RESTful Web Services: Principles, Patterns, Emerging Technologies

Cesare Pautasso
Faculty of Informatics
University of Lugano
c.pautasso@ieee.org

Erik Wilde
School of Information
UC Berkeley
dret@berkeley.edu

ABSTRACT

Recent technology trends in Web services indicate that a solution eliminating the perceived complexity of the WS-* standard technology stack may be in sight: advocates of *Representational State Transfer (REST)* have come to believe that their ideas explaining why the World Wide Web works are just as applicable to solve enterprise application integration problems and to radically simplify the plumbing required to implement a *Service-Oriented Architecture (SOA)*. In this tutorial we give an introduction to the REST architectural style as the foundation for RESTful Web services. The tutorial starts from the basic design principles of REST and how they are applied to service oriented computing. Service-orientation concentrates on identifying self-contained units of functionality, which should then be exposed as easily reusable and repurposable services. This tutorial focuses not on the identification of those units, but on how to design the services representing them. We explain how decisions on the SOA level already shape the architectural style that will be used for the eventual IT architecture, and how the SOA process itself has to be controlled to yield services which can then be implemented RESTfully. We do not claim that REST is the only architectural style that can be used for SOA design, but we do argue that it does have distinct advantages for loosely coupled services and massive scale, and that any SOA approach already has to be specifically RESTful on the business level to yield meaningful input for IT architecture design.

Categories and Subject Descriptors

H.3.5 [**Information Storage and Retrieval**]: Online Information Services—*Web-based services, Data sharing*

General Terms

Design, Documentation, Languages

Keywords

SOA, REST, Web Services, Patterns

1. AIMS/LEARNING OBJECTIVES

The primary goal of this tutorial to close the gap between the high-level concept of *Service-Oriented Architecture (SOA)*, and the question of how to implement such an architecture once services have been identified. Colloquially, it is often assumed that "services" in a Web-oriented are implemented as "Web services", and these are often exclusively perceived as using the SOAP/WS-* stack of protocols. Our goal is to describe that "Web services" can also use other technologies, such as HTTP correctly used following the principles of the REST architectural style. Furthermore, we will explain how a disciplined process can lead from the business level, which is mainly about identifying services on an abstract level, to an IT architecture, and that it is important to not impose architectural constraints (such as defining service in a function-oriented way rather than in a resource-oriented way) too early in the process. Web Services have been of increasing interest in the past years. While "Web Services" were first defined as machine-accessible services based on Web technologies, the term quickly was perceived as exclusively referring to SOAP-based services, which mirror the traditional IT integration style of distributed objects with messaging technologies layered on top of Web technologies [7, 3]. This tutorial focuses on an alternative approach towards the design and implementation of Web Services, based on the architectural style of the Web itself, *Representational State Transfer (REST)* [2]. Thus, the tutorial learning objectives specifically include giving an in depth understanding on the design principles and patterns behind RESTful Web Services, as well as how the two different styles of Web Services compare [5], and why the decision of SOAP vs. REST is fundamental and needs to be made early in any SOA project. We focus on the question of how to do SOA with REST, and do so by explaining and highlighting the fundamental differences between the more tightly coupled "integration" style of distributed objects, and the more loosely coupled "cooperation" style of REST. Thus, attendees will gain a new perspective that highlights the importance of following Web principles to support the design of Web Services and learn why not all Web Services are made equal and that only RESTful ones are really "of the Web, " whereas others are just implemented "on the Web." [6]

2. TUTORIAL DESCRIPTION

There are two main explanations of this ongoing debate between the RPC style and the REST style. The first explanation can be given in technical terms, in the sense that RPC style middleware, based on the idea of building distributed systems, has been used for a long time and has become the predominant architectural style in enterprise IT. REST and its approach of building information systems based on loose

WWW 2010, April 26–30, 2010, Raleigh, North Carolina, USA.
ACM 978-1-60558-799-8/10/04.

coupling has a different background, and even though its success on the Web is evident, it still often is pictured as being inappropriate for enterprise IT architectures. The second explanation is that RPC vs. REST is an instance of sustaining vs. disruptive innovation, and that the adoption curve for something as heavyweight as enterprise IT architectural styles is long, and thus it will take a long time to see whether REST in the end will be able to deliver better solutions at lower cost. In either case, it can be safely assumed that REST is not a substitute for RPC-oriented middleware, but an alternative style which could be considered as an alternative approach for designing a system architecture. Thus, our main goal is not to take sides in the ongoing SOAP vs. REST debate, but to explain why REST is an interesting alternative for SOA, and how it influences SOA on various levels.

While SOA is mainly about business-level processes, it inevitably produces "models" of system architectures, and in many cases these models already have a bias towards certain modeling styles. REST is an "architectural style" and therefore already comes into play when producing these high-level models (which often do not use any formal modeling approach). The main question is how any SOA project is defining a what a service is, and how that decision is reflected in the model. The two main approaches are the viewpoint of *distributed objects*, or that of *document exchanges*. In the *distributed objects* style, the two main methods which are later used for defining the interactions with services are *Remote Procedure Calls (RPC)* and *message passing*, but both approaches are based on the same assumption that a service essentially is a function-oriented interface exposed through distributed object technology. In the *document exchanges* style, services are accepting and generating self-contained description of service invocations and results, but there is no underlying assumption of interacting with a stateful remote object that represents a service invocation. Instead, the service invocation carries no state beyond one interaction. This radically changes the model of how services have to be designed, and how to interact with them. It also radically changes the way in which services can scale, because there is no need to keep state on the service side.

3. PRESENTER BIOGRAPHY

- *Cesare Pautasso* is assistant professor in the new Faculty of Informatics at the University of Lugano, Switzerland. Previously he was a researcher at the IBM Zurich Research Lab and a senior researcher at ETH Zurich. His research focuses on building experimental systems to explore the intersection of model-driven software composition techniques, business process modeling languages [4]. Recently he has developed an interest in Web 2.0 Mashups [1]. He is the lead architect of JOpera, a powerful rapid service composition tool for Eclipse. His teaching and training activities both in academia and in industry cover advanced topics related to Web Development, Middleware, Service Oriented Architectures and emerging Web services technologies.
- *Erik Wilde* is adjunct professor at UC Berkeley's School of Information. His general area of interest are open information systems. His main expertise are Web technologies and, more specifically, XML-related technologies [11, 12]. Erik has also worked on how to expose linked information in a lightweight way [13], and on the general question of how to better integrate the Web perspective into how Web-based applications are built [8, 9, 10].

4. REFERENCES

[1] Bioern Biörnstad and Cesare Pautasso. Let it flow: Building Mashups with Data Processing Pipelines. In *Proc. of the 1st International Workshop on Web APIs and Services Mashups (Mashups'07) at ICSOC 2007*, volume 4907 of *LNCS*, pages 15–28, Vienna, Austria, September 2007.

[2] Roy Thomas Fielding. *Architectural Styles and the Design of Network-based Software Architectures*. PhD thesis, University of California, Irvine, Irvine, California, 2000.

[3] Ken Laskey, Philippe Le Hègaret, and Eric Newcomer, editors. *Workshop on Web of Services for Enterprise Computing*. W3C, February 2007. `http://www.w3.org/2007/01/wos-ec-program.html`.

[4] Cesare Pautasso. BPEL for REST. In *Proc. of the 6th International Conf. on Business Process Management (BPM 2008)*, volume 5240 of *LNCS*, pages 278–293, September 2008.

[5] Cesare Pautasso, Olaf Zimmermann, and Frank Leymann. RESTful Web Services vs. "Big" Web Services: Making the Right Architectural Decision. In Jinpeng Huai, Robin Chen, Hsiao-Wuen Hon, Yunhao Liu, Wei-Ying Ma, Andrew Tomkins, and Xiaodong Zhang, editors, *17th International World Wide Web Conference*, pages 805–814, Beijing, China, April 2008. ACM Press.

[6] Steve Vinoski. RPC and REST: Dilemma, Disruption, and Displacement. *IEEE Internet Computing*, 12(5):92–95, September 2008.

[7] Werner Vogels. Web Services are not Distributed Objects. *IEEE Internet Computing*, 7(6):59–66, December 2003.

[8] Erik Wilde. Declarative Web 2.0. In Weide Chang and James B. D. Joshi, editors, *2007 IEEE International Conference on Information Reuse and Integration*, pages 612–617, Las Vegas, Nevada, August 2007.

[9] Erik Wilde. The Plain Web. In *First International Workshop on Understanding Web Evolution (WebEvolve2008)*, pages 79–83, Beijing, China, April 2008.

[10] Erik Wilde and Martin Gaedke. Web Engineering Revisited. In *2008 British Computer Society (BCS) Conference on Visions of Computer Science*, London, UK, September 2008.

[11] Erik Wilde and Robert J. Glushko. Document Design Matters. *Communications of the ACM*, 51(10):43–49, October 2008.

[12] Erik Wilde and Robert J. Glushko. XML Fever. *Communications of the ACM*, 51(7):40–46, July 2008.

[13] Erik Wilde and Yiming Liu. Lightweight Linked Data. In *2008 IEEE International Conference on Information Reuse and Integration*, Las Vegas, Nevada, July 2008.