Offline execution in workflow-enabled Web applications

Edgar Gonçalves
Instituto Superior Técnico
Universidade Técnica de Lisboa, Portugal
edgar.goncalves@gmail.com

António Menezes Leitão
Instituto Superior Técnico
Universidade Técnica de Lisboa, Portugal
aml@gia.ist.utl.pt

Abstract

Today, more and more organizations prefer Web applications to perform their business processes, because these allow collaborative work between several users. However, many tasks must be done while being offline. This goes against today's Web architecture, where a remote server must be contacted for each performed operation. We propose a methodology to develop functionalities that can be executed offline within workflow-enabled Web applications. With these functionalities a user will be able to retrieve his pending work from the server, go offline, perform his activities in the browser and, when he gets back online, synchronize his work with the server, continuing the workflow. The development of such functionalities will take advantage of the organization-specific workflow definitions to manage the application state.

1 Introduction

Nowadays Web based interfaces are taking control over most organization's applications. To better control their operations, workflow management systems can be used to handle business processes. It is, thus, appropriate to want a Web application to control these processes. However, certain activities must take place outside the office, frequently while disconnected from the Internet. It is often desired that users enroll in such activities without leaving the application (e.g., in a laptop), just as if they were facing a common desktop application. To achieve this, Web applications need to provide support for disconnected operations. The fact that this scenario is not contemplated in today's Web application's architecture is our main motivation for the proposal of a new suitable one.

The purpose of this thesis is to enhance workflow-enabled Web applications with offline work capabilities. To this end, a development methodology will be designed and a compliant framework will be implemented. Using this methodology, the final application will support both connected and offline functionalities, without requiring the programmer to explicitly write separate programs for each one of them. From the end-user standpoint, we want him to access the same interface he would if he was connected to the server, not leaving the same vanilla browser (i.e., without any extension), while being disconnected from the Internet.

The requirement of working with Web applications raises several architectural challenges. These applications are inherently built around a client-server link. Also, it is generally assumed that the server holds much (if not all) of the application state. However, a disconnected client needs to manage all the necessary state for the desired functionality. This state comprehends both the application execution and the domain data. The first is hard to exchange between the client and the server due to many language internal details differing on each side, while the latter is usually associated with transactional operations — collaborative work usually causes concurrent accesses to the same information — and access control issues, since not all information is available to a given user.

Aside the aforementioned architectural issues, there is one final, but essential step in the offline work: the server synchronization of all changes made in the client. This may not be a one-step operation, as conflicts may arise that need user input to solve. There will have to be domain-specific merge rules associated to each conflicting data type.

One important characteristic of the considered applications is that they are defined as workflows, which are analyzed and processed by a workflow engine on the server. This means that, at any point in time, it is possible to know exactly where, application-execution-wise, is the user interaction at. We intend to exploit this in order to efficiently transfer execution state between the client and the server. Additionally, the workflow definition will also help us locate, in a given application, all the functionalities available to the user while disconnected from the Internet (i.e., without collaborating with the server and/or other parties).

In the following section we describe the state-of-the-art on this subject. We proceed with a description of this work's research, its goals and the proposed methods to achieve

them. Afterward we mention the current state of our research, followed by our work plan. We finally present the conclusions of this thesis.

2 State-of-the-art

More than a decade ago the Web was made almost exclusively by pure HTML documents, that is, static files with links to other HTML documents. Several approaches[7, 8] enabled offline browsing, but their main purpose was semantic information retrieval, where documents were prefetched by the browser, upon a user search, so that they were available even while disconnected from the Web. Studies[13] were made to prove that the user experience was greatly improved when (1) their usage pattern was detected automatically and the documents were pre-fetched automatically (i.e., without user action), and (2) this fetching was made by a background agent, allowing the user to keep searching normally.

Offline execution isn't new. In fact, desktop applications have been providing offline work for several decades. Their features range from a simple mail client (that can, for instance, pre-fetch all mail from servers and postpone mail sending), to distributed file systems. For instance, CODA[9] allows transparent disconnected file operations, high performance achieved via client side persistent caching, continued operation during partial network failures and a conflict resolution system via file merging. Its merge system is similar to what an offline Web application will need, but it is built only for files and directories, with well known semantics. This wouldn't fit in a data intensive application, where several domain-specific conflicts must be handled.

The offline execution has recently gained momentum in the development community: Apollo[11] can be used to transfer a Web application to the desktop, being able to operate without the need for a remote server. It also breaks the browsing experience, since the user gains a new desktop application. Google Gears[5] has a different approach: it provides browser capabilities to store persistent data on the client-side and to handle online and offline events differently. However deciding whether an operation can/should be used is out of the tool's scope - just like the synchronization of application data between the client and the server.

The first Web applications relied on a thin client model, persisting execution state in the server between user requests. The latest evolutions on client technology have been bringing more functionality to the client-side, allowing the execution to change its state without contacting the server. It is, thus, necessary to synchronize client changes with the server. This is usually done encoding data in HTTP requests parameters. The strategy of passing continuations to express the current execution state[15] appeared to solve the lack of integration of this synchronization process within the server-side programming language paradigms. Specific Web servers were built based on this strategy, using high-level languages (notably, Cocoon[1], PLT Scheme[10] and Termite[4]. This was not only an elegant solution to the exchange of local data between the client and the server (automatically dealing with the (un)marshaling of objects), but also offered the possibility to program client-server interactions, possibly spanning several HTTP requests, in a single function definition. Continuations are not, though, a perfect solution. In the absence of a good resource management strategy execution state exchange via continuations may be the cause for unscalable Web servers.

In spite of browser improvements, the client-side widely used language, ECMAScript, is still limited when compared to a full-fledged server-side language. Developing the server-side and the client-side of the application becomes problematic if we're using two different languages. Recently some attempts were made to cover this gap, notably GWT [6], XML11[14] and Morfik [12] . They allow a programmer to build client-side applications coding only in high-level languages, such as Java. The framework is able to write all necessary HTML and ECMAScript, from the application logic to its user interface. This leaves the programmer with a program (written in a single language) with pieces of code that control application execution in both environments.

3 Research objectives and approach

The main goal of this thesis consists in the creation of a new methodology for the development of workflow-enabled Web applications containing certain functionalities that may be usable while offline. A Web application specification shall include a workflow description of all its functionalities; each task may have a personalized user interface, as well as an associated behavior. A development framework will automatically produce both offline and online programs for these tasks descriptions. The programmer will also be able to specify both task and domain model restrictions (e.g., forcing a given information to be accessed in the server, only). Finally, the programmer may have to define strategies to allow a user to recover from a conflict in the synchronization stage. These strategies will be associated with the domain model contents and/or with specific workflow tasks. This step may not be fully automated by the framework, since each conflict type is generally application specific. The remaining steps of the development process using the proposed methodology shall not differ from the development of current Web applications.

The helper framework will benefit from a running workflow engine (on the server) to know, at each moment, what workflows are available to the present user. It will also use

the engine to know about the existing data flows between tasks. In the synchronization stage, the engine will update the workflow instances according to the information from the client, as well as inform about conflicting situations.

To produce such a methodology, three important research topics must be addressed:

1. **Offline module identification**: The functionalities that the user is going to need for his work are a subset of the entire application. They must be well identified, along with the data items needed for their execution. What makes a given functionality suitable to be operated offline? When access control is concerned, what guarantees of data privacy can be given in this process? The programmer will only be responsible for describing both data and workflow tasks, so that module detection may be done automatically.

2. **Offline application creation**: The client-side application must be generated, and transfered to the client. How can both online and offline versions of a functionality be generated from a single source? How will the offline application save its alterations to be synchronized later? This step will be fully automated, requiring little or no effort from the programmer.

3. **Changes synchronization**: Each modification the user makes while offline must eventually be propagated to the server. However, each existing conflict (originated by concurrent data modifications) must be handled so that the user may gracefully decide what to do with his work. How is this synchronization going to be integrated with the transactional scheme of the server-side application?

3.1 Research method

The research to support this thesis will be based on a literature analysis of existing approaches that aim to (partially) solve the problem in hands. This will lead to the creation of a new software architecture and a development methodology, based on workflow specifications together with domain models. These models will be enriched with data security information (e.g., access control), as well as with specifications of conflict resolution strategies. A prototype framework will be developed to support this methodology. This, along with required notations specifications, shall be used to implement the desired Web applications. The framework will shorten the development process as the offline code is to be created automatically, from the discovery of available offline modules to the conflict detection and recovery system.

The prototype simulation will allow us to validate of the methodology, measuring its success with real workflows and user(s) interactions. Afterward the methodology shall be applied in the Fenix[3] academic information management system, where the Worksco[2] workflow engine will be used. This will constitute a case study, as well as a source of feedback: elected, in-production modules will be adapted to be able to be operated offline, thus allowing an iterative fine-tune of the framework.

4 Current work and preliminary results

Having delved into the literary research, we have already specified an algorithm to partition a workflow in a sub-workflow capable of being executed offline by a given user. The algorithm consists in the computation of the connected graph from an initial task, given as input a model of the workflow based on annotated directed graphs. The set of initial tasks is provided by the workflow engine at any given moment, so the framework can easily compute a set of offline executable workflows for the user. Restrictions to specific tasks that must be performed while connected to the server are modeled via annotations in the nodes that represent each task.

5 Work plan and implications

The work plan for this PhD is as follows:

Prototype development (6 months) including research on data synchronization and client-side code generation;

Prototype application (12 months) integrating it with Fenix;

Evaluation (6 months) by means of stress tests with multiple usage scenarios, and user feedback collection;

Documentation (24 months) is a continuous process, and has already been started.

In the end, we aim to give the following contributions:

- A methodology description on how to include offline execution in an (already existing) Web application;
- A prototype of a framework that implements the methodology;
- Adaptation of an academic information management system to use the proposed methodology;
- Articles will be written with emphasis on the Fenix integration, on offline execution (based on the prototype) and on execution state exchange using a workflow engine;

6 Conclusions

This thesis aims to enable offline work in workflow-enabled Web applications. To achieve this, a development methodology will be proposed, grounded on the application's workflow definitions and domain model. The final framework will automate the production of the offline portion of the application, with focus on offline modules detection, conflict resolution and data synchronization. The proposed methodology will be applied to Fenix, where it will be tested with heavily used scenarios.

References

[1] A. Belapurkar. Use continuations to develop complex web applications — a programming paradigm to simplify mvc for the web. Technical report, IBM, 2004.

[2] S. M. M. Fernandes. A workflow virtualmachine. Master's thesis, Instituto Superior Técnico — Universidade Técnica de Lisboa, 2005.

[3] Project fénix. http://fenix-ashes.ist.utl.pt/.

[4] G. Germain. Concurrency oriented programming in termite scheme. In *ERLANG '06: Proceedings of the 2006 ACM SIGPLAN workshop on Erlang*, New York, NY, USA, 2006. ACM Press.

[5] Google. Google gears. http://code.google.com/apis/gears/.

[6] Google. Google web toolkit. http://code.google.com/webtoolkit/.

[7] A. Joshi, S. Weerawarana, and E. Houstis. On disconnected browsing of distributed information. *Proc. Seventh IEEE Intl. Workshop on Research Issues in Data Engineering (RIDE)*, 1997.

[8] R. Kavasseri, T. Keating, M. Wittman, A. Joshi, and S. Weerawarana. Web Intelligent Query-Disconnected Web Browsing using Cooperative Techniques. *Proc. 1st. IFCIS Intl. Conf. on Cooperative Information Systems*, 1996.

[9] J. KISTLER and M. SATYANARAYANAN. Disconnected Operation in the Coda File System. *ACM Transactions on Computer Systems*, 10(1), 1992.

[10] S. Krishnamurthi, P. W. Hopkins, J. McCarthy, P. T. Graunke, G. Pettyjohn, and M. Felleisen. Implementation and use of the plt scheme web server. *Journal of Higher-Order and Symbolic Computing*, 2007.

[11] A. Labs. Apollo. http://labs.adobe.com/wiki/index.php/Apollo.

[12] Morfik. Webos appsbuilder. http://www.morfik.com/.

[13] J. Pitkow, M. Recker, G. I. of Technology, Visualization, and U. C. Graphics. *Integrating Bottom-up and Top-down Analysis for Intelligent Hypertext*. Graphics, Visualization & Usability Center, Georgia Institute of Technology, 1994.

[14] A. Puder. Xml11: an abstract windowing protocol. *Sci. Comput. Program.*, 59(1-2):97–108, 2006.

[15] C. Queinnec. The influence of browsers on evaluators or, continuations to program web servers. *Proceedings of the fifth ACM SIGPLAN international conference on Functional programming*, 2000.