10-2007

Swashup: Situational Web Applications Mashups

E. Michael Maximilien
Ajith Harshana Ranabahu
Wright State University - Main Campus
Stefan Tai

Follow this and additional works at: http://corescholar.libraries.wright.edu/knoesis

Part of the Bioinformatics Commons, Communication Technology and New Media Commons, Databases and Information Systems Commons, OS and Networks Commons, and the Science and Technology Studies Commons

Repository Citation

This Conference Proceeding is brought to you for free and open access by the The Ohio Center of Excellence in Knowledge-Enabled Computing (Kno.e.sis) at CORE Scholar. It has been accepted for inclusion in Kno.e.sis Publications by an authorized administrator of CORE Scholar. For more information, please contact corescholar@www.libraries.wright.edu.
Swashup: Situational Web Applications and Mashups
E. M. Maximilien\(^1\), A. Ranabahu\(^2\), and S. Tai\(^3\)

\(^1\)IBM Almaden Research Center, \(^2\)Wright State University, and \(^3\)IBM T.J. Watson Research Center

Abstract

Distributed programming has shifted from private networks to the Internet using heterogeneous Web APIs. This enables the creation of composed services exposing user interfaces, i.e., mashups. However, this programmable Web lacks unified models that can facilitate mashup creation and deployments. This poster demonstrates a platform to facilitate Web 2.0 mashups using a mashup domain-specific language (DSL) and a collection of Web 2.0 tools and APIs to facilitate writing, sharing, and deploying mashups created in the DSL. The platform leverages and is implemented in Ruby on Rails.

Motivation and architecture

- Key problems in building Web 2.0 mashups
  - Heterogeneous data and service types (e.g., REST, RSS, Atom)
  - Different invocation sequences requirements for each service
  - Need to cache data for performance
  - Need to invoke services asynchronous
  - Need for rich user interfaces (e.g., AJAX-style interfaces)
  - Ruby on Rails (RoR) framework can be used to address issues

- Domain-Specific Languages allows
  - Define concepts at domain-level
  - High-level declarative programming
  - Reduce code to what is necessary

- Metaprogramming to translate DSL into underlying RoR code
- DSL BNF is generally useful (i.e., implemented in other languages)

Domain-specific language (DSL)

- Takes advantage of Ruby’s support for creating DSLs
- Allows developers to program at higher-level of abstraction
- DSL includes primitive to represent aspects of any mashups
- Complete Backus-Naur form for the mashup DSL available
- DSL constructs are indexed for search and reuse
- Each recipe do
  - data for each API data element
  - api for each service (only methods used)
  - service to bind to endpoint
  - data mediation for data transformations
  - mashups do
    - steps or interactions between services
    - wirings or interactions with end-users
    - views for each wiring
    - Iterate
    - Each DSL construct supports desc and tag or tags
    - Expose any data as REST, RSS, or Atom APIs
    - Deploy and share
- Examples
  - YouTube’s APP service and Flickr’s REST API mashup
  - Google News Atom feed with Stanford’s events RSS feed
- Many more ...

Platform tools

- Swashup mashup creator
- Available on IBM’s alphaWorks services December 2007

Conclusion and future

The DSL is essentially the glue code that enables composition of Web services while also giving some structure to the tasks of a mashup designer. Implementation using RoR gives us a rich substrate to enable rapid and sophisticated mashups. Additionally Ruby’s excellent support for DSL results in a syntax that is natural and easy to read. Future considerations include:
- Support JSON for data construct
- SOAP/WSIDL services
- Manager application to manage deployments (local or remote)
- Simple wizard for service workflows
- Simple wizard for data mediation generation
- Support for advanced API metadata via microformats
- Available on IBM’s alphaWorks services December 2007

References