

Mark D. Hawker

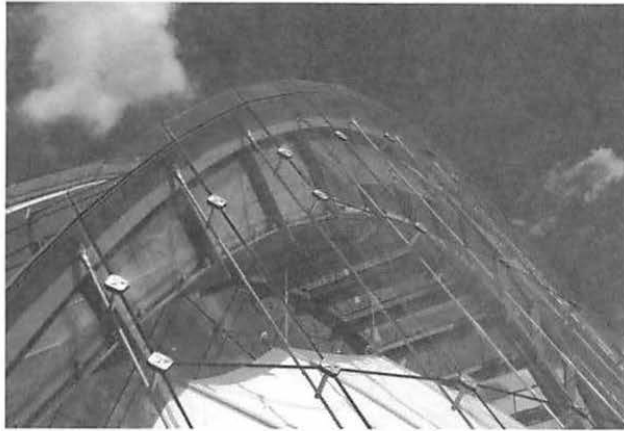


The Developer's Guide to Social Programming

Building Social Context Using Facebook, Google Friend Connect, and the Twitter API

Developer's Library





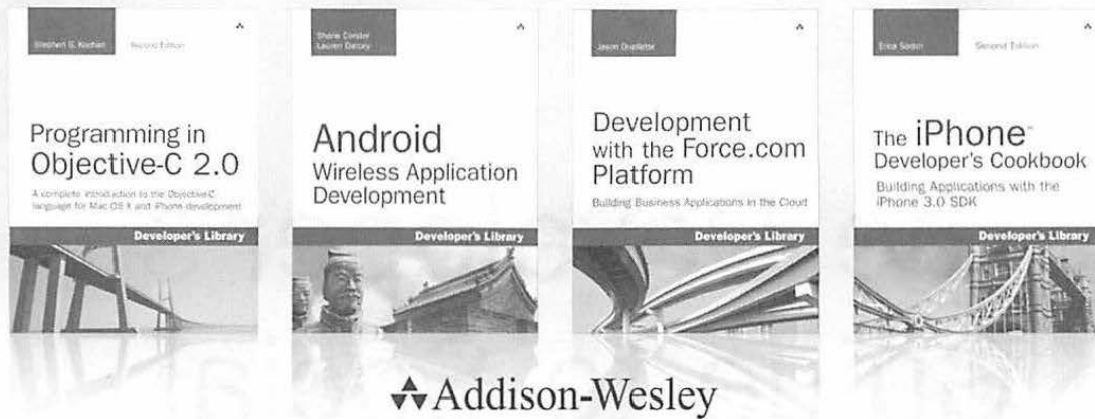
This book's cover shows **The Davies Alpine House** at Royal Botanic Gardens, Kew Gardens, near London, UK. Opened in 2006, this is a remarkably unusual greenhouse. Rather than generating warmth, it provides cool temperatures, breezes, and low humidity—ideal conditions for Kew Gardens' alpine plants. Moreover, it does so using passive, sustainable techniques, requiring neither air conditioning nor wind pumps.

Architected by the award-winning firm Wilkinson Eyre, The Davies Alpine House shades its plants through fan-shaped structures reminiscent of a peacock's tail. Beneath, fans blow air through a cool concrete "labyrinth" that is structurally analogous to the cooling nests built by termite ants. Once cooled, air flows into the building; as it rises and warms, it escapes through vents in the roof.

The result is a structure that is aesthetically compelling, superbly energy efficient—and a glorious home for plants ranging from tulips to thyme.

The Developer's Guide to Social Programming

Developer's Library Series



Visit **developers-library.com** for a complete list of available products

The **Developer's Library Series** from Addison-Wesley provides practicing programmers with unique, high-quality references and tutorials on the latest programming languages and technologies they use in their daily work. All books in the Developer's Library are written by expert technology practitioners who are exceptionally skilled at organizing and presenting information in a way that's useful for other programmers.

Developer's Library books cover a wide range of topics, from open-source programming languages and databases, Linux programming, Microsoft, and Java, to Web development, social networking platforms, Mac/iPhone programming, and Android programming.

PEARSON

Addison-Wesley

Cisco Press

EXAM/CRAM

IBM Press

QUE

PRENTICE HALL

SAMS

Safari Books Online

The Developer's Guide to Social Programming

Building Social Context Using
Facebook, Google Friend
Connect, and the Twitter API

Mark D. Hawker

◆Addison-Wesley

Upper Saddle River, NJ • Boston • Indianapolis • San Francisco
New York • Toronto • Montreal • London • Munich • Paris • Madrid
Cape Town • Sydney • Tokyo • Singapore • Mexico City

Many of the designations used by manufacturers and sellers to distinguish their products are claimed as trademarks. Where those designations appear in this book, and the publisher was aware of a trademark claim, the designations have been printed with initial capital letters or in all capitals.

The author and publisher have taken care in the preparation of this book, but make no expressed or implied warranty of any kind and assume no responsibility for errors or omissions. No liability is assumed for incidental or consequential damages in connection with or arising out of the use of the information or programs contained herein.

The publisher offers excellent discounts on this book when ordered in quantity for bulk purchases or special sales, which may include electronic versions and/or custom covers and content particular to your business, training goals, marketing focus, and branding interests. For more information, please contact:

U.S. Corporate and Government Sales
(800) 382-3419
corpsales@pearsontechgroup.com

For sales outside the United States please contact:

International Sales
international@pearson.com

Visit us on the Web: informit.com/aw

Library of Congress Cataloging-in-Publication Data:

Hawker, Mark D.

The developer's guide to social programming : building social context using Facebook, Google friend connect, and the Twitter API / Mark D. Hawker.

p. cm.

ISBN 978-0-321-68077-8 (pbk. : alk. paper) 1. Online social networks. 2. Entertainment computing. 3. Internet programming. 4. Google. 5. Facebook (Electronic resource) 6. Twitter. I. Title.

HM742.H39 2010
006.7'54—dc22

2010020866

Copyright © 2011 Pearson Education, Inc.

All rights reserved. Printed in the United States of America. This publication is protected by copyright, and permission must be obtained from the publisher prior to any prohibited reproduction, storage in a retrieval system, or transmission in any form or by any means, electronic, mechanical, photocopying, recording, or likewise. For information regarding permissions, write to:

Pearson Education, Inc
Rights and Contracts Department
501 Boylston Street, Suite 900
Boston, MA 02116
Fax (617) 671 3447

ISBN-13: 978-0-321-68077-8

ISBN-10: 0-321-68077-4

Text printed in the United States on recycled paper at RR Donnelley Crawfordsville in Crawfordsville, Indiana.

First printing, August 2010

Handling GET Parameters

The one downside to using CodeIgniter is its inability to handle GET parameters without seriously affecting the complexity of the framework. This is particularly important when using Twitter because it sends an `oauth_token` parameter back to your applications. A workaround will be used that just reformats the Twitter callback as a POST parameter to work with inside your applications.

CodeIgniter uses the Model-View-Controller (MVC) architectural pattern, which separates application logic from presentation and database extraction. In case you have not used MVC before, this section includes a brief overview with examples and prompts of how to set up the first stages of Sprog. Even for beginners, it will take only a matter of hours before you start to appreciate the benefits of MVC and wonder why you have never used it before! For experienced programmers, you can use the more advanced features of CodeIgniter, such as benchmarking, File Transfer Protocol (FTP), and caching, to increase performance of large-scale applications.

The Model-View-Controller Architectural Design

The MVC pattern is one of the most commonly used architectures used in web applications today. The clearest benefit of MVC is its separation of presentation and application logic, which is akin to the separation of HTML and CSS. For example, although your data-retrieval functionality might be the same you may want to present content to a number of devices from laptops to cell phones. What MVC enables is the ability to make maximum reuse of existing code that is easier to test and build upon. Applications are divided into three components:

- The **model** (or **models**) is the conduit between a controller and your application's data store.
- The **controller** (or **controllers**) manages user requests such as GET or POST operations and in turn requests data from a model and then sends it to a view for presentation. In practice, a controller will contain all the logic for handling form inputs or requests to view specific pages. You may want to use a master controller, which connects to multiple other controllers to handle requests from multiple devices.
- The **view** (or **views**) handles data passed via the controller, and then presents the data onscreen or to the requesting device. A view should be parsing only prefetched data and should not make requests to your data store (because that is the role of a model). Often, you can use a template engine such as Smarty (<http://www.smarty.net/>) to promote greater reuse of views across multiple devices.

The remainder of this chapter and Chapter 13 focus on MVC using a web application framework called CodeIgniter. The benefits of a framework are that most of the complexity of using common features such as working with databases, sessions, and form handling are already coded for you. All that you then need to do is configure the framework to your specific needs without worrying about creating your own potentially error-prone

functions. Many of the frameworks available today have been tested extensively and are updated continually when new vulnerabilities are discovered.

Installing, Configuring, and Exploring CodeIgniter

You can download CodeIgniter from <http://codeigniter.com/downloads/>. It requires PHP 4.3.2+ and supports MySQL (4.1+), MySQLi, MS SQL, Postgres, Oracle, SQLite, and ODBC database platforms. Because you are using the twitter-async library, it is assumed that you are working with PHP 5.2+. The latest version of CodeIgniter at the time of this writing, and that is included with source code examples, is 1.7.2. After you have downloaded the framework, unzipped it to a location of your choice, and renamed the base folder to `codeigniter`, you will notice a directory structure similar to the following:

- **system**

The `system` directory contains all the files required to make CodeIgniter work. It includes directories for saving an application cache, for storing logs, and for storing core helpers and libraries. Each helper file is just a collection of functions in a particular category that help you with specific tasks. For example, URL Helpers assist with creating links, Form Helpers help you create form elements, and Text Helpers perform various text-formatting routines. Libraries, on the other hand, are suites of functions that perform tasks such as maintaining a database, uploading files, and sending e-mails. You will also see a `plug-in` directory, which is used for installing code created by the CodeIgniter community to use in your applications.

- **system\application**

One of the most important directories in `system` is `application`. This directory stores all the models, views, and controllers used in your applications. It also stores configuration and error page details. Although not explored in this book, it is possible to have multiple applications running from a single CodeIgniter installation.

- **user_guide**

The `user_guide` directory should be exactly the same as the one that can be found at http://codeigniter.com/user_guide/, and so you can delete this if you want to save space on your web server.

- **index.php**

This is the main file that runs all the CodeIgniter functions and is where you can change the names of the `system` and `application` directories for enhanced security.

- **license.txt**

The license associated that must be adhered to when using the framework.

The `system` directory contains all the CodeIgniter functionality, such as libraries and helpers, which are explored later in this section. You will also notice an `application` directory, which is where all the models, views, and controllers are stored. For security reasons, this directory should be moved up one level alongside the `system` and `user_guide` directories. Because you cannot place static files such as images, JavaScript, or CSS within