A Conference Management System Based on the iData Toolkit

Rinus Plasmeijer and Peter Achten

Software Technology, Nijmegen Institute for Computing and Information Sciences,
Radboud University Nijmegen
{rinus,P.Achten}@cs.ru.nl

Abstract. The iData Toolkit is a purely functional toolkit for the Clean programming language to create highly dynamic, interactive, thin client web applications on a high level of abstraction. Its main building block is the iData element. With this element the programming effort of the application programmer is reduced significantly because it takes care of state handling, rendering, user interaction, and storage management automatically. In this paper we show that it can be used for even more tasks: handle *destructively shared model data*, perform *version management*, and *state consistency management*. This can be done entirely on top of the iData Toolkit. The toolkit comes with a new programming paradigm. We illustrate the extended power of the toolkit and programming paradigm by a case study of a conference management system.

1 Introduction

The purely functional language Clean has a library to create highly dynamic, interactive, thin client web applications on a high level of abstraction. This library is the iData Toolkit [11,13,12].It is based on the language support for generic programming [2,3]. The toolkit's main building block is the iData element, which is a versatile unit that automates a great deal of things for the programmer:

- it manages a state of arbitrary type;
- it renders an HTML form representation of its state;
- it handles user actions made with these forms in a type safe way;
- it stores its state either in the page or at the server side on disk.

Web applications are created by interconnecting an arbitrary collection of iData elements via their states and rendered forms. In the past years we have obtained experience in programming applications with iData elements, and their desktop GUI predecessors, the GEC elements of the GEC Toolkit [1]. This has resulted in a new programming paradigm. In the iData Toolkit programming paradigm the application programmer *models* the application as an *information system*, by identifying the entities and entity-relations and specify them as pure functional data structures and pure functions. The generic power of the toolkit is used subsequently to handle as much as possible automatically. Human intervention is still required, but the power of generic programming is that it allows application programmers to specialize the generic scheme where needed.

Z. Horváth, V. Zsók, and A. Butterfield (Eds.): IFL 2006, LNCS 4449, pp. 108–125, 2007.

When constructing programs with the programming paradigm, it turns out that the 'classic' version of the toolkit has a number of limitations:

- Model types are pure functional data structures. Although functional languages can define and handle *shared* data structures, they cannot handle *destructively shared* data because this destroys referential transparency. However, in information systems destructive sharing is a natural phenomenon, because data should not be stored redundantly. Hence, an iData Toolkit application programmer can not model destructive sharing directly, but instead has to program this on top of the functional data structures and for each and every edit operation. This is cumbersome, error-prone, and an example of boilerplate code that should be automated once and for all.
- It is important in multi-user web applications with several persistent shared states to manage *versions* of these states correctly. Again, the programmer might be able to program this, but it should be dealt with once and for all.
- The final limitation concerns the *consistency of states*. The iData Toolkit is *edit driven*, i.e.: it reacts to (type safe) edit operations of the application user who can alter a part of the state of one of the iData elements. In general, it may well be the case that during a sequence of edit operations, the set of states is *inconsistent*. In that case, the application should not commit this configuration of states to disk, but rather work on a local version.

In this paper we show that the above concerns can be handled automatically by the iData elements, on top of the 'classic' iData Toolkit. We believe that this provides further evidence to the fact that iData elements form a powerful abstraction mechanism to create highly interactive and dynamic web applications with. We illustrate the use of the new techniques by studying the case of a *conference management system*. Conference management systems are software systems that support conference managers, programme committee members, and authors with a number of tasks, such as the electronic paper submission process, paper distribution and reviewing process, deadline management, and the paper discussion process. They serve as a good example of the domain of web applications that suffer from the limitations that have been presented above. We show that the resulting system widens the application domain of the toolkit while still adhering to its programming paradigm.

This paper is structured as follows: we first briefly present the iData Toolkit in Sect. 2. Next, in Sect. 3, we discuss the case study of a conference management system. Implementation details are presented in Sect. 4. Finally, related work is discussed in Sect. 5, and we conclude in Sect. 6.

2 The iData Toolkit

In this section we present the 'classic' iData Toolkit, i.e. the toolkit without the extensions that are discussed in the next sections. First, we give an informal explanation of iData elements, which are the building blocks of the iData Toolkit (Sect. 2.1). Second, we present the programming paradigm (Sect. 2.2).