

Apptacular Documentation




What is Apptacular

Apptacular is an application scaffolder, that introspects a database, and generates an application that is capable of displaying, searching, and editing the underlying database. It also give you the ability to customize many elements of the created application.

Additionally, it can generate remote API's. These remote services can be easily consumed using Flex for Flash Player or the AIR runtime. In short, you can quickly wire up your client app with your backend using other Adobe tools. These services are also accessible in your AJAX or Web 2.0 or HTML 5 or HTML/JS/CSS application.

Finally, if you are looking to get into Unit testing, Apptacular can generate unit tests for you. Additionally, it can wire together all of the supporting code to help you get started quickly with unit testing using the MXUnit testing framework.

Installation

1. In the Extensions view, click  on the plus sign  and select the the Apptacular.zip file to install.
2. The Extension Install wizard guides you through the installation.
3. Select the Server. If no server is configured, click Add Server to add a new server.
4. Enter the path to the ColdFusion web root (if the path is not automatically populated when selecting the server). For example,  *C:\ColdFusion9\wwwroot*
5. Select a folder within the web root to install the extension. The archive file is extracted to the install location.

For best results, install in a folder named *apptacular* under your webroot.

6. Click Install.

Use

Creating an Apptacular Application

From a Database

From a Project

On Project Creation

Updating Apptacular Applications

Updating Global Application Settings

Updating Database Items Settings

Apptacular Features

Table name tweaking

Prefixes

Deplurizaztion

UI tweaking

Image Interfaces

Virtual Columns

Using Remote Services with Flex

Getting Started with Unit Testing

Using Inheritance and Super Classes

One of the problems with using scaffolding is that your business model may change after you've customized your files with new functions and properties. Many scaffolding solutions use Inheritance to fix this. The scaffolder will regenerate dynamic files which are in turn extended by static files that are generated once, but then never touched again.

That can be done with services using the Application Configuration *Generate Super Service that are extendable by services that you can edit.*

That's not really possible using ColdFusion ORM Entities. However ColdFusion 9.0.1 introduced the ability to create unpersisted superclasses that are in turn extended by your actual ORM entities. This can be done using the Application Configuration *Generate Super Entities that can be edited but are extended by static entities.*

Configuration Reference

Table Options

Entity Name

A code friendly name of the object usefull when your table is named 'tbl_author_active'

Display Name

A pretty name, not at all like 'tbl_author_active'

Plural Display Name

The pretty name of the plural.

Foreign Key Label

The column to be used as a reference in related object ui.

Order by

The column to order all reference to these objects.

Plural

The plural of the entityname, used in relationships

Should this table get wired up?

Whether or not this table should have interfaces built for it.

Is this table a join table?

Whether or not this table is the join table of a many to many relationship

Configuration Options**Application Basics****Lock Application**

Setting this to true will prevent any new files from being written or modified by the Apptacular extension

Create Application.cfc

Whether or not Apptacular should create a Application.cfc

Create ORM Entities

Whether or not Apptacular should create entity files

Create Views

Whether or not Apptacular should create view files (index.cfm, plus tabname.cfm, and custom tags.)

Create Services

Whether or not Apptacular should create services

Create Login Protection

Whether or not Apptacular should wire up login framework.

OverwriteDataModel

Setting this to true will allow config XML to become authoritative. Don't do this until you are mostly done messing around with model.

Path Information**Root CFC Path**

The CFC format path that corresponds to the rootFilePath

Root File Path

The root directory under which all Apptacular files will be written.

CSS Folder

The subfolder under the root where the css files and accompanying images will be written.

Custom Tag Folder

The subfolder under the root where the custom tags will be written.

ORM Entity Folder

The subfolder under the root where the entity CFCs will be written.

Service Folder

The subfolder under the root where the service CFCs will be written.

Application Folder

The subfolder under the root where the application files will be written.

Service Methods

Get method name

The name of the method in a service that retrieves a single record.

Update method name

The name of the method in a service that updates a single record.

Delete method name

The name of the method in a service that deletes a single record.

List method name

The name of the method in a service that retrieves all records.

List Paged method name

The name of the method in a service that retrieves all records, in pages.

Search method name

The name of the method in a service searches for records.

Search Paged method name

The name of the method in a service searches for records, in pages.

Init method name

The name of the method in a service that acts as the constructor.

Count method name

The name of the method in a service that returns a count of records in the underlying table.

Misc

Service Access

Whether or not your services to be accessible or not. [Options: public, remote]

Return Queries from Services instead of ORM Arrays

Whether or not generated application return queries instead of arrays of ORM objects.

CFCFormat

Whether or not your CFCs are all CFML or all CFScript. [Options: CFML, CFSCRIPT]

Wire One to Many relationships in views

Whether or not Apptacular should write out oneToMany interfaces in view. Good to turn off if you have tables with 1000's of records wired to a oneToMany.

Log SQL

Whether or not generated application should log Hibernate SQL operations to the console.

Depluralize

Whether or not Apptacular should depluralize your entity names. Experimental, may not yeild perfect results.

Generate Super Service that are extendable by services that you can edit.

Whether or not to generate dynamic super services that are extended by a static, editable, base class.

Generate Super Entities that can be edited but are extended by static entities.

Whether or not to generate editable static super services that are extended by a dynamic, non-editable, base class.

Ormsettings DBCreate

How should ORM react to new CFC's, should it create new tables, or ignore them.

Valid options: none, update (apptacular default), dropcreate

Formats

Date Format

The format in which to display dates in views.

Time Format

The format in which to display times in views.

CSS File Name

The name of the CSS file to use. Helpful to replace Apptacular regenerated CSS with your own.

Magic Fields

Created on String

The magic string for DateTime of object creation

Updated on String

The magic string for DateTime of object last update

MXUnit Settings

Create Unit Tests

Whether or not Apptacular should write default unit tests.

MXUnit Framework File Path

The directory where the MXUnit files will be found.

Unit Test Folder

The subfolder under the root where the test cases will be written.

Virtual Column Options

name

The code name of the property to represent this column.

displayName

A pretty name, not at all like 'varchar_author_is_active'

getterCode

The code that goes in the getVirtualColumn method.

Type

The ColdFusion datatype

uiType

The type to generate ui's for.

Datasource Options

Display Name

A pretty name, not at all like 'db_blog1_mysql'

Prefix

An prefix on all of the tables in the database.

Column Options

name

The code name of the property to represent this column.

column

The real name of the column in the database.

displayName

A pretty name, not at all like 'varchar_author_is_active'

uiType

The type to generate ui's for.

includeInEntity

Whether or not to include this column in an entity