

Data-Centric Systems and Applications

Series Editors

M.J. Carey

S. Ceri

Editorial Board

P. Bernstein

U. Dayal

C. Faloutsos

J.C. Freytag

G. Gardarin

W. Jonker

V. Krishnamurthy

M.-A. Neimat

P. Valduriez

G. Weikum

K.-Y. Whang

J. Widom

For further volumes:

<http://www.springer.com/series/5258>

Florian Daniel • Maristella Matera

Mashups

Concepts, Models and Architectures



Springer

Florian Daniel
Università di Trento
Povo, Trento
Italy

Maristella Matera
Politecnico di Milano
Milano, Italy

ISBN 978-3-642-55048-5 ISBN 978-3-642-55049-2 (eBook)

DOI 10.1007/978-3-642-55049-2

Springer Heidelberg New York Dordrecht London

Library of Congress Control Number: 2014943453

© Springer-Verlag Berlin Heidelberg 2014

This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed. Exempted from this legal reservation are brief excerpts in connection with reviews or scholarly analysis or material supplied specifically for the purpose of being entered and executed on a computer system, for exclusive use by the purchaser of the work. Duplication of this publication or parts thereof is permitted only under the provisions of the Copyright Law of the Publisher's location, in its current version, and permission for use must always be obtained from Springer. Permissions for use may be obtained through RightsLink at the Copyright Clearance Center. Violations are liable to prosecution under the respective Copyright Law.

The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

While the advice and information in this book are believed to be true and accurate at the date of publication, neither the authors nor the editors nor the publisher can accept any legal responsibility for any errors or omissions that may be made. The publisher makes no warranty, express or implied, with respect to the material contained herein.

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

To Cinzia, my family, my friends.
Florian

To Francesca, Alessandro, Fabio.
Maristella

Foreword

During the last few years, we have witnessed several computing advances that are transforming the Internet into a global development and deployment platform, including Web services, social software, the Internet of things, and cloud computing. Web services enable modular and uniform access to heterogeneous and Internet-accessible resources. Social software technologies provide a Web-scale sharing platform and support participants' collaboration. Cloud computing is transforming both software and hardware resources into elastic resources that deliver on-demand through Web services. The Internet of things provides monitoring and control services over appliances, mobile devices, sensors, vehicles, and consumer electronic devices.

In a nutshell, the new computing environment enabled by advances in the above areas consists of a network of services representing data sources, software as well as hardware resources, and networked devices distributed over the Internet. These services are in general accessible through Application Programming Interfaces (APIs) in order to allow third-party developers to write auxiliary or satellite apps that add new uses to the original service, enrich its features and accessibility, enhance its agility, and accelerate overall development. For instance, in modern enterprise application development, access to internal data and applications is provided through APIs.

A key programming paradigm of APIs and service-based applications is composition (also called service aggregation). The benefit of service aggregation originates from the added value generated by the possible interactions and from the large scale rather than from the capabilities of its individual component services separately. This offers tremendous automation opportunities in a variety of application domains. Web services or API mashups emerged as a mainstream service aggregation approach over the last few years. Domain-specific as well as general-purpose mashup platforms and tools flourished with the growth of APIs and services. Primary aggregation tasks supported by mashups focus on data aggregation from prebuilt components. However, tasks such as data visualization pipelines or UI widget-based portal development are also common in mashups.

It should be noted that mashups target both professional and end-user programmers. For instance, within an organization, programmers using mashups to build integrations over Web services include developers, consultants, and business analysts or business process experts. End-user programmers (e.g., skilled knowledge workers, domain experts) often need to access and integrate information from various sources to perform their routine tasks. These programmers should—as is the case with professional programmers—also be able to benefit from the power of service-oriented architectures (SOAs) and APIs.

The authors provide a very informative discussion on the state of the art in mashups. They introduce the necessary background and survey the main issues, solutions, tools, platforms, and applications. They also clarify the fundamental concepts behind mashups and illustrate the proposed concepts and techniques in different application domains. This book is timely, provides a thorough scientific investigation, and also has practical relevance in the general area of composition and mashups. It is of particular interest to researchers and professionals wishing to learn about relevant concepts and techniques in service mashups, composition, and end-user programming.

Sydney, NSW, Australia
February 2014

Boualem Benatallah

Preface

By now, the Web has completed its evolution from the one-way communication medium of the early 1990s, dominated by developers and information providers (the so-called Web 1.0), into a fully distributed and democratic communication platform that involves developers, information providers, and consumers alike (the so-called Web 2.0). Many factors have contributed to this evolution. The emergence of the SOA has provided a huge number of distributed and reusable services and increased interoperability among applications. Cloud computing, with its Software as a Service (SaaS) paradigm, has changed the way applications are distributed and consumed. HTML 5 with its Web Sockets has turned traditional client-server Web architectures into full-fledged, distributed programming environments. And thanks to the increasing availability of sophisticated mobile devices, such as smartphones or tablets, ubiquitous Web access has not only become a reality but simply common practice.

In parallel to these technological advances, societal changes have reshaped the nature of consumers and end users acting in the Web. While only 10 years ago the average Web user was barely able to navigate content through static Web applications, today the same users actively contribute to the success of the Web by generating content, feedbacks, ratings, and similar. That is, the typical Web user of today is simply more knowledgeable in terms of IT competences and more open to experimentation with software than the Web users of only a few years ago.

In this technological and societal context, *Web mashups* have emerged as an innovative software trend that reinterprets existing building blocks of the Web and leverages on novel perspectives and competences. Mashups allow one to go beyond the utility of individual components by composing them in novel, value-adding manners. But their strength also derives from the appeal they exert on nonprogrammers (e.g., thanks to the presence of user interface components and widgets that are intuitive even to common Web users) and their potential to turn nonprogrammers into developers. In fact, while initially mashups were merely developed manually by skilled programmers, the phenomenon inspired a new generation of tools for the computer-assisted composition of mashups, starting from heterogeneous components available on the Web. Differently, for example, from

Web service composition tools, mashup tools bet on simplicity, abstraction, and ease of development, rather than on completeness of features or low-level details. As a result, in many cases by now mashup tools represent a viable alternative to Web service composition instruments, such as BPEL editors and engines. In short, they are another step toward the dream of the “programmable Web.”

Mashups have been attracting a lot of attention from Web developers and programmers, yet the huge number of scientific publications on mashups and related topics testifies also a strong interest in the phenomenon from the academic community. We personally experience this interest every day in our own work, with our own research on mashups and by teaching enthusiastic Ph.D. students and collaborating in industrial and EU-funded research projects. This is the context in which the idea of writing this book was born: Increasingly, we noticed that it was difficult to recommend a reading list or a good book to our students, colleagues, or the various people who ask us for material to get acquainted with the topic and learn its fundamental concepts. Before writing this book, we were just able to suggest some (few) scientific papers dealing with specific theoretical issues and a number of different books focusing on how to exploit generic Web technologies and programming languages to program mashups.

This book aims to fill this gap and to become a reference for mashups. It specifically aims to provide a systematic view on mashups, on the main concepts and techniques underlying mashup design and development, on the synergy among the involved models at different levels of abstraction, and on the way models materialize into composition paradigms and architectures of corresponding development tools. As such, the book aims to provide an academic perspective on the problem. However, it does not disregard the necessary technologies. The first part of the book, therefore, introduces the theoretical and technological foundations that are the basis for designing and developing mashups, as well as for designing tools that can aid mashup development. The second part then focuses more specifically on mashups and their various aspects. It discusses a set of core component technologies, core approaches, and architectural patterns for mashups and mashup tools, with a particular emphasis on tool-aided mashup development exploiting model-driven architectures. Development processes for mashups are also discussed, and particular emphasis is devoted to composition paradigms for the End-User Development (EUD) of mashups and quality issues.

In our explanations, we especially focus on *concepts*, *models*, and *architectures* in an attempt to look at the mashup phenomenon from a scientific perspective and to highlight the conceptual issues that underlie these applications, which oftentimes are independent of the particular technologies adopted by mashup components or the mashups themselves and, thus, of more general validity. Technologies—especially new ones—typically attract the attention of the passionate programmers, while they also risk to distract them from the core problems to be solved, if these are not properly understood and mastered before starting a new implementation. With this book, we therefore would like to help the reader understand and focus on what we think are the core problems.

Writing a book like this is always a trade-off between completeness and time. The area of mashups is vast, and covering all aspects is not possible. With this book, we did our best to identify the right trade-off by concentrating our attention to those aspects we are most familiar with. As a result, we, for instance, privilege model-driven development and composition-centric approaches, while we acknowledge that there is a variety of other mashup development approaches that would deserve more attention. For example, in this respect we can think of the use of natural language to develop mashups, or of scripting languages, or similar. It was also our initial intention to dedicate one chapter to the comparison of the various mashup tools and platforms that are available online today. Throughout the book, we make extensive use of examples of mashup tools and platforms; yet, we were not able to systematically survey the complete spectrum of existing tools. Similarly, with its scientific perspective on the problem, the book does not specifically study the different market models or the economic value of mashups and mashup practices, for example, in everyday software development practices. Unfortunately, as of today only little can be learned from the literature available. We are however confident that the material we cover is representative of the mashup phenomenon and that it provides significant insight into both concepts and technologies. We can hopefully fill the aspects we miss with other publications on the topic or—why not!—with a second edition of this book.

Intended Audience

The book has the following *goals*:

- To introduce the readers to mashups and to explain what mashups are, what their benefit is, who the possible users are, how they can be used and developed in practice, and how they relate to existing technologies that characterize the modern Web
- To help the readers understand how the presented technologies, models, and architectures can be used to engineer their own mashups as well as the tools aiding mashup development in different contexts and for different target users—from expert-oriented data and application integration to end-user composition of UI components
- To convey the salient research aspects related to mashups and to show how they relate to research areas like data integration, service-oriented computing, and Web engineering, while also highlighting the still open research challenges and opportunities

Given these goals, the book targets a wide range of potential *readers*:

- *Students and teachers*: The book overviews technologies, solutions, and methodologies in the area of Web engineering, service-oriented computing, and data and application integration and provides the necessary knowledge to understand

and develop mashups and mashup tools. As such, it can serve as a textbook for courses on Web and Internet Technologies, Web Engineering, and Web Information Systems, as well as for self-teaching.

- *Researchers and academics*: The book provides researchers, especially in the areas of Web engineering, service-oriented computing, and data and knowledge management, with a systematic overview of research aspects, conceptual models, and approaches to develop mashups and mashup tools. They will understand the open issues and challenges related to technologies, methodologies, and tools.
- *Practitioners and software architects*: The book does not provide low-level details on mashup development. Yet, we believe it is a valuable instrument to practitioners or software architects who are new to the topic and, as such, would like to understand what mashup development is about. The book can guide them toward an understanding of the main problems to be solved and the key concepts to know, as well as toward identifying possible solutions and new development practices for their software projects.
- *CIOs and IT decision makers*: With the growing hype of enterprise mashup platforms, on the one hand, and the difficulty to make return on investment on the money spent for turning their IT infrastructures into service-based ecosystems, on the other hand, the book will help CIOs and IT decision makers understand the benefits that mashups and mashup tools can bring to them and to assess the value of mashup-based integration practices to their company.

We tried to make the book self-contained by providing not only foreground information on mashups but also the necessary background information that is needed to understand the presented material. Although in Part I we briefly discuss the most relevant technologies involved in the development of mashups, it is not the intention of this book to teach how to use them in practice, for example, to program a concrete piece of code. Prior knowledge of Web technologies, such as HTML, CSS, JavaScript, and server-side languages like Java or PHP, is therefore beneficial but not strictly mandatory. Knowledge on programming and on the basic principles of Web and software engineering is however recommended for a in-depth comprehension of the presented concepts.

Accompanying Material

A book is by definition a static snapshot of a given topic taken in a given instant of time and, more so, even a limited one. In order to provide the reader with additional material that would not have fit into this book and with updated information regarding mashups, we therefore set up a dedicated Web site that complements the book with a collection of online resources. The Web site can be accessed via the following URL:

<http://www.floriandaniel.it/mashupsbook/>

The Web site aims to provide *teaching material* for teachers and professors who want to use the book as a textbook in their classes or for self-learning by students. Specifically, it provides access to the figures of the book and additional material for each topic covered in the book. It further provides students, practitioners, and researchers with additional links to *external online resources* that allow them to deepen their understanding of the field and to learn more about the technologies involved in mashup development. The Web site also provides researchers with updated links to *scientific publications* regarding mashup development and related topics.

We also would like to use the Web site as an instrument to *collect feedback* from the readers and to answer possible questions, thereby establishing a positive feedback loop from the readers to the authors. A suitable comments section allows the readers and authors to communicate and discuss.

Acknowledgments

This book is the result of several years of work in the context of mashups and of about 2 years of writing. It summarizes our experience and personal view on the topic, which were of course not possible without the help from a variety of different people who, directly or indirectly, contributed to enhance our experience as well as the software products and prototypes produced together and discussed in this book.

We would like to thank our Ph.D. students who wrote their dissertations on mashups: Matteo Picozzi proved that mashup development can be afforded even by laypeople if adequate modeling abstractions and interactive composition paradigms are provided. Soudip Roy Chowdhury nicely showed how to ease and speed up mashup development with the recommendation of mashup model patterns. Carlos Rodríguez did a great job with the mining of meaningful mashup model patterns from repositories of existing mashup models. Stefano Soi worked hard on the conceptual development approach for mashup languages and platforms. Stefano Tranquillini contributed to the orchestration of distributed user interfaces with the MarcoFlow project, sponsored by Huawei Technologies, China.

Throughout all these years, we have also had the pleasure to work with several BSc and MSc students at the Politecnico di Milano and the University of Trento; we would like to thank them for their commitment and creativity in the projects and diploma theses they developed together with us.

Part of our research on mashups was funded by and conducted in the context of research projects. We mention here the ERC project SeCo (Search Computing), which investigated how the integration of data services can be exploited for answering complex search queries, and the EU FP7 project OMELETTE, which focused on enabling the development of telco mashups. Our thanks go to all the partners and the people involved in these projects.

We deeply thank Boualem Benatallah, Fabio Casati, and Jin Yu for kicking off with us this line of research with the publication, in 2007, of our joint papers on

UI Integration, which then turned into research on mashups. Jin Yu implemented the very first prototype of a mashup approach, Mixup, which helped us prove that the integration of UI components, never investigated in these terms before, was possible. Together with Fabio Casati, Mixup then evolved into MashArt, a tool characterized by what we call “Universal Integration.” From Mixup the research line on the EUD of mashups also originated; it resulted in a series of tools embodying “live” composition paradigms and collaboration mechanisms. We would also like to thank Cinzia Cappiello for inspiring and helping us with the works on quality in mashups, and Maria Francesca Costabile and the IVU (Interaction, Visualization, and Usability) research group in Bari for the valuable contributions to our work on the EUD of mashups.

Our special thanks go, of course, to Boualem Benatallah for kindly agreeing to write a foreword to this book. We are proud to have his endorsement for this work.

Finally, we would like to thank Ralf Gerstner of Springer for welcoming so warmly the idea of this book and for all the patience he has had with us waiting for the book to be written. We would also like to express our gratitude to the reviewers (Cesare Pautasso and an anonymous reviewer) for the competent, critical, and constructive feedback. We seriously took their suggestions into account and believe the book has noticeably improved since its first version.

Povo, Italy
Milano, Italy
February 2014

Florian Daniel
Maristella Matera

Contents

1	Introduction	1
1.1	Concepts and Definitions	1
1.2	Mashup Types	4
1.3	Benefits and Benefitters	7
1.4	The Research Perspective	9
1.5	This Book	11
 Part I Fundamentals		
2	Data and Application Integration.....	15
2.1	Introduction.....	15
2.2	The Integration Problem: A Scenario	17
2.3	Data Integration	18
2.3.1	Materialized Versus Virtual Data Integration	18
2.3.2	The Mediator-Wrapper Architecture	19
2.3.3	Peer-to-Peer Data Management Systems	23
2.3.4	Ontology-Based Data Integration	24
2.3.5	Lightweight Data Integration	25
2.4	Application Integration	27
2.4.1	Middleware and Application Integration	27
2.4.2	Workflow Management Systems.....	29
2.4.3	Web Services.....	31
2.4.4	Service-Oriented Computing.....	32
2.5	Cloud Computing	36
2.5.1	Virtualization	37
2.5.2	Cloud Architectures	38
2.6	Summary and Bibliographic Notes	39

3	Web Technologies	41
3.1	Introduction	41
3.2	The Internet	44
3.2.1	Internet Topology	44
3.2.2	The TCP/IP Protocol Stack	45
3.2.3	The HTTP Protocol	46
3.3	The Hypertext Markup Language	47
3.3.1	HTML 5	50
3.4	Cascading Style Sheets	51
3.5	Client-Side Scripting	53
3.6	Client-Side Business Logic and AJAX	55
3.7	Server-Side Business Logic	57
3.7.1	Servlets	58
3.7.2	Server-Side Scripting	58
3.7.3	Multitiered Web Architectures	61
3.8	Model-View-Controller Pattern	64
3.9	Data Representation Formats	66
3.9.1	eXtensible Markup Language	66
3.9.2	JavaScript Object Notation	67
3.10	Summary and Bibliographic Notes	68
4	Model-Driven Software Development	71
4.1	Introduction	71
4.2	Model-Driven Design	74
4.2.1	Conceptual Modeling	74
4.2.2	Model-Driven Architecture	76
4.2.3	Architecture-Centric MDSD	78
4.3	Metamodeling	79
4.3.1	The Metalevels	79
4.3.2	Metamodels, MOF, and UML Profiles	81
4.3.3	Modeling Syntax	85
4.4	From the Model to the Application	87
4.4.1	Model-to-Model Transformations	87
4.4.2	Code Generation	88
4.4.3	Model Interpretation	91
4.5	Summary and Bibliographic Notes	91
 Part II Mashups		
5	Mashup Components	97
5.1	Introduction	97
5.2	Modeling Components	98
5.2.1	Basic Component Model	99
5.2.2	Component Characteristics	100

5.3	Logic Components	107
5.3.1	Web Services.....	107
5.3.2	RESTful Web Services	110
5.3.3	JavaScript APIs and Libraries.....	113
5.3.4	Device APIs.....	114
5.3.5	API Extraction	115
5.4	Data Components	116
5.4.1	Really Simple Syndication	116
5.4.2	Atom	118
5.4.3	XML, JSON, CSV, and Similar	119
5.4.4	Web Data Extraction	120
5.4.5	Microformats and Linked Data	121
5.5	User Interface Components	124
5.5.1	Code Snippets and JavaScript UI Libraries	124
5.5.2	Java Portlets.....	125
5.5.3	Widgets and Gadgets	127
5.5.4	Web Clipping and UI Component Extraction	130
5.6	Real-Time Streaming Components.....	131
5.6.1	Multimedia Resources.....	131
5.6.2	Telco APIs	132
5.7	Summary and Bibliographic Notes.....	134
6	Mashups	137
6.1	Introduction.....	137
6.2	Modeling Mashups	140
6.2.1	Basic Mashup Model	141
6.2.2	Mashup Characteristics.....	142
6.3	Data Mashups	148
6.3.1	Point-to-Point Data Mashups	151
6.3.2	Centrally Mediated Data Mashups.....	152
6.3.3	Data Mashups with External Data Processing Logic	154
6.4	Logic Mashups	155
6.4.1	Stateless Logic Mashups	158
6.4.2	Long-Living Logic Mashups.....	159
6.5	User Interface Mashups	160
6.5.1	HTML UI Mashups.....	163
6.5.2	Wrapped UI Mashups	165
6.5.3	Container-Based UI Mashups	167
6.6	Hybrid Mashups	174
6.7	Summary and Bibliographic Notes	180
7	Advanced Mashups.....	183
7.1	Introduction.....	183
7.2	Multiuser Mashups	185
7.2.1	Concurrent Mashup Components	185
7.2.2	Concurrent Mashups.....	186
7.2.3	Process Mashups.....	188

7.3	Mobile Mashups	191
7.4	Telco Mashups	194
7.5	Enterprise Mashups	196
7.6	Summary and Bibliographic Notes	200
8	Tool-Aided Mashup Development	201
8.1	Introduction	201
8.2	Mashup Modeling Concerns	204
8.3	Abstracting Components	206
8.4	Mashup Metamodels	209
8.4.1	A Simple Example	209
8.4.2	Yahoo! Pipes	212
8.4.3	mashArt	214
8.5	Mashup Languages	217
8.5.1	Enterprise Mashup Markup Language	217
8.5.2	Open Mashup Description Language	219
8.6	Developing Mashup Languages	221
8.6.1	Conceptual Development	222
8.6.2	Usage Example	223
8.7	Mashup Platforms	225
8.7.1	Mashup Development	226
8.7.2	Mashup Execution and Operation	230
8.7.3	Mashup Management	233
8.8	Summary and Bibliographic Notes	234
9	Mashups and End-User Development	237
9.1	Introduction	237
9.2	User-Driven Innovation	239
9.3	EUD Mashup Scenarios	240
9.4	Lightweight Development Process	243
9.4.1	Component Discovery and Selection	244
9.4.2	Mashup Composition	244
9.4.3	Usage and Maintenance	245
9.5	EUD Dimensions for Mashup Tools	245
9.5.1	Representation	246
9.5.2	Domain Specificity	251
9.5.3	Assistance Capability	257
9.6	Guidelines for EUD of Mashups	265
9.7	Summary and Bibliographic Notes	267
10	Quality in Mashup Development	269
10.1	Introduction	269
10.2	Component Quality	271
10.2.1	API Quality	272
10.2.2	Component Data Quality	277
10.2.3	Presentation Quality	278

10.3	Mashup Quality	279
10.3.1	Characterizing the Integration Sets	281
10.3.2	Roles of Mashup Components	282
10.3.3	Composition Quality	284
10.3.4	Information Quality	287
10.3.5	Presentation Quality	289
10.4	Summary and Bibliographic Notes	290
11	Outlook	293
11.1	Summary.....	293
11.2	Outlook	294
11.2.1	Growth	294
11.2.2	Threats	295
11.2.3	Directions	296
11.3	The Fate of Mashups	297
	References.....	299
	Index.....	313

Mashups

Concepts, Models and Architectures

Daniel, F.; Matera, M.

2014, XIX, 319 p. 119 illus., 2 illus. in color., Hardcover

ISBN: 978-3-642-55048-5