BUILDING TITANIUM WITH ALLOY

February 2014

ALLOY + TITANIUM

WHO WE ARE

M&B

MAKE & BUILD

THE ARTERY · ATLANTA, GA

Make & Build
Appcelerator Partner
Atlanta Titanium Developers sponsor

Andrew Zuercher
Chief Innovation Officer
@bahzuercher
az@makeandbuild.com

WHAT IS TITANIUM?

TITANIUM ALLOYS

Titanium alloys are metals which contain a mixture of titanium and other chemical elements. Such alloys have very high tensile strength and toughness (even at extreme temperatures). They are light in weight, have extraordinary corrosion resistance and the ability to withstand extreme temperatures.

ALLOY IS AN MVC APPLICATION...

BY APPCELERATOR FOR TITANIUM

Alloy will allow **Titanium** developers, old and new, to develop cross-platform mobile applications easier and more effectively than ever. The separation of concerns will increase the scalability of your apps. Titanium best practices are generated for you under the hood, making your apps of the highest quality across platforms. The markup, styles, themes, and mountain of other Alloy features will take your productivity to a whole new level.

It's kind of a big deal.

TITANIUM 3.0

TITANIUM 3.x

- CLI

Alloy concepts

- Views (xml)
- Controllers (javascript)
- Models & Migrations (javascript)
- Styles (tss), Themes
- Assets (resources)
- Widgets
- I18n
- Config (config.json), Build configuration
- (alloy.jmk), Initializer (alloy.js)

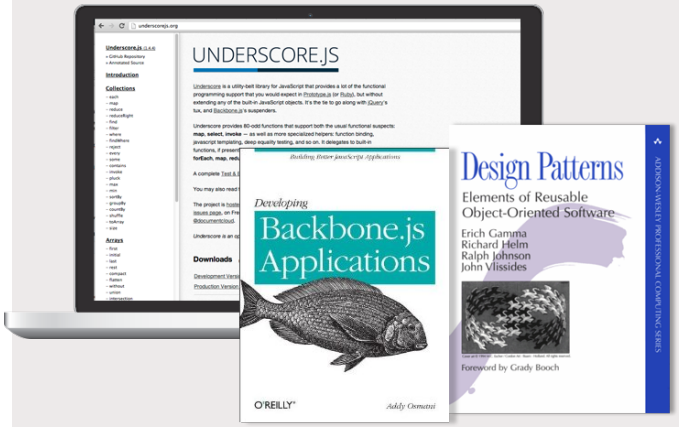
HOMEWORK

SUGGESTED READING

- Developing Backbone.js Applications, backbonejs.org
- Underscore.js
- Design Patterns, MVP (Model-View-Presenter)

Pre-requirements

- Install titanium 3.2
- Install npm http://nodejs.org/#download



PAGE TITLE GOES HERE

CREATE THE PROJECT

```
titanium create --platforms ios,android --id com.mnb.fantasy --name
fantasypool--workspace-dir .
```

ADD ALLOY

```
sudo npm install -g alloy

cd fantasypool
alloy new
```

RUN THE PROJECT

```
titanium build-p ios
```

STEP THROUGH THE PROJECT STUCTURE

Index

app/controllers/index.js

app/views/index.xml

app/styles/index.tss

Note the contents in Resources/

Regenerated on each build

Only includes platform specific items

Resources/alloy/controllers/index.js

Merges in styling and view definition

Uses Backbone.Events extends BaseController

Views

PassiveView approach
XML Based with Ti.UI.xxx component to TAG mappings (Window, Label)
onXxx event (onClick) bindings to Controller methods (doClick)
selector attributes => id, class, etc.

```
<Alloy>
    <Window class="container">
        <Label id="label" onClick="doClick">Hello, wassup</Label>
    </Window>
</Alloy>
```

formFactor and platform allow for specialization

```
<Alloy>
    <Window class="container">
        <View height="50" width="200" bottom="10" backgroundColor="#cdcdcd">
            <Label class="platformLabel" platform="android” formFactor="tablet">
                   android tablet
             </Label>
        </View>
    </Window>
</Alloy>
```

Use the Require Tag to Import Other Composite Views

Index.xml

```
<Alloy>
    <Window>
        <View id="top" />
        <Require src="theRest"/>
    </Window>
</Alloy>
```

theRest.xml

```
<Alloy>
    <Require src="bottom" id="bottom"/>
</Alloy>
```

bottom.xml

```
<Alloy>
    <View>
        <Button id="b"></Button>
    </View>

</Alloy>
```

TSS Styling

- Application wide styles (app.tss) and view specific (index.tss)
- Selector based - Class (".container"), Type ("Label"), and ID ("#label"), FormFactor/Platform

Index.tss

```
".container": {
    backgroundColor:"white"
},
"Label": {
    width: Ti.UI.SIZE,
    height: Ti.UI.SIZE,
    color: "#000"
}
"#label": {
    font: {
        fontWeight: "bold"
    }
}
"Label[platform=ios formFactor=handheld]": {
    backgroundColor: "#f00",
    text: 'iPhone'
}
```

Controller wiring

Glue up the views & invoke Non-UI components

Use of the $ context

```
function doClick(e) {
    alert($.label.text);
}
$.index.open();
```

exports.baseController defines base class

```
var controller = Alloy.createController('a');
controller.setDelegate($);

exports.showAlert = function() {
    alert($.t.text);
};

function doClick(e) {
     controller.getView().open();
};

$.index.open();
```

Index.js

```
var delegate;

exports.setDelegate = function(o) {
    delegate = o;
};

function doClick(e) {
     delegate.showAlert();
};
```

a.js

exports.baseController defines base class

Conditional Code

Conditional Code - like macros in c/c++

- OS_xx - ex: OS_IOS
- ENV_xx - ex: ENV_DEV

```
if (OS_IOS)
{
    var closeButton = Ti.UI.createButton({
        title: 'Close',
        style: Ti.UI.iPhone.SystemButtonStyle.PLAIN
    });

    closeButton.addEventListener('click', function(){
        $.window.close();
    });

    $.window.leftNavButton = closeButton;
}
```

Create Model

Use generators

```
alloy generate model player sql name:text team:text position:text
```

Create Migration

Create migration

```
alloy generate migration player
```

Migration implements up and down (like RoR)

```
 migration.up = function(migrator) {
    migrator.createTable({
        "columns": {
            "name":"TEXT",
            "position":"TEXT",
            "team":"TEXT"
        }
    });
};
migration.down = function(migrator) {
    migrator.dropTable("player");
};
```

Sync Adapters - Used for mapping Models to remote data

Data Binding

index.html

```
Alloy.Collections.player.fetch();
$.index.open();
```

index.js

```
<Alloy>
    <Collection src="player"/>
    <Window>
        <ScrollView id="scroll" dataCollection="player">
            <Require src="entry"/>
        </ScrollView>
    </Window>
</Alloy>
```

entry.xml

```
<Alloy>
    <View id="container">
        <Label id="name" text="{name}"/>
        <Label id="position" text="{position}"/>
        <Label id="team" text="{team}"/>
    </View>
</Alloy>
```

Examples

Clone the project

```
git clone git@github.com:appcelerator/alloy.git
```

Install Jake and other NPMS

```
sudo npm install -g jake
sudo npm install -g colors
sudo npm install -g wrench
sudo npm install -g jsonlint
sudo npm install -g xmldom
```

Run the project

```
cd alloy/test/apps
jake app:run dir=basics/simple
```

Guides

The Guides are Very Good

http://docs.appcelerator.com/titanium/latest/#!/guide/Alloy_Framework

Check out the source

https://github.com/appcelerator/alloy

and the sample apps there....

Suggestions

- Slight ramp up, extremely powerful however ==> worth the investment
- When something goes wrong, its hard to troubleshoot source - you can debug generated Resources, but still not the same
- Check out the alloy google group:

  https://groups.google.com/forum/?fromgroups#!forum/appc-ti-alloy

Questions?

Q&A