Developing Mobile Ajax/Web 2.0 Applications Using the java.net Open-Source Project Orbit

Max Carlson
Co-founder and Lead Runtime Architect
http://openlaszlo.org

Hinkmond Wong
Sr. Staff Engineer
Sun Microsystems, Inc.

http://orbit.dev.java.net

TS-5699

Goal of This Talk

What You Will Gain

Learn to develop mobile Ajax/Web 2.0 style applications with code examples and demos using OpenLaszlo on Java™ Platform, Micro Edition (Java ME platform) technology devices with the Project Orbit LzPlayer.

Updated Slides

https://phoneme.dev.java.net/files/documents/

Agenda

Introduction to Java ME Technology as AJAX/Web 2.0 Platform

Introduction to AJAX/Web 2.0 Using OpenLaszlo

Project Orbit Design Overview

Running LZX Application Using Project Orbit

Summary

Agenda

Introduction to Java ME Technology as AJAX/Web 2.0 Platform

Introduction to AJAX/Web 2.0 Using OpenLaszlo

Project Orbit Design Overview

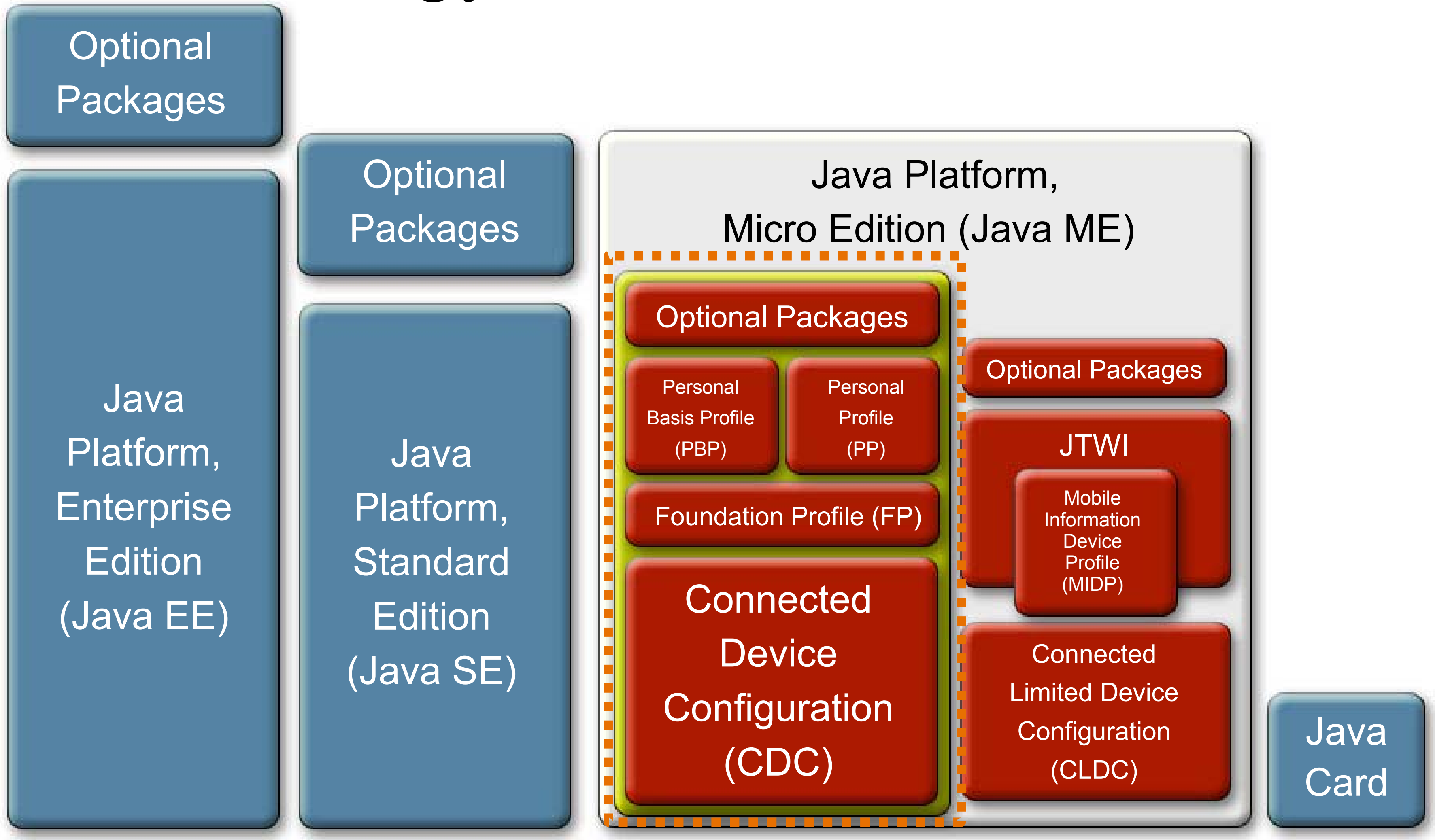
Running LZX Application Using Project Orbit

Summary

Introduction to Java ME Technology as AJAX/Web 2.0 platform

- Limitations of AJAX and Web 2.0 on mobile Web browsers
  - No widespread full Flash capabilities or JavaScript™ technology
  - Limited screen size, memory, and CPU power
- Java ME technology addresses these limitations
  - Stripped down VM and libraries tuned for reduced capacity devices
  - Primarily subset APIs
  - New APIs specifically for mobile applications
    - Can run Rhino engine to give JavaScript technology support
    - Enables Project Orbit to play LZX content

Introduction to Java ME Technology


Optional Packages
Java Platform, Enterprise Edition (Java EE)
Optional Packages
Java Platform, Standard Edition (Java SE)
Java Platform, Micro Edition (Java ME)
Optional Packages
Personal Basis Profile (PBP)
Personal Profile (PP)
Foundation Profile (FP)
Connected Device Configuration (CDC)
Optional Packages
JTWI
Mobile Information Device Profile (MIDP)
Connected Limited Device Configuration (CLDC)
Java Card

Introduction to Java ME Technology

Open Source Java Technology

- Open sourcing for Java ME platform and Java SE platform
- Not opening the Java platform brand; hence,
  - **Java ME technology → phoneME™ technology**
- License: GPLv2
- Mobile and Embedded Community
  - phoneME technology: VMs and class libraries
  - CqME™ software: test tools for compatibility and quality
  - ME Application Developers

Agenda

Introduction to Java ME Technology as AJAX/Web 2.0 Platform

Introduction to AJAX/Web 2.0 Using OpenLaszlo

Project Orbit Design Overview

Running LZX Application Using Project Orbit

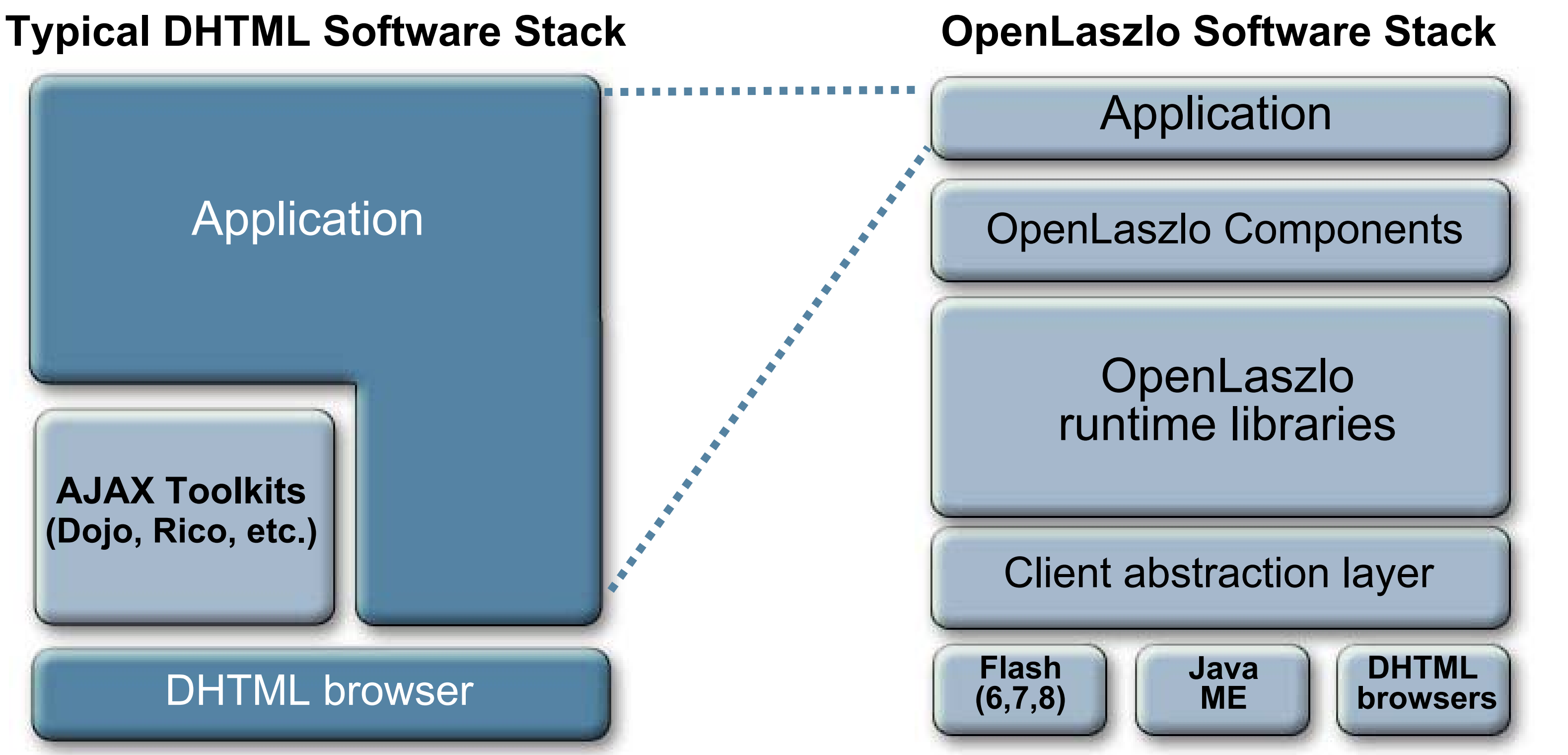
Summary

Introduction to OpenLaszlo

- AJAX and Web 2.0
  - Rich, Web-based applications
  - JavaScript technology and XML
- OpenLaszlo
  - Open Source Rich Internet Application (RIA) Framework
  - Rich and robust application delivery
  - Multi-runtime: Rendered in Flash, DHTML, or Java ME Technology

Introduction to OpenLaszlo

AJAX toolkits vs. OpenLaszlo for advanced apps



- Lack of high-level framework and rich component library means more code, complexity and less functionality
- Browser compatibility is a major concern

- Richness of components and framework reduces code for sophisticated apps
- Abstraction layer insulates developer from browser/runtime idiosyncrasies

Introduction to OpenLaszlo

Developing with OpenLaszlo

- XML-based
  - Use your favorite editor
  - Source-control
  - Library mechanism (for modularization)
- Familiar methodology for software engineers
- Standard OOP features
  - Attributes and methods
  - Class definitions with inheritance
  - Familiar design patterns apply
- Emphasis on declarative constructs

Introduction to OpenLaszlo

LZX: Laszlo’s XML Application Description Language

- Runtime independent tags and APIs
  - No ActionScript, DOM, etc.
- Interface with server via XML over HTTP, SOAP, XML-RPC, and Java API for XML-based RPC (JAX-RPC)
- Runtime constraint system
- Hierarchical data binding with XPath
- Media, streaming support
- Extensible UI component framework
- “XML all the way down”

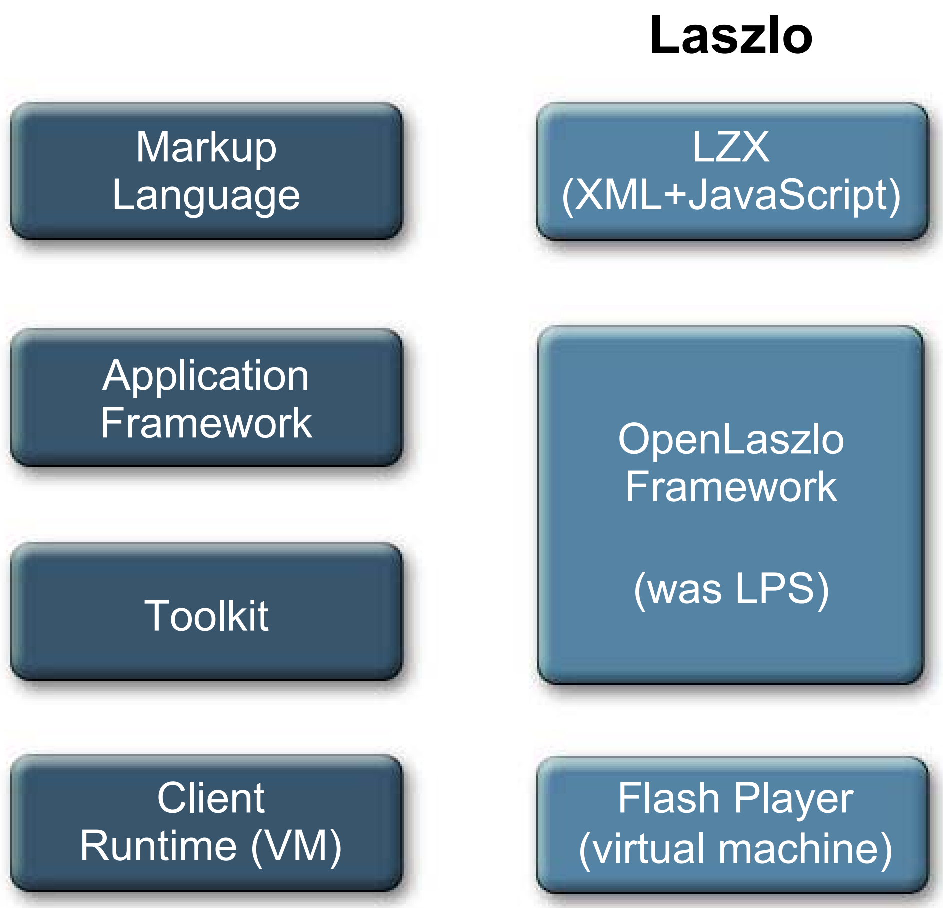
Introduction to OpenLaszlo

Rich client development hierarchy

Markup Language	Generic, abstract description of user experience; independent of specific application type
Application Framework	Integrated methods, patterns and approaches for application layout and functionality
Toolkit	Reusable functions for common uses of a given client runtime
Client Runtime (VM)	Basic functionality for application rendering, interaction and networking

Introduction to OpenLaszlo

Block diagram of components



Introduction to OpenLaszlo

- “Legals” is the code name for our multi-runtime architecture
- Currently we target three distinct runtimes
  - Flash 7 and 8
  - Java ME
  - DHTML (“A” list browsers: IE6, FF1.5, Safari 1.3, Project Orbit [Java ME technology])
- Current status
  - Multiple runtime architecture in place
  - Runtime-specific kernels built for DHTML, SWF 7/8 and Java ME
  - Released 4.0
- Contributors welcome!

DEMO

Laszlo language overview and demos

Consistency across browser brands and versions
Integrated Media Support
Animation
Audio
Video

Flash or DHTML: You choose, we’re (uniquely) there

Flash Advantages:

- Media-centric applications
- De-facto standard plug-in: over 98% penetration of Flash 6+
- Consistency across browser brands and versions
- Integrated Media Support
  - Animation
  - Audio
  - Video

AJAX Advantages:

- Open standard
  - No plug-in required
- Better browser integration
- Seamless integration with HTML
  - Text rendering
  - Allows embedding plug-ins, including Flash

Introduction to OpenLaszlo

OpenLaszlo community update

- Sources now publicly available
  - Subversion repository: http://svn.openlaszlo.org
  - 3.x and 4.x branches are available in openlaszlo/
  - Sample code (photoblox et. al.) in labs/
  - Several side projects happening on branches in svn
  - We’ll soon be qualifying external committers
- So get involved! Help us create the most powerful unified language for cross-runtime web application development
  - Go to www.openlaszlo.org and click on “Community” → “Be a contributor” to learn more

Agenda

Introduction to Java ME Technology as AJAX/Web 2.0 Platform

Introduction to AJAX/Web 2.0 Using OpenLaszlo

Project Orbit Design Overview

Running LZX Application Using Project Orbit

Summary

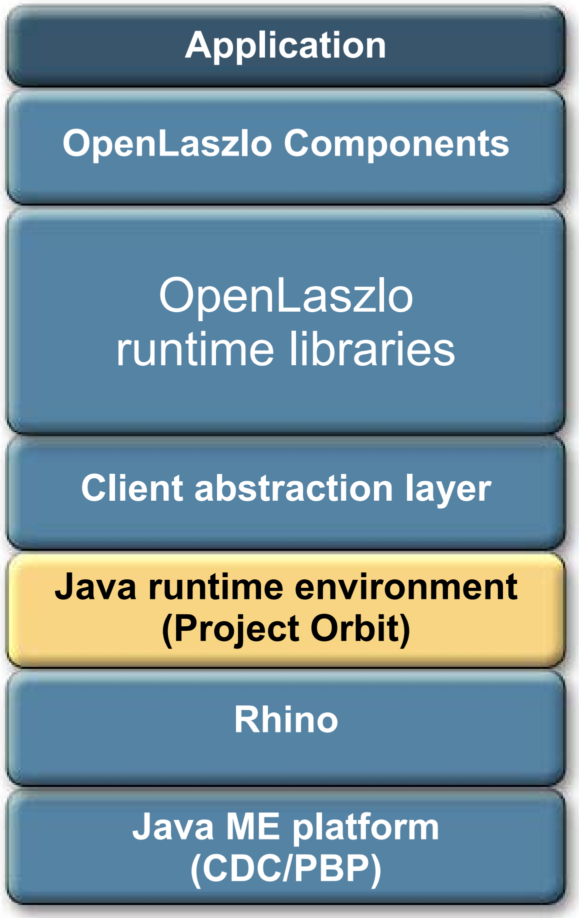
Project Orbit Design Overview

Why Project Orbit?

- Need a way to render AJAX/Web 2.0 style content on Java ME platform devices
- Re-uses DHTML runtime from OpenLaszlo to “view” or “play” content on Java ME technology
- Leverages Rhino engine which functions on top of Java ME platform CDC
- AJAX + Java ME platform = Project Orbit

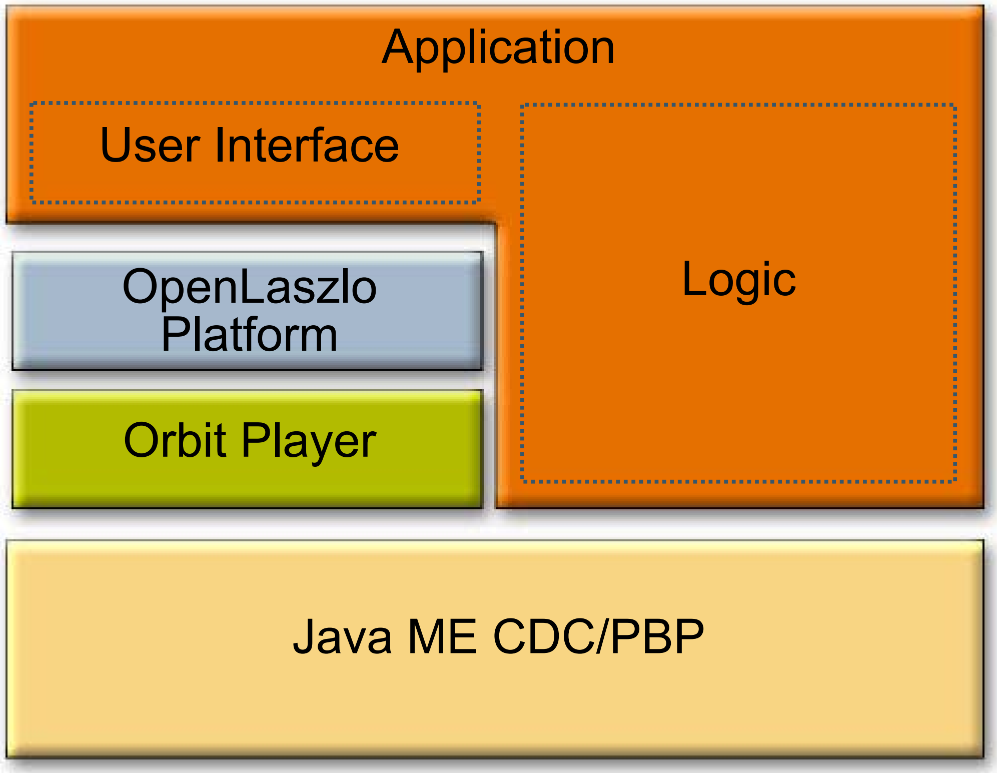
Project Orbit Design Overview

Project Orbit architecture


Application
OpenLaszlo Components
OpenLaszlo runtime libraries
Client abstraction layer
Java runtime environment (Project Orbit)
Rhino
Java ME platform (CDC/PBP)

Project Orbit Design Overview

Orbit player usage



Project Orbit Design Overview

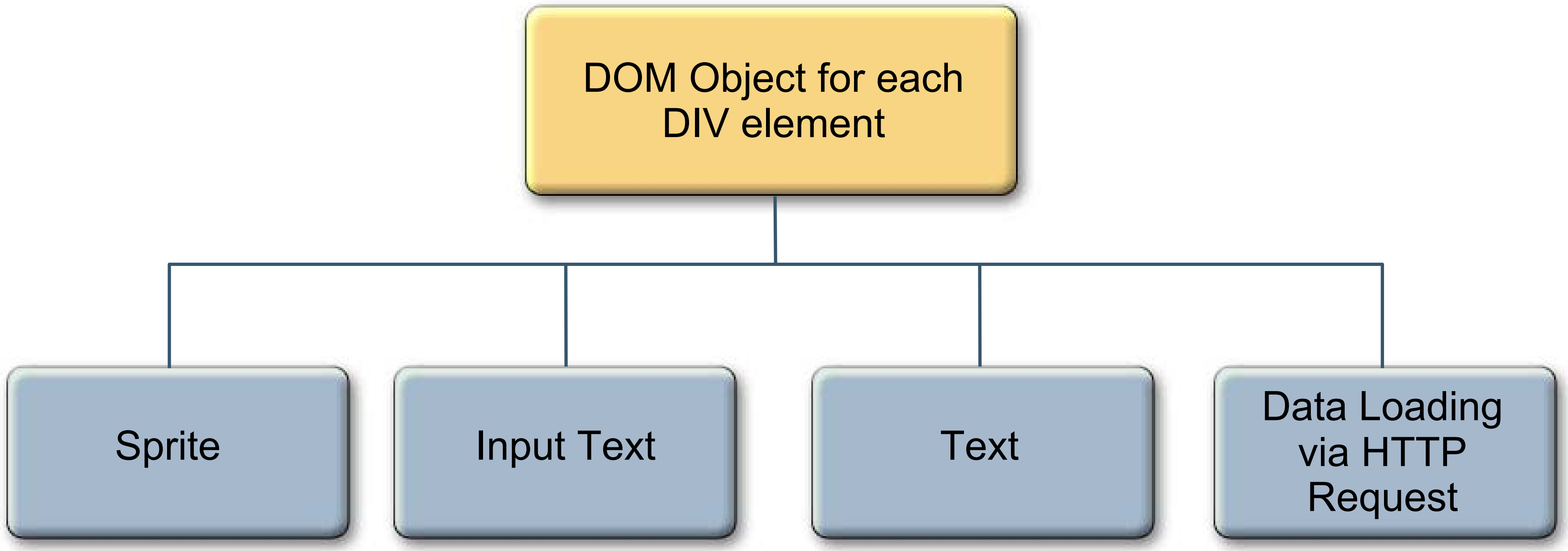
- Mini-DHTML Runtime
- DOM Objects: resizable, repositionable, web standard images, text, text field (input), and data loading

Project Orbit Design Overview

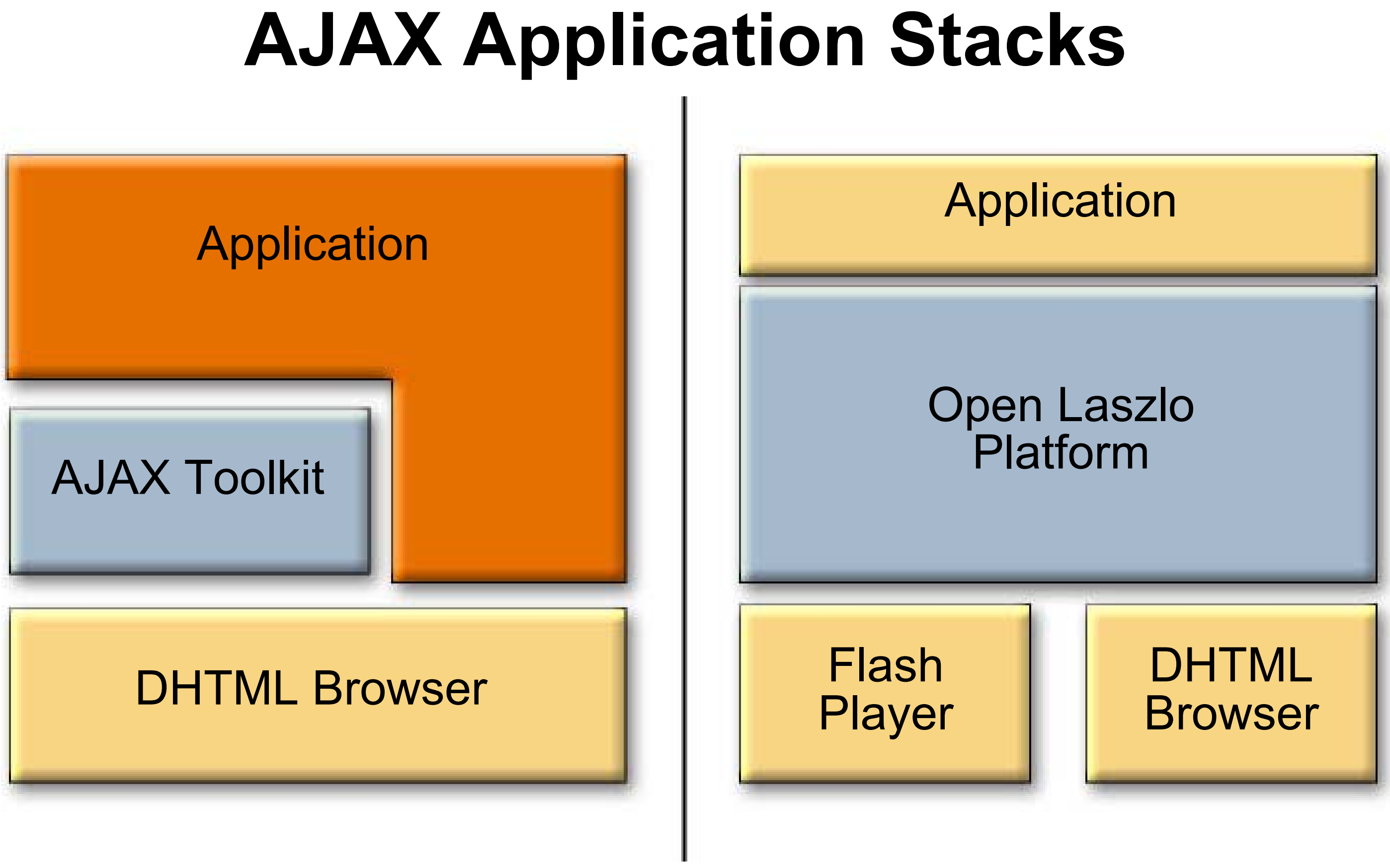
- DIV elements map to Java objects
  - Sprite (LZX primitive) to Java 2D™ API Image Object mapping
    - Dynamically created
    - Background color ex. Maps to Java 2D API fillColor()
    - Images from files (GIF, JPEG, PNG, etc.)
- Input text to TextField mapping
- Text mapping
- Data Loading XTML/HTTP Request, thinner wrapper

Project Orbit Design Overview

Project Orbit DOM Object


DOM Object for each DIV element
Sprite
Input Text
Text
Data Loading via HTTP Request

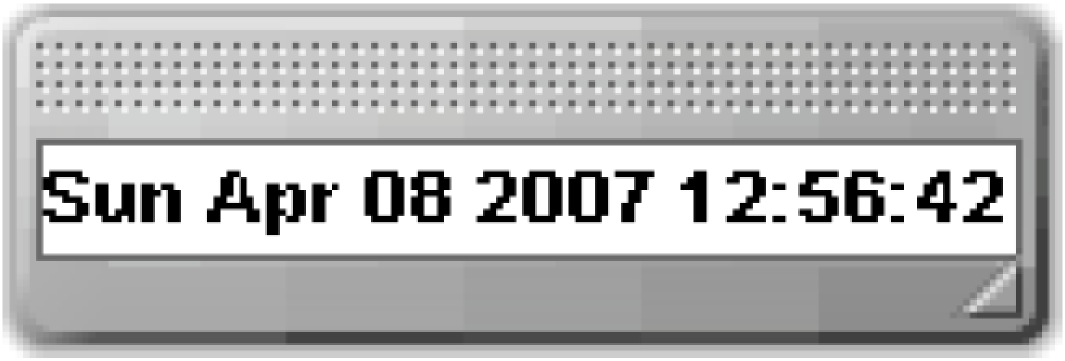
Project Orbit Design Overview



Project Orbit Design Overview

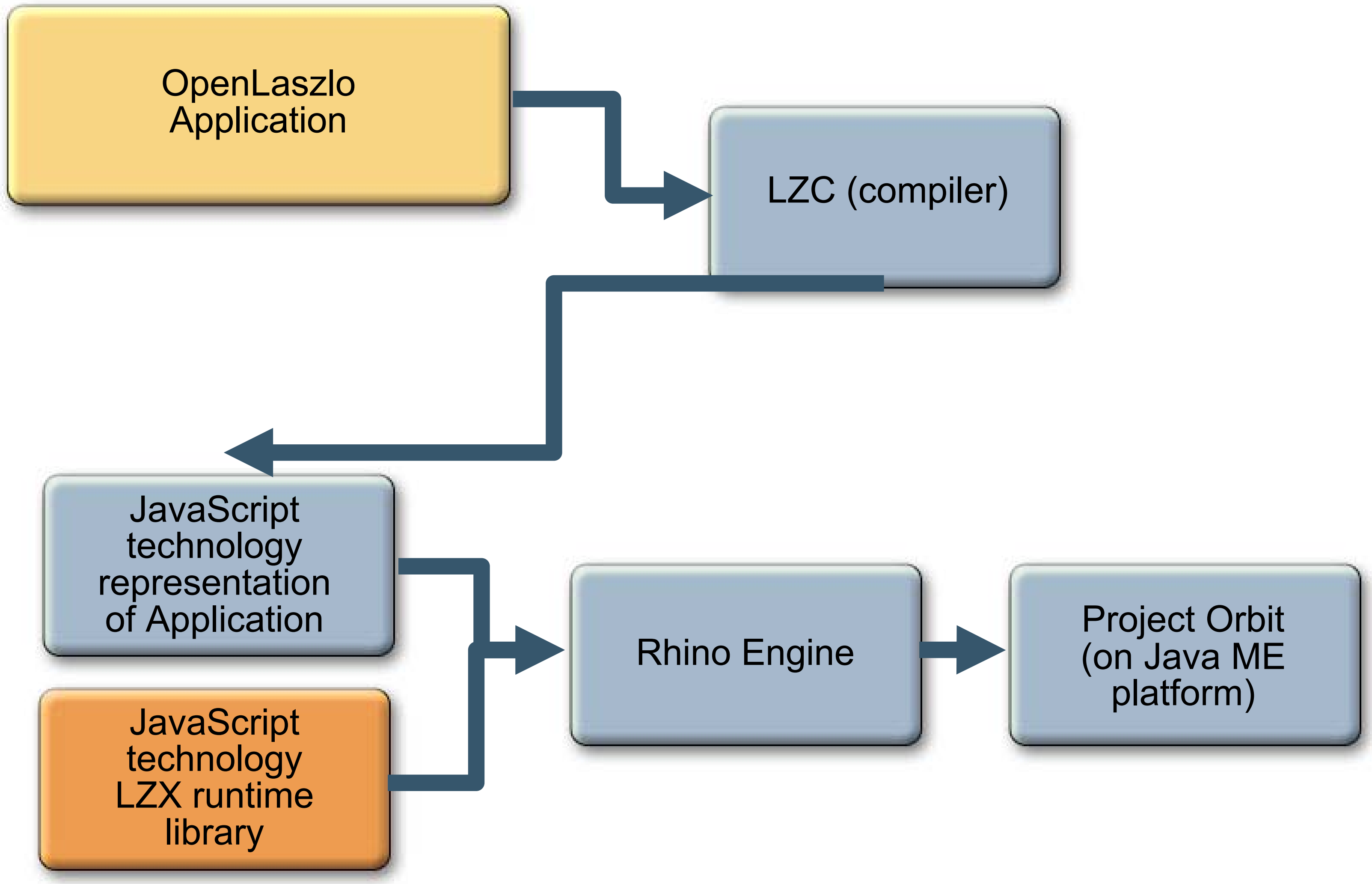
Sample OpenLaszlo Application

```
<canvas>
  <window id="vw" oninit="setTimeout('vw.updateText()', 1000)">
    <text id="lbl" fontstyle="bold" resize="true"/>
    <method name="updateText">
      lbl.setText(new Date())
      setTimeout('vw.updateText()', 1000)
    </method>
  </window>
</canvas>
```



Sun Apr 08 2007 12:56:42

Project Orbit Design Overview


OpenLaszlo Application
LZC (compiler)
JavaScript technology representation of Application
JavaScript technology LZX runtime library
Rhino Engine
Project Orbit (on Java ME platform)

Project Orbit Design Overview

- **Step #1:**

  Write LZX application (like the example in slide #28)

- **Step #2:**

  LZX app is compiled to JavaScript technology using LZC Laszlo compiler

- **Step #3:**

  Project Orbit uses Rhino engine to interpret JavaScript technology LZX runtime libraries plus LZX app (compiled to JavaScript technology from example in slide #28) to become translated to Java bytecodes

Project Orbit Design Overview

- Rendering graphics
  - Mapping LZX Sprites and Text to Java ME platform
  - Sprites → Java Platform Image objects
    - Rendered by drawImage() calls (Java ME Platform Personal Basis Profile)
  - Text → Java Platform Text strings
    - Rendered by drawString() calls (Java ME Platform Personal Basis Profile
  - InputText → Java TextField
    - Focus, blurring and selection not implemented yet
  - Pointer/Keyboard events → Maps to AWT events

DEMO

Project Orbit code walkthrough

Step through code from slide #28

Sample OpenLaszlo application

Agenda

Introduction to Java ME Technology as AJAX/Web 2.0 Platform

Introduction to AJAX/Web 2.0 Using OpenLaszlo

Project Orbit Design Overview

Running LZX Application Using Project Orbit

Summary

Running LZX Application Using Project Orbit

```
<!--sample.21.4.lzx-->


<canvas height="100" width="500">

  <view bgcolor="red" width="100" height="100"
onclick="this.myAnimator.doStart()">

    <animator name="myAnimator" attribute="x" to="100"
duration="1000" start="false"/>

  </view>

</canvas>
```

Running LZX Application Using Project Orbit

- Each LZX line compiled by LZC compiler
  - Compiled to JavaScript technology
- Will be use by Rhino engine on top of Java ME platform stack
- Will use LZX runtime library where graphics library routines and other useful utilities exist in JavaScript technology
- Java ME Platform VM used to execute behavior
- Java ME Platform Personal Basis Profile Graphics used for rendering

Running LZX Application Using Project Orbit (Raw Compiled JavaScript Technology)

```
var $dhtml = true;
...
canvas = LzCanvas.make({__LZproxied: "true", bgcolor:
16777215, build: "x.x.x.x"
, embedfonts: true, fontname: "Verdana,Vera,sans-serif",
fontsize: 11, fontstyle
: "plain", height: 100, lpsrelease: "Latest", lpsversion:
"4.0.x", runtime: "dht
ml", width: 500});
LzInstantiateView({attrs: {$events: {onclick: function ()
{
this.myAnimator.doStart()
}}, bgcolor: 16711680, clickable: true, height: 100,
width: 100}, children: [{at
trs: {attribute: "x", duration: "1000", name:
"myAnimator", start: false, to: 10
0}, name: "animator"}], name: "view"}, 2);
canvas.initDone()
```

DEMO

Sample.21.4.lzx: Walkthrough of Animator Red Box

Example from OpenLaszlo Web site that can also run in Project Orbit

Animation of a square that moves across the screen

Running LZX Application Using Project Orbit

- Walkthrough of sample.21.4.lzx app
- XML tags converted to DOM objects
- LZC compilation to JavaScript technology functionality
- Project Orbit takes JavaScript technology of sample.21.4 and LZX runtime to run in Rhino engine
  - Canvas is initialized
  - Animator is initialized
  - App is started

Running LZX Application Using Project Orbit

- Using Project Orbit to develop your own LZX apps
- Writing your LZX app
- Testing using Project Orbit
- Testing also using a desktop Web browser
- Deploying your LZX app using Project Orbit
- For more information:
  - https://orbit.dev.java.net

DEMO

Project Orbit with LzPix

Show the same OpenLaszlo app that ran on the desktop running in a Java ME Platform CDC environment instead

Agenda

Introduction to Java ME Technology as AJAX/Web 2.0 Platform

Introduction to AJAX/Web 2.0 Using OpenLaszlo

Project Orbit Design Overview

Running LZX Application Using Project Orbit

Summary

Summary

Project Orbit status

- Subset of DHTML/AJAX browser
  - Useful to extend beyond this (add more HTML/DHTML support)
  - More contributors can help extend
- Future directions
  - Create new runtime to map directly Java ME platform
  - No intermediate DHTML, directly Java ME bytecode
  - Run pre-compiled JavaScript technology to Java ME bytecode
    - Maybe leverage Rhino pre-compiler
    - Laszlo compiler to generate (bytecode generation)
    - Address MIDP (1.2 billion handsets)

Summary

The future of Project Orbit

- Hardware support
  - TV set-top boxes
  - Blu-ray Disc players (ex. Sony PlayStation 3)
  - Nokia Communicator and Internet Tablets
- Get involved!
  - Project homepage:
    - http://orbit.dev.java.net
  - Download source:
    - http://orbit.dav.java.net/svn/orbit
  - Contribute!

Q&A

Max Carlson
Co-founder and Lead Runtime Architect
http://openlaszlo.org

Hinkmond Wong
Sr. Staff Engineer
Sun Microsystems, Inc.

http://orbit.dev.java.net

Developing Mobile Ajax/Web 2.0 Applications, Using the java.net Open-Source Project Orbit

Max Carlson
Co-founder and Lead Runtime Architect
http://openlaszlo.org

Hinkmond Wong
Sr. Staff Engineer
Sun Microsystems, Inc.

http://orbit.dev.java.net

TS-5699