Computer Science at Kent

Implementation and Application of Functional Languages
19th International Symposium, IFL 2007

Olaf Chitil (Ed.)

Freiburg, Germany, 27th-29th September 2007

Technical Report No. 12-07
September 2007

Published by the Computing Laboratory,
University of Kent, Canterbury, Kent, CT2 7NF, UK

Preface

The 19th International Symposium on Implementation and Application of Functional Languages (IFL 2007) is held at Freiburg, Germany, on the 27th to the 29th September 2007. Local organiser is the Programming Languages Group of the Department of Computer Science of the University of Freiburg.

IFL brings together researchers active in the area of functional programming, with an emphasis on the implementation and application of the same. IFL provides an annual open forum for researchers who wish to present and discuss new ideas and concepts, work in progress, preliminary results, etc. IFL has been held throughout Europe in the Netherlands, United Kingdom, Germany, Sweden, Spain, Ireland and Hungary. This year for the first time IFL is co-located with the International Conference on Functional Programming (ICFP). A record number of 44 papers have been submitted for these draft proceedings. By the time of printing 73 researchers had registered for attendance at the symposium.

Following tradition, two proceedings are to be published: the draft proceedings used at the symposium (this document), released as a technical report of the Computing Laboratory of the University of Kent, and the post-symposium proceedings based on revised papers. The draft proceedings are un-refereed and provide a useful reference to the delegates at the symposium. All participants who give talks at the symposium are invited to submit revised papers for review after the symposium, to normal conference standards. The post-symposium proceedings of selected revised papers will be published by Springer-Verlag in its Lecture Notes in Computer Science (LNCS) series.

Olaf Chitil
Programme Chair
University of Kent
September 2007

Local Organisers

Markus Degen
Peter Thiemann
Stefan Wehr

Supported by Deutsche Forschungsgemeinschaft (DFG)

Table of Contents

Index

Transparant Ajax and Client-Site Evaluation of iTasks

– Draft Version –

Rinus Plasmeijer, Jan Martin Jansen, Pieter Koopman, and Peter Achten

Radboud University Nijmegen, Netherlands
rinus@cs.ru.nl, jm.jansen.04@nlda.nl, pieter@cs.ru.nl, p.achten@cs.ru.nl

Abstract

The iTask system is a combinator library written in Clean which allows the specification of multi-user workflow systems for the web. The iTask system generates forms that have to be filled in and submitted by the user. Each user has a set of tasks that can be processed in any order. The submission of a form might terminate existing tasks or create new tasks for the user herself as well as for other users. As a consequence a single event can cause a very complex state change on the server and can effect the work of many other users.

The advantages of using a browser as interface to a workflow system created with the iTask library is that no software has to be installed at the client site and that the look and feel of the GUI is familiar to every user. A drawback of this architecture is that the response might become rather slow when there are many users and many tasks. For each and every event on the client a message is sent to the server over the world wide web. The server processes the event and generates a new web-page for the user containing all her new tasks. For the calculation of the set of new tasks, the state of all other tasks has to be examined. Due to the delay of the world wide web and the creation, transportation and rendering of the complete new page by the browser, the response of a workflow system can become relatively slow.

In this paper we present two solutions for dealing with this performance problem.

First we introduce a combinator 'UseAjax' that cause the workflow system to use Ajax technology for handling a (sub)task. This has as consequence that only a part of the web page is updated instead of the creation, sending and rendering of an entire new page. The advantage of this extension is not only a smoother reaction in the browser on changes being made. Also the efficiency for large workflow systems is commonly improved in this way because most of the time only a smaller, for this (sub)task relevant, part of the current state needs to be recalculated.

For the definition of the workflow system a single annotation 'UseAjax' is sufficient. The implementation of this feature in the iTask library requires a Java script that runs on the client as well as a call-back function that handles the event. For the implementation this requires the possibility to store Clean functions temporarily in a web page as well as the possibility to store them in a persistent store at the server site such as in a file or in a database.

The second extension is another annotation, 'OnTheClient', which allows client site evaluation of tasks.

Since no call at all has to be made to the server when such a task is evaluated, there is no web communication overhead anymore as is the case when Ajax technology is being used. For the implementation of this feature one needs to be able to execute the tasks specified in Clean in the browser at the client site. We realize this with an interpreter for Clean code running in the browser. Therefore Clean is compiled to Sapl and this code is loaded into the browser together with the compact and efficient Sapl interpreter. Of course, code interpreted by the Sapl interpreter running in the browser is not as efficient as the execution of compiled Clean code at the server. So, there is also an efficiency penalty when 'OnTheClient' is chosen instead of 'UseAjax'.

By choosing one of the annotations, the programmer can define which evaluation method preferably should be used for a certain (set of) tasks.

Whenever evaluation 'OnTheClient' is not possible for some reason (e.g. when a database needs to be inspected on the server) the system can automatically decide to 'UseAjax' instead.