A Domain Specific Language for Enterprise Grade Cloud-Mobile Hybrid Applications

The MobiCloud II Experience

Ajith Ranabahu*, Michael Maximilien**, Amit Sheth*, Krishnaprasad Thirunarayan

*Kno.e.sis Center, Wright State University, Dayton OH, USA
**IBM Research, San Jose CA, USA

The 11th Workshop on Domain-Specific Modeling @ SPLASH 2011
23rd and 24th October 2011
Outline

• What is MobiCloud
  • A quick introduction
• The MobiCloud DSL
  • MVC pattern based domain modeling
• Extending the MobiCloud DSL with Enterprise Features
• Demonstration
• Lessons learnt
• Future Directions and Work In Progress
What is MobiCloud?

A Cloud-Mobile Hybrid application generator

- Front-end runs on a mobile device
  - Smart phone / tablet etc
- Back-end deployed on a cloud
  - Amazon EC2 / Google App Engine
- Both components needed for the full experience
Why MobiCloud?

• Cloud-Mobile hybrids are hard to develop
  • Many complications
• Current practice is to treat the components as separate projects
  • Increases effort, decreases portability & drives up the cost
• Portability (both front-end and back-end) is important
  • Hard to cater for the large number of platforms
A better approach is needed to develop cloud-mobile hybrid applications, while maintaining portability.
Modeling a Cloud-Mobile Hybrid (CMH)

- A CMH is effectively *functionally Monolithic*
  - Except there is a service layer in between!
- A CMH can be modeled as a single conceptual unit
The MVC Design Pattern

- Model
- View
- Controller

23rd October 2011
CMH Applications can be nicely decomposed into the MVC Design
A Simple Example

recipe(:todolist) do
  metadata({:id => 'todoapp'}) # model
  model(:task, {:time => :date, :location => :string, :description => :string, :name => :string}) # controller
  controller(:todohandler) do
    action(:create, :task)
    action(:retrieve, :task)
  end
  # view
  view(:add_task, {:models => [:task], :controller => :todohandler, :action => :create})
  view(:show_tasks, {:models => [:task], :controller => :todohandler, :action => :retrieve})
end

Metadata – details that need to be attached to the whole application

Models

Controllers

Views
Graphical Representation
Generated Application Components

- Model
- View
- Controller
- RESTful Service Client
- RESTful Service Implementation
- Server side handler
- Persistent Storage
- Task Data Structures
- Task Data Structure
- UI
- Mobile Device
- Cloud
Designing Extensions for MobiCloud

• The base language is very limited
  • How can we add extra capabilities, keeping the MVC structure intact?

• Introduce an Extensions mechanism
  • Predefined models, views and/or controllers with specific capabilities
  • Insert platform specific code at predefined extension points
A Simple Extension – Fetching from a URL

recipe :http_fetch do
  # Generic http extension
  extensions ['http']
  # metadata
  metadata({:id => "ajithssimpleapp"})
  # models
  model :time_value,{:ts => :int}
  # controllers
  controller :time_manager do
    # fetch & display time from yahoo
    action :fetch_time,:time_value,{:type=>'http',
      :url => 'http://developer.yahooapis.com/....../getTime',
      :params => {:appid => 'o6fGNQ3V34GxD......OaFr'},
      :return_mapping => {:/ts=> '/Result/Timestamp'},
      :action_forward => :retrieve}
  ...

23rd October 2011
Enterprise Integration with Extensions

• Integrate secured data sources like Salesforce with MobiCloud
• Not easy!
  • Data security
    • Salesforce enforces OAuth
  • Dependent data structures
    • “User” comes under “Organization”
  • Special configurations and required call back endpoints
    • https call back endpoint required for OAuth
Demonstration
Lessons Learnt

• Developers are hesitant to use a top-down approach unless there is an extreme improvement in productivity and/or convenience
• Graphical abstractions are important to support adoption
Big Picture and Future Directions

• Using DSLs as the primary means to overcome issues of Cloud application portability
• Fits in with a middleware layer for complete independence in
  • Development
  • Deployment
  • Management of cloud applications
Big Picture and Future Directions (Cont)

• More extensions
  • Google and other third party API integrations

• Extensions as graphical abstractions
  • The graphical mode only supports the base language

• Mixing in other languages for special functions
  • UI enhancements
    • XAML / CSS ?
Visit us on the Web at http://mobicloud.knoesis.org

More documentation, videos and details at http://wiki.knoesis.org/index.php/MobiCloud_Web_UI
Questions
Thank you
Extra : Extension Architecture

Diagram:
- DSL Script
- Parser
- Global Extension Processor
- Targeted extension processor
- Targeted extension processor
- Target specific Template Processor
- Target specific Template Processor
- Augmented Semantic Model