JSR proposal: Enhanced Hybrid APIs

OPENTECH

Introduction

“HTML5 is not the future of apps. While developers dream of 'write once run everywhere' the fragmented support for and limited APIs within HTML5 make this impossible.”

Josh Martin, analyst at Strategy Analytics, 2012

> We are now in a bit of a disillusionment phase for HTML5 as early adopters push the boundaries of the capabilities and sometimes fail. Anyone who thought that HTML5 was a panacea for mobile development is probably reconsidering. Having said that, the mobile Web will have an important place at the table and some of these projects that pushed the envelope will help push the technology forward.

Al Hilwa, analyst at IDC, 2012

“We burned through two years on that, It probably was the biggest strategic mistake we made.”

Mark Zuckerberg, Facebook CEO, 2012

Usability issues

“

While a mobile site is good, a mobile app is even better. We measured a success rate of **76% when people used mobile apps**, which is much **higher than the 64%** recorded for mobile-specific websites.

[…]As of this writing, there's no contest: **ship mobile apps if you can afford it**. Our usability studies with mobile devices clearly show that users perform better with apps than with mobile sites.

The empirical data is really all you need to know. **It's a fact that apps beat mobile sites in testing**. To plan a mobile strategy, you don't need to know why the winner is best, but I'll try to explain it anyway.

Mobile applications are more usable than mobile-optimized websites because **only limited optimization is possible during website design**. An app can target the specific limitations and abilities of each individual device much better than a website can while running inside a browser […].

Jacob Nielsen 13/02/2012

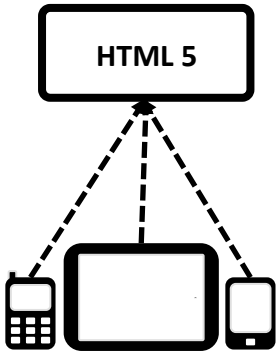
”

Usability issues

NATIVE APP

PROS:
- Native app are fluid and reactive offering an excellent UX

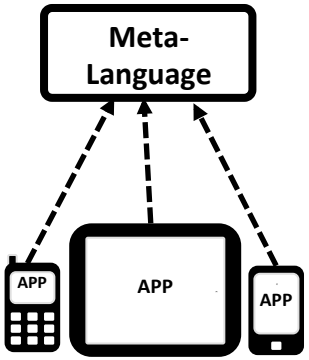
CONS:
- Limited flexibility



HTML5 HYBRID APP

PROS:
- Unique development
- No Distribution approval
- Full coverage of mobile devices

CONS:
- Sub-optimal UX
- Limited control



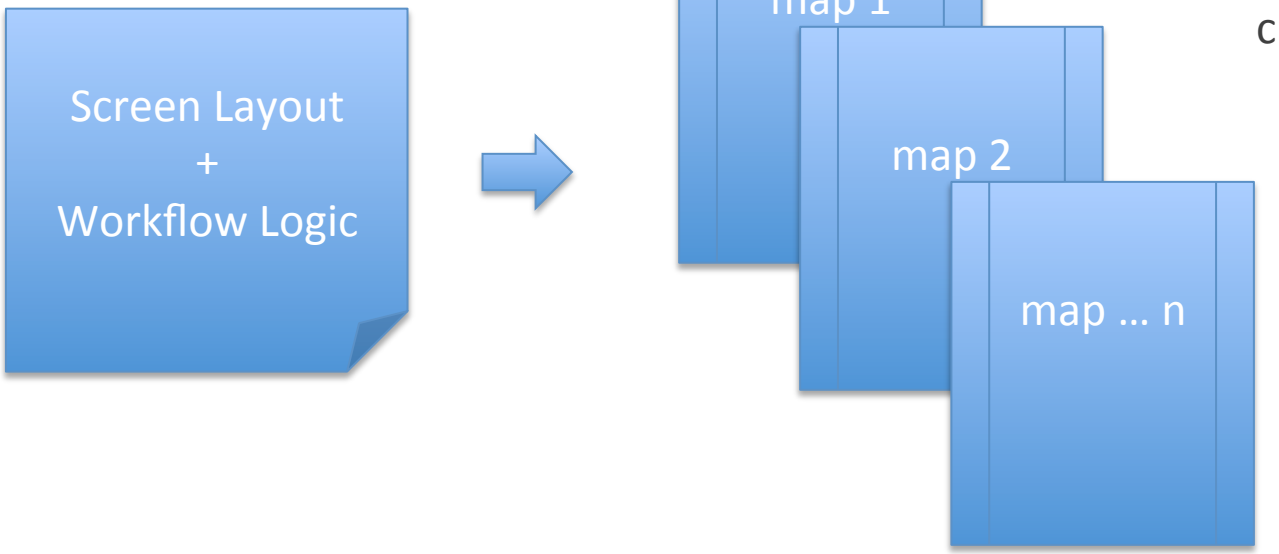
- With **Enhanced Hybrid Apps** we mean a new paradigm that mixes pros of hybrid and native approaches, therefore achieving:
- User Experience same as Native one
- Development “server side”, leveraging on Java EE
- The result is a native app completely configurable on server side, instantly available cross platform

Design

Enhanced Hybrid APIs

What are:

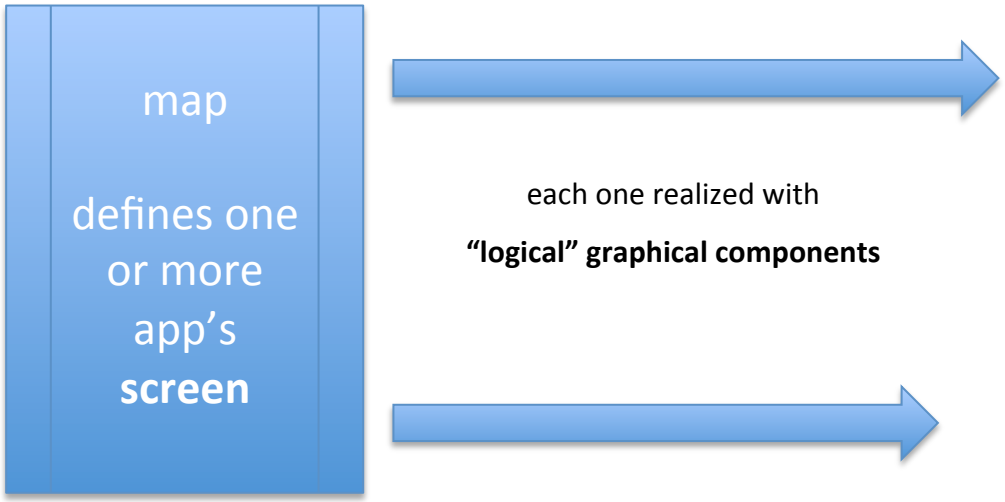
- Java APIs on top of Java EE used to write a Enhanced Hybrid Apps
- The APIs allow to "remotize", entirely or partially, both **model** and **control** layers (not just the **presentation** layer)
- They allow reuse of the business logic implemented on server side



set of «maps»

- Defined as Java beans
- Not dependent on the client side platform

Presentation Layer



Basic components

- buttons, label, textfield, images, container, etc.
- translated in a single native component

Complex components

- Cartesian graph, carousel, etc.
- Ad-Hoc components
- Translated in a complex composition of native components

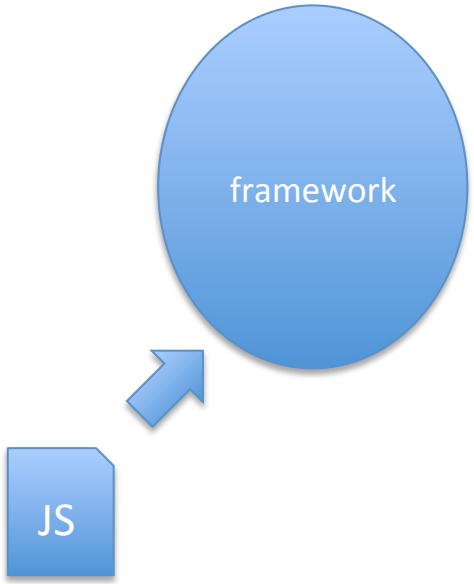
- **Each component is reconfigurable on sever side**. Ex.: for a button it is possibile to configure its content (image or text), background, and actions
- **Look and Feel and performance** of the resulting Apps are same as native

Control Layer

Enhanced Hybrid APIs use Java Actions (whose behavior is defined within the JSR) and JavaScript as control layer.

JS is used exclusively as a control tool in order to allow real-time behaviours:

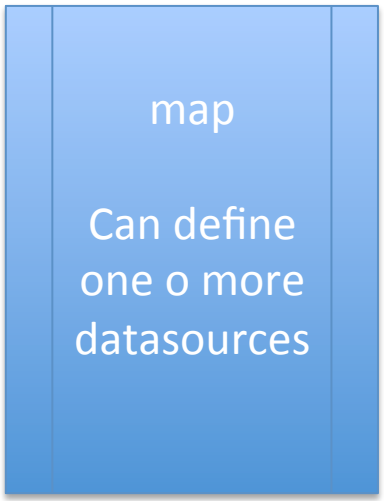
- Evaluation or valorization of component's state
- A JavaScript interact with each component only through actions



Pros:

- Uniform interaction from a specific device to the presentation layer.
- Each update of internal component state is delegated to the component iteslf.
- Independency from the rendering engine.

Model Layer



Datasource

- Represent a data bean
- Is contained in maps
- added or redefined by server interactions
- can be used by components or to setup a new request

Proposal Form

Proposal Form (1/3)

Please describe the proposed Specification

There is a huge demand by Java Developers (both individuals and large corporations) for leveraging on Java Development skills to deploy multiplatform mobile applications

What is the target Java platform?

Java EE

Please provide details here for which platform editions are being targeted by this JSR, and how this JSR has considered the relationship with the other platform editions.

Through the implementation of Java ME drivers, the specification could also revamp the Java ME platform, even though this is not the primary objective of the JSR, but a possible side effect.

Proposal Form (2/3)

What need of the Java community will be addressed by the proposed specification?

Writing once and running anywhere

Why isn't this need met by existing specifications?

Because with the spread of new mobile platforms the market is highly fragmented

Please give a short description of the underlying technology or technologies

The JSR requires Java EE in order to leverage on the existing workflows to build its functionalities: session management and HTTP communication are required by this JSR.

Does the proposed specification have any dependencies on specific operating systems, CPUs, or I/O devices that you know of?

No.

Proposal Form (3/3)

Are there any security issues that cannot be addressed by the current security model?

No

Are there any internationalization or localization issues?

No new issues, the localization can be managed through standard Java APIs

Are there any existing specifications that might be rendered obsolete, deprecated, or in need of revision as a result of this work?

No

Please describe the anticipated schedule for the development of this specification

JSR submittal: Jan 2013
Early Draft Review: May 2013
Public Draft Review: Aug 2013
Proposed Final Draft: Oct 2013
Final Approval Ballot: Dec 2013

Demo

Example of a Enhanced Hybrid App

FAQ

FAQ

Why handling this proposal within the JCP?

There is a huge demand by Java Developers (both individuals and large corporations) for leveraging on Java Development skills to deploy multiplatform mobile applications

How to deal with native implementations on different target platforms (iOS, Android, Windows Phone, Windows 8, BlackBerry, Java ME)?

Same as what has been done with JDBC: native drivers are implemented on each platform, and are not strictly part of the JSR

How to test compliancy?

Compliancy test is done on the meta-data generated by the APIs, not on the User Interface produced by the native drivers