Reactive User Interfaces in the Web

M.Sc. Thesis
University of Turku
Department of Information Technology
Computer Science
2015
Johannes Dahlström

Supervisors:
First Supervisor
Second Supervisor

TURUN YLIOPISTO
Informaatioteknologian laitos

JOHANNES DAHLSTRÖM: Reactive User Interfaces in the Web

Pro gradu, 9 s., 0 liites.
Tietojenkäsittelytiede
Kuukausi 2015

Tarkempia ohjeita tiivistelmäsivun laadintaan löytyy opiskelijan yleisoppaasta, josta alla lyhyt katkelma.

Asiasanat: tähän, lista, avainsanoista

UNIVERSITY OF TURKU
Department of Information Technology

JOHANNES DAHLSTRÖM: Reactive User Interfaces in the Web

M.Sc. Thesis, 9 p., 0 app. p.
Computer Science
Month 2015

If your thesis is in english this might also be required???

Keywords: list, of, keywords

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Interactive software systems have become progressively more complex, driven by the demand for richer user interaction along with the growing multimedia capabilities of modern hardware. Applications are increasingly being written for the World Wide Web platform, and the inherently distributed nature of web applications brings forth its own challenges. Not only are they required to serve an ever-increasing number of users, but also, more and more commonly, to facilitate interaction *between* users.

Complex interaction requires complex user interfaces. In object-oriented programming, the usual means of implementing a user interface is to make use of the Observer design pattern. In this pattern, interested objects can register themselves as *observers*, or listeners, of events, such as mouse clicks or key presses, sent by user interface elements. When an observer is notified of an event, it reacts in an appropriate manner by initiating a computation or otherwise changing the application state. Similarly, the user interface may react to events triggered by some underlying computation, signaling the user that something requires his or her attention.

The traditional observer pattern has several disadvantages. Observers are difficult to compose and reuse, leading to instances of partial or complete code duplication. Events often lack important context, making it necessary to manually keep track of the program state and share the bookkeeping between different observers. The resulting code typically forms a complex and fragile state machine, making it difficult to reason about its behavior and correctness.

In the recent years, several mainstream object-oriented programming languages used in the industry have adopted concepts traditionally belonging to the relatively academic realm of functional programming. These include functions as first-class values; anonymous functions (lambdas); and combinators such as map, filter, and reduce. One impetus for this paradigm shift has been the constantly increasing importance of concurrency and parallelism in software. Correctly managing and reasoning about mutable state shared between concurrent threads of execution is notoriously difficult, and the general disposition towards immutable state in functional programming has proven to be a useful basis for building better concurrency abstractions.

Reactive programming is a programming style centered on the concept of propagating change. In a reactive system, a variable can be bound to other variables so that its value changes automatically as a response to value changes in other components of the bound system. A common example of such reactivity is a spreadsheet application, where the value of a cell can be a formula referring to several other cells. The displayed value of the cell is refreshed whenever the value of a referenced cell changes. Another name for this approach is dataflow programming.

One way of improving upon the observer pattern is to elevate the abstract concept of an event stream to a first-class data type. Event streams, or *observables*, can then be merged, transformed, and otherwise manipulated in a declarative manner using the familiar set of functional tools.

Vaadin is a web application framework written in Java, aiming to provide a rich set of user interface components facilitating rapid application development. It also contains a data binding layer for propagating input and output between the user and a data model. Both the user interface and data binding are designed in terms of the traditional observer pattern.

This thesis seeks to answer the question of whether reactive programming techniques are useful in writing user interfaces in Vaadin. Furthermore, it seeks to analyze whether some of these techniques should be adopted by Vaadin itself instead of simply being built on top of the framework.

Chapter 2

Programming Web Applications

Chapter 3

Reactive Programming

Chapter 4

Case Study: Reactive Vaadin

Chapter 5

Validation

Chapter 6

Conclusions

References