elements and nested data sources into them. According to these specifications will add functionality to the interpreter to create a Smarty-objects, which allows you to create these objects when the parent be-
comes the current state and remove them when the state no longer be current. This will reduce the complexity of the preparation Smarty-objects and arrays when handling JSON, as well as set the model transformation operations JSON-data for further processing to specify procedures for processing JSON. Specification example is shown in Fig. 4.

For specifications Smarty-objects need to enter a special symbol \texttt{smt}, denoting a graphical notation model description, it specifies the name of the object and the required template to display the result. In the description of the source \texttt{src}, the previously entered for the sources of dynamic DOM-objects (Mironov & Gusarenko, 2012a; Mironov & Gusarenko, 2012b) add task construction source type type = "json" and type = "array", as well as links on the path to which the interpreter retrieves required content. Further implementation of the idea to reduce the complexity of the issues requires subsequent studies to provide the data sources.

JSON-content into Smarty-objects. Connecting JSON sources along with XML content actual if we consider this problem based on the fact that in comparison with XML it has a much more compact form of writing, the exchange of the results of executing HTTP-request object using XmlHttpRequest, technology used to implement AJAX, between client and the server, or just on the server. And it can be an array or an object in memory that is loaded via object XmlHttpRequest. If we use the JSON-content from the server, requires specification of the path on which the file resides. In such conditions SODB requires tools capable to handle and display the JSON documents stored in memory and the external memory. To illustrate the proposed concept as a practical example of using Smarty-objects provides an example solving the problem of using JSON, similar to that given in (Mironov & Gusarenko, 2012b) with the solution of task displaying information about students and submission subjects from SODB by using dynamic DOM-objects (Dekeyser, Hidders & Paredaens, 2004; Herskovits & Chen, 2008; Jea, Chang & Chen, 2009; Kudrass & Conrad, 2002; Nassis, Dillon, Rajagopalapillai & Rahayu, 2006; Batory, 2006; Dejanovic, Milosavljevic, Perisic & Tumbas, 2010).
2.2. Example of using Smarty-objects

To illustrate the proposed concept as a practical example of using Smarty-objects provides an example solving the problem of using JSON, similar to that given in (Mironov & Gusarenko, 2012b) with the solution of task displaying information about students and submission subjects from SODB by using dynamic DOM-objects (Dekeyser, Hidders & Paredaens, 2004; Herskovits & Chen, 2008; Jea, Chang & Chen, 2009; Kudrass & Conrad, 2002; Nassis, Dillon, Rajagopalapillai & Rahayu, 2006; Batory, 2006; Dejanovic, Milosavljevic, Perisic & Tumbas, 2010).

**Benchmark data.** Data about students and passing subjects are stored in files JSON - stud.json, predm.json, sdacha.json, are located in the JSON server SODB. Figure 5 shows the conceptual model of these documents and test instances of their contents. Document stud.json (Fig. 5a) in the root object "Students" can contain multiple nested instances of "student" with the attributes of the "code" (code for the student) and "name" (last name, first name of the student), and the attribute "code" is an identifier. Instance of this document (see Fig. 5, a) contains information about the three students: Ivanov, Petrov and Sidorov. Similarly, the JSON-document predm.json (Fig. 5b) in the root of the object "object" can contain multiple nested instances of "object" with attributes "code" (code of the object) and "titles" (the name of the object), and the attribute "code" is an identifier. Model instance of this document (see Fig. 2.10b) contains information about three subjects: "system-theory", "mathologic" and "cryptography." Document sdacha.json (Fig. 5c) in the root object "Delivery" can contain multiple nested instances of "surrender", the relevant student sitting some particular subject, with the attributes of "codeSt" (code for the student), "codePr" (code of the object ) and "evaluation" (student's grade on the subject), and a pair of attributes "codeSt, codePr" is a composite identifier. Model instance of this document (see Fig. 5c) contains information about the five passings exam: three passings exam Ivanov and two Petrov's (Sidorov did not pass any subject).

**User interface.** The above example is the mapping of data generated by Smarty-object based on JSON-data. The initial state of the request form displays the names of the student for whom using Smarty-object data is requested from all passings the selected student. The result of the selection is displayed in the browser. Fig. 6 shows examples of the forms that are generated by the choice of the student, in this case the selected student Ivanov, who has a choice of 3 results obtained with the passing scores on the "system-theory", " mathlogic " and "cryptography". The first form of Fig. 6 shows a list of students in a situation of "StudentsList" available in JSON-file, the second form is a result of pressing the "SelectStudent" button in a situation of "SelectedStudent". Return to the previous situation occurs when you press the button to the "StudentsList". Button "ToSubject" in both forms are provided to move to the items and request information from the JSON-source to output the number of passings to the chosen subject. Similar forms are also provided for situations "SubjectsList" and "SelectedSubjects". To select one of the items provided for the radio buttons. The purpose of this interface the user to send the appropriate forms in each of the situations, as well as presentation of data connection information from the JSON-sources.
**Dynamic model.** The basis of the application is a model of a new layer of abstraction, which is equipped by new elements and attributes. The dynamic model of this application was considered in (Mironov & Gusarenko, 2012a; Mironov & Gusarenko, 2012b), the main distinguishing features of the Web application model is described in this paper is to define the JSON-documents with elements of "doc", before using this structure describes only the XML-documents in external memory server SODB. Introduced a new type of specification tpl-templates Smarty to display objects. Examples of

**Fig 5.** Data for forming Smarty-object based on the test instances of JSON-objects

**Fig 6.** Examples of form views formed on the basis of test data from the JSON-source using Smarty-object
these agents are shown in Fig. 7 in a "Students-Subjects". The design, denoted by the symbol \texttt{tpl}, tells the interpreter that the template is requested in the sub-state "SelectedStudent" submodels "Students" demands the withdrawal of Smarty-object "Studademicachievement" using tpl-template "Studademicachievement", which is on the external memory in the folder templates Smarty TPL/uspevstud.tpl. Each Smarty-object may be one or more nested JSON-sources, in addition, they may be of different origin, for this type of data has been entered in the type which indicates where it can be accepted for processing content. The most common type of data is source used in the application, is an array, in the model it is referred to as the array. As extra type used in the application has decoded JSON-object in Smarty and output by the receiver content designates a symbol \texttt{rcv} which provides for a new method of data output method = "tpl". Output tpl-method assumes output as tpl Smarty-object using the compiled template tpl.

**Content model for loading Smarty-object.** Content loaded into Smarty-object based on data JSON-sources. The data model is shown in Fig. 8 Data for download
Smarty-object "StudentsList" for students to choose from a document doc "StudentsList" link path = "JSON/stud.json". Information for loading into Smarty-object "SubjectsList" for students to choose from a document doc "SubjectsList" link path = "JSON/ predm.json". To form the final content (see Fig. 8 c, d), the information transferred in the situation of the selected students are selected by the student code and subject to these codes are selected as needed to execute the query for information delivery, such as the name of the student specialty group, the score for the situation "StudentSelected" (see Fig. 8 a, b), as to the situation "SubjectSelected" selected titles for more information, cycle specials. Key elements can be moved in JSON-array of attributes, referred to @attributes in this case the choice of may be produced by attributes in more detail is shown in (Fig. 5 a, b, c).

**Using tpl-templates to display content Smarty-object** The essence use of tpl-patterns is Smarty-forming object and bring it to the final status of content, the desired content, this means that in the situation when all processing procedures are completed and the content is adjusted to the level desired result is its output. Displays results in the client's browser is done by rendering the object in the tpl-template. Tpl-template contains the specifications in the programming language PHP. For interpreter model structure used instead stylesheet construction tpl, which specifies the name of the template to be used for content display. Method tpl used as an alternative method for xslt-transformation. Initial situation contains a list of students, among which we selected the correct student, for detailed information on delivery and subjects. List of students formed on the basis of document doc:StudentsList. For the initial situation "Students" there is a template tpl:StudentsList. The template is compiled, showing each student conceptual model compilation template tpl:StudentsList shown in Fig. 9a. Student code can be selected from both the attributes and properties of the object "Student".

In carrying out each of a FOREACH template element is added to INPUT, which contains the identification information of each student. INPUT element is of type type = "radio", which is displayed in the browser as a set of radio buttons to select one of the students and the button BUTTON to move to the next state "SelectedStudent" reflects the information on the performance of the selected student for this situation, there is a template tpl:Studademicachievement.

Template tpl:Studademicachievement displays the contents of Smarty-object for the selected student conceptual model template compilation is shown in Fig. 9b. The result of the compilation of the second template "Studademicachievement" is a list of elements to partition LI donations in each subject separated from the assessment for each subject triple dash.
Fig 8. Contents Smarty-object formed on the basis of information about students, subjects and submission from JSON-sources

Fig 9. Conceptual models compile time TPL sent to the browser based on the test data from the JSON-Smarty-object: a - Creating a list of students; b - Creating a list of selected student passings
Conclusion

In this paper we consider the problem of using JSON-content in situationally-oriented databases, and introduced into the dynamic model needed for the process formation of JSON-content. Formed the essence of the concept, which is to use the model of a higher level of abstraction for designing applications using states in Smarty-objects to display the JSON-function results. The examples described and used in the model are considered, excluding the particular implementation in the server platform.

Concept also provides bind Smarty-objects to the states of the dynamic model, and their creation, the interpretation is performed automatically, it loads filtering and purification, as well as merging with the defined conditions. JSON-final result is displayed using a prepared tpl-template. The main distinguishing features of this concept are:

• The elements of the state of the dynamic model of situational-oriented databases contain the associated Smarty-objects specifiable by means of introduced in this paper, smarty-elements;
• During interpretation of the dynamic model objects are created automatically, JSON-loaded content, then in accordance with the specified processing conditions formed the resulting content;
• Dynamic model supplemented by additional means for connecting and processing JSON-content situationally-oriented database. According with this concept will lead to the fact that the analyst without involving a programmer will be able to design applications without manual programming, connecting in a dynamic model of JSON-sources and specify the conditions for them to get the result of processing the content and set the template to display it. Further development and implementation of this concept is related to a software implementation on a platform HSMI, to handle HSM dynamic model SODB.

Acknowledgments

This work was supported by RFBR grant 10-07-00167.

References


O’Connor, B. D., Merriman, B., & Nelson, S. F. (2010). SeqWare Query Engine: stor-
ing and searching sequence data in the cloud. Bmc Bioinformatics, 11, 9.
Takahashi, K. (2013). Design and evaluation of lifelog mashup platform with NoSQL database (pp. 133–139). Presented at the Proc. of 15th International Conference on Information Integration and Web-Based Applications and Services, iiWAS’2013, Vienna, Austria.