Cappuccino

+

Good Practices

#CappCon2016

Christophe Serafin

- Software Engineer
  at **Nuage Networks**
  http://nuagenetworks.net/

- Working with **Cappuccino**
  for 3 years
  http://www.cappuccino-project.org/

Summary

- Structure & Organization of a big application
- Use Git submodules for libraries
- Cappuccino environments
- Organize your xib files
- Compile & deploy an application

Structure & Organization of a big application

Organize your files

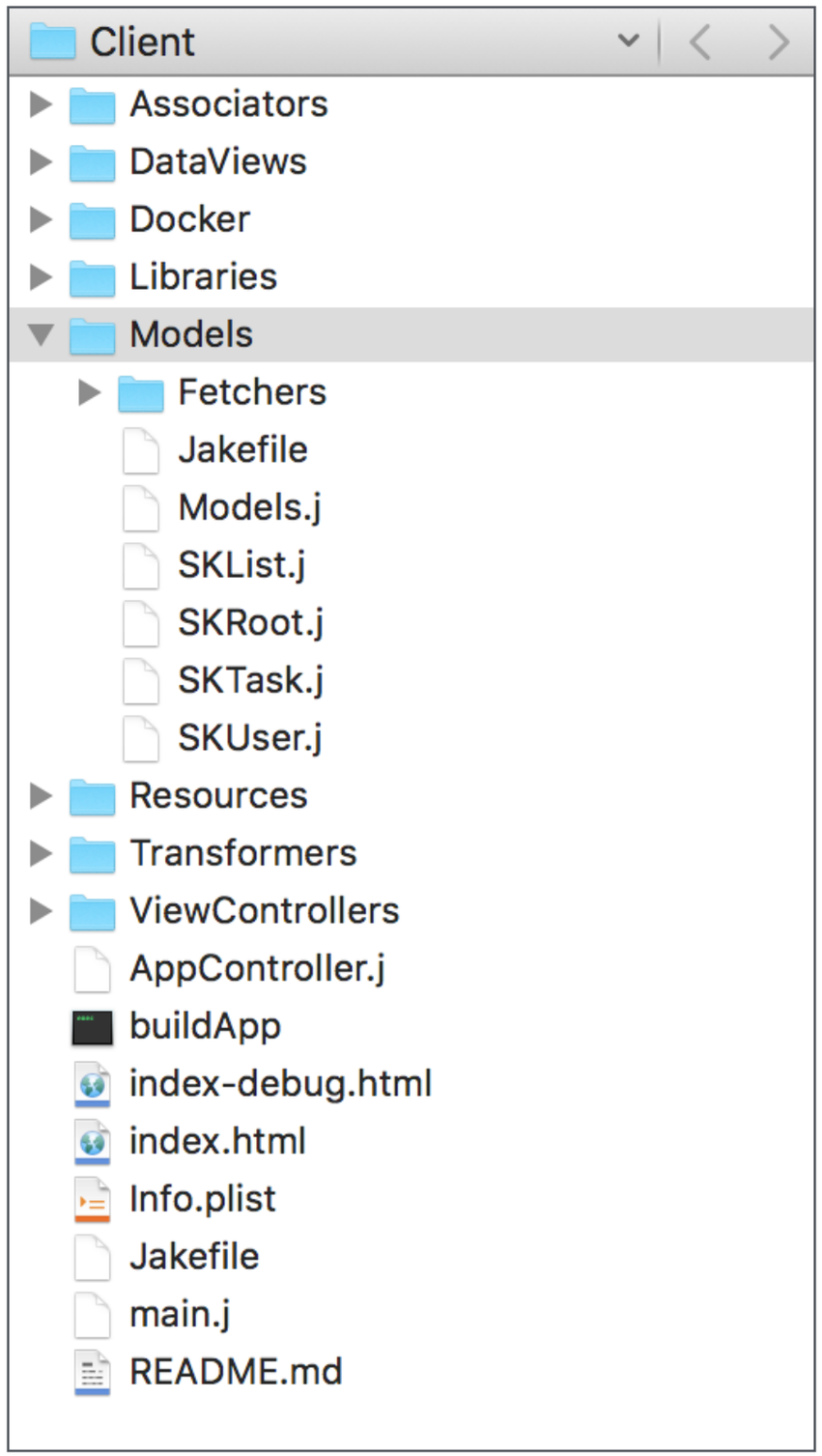
- Keep folder hierarchy **simple**

    1. Models - define your object & relations
    2. Resources - holds your .xib files and images
    3. ViewControllers - Define your controllers
    4. Libraries - Contains all external libraries
    5. Frameworks - Symlink or copy from your $CAPP_BUILD

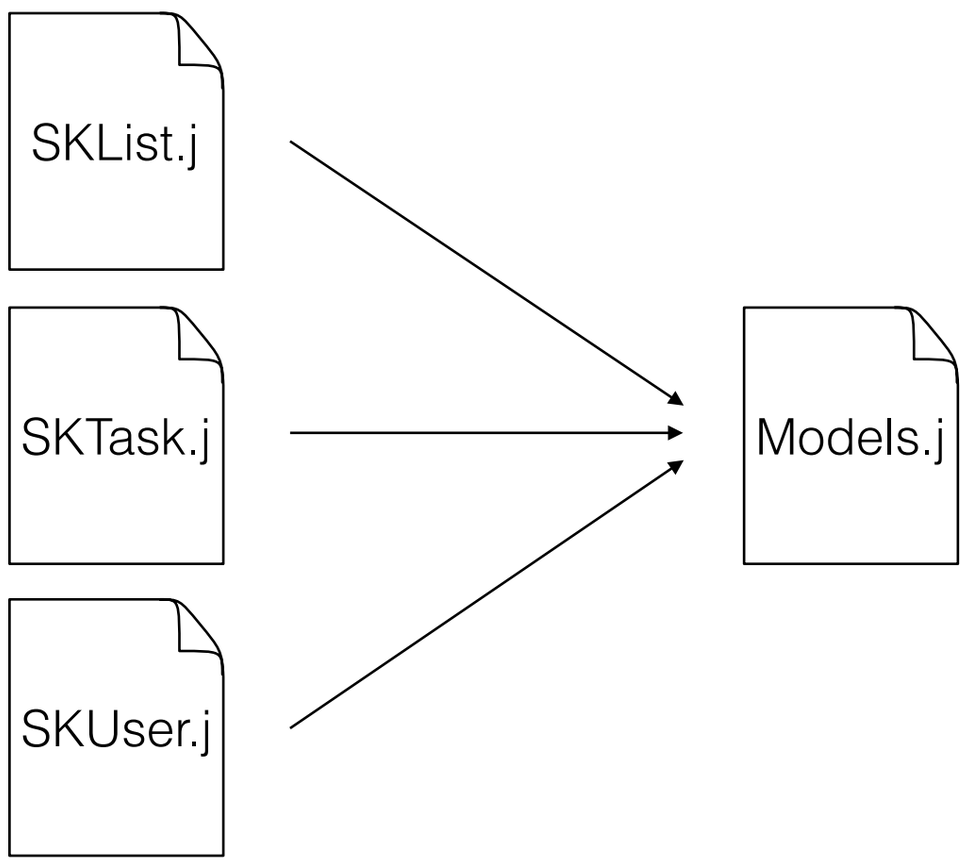
- Create **generic** folders if necessary

    Examples: DataViews, Associators, Abstracts etc.

Organize your files



Gather your files



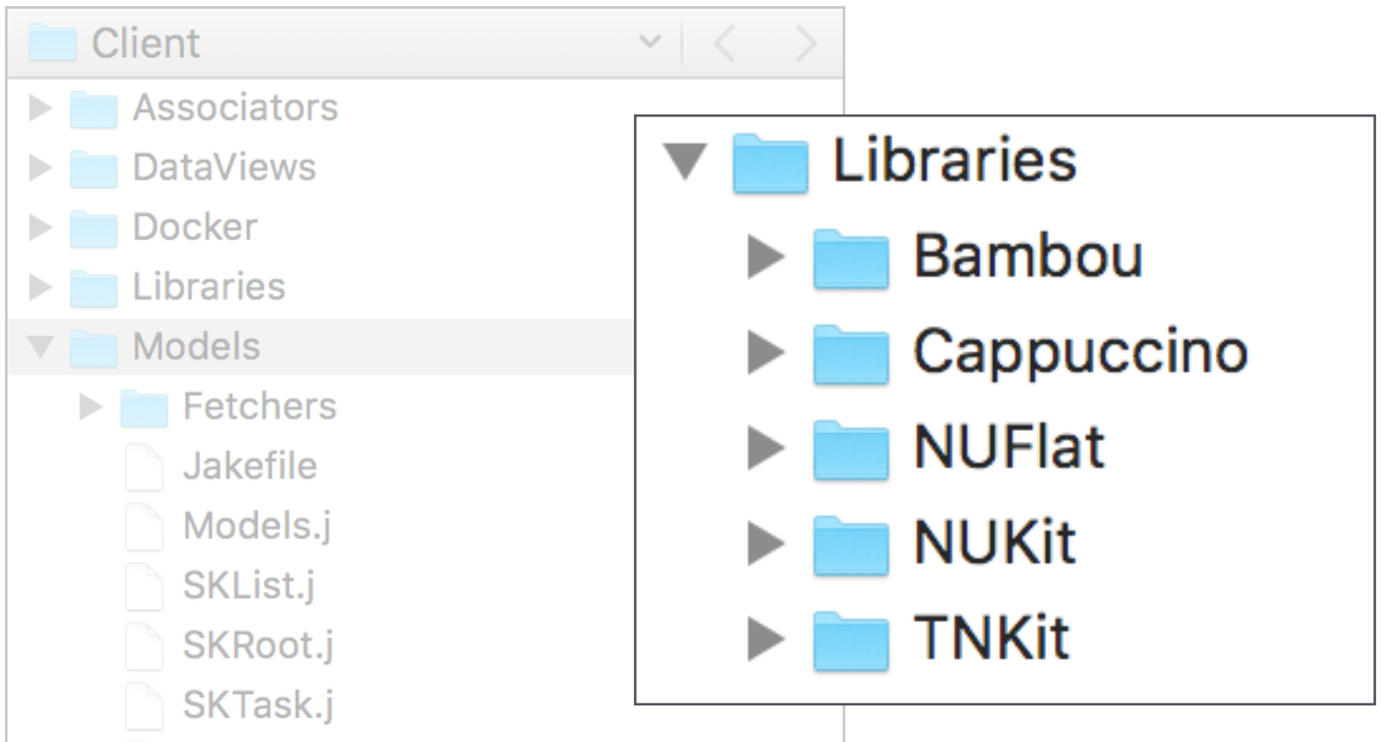

```
@import "SKList.j"
@import "SKTask.j"
@import "SKUser.j"
```

https://github.com/nuagenetworks/monostack-example/tree/master/Client

Git Submodules

Libraries

- Improves **reusability**
- Use them as **submodules** in your application
- Open your mind and **share** your code to the community



Advantages

- Submodule is represented as a **commit hash** in your code

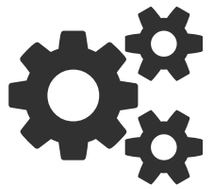
```
$ git status
On branch master
Your branch is up-to-date with 'origin/master'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)
  (commit or discard the untracked or modified content in submodules)

  modified:   Libraries/NUKit (modified content)
```

- Link your code to a specific branch or version of a library to manage **multiple versions** of your product

Branching model

Product	*Libraries*
master	Bambou @ 48d1e82 Cappuccino @ 7c21ed2 NUKit @ a2a0538
release	Bambou @ 7c4e22c Cappuccino @ 7c21ed2 NUKit @ 0034ca1

Cappuccino Environments



Cappuccino Environments

- Create a **dedicated** environment for each branch of your code
- Works exactly the same as **virtualenv** for python
- **capp_env** command installed with Cappuccino

Set up your environments

```
// Add cappenvs environments at your product root level
$ mkdir .cappenvs
$ mkdir .cappenvs/environments
$ cd .cappenvs/environments
```

```
// Create an environment for master and release branches
$ capp_env -p master
$ capp_env -p release

// Activate an environment will update your current PATH
$ cd master
$ source bin/activate
```

Auto-Cappenvs

Script to switch from an environment to another depending on the current branch of your code

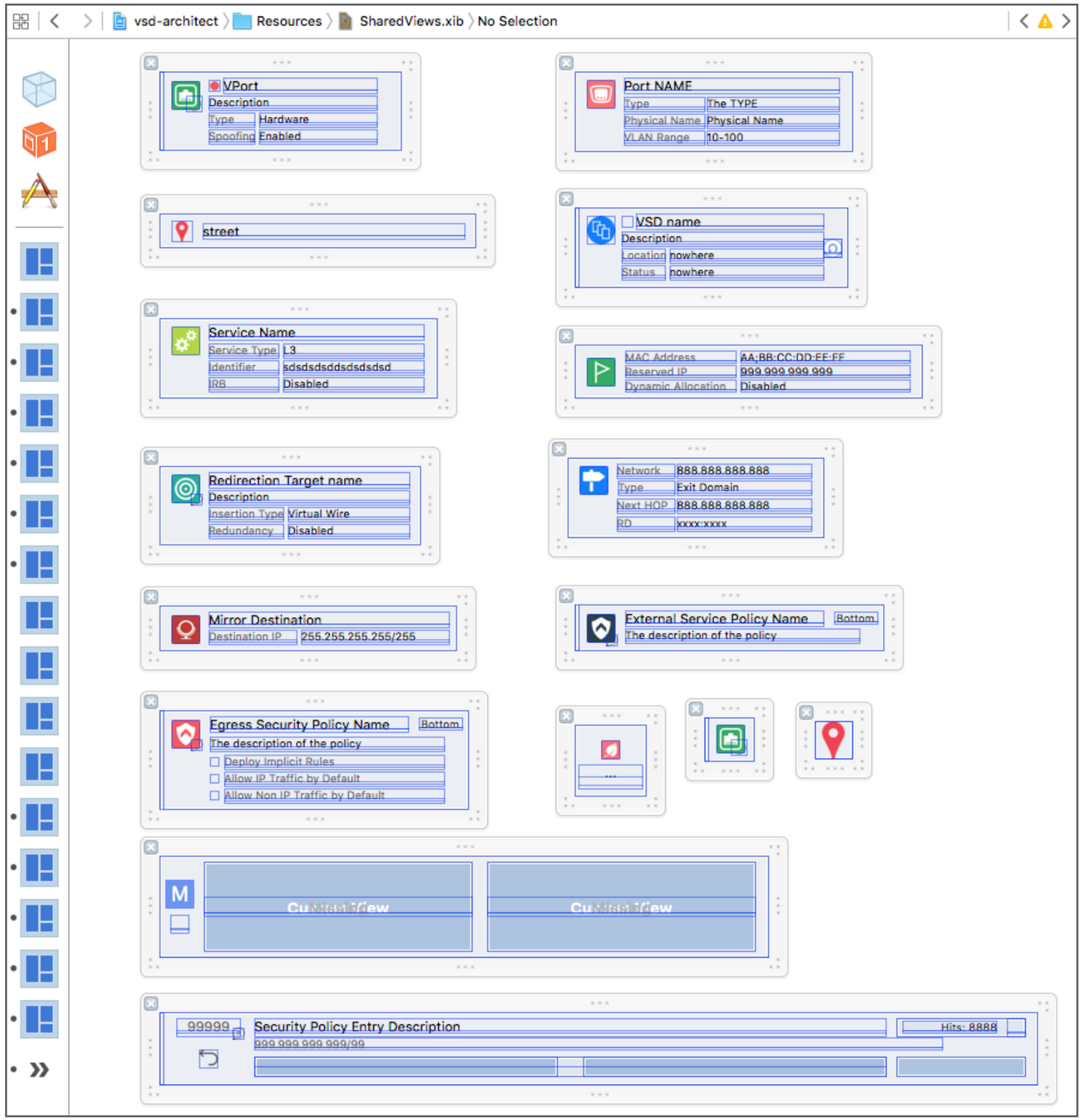
```
$ cd Client
(master)$

(master)$ git co release
  Switched to branch ‘release’

(release)$ git co master
  Switched to branch ‘master’

(master)$
```

Organize your xib files

Shared Views

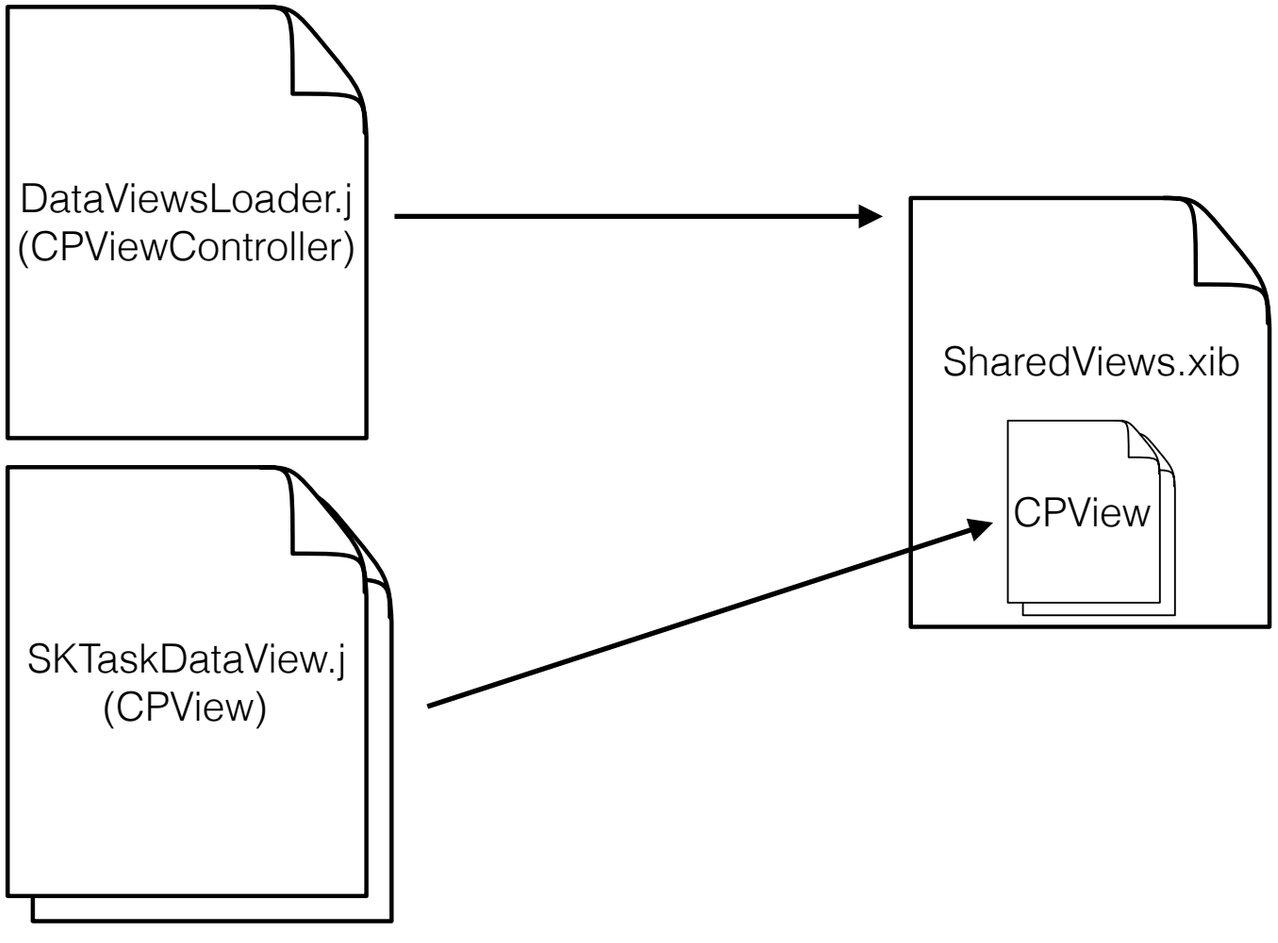
- Certain **view parts** can be used multiple times in your product
- **Reduce cib files** that should be loaded in your application
- Gather shared views in a **single xib file**

Shared Views



Shared Views

- Create a **CPViewController** that contains multiple **CPView**
- Declare an @outlet in an object that inherits from AbstractDataViewLoader (NUKit)



Compile & Deploy an application

Compile & Deploy

- **objjc** is transforming Objective-J code to Javascript (extention .o)
- **flatten** creates a single-compiled file for the whole application (extension .sj)
- **press** tries to reduce the size by removing the unused parts
- **jake deploy** is doing everything for you !

Jake

- jake install
  *Builds and installs Cappuccino into the directory **$CAPP_BUILD***

- jake debug / release
  Builds the debug / release version and place the generated files in Build subdirectory

- jake clean
  Cleans the Build subdirectory

- jake deploy
  Builds the release version and run the tool chain to provide an optimized version of the application (Build/Deployment/.ready)*

buildApp Script

```
$ ./buildApp -h
Usage: buildApp [options]
Options:
  -h, --help            Show this help message and exit
  -c, --cappuccino      Build and install Cappuccino
  -k, --nukit           Build and deploy NUKit
  -d, --project         Build and deploy project
  -a, --all             Build and deploy everything without Cappuccino
  -E, --everything      Build and deploy everything + Cappuccino
  -L, --libraries       Build all libraries
  -v, --verbose         Print commands output
  -C, --clean           Clean all libraries and project
  --clobber             Clean all libraries, project and cappuccino
  --debug               Generate a debug deployment build
```

https://github.com/nuagenetworks/nukit/tree/bambou/Tools/nukit

Other Tips

Tips

- Organize your code using **pragma** sections

```
#pragma mark -
#pragma mark Initialization | API | Utilities | Actions | Overrides | Delegates
```

- Create and respect a naming **convention**

```
@outlet fieldName
@implementation SKTask
- (CPArray)permissionsForObject:(id)anObject
- (@action)changeCheckboxStatus:(id)aSender
```

- Write **conditions** as variables

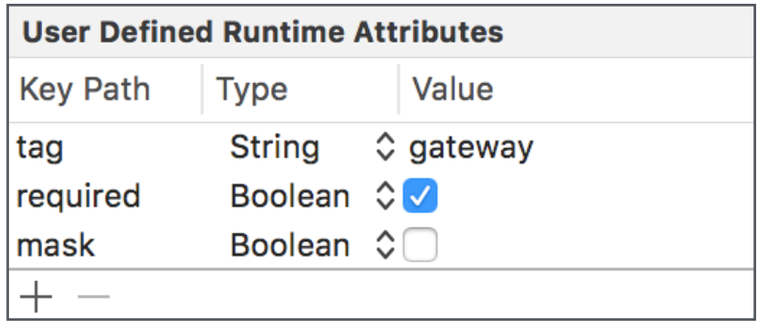
```
var shouldHide = [user hasAuthorization] && ([anObject isValid]  || [anObject address

[buttonCountry setHidden:  shouldHide];
```

- Define **delegates methods** to provide a flexible API

```
[_delegate moduleContext:self willManageObject:_editedObject];
// …
[_delegate moduleContext:self didManageObject:_editedObject];
```

Tips

- Use **User Defined Runtime Attributes** to avoid @outlet declaration



- Optimize your medias
  Create a **Flat Theme** and use **imageOptim**

- Avoid **xib conflicts**

- **Git ignore** autogenerated files
  Frameworks, cib, Build…

- **Never** write @import <AppKit/AppKit.j>

- Join us on **Gitter** and follow **Cappuccino News**

Thanks !

#CappCon2016