On the Design and Implementation of Interactive XML Applications

Jeff Brown, University of North Carolina Wilmington, USA
Rebecca Brown, University of North Carolina Chapel Hill, USA
Chris Velado, University of North Carolina Wilmington, USA
Ron Vetter, University of North Carolina Wilmington, USA

ABSTRACT

This paper describes issues and challenges in the design and implementation of interactive client-server applications where program logic is expressed in terms of an extensible markup language (XML) document. Although the technique was originally developed for creating interactive short message service (SMS) applications, it has expanded and is used for developing interactive web applications. XML-Interactive (or XML-I) defines the program states and corresponding actions. Because many interactive applications require sustained communication between the client and the underlying information service, XML-I has support for session management. This allows state information to be managed in a dynamic way. The paper describes several applications that are implemented using XML-I and discusses design issues. The software framework has been implemented in a Java environment.

Keywords: Extensible Markup Language, Short Message Service, SMS, Software Architecture, XML

INTRODUCTION

Client-server applications are often written as customized software that is dedicated to providing the information service desired. Each new service is a new application (or procedure) that implements the service. Like the basic web protocol (http), the client-server environment is often stateless – meaning that the environment won’t do anything to preserve the session state. Each request and response is a new one and not related to any other. Therefore when writing interactive client-server applications, session management is a major part of the effort since one must implement stateful sessions in a stateless environment.

Multi-tier web development architectures have become commonplace in modern web applications. A multi-tier, or more specifically a 3-tier, architecture refers to a client-server architecture in which the presentation, application processing, and the data management layers are

DOI: 10.4018/ijirr.2011010102
logically separate processes (Wikipedia, n.d.). The idea is to modularize the software application into functional units with well-defined interfaces so that changes and upgrades to any software layer (tier) can be made independently. Web application frameworks, such as Java Enterprise Edition (J2EE) and Ruby on Rails, are often used to support the development of dynamic web sites, web applications, and web services. These frameworks help alleviate the overhead associated with common activities performed in web development, such as providing libraries for data access, security, and session management (Wikipedia, n.d.).

In a recent article, a flexible framework for developing interactive short message service (SMS) applications was presented (Brown et al., 2010). This framework was called XML Interactive or XML-I. In this paper, we show how XML-I can be used not just for creating interactive SMS applications but also for creating any interactive client-server web application that requires state information and state control. Although this work is a relatively new, it builds upon the established work of more general web application frameworks. With XML-I, program logic is expressed in terms of an XML document which contains the program states and corresponding actions. Because many interactive applications require sustained communication between the client and the underlying information service, XML-I also has support for session management. This allows state information to be managed in a dynamic way. This paper describes several applications that have been implemented using XML-I and discusses design issues. The software framework has been implemented in a Java environment.

WHAT IS XML-I

XML Interactive is a specification, in XML format, of a software application. Thus, XML-I can be used for both the conceptual and the actual design of interactive SMS and web applications. The software framework of XML-I contains properties that can be easily implemented by simple programs in a web server environment, using Java’s application programming interface (API). By establishing XML-I in a multi-tier architecture, web services can be extended to create complex client-server conversations based on the input received from a mobile originated (MO) SMS message or more traditional web (HTTP) request.

Before the introduction of XML-I, many interactive SMS services were implemented by responding to a single and specific keyword contained in the MO message. The SMS applications would generate mobile terminated (MT) messages based upon the specific keyword received. With XML-I, users are able to carry out multifaceted conversations with the underlying information service without the need to specify a keyword for each client-server interaction. For example, a user might navigate through several menu options (responding via SMS messages) in order to request more specific information from a large data set.

Just as standard XML is based on tree structures, XML-I code contains specific nodes and attributes that control the navigation within the XML along with key commands to enable client interaction. When a user sends a message, the content within that message determines the new “state” of the application. Moving to a new state will determine what actions will take place. Basic functionality such as switching state or terminating conversations is also part of this system. Changing an existing application is easy – one simply updates the states and actions in the XML document.

SMS APPLICATION – DRUG TRIAL PRESCREENING

Mobile Education LLC has established the hardware and software infrastructure needed for implementing interactive SMS applications (Brown & Vetter, 2010). One of these applications involves prescreening applicants who are interested in participating in experimental drug trials. The basic idea behind this application can be described as follows.