Nexus Core 3.0

Simple is better

Author: Mike Zhang

Email: [winsource6@hotmail.com](mailto:winsource6@hotmail.com)

Create Date: 05 August 2014

Last Update Date: 10 November 2014

Nexus Core 3.0

# Life cycle

## Request Handler

* Module Handle
  + Domain check
  + Redirect to main domain from subdomain

## Global Application Start

* Nexus Engine Startup
  + Dependency engine setup (AutoFac)
  + Cache engine setup (not decide yet, use normal memory for now)
  + Initial setup
* Model Binder (Consider)
  + Model Metadata Provider (Consider)
* Custom View Engine
  + Add custom view template path
* Route Builder
  + Add Route Constrains

## Base Controller

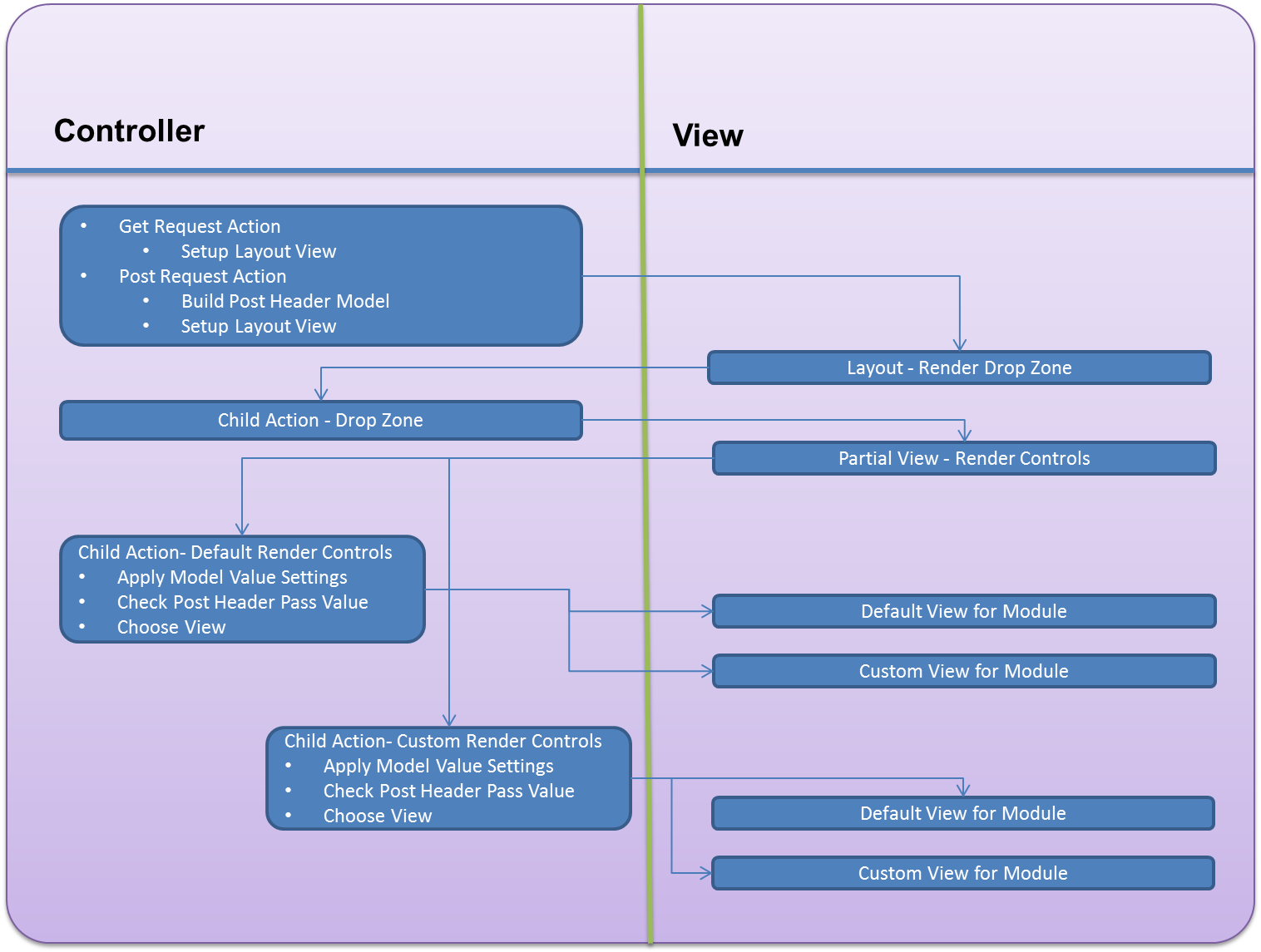
* Engine Context
  + Site Info Builder
  + Page Info Builder
  + Required JavaScript Builder
  + Required Css Builder
  + Page Module Model Builder
  + 301 handler
  + 404 handler

## Controller and View

* Controller
  + Get Request Action
  + Post Request Action
  + Error Request Action
  + Authentication Login Action
  + Default Drop Zone Child Action
  + Default Render Module Child Action
* View
  + Load Layout
  + Loop Drop Zone
  + Loop Modules in Drop Zone
  + Partial View for module default display
* Custom View
  + Partial View for module custom display

## Client Side JavaScript

* Default JavaScript
  + Bootstrap
  + Angular
  + Angular Master Controller
  + Http Ajax post
* Custom JavaScript
  + Depend on page



# Source Tree

1. *An element of data in the tree is called a node.*
2. *A node has either one parent or none.*
3. *If a node doesn't have a parent, it's called a root node.*
4. *If a node has a parent, the node is referred to as the child of the parent node.*
5. *A node can have an arbitrary number of children.*
6. *A node can't be the parent of itself.*

## Concept

* Master Root (**MasterRoot**)
  + Clients (**Client**)
    - Websites (**WebsiteRoot**)
      * Website1 (**Website**)
        + Menu (**MenuRoot**)

Menu1 (**Menu**)

Homepage (**MenuItem**)

Products

Product1

Product2

About Us

Contact

Menu2 (**Menu**)

Spring Landing page

* + - * + Pages (**PageRoot**)

Layout1 (**Layout**)

Page1 (**Page**)

Page7

Page8

Page2

Page3

Layout2

Page4

Page5

Page9

Page10

Page6

* + - * Website2 (**Website**)
    - Modules (**ModuleRoot**)
      * Module1 (**Module**)
      * Module2
    - Contents (**ContentRoot**)
      * Category1 (**Category**)

# Page Layout and Inherit

## Create

* A page can be create
* A page must have a parent node
* A page can inherit from layout
* A page can inherit from page
* A page cannot inherit from their own child page
* A page cannot inherit from their own child page in chain

### Modify

* A page can change parent to layout
* A page can change parent to page
* A page cannot change parent to their own child page
* A page cannot change parent to their own child page in chain

### Delete

* A page can be deleted
* A page cannot be deleted when it has child page
* A page cannot be deleted when it has been used in web node

## Display Page Layout Structure

### Basic Rules

* A control can be dropped on PlaceHolder and Col and Container and Row
* A Col can be dropped on PlaceHolder and Row
* A Row can be dropped on PlaceHolder and Container
* A Container can be dropped on PlaceHolder only

### Sample 1

* Page
  + Container (**PlaceHolder**)
    - Row (**PlaceHolder/Container**)
      * Col(**PlaceHolder/Container/Row**)
        + Controls(**PlaceHolder/Container/Row/Col**)

### Sample 2

* Page
  + ParentContainer(**PlaceHolder**)
    - ParentRow(**PlaceHolder/Container**)
      * Col(**PlaceHolder/Container/Row**)
        + Controls(**PlaceHolder/Container/Row/Col**)
      * ParentCol(**PlaceHolder/Container/Row**)
        + ParentControls(**PlaceHolder/Container/Row/Col**)
        + Controls(**PlaceHolder/Container/Row/Col**)
        + Controls(**PlaceHolder/Container/Row/Col**)
      * Col(**PlaceHolder/Container/Row**)
        + Controls(**PlaceHolder/Container/Row/Col**)

### Sample 3

* Page
  + Controls (**PlaceHolder**)
  + Col (**PlaceHolder**)
    - Controls (**PlaceHolder/Col**)
  + Row (**PlaceHolder**)
    - Col(**PlaceHolder/Row**)
      * Controls (**PlaceHolder/Row/Col**)

# Business Logic Layer Service

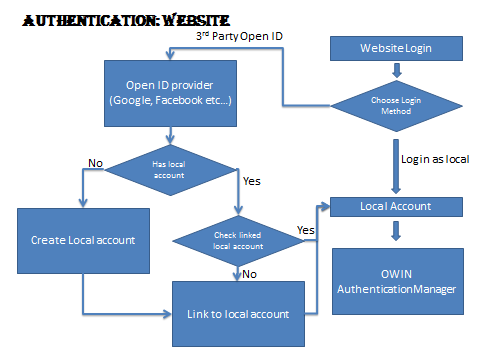
## Data Service Types

* Primitive: CRUD service for one entity only
* Aggregate: Composite service has join with multi entities, can see all Primitive Service
* Component : Component service with model container from both aggregate and primitive model

# User membership and Authentication

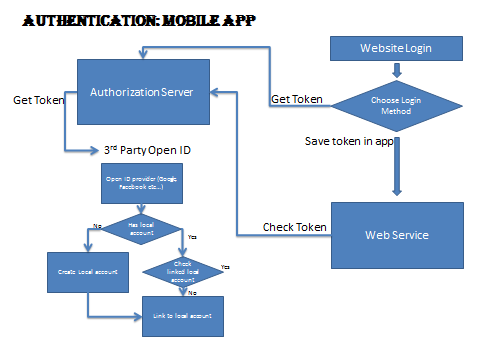
## Website Login

Owin Authentication Manager will be used to manage for local user identity. Any other open ID provider will linked to a local account.



## Mobile App Login

Mobile App login will able to access via Web API service only, two authorization method will be used in Service. Application Token and User login Token, The final decision haven’t been made yet.



## Admin Security

In Admin section permission is managed by User Group and individual user. Master Admin can manage all user groups and users from all website. Site admin can manage all user groups and black list from his websites

From Source tree user group permission is inherited from parent if not defined on current level. All user group or users created for source tree will always create default permission on root node.

# Log and Error Handling

Error handing works close with Logger and LogCode. Depend on log code profile to decide which to log

## Log Code Category

* 1-199 System Critical Error
* 200-299 System Error
* 300-399 System Warning
* 400-499 System Information
* 500-599 Web page Error
* 600-699 Web page Warning
* 700-799 Web page Information
* 800-899 Module Error
* 900-999 Module Warning
* 1000-1099 Module Information