Providing Session Management as Core Business Service

Ismail Ari, Jun Li, Riddhiman Ghosh, Mohamed Dekhil
Hewlett-Packard Laboratories, 1501 Page Mill Rd. Palo Alto, CA, 94304, USA
ismail.ari@hp.com, jun.li@hp.com, riddhiman.ghosh@hp.com, mohamed.dekhil@hp.com

ABSTRACT
It is extremely hard for a global organization with services over multiple channels to capture a consistent and unified view of its data, services, and interactions. While SOA and web services are addressing integration and interoperability problems, it is painful for an operational organization with legacy systems to quickly switch to service-based methods. We need methods to combine advantages of traditional (*i.e.* web, desktop, or mobile) application development environments and service-based deployments.

In this paper, we focus on the design and implementation of session management as a core service to support business processes and go beyond application-specific sessions and web sessions. We develop local session components for different platforms and complement them with a remote "session service" that is independent of applications and platforms. We aim to close the gap between the two worlds by combining their performance, availability and interoperability advantages.

Categories and Subject Descriptors
D.2.13 [**Software Engineering**]: Reusable software–*reuse models.* H.3.5 [**Information Storage and Retrieval**]: Online information services– *web-based services.*

General Terms
Design, Management, Performance, Reliability.

Keywords
Session service, data serialization, multi-channel integration.

1. INTRODUCTION
Today, many complex tasks, projects or business workflows are completed using information from multiple sources and in a collaborative fashion. For example, consumers make their buying decisions using multiple channels such as web sites, printed-ads, mobile devices, store kiosks and their decisions are affected by family members, friends, and online reviews. It is extremely challenging for today's global enterprises with disparate silos of Information Technology (IT) applications to give their customers a consistent view of offered services and get a unified view of business state while customers are interacting with their systems.

SOA paradigm and Web Services (WS-*) efforts are addressing these business and IT integration problems. Therefore, many organizations are already along the way for SOA migration of their IT infrastructures. However, several real-world challenges slow down the SOA adoption. These challenges include dependence of business continuity on legacy systems, disconnected computing needs, pains of converting existing functions into web services, the lack of common schemas and rich set of reusable enterprise services, and finally the lack of trust to "service providers" for guaranteed availability, performance, privacy or security of services and at "reasonable" prices. Our design methodology addresses some of these concerns.

In many desktop applications a session module is embedded in the application, but it cannot be reused with other applications. HTTP sessions have enabled different web browser based applications to track state of multiple user sessions concurrently and solved many E-commerce problems (as HTTP protocol is stateless). Yet, HTTP sessions are not available for PC or mobile desktop applications. Furthermore, with web sessions customers and business owners have no control over how, when, or where a session's state will be stored and then reloaded. Our session service allows such choices and enables multi-channel availability. Recently, WS-Resource Framework (WSRF) [1,2] and WS-Context [4,3] specification were proposed to standardize representation of service state. While these proposals enable sharing of state among web services, we believe that legacy, performance, trust, *etc.* issues mentioned above will limit adoption of solely web-based state management approaches for many business processes. Our approach uses local components matched by a remote session service. The remote component is currently implemented using web services and it can be rendered to comply with WSRF in the future.

2. SESSION SERVICE DESIGN
Figure 1 shows the high-level design of our session service and the motivating retail business use cases. The local session managers coexist with traditional application development environments such as desktop, web, or mobile platforms to provide them with features such as serialization of session state, fast local storage, session timing, handling of session events, and communication with the remote session service. The remote

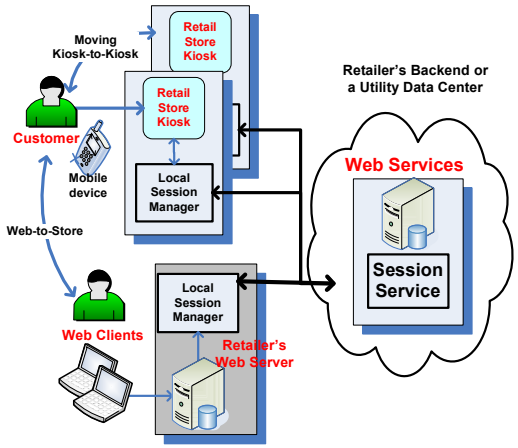

Figure 1. Session manager design and retail use cases.

Copyright is held by the author/owner(s).
WWW 2007, May 8--12, 2007, Banff, Alberta, Canada.
ACM 978-1-59593-654-7/07/0005.

component resides at the backend and provides interoperability, reliable long-term storage of enterprise-scale data, multi-channel availability, and data mining services. Customers can seamlessly move between channels to complete their tasks.

Users with privacy concerns can use their personal mobile devices to carry state. By separating the design concerns, we can combine the ease of integration and performance advantages of the local component with the availability, reliability of a service-based remote component at the backend. This way we aim to close the gap between traditional and service-based deployments.

2.1 Design Details

A single instance of a local **`SessionManager`** class is created when a local application is started or the web server receives its first request. The session manager checks whether a previous session exists for the given userid or sessionid. If so, it loads this session either from a local resource (file, table) or imports it from the remote service. If no previous session exists, a new session is created. **`SessionState`** classes serve as containers for application states (*e.g.* **`Session["ShoppingCart"]=cart`**) similar to web sessions, but in our case a session can be saved and loaded at any channel. They know *where*, *how* and *when* to store data. For example, the state can be stored in memory temporarily, in a file/database table, or in web server's session context (where). XML, binary, or other data serialization types can be used (how). Finally, state synchronization with remote session service can be immediate, deferred or never based on the policy (when). The remote session service has two basic methods– **`ExportSession`** and **`ImportSession`**– (Fig. 2) which can be used by local session managers and other application-specific web services to persist session state and load it back, respectively.

2.2 Use Cases for Session Service

Session manager deals with different dimensions of user sessions. The first is multi-channel application continuity. Any serializable application state can be saved and reloaded at the same or different channels. For retail-related business applications (*e.g.* in Fig.2) state may include shopping carts, shopping lists, recently viewed items, purchase history, privacy/security settings, and UI customizations. Another aspect of session management is the timing of sessions, detecting inactivity, pausing and resuming a session (*e.g.* during a training). Group management is another aspect. The current state of a session can be shared among users as a whole or in finer-granularity per application state (*e.g.* just share my baby registry). Both application designers and customers can have control over the policies such as what is stored for how long and who gets to see what (*i.e.* access and authorization control). A shopping session can be started by one person (wife) and continued by another (husband) at another channel. It can be transferred from one role (clerk) to another (manager).

Session manager supports core session/state-related features, which can be used by applications (*e.g.* Business Intelligence) and other core services (*e.g. authentication & session tokens*). The collected state can be data-mined either offline or in real-time to provide personalized services. Using session data we can report channel usage, conversion rates, channel spending, popular items, *etc.* which lead to characterization of multi-channel shopping behavior. Thus, we introduce a collection of new metrics, which are not widely available, but desperately needed by businesses today. Existing session managers do not support these features lacking the ability to become business-integrated.

3. PROOF OF CONCEPT

The session manager is implemented using .NET and its operation is demonstrated in a retail setting including an in-store kiosk and a mobile/desktop retail program to constitute different shopping channels as shown in Figure 2. We use the retail kiosk and then save the shopping cart and viewed coupons into session service. Next, we access and update the carts using the mobile/desktop application as a proof of concept and interoperability.

Figure 2. Screenshots of retail applications that use our session service implementation.

4. CONCLUSIONS AND FUTURE WORK

We described the design of session management that supports business processes with both disconnected and networked parts. We compared our work at a high-level to some existing solutions and showed a proof of concept implementation. We demonstrated a multi-channel integration use case for a retail application, which is known to affect customer loyalty. We are currently making performance evaluations, integrating session with authentication-authorization services, addressing mobility issues, and adding Join/LeaveSession functionality for collaborative applications.

5. REFERENCES

[1] Czajkowski, K., et al. 2004. The WS-Resource Framework. http://www.globus.org/wsrf/

[2] Geoffrey Fox, "Grids of Grids of Simple Services," Computing in Science and Engineering, vol. 06, no. 4, pp. 84-87, Jul/Aug, 2004.

[3] Hildebrand, H., Karmarkar, A., Little, M., Pavlik, G.: Session Modeling for Web Services. In IEEE ECOWS 2005.

[4] Little, M., Newcomer, E. and Pavlik, G. Web Services Context Specification (WS-Context). OASIS Committee Draft v.0.8. 2004.