

# AppPress Developer Guide

*This document is in Active Development and being changed frequently.*

Copyright 2015 SysMates Technologies Pte. Ltd.  
www.sysmates.com

## Online Resources

## Support

For Support: [support@AppPress.tech](mailto:support@AppPress.tech)

Forum: [AppPress Forum](#)

## Downloads

[AppPress Solution Template](#) 22nd Jan 2016.

[AppPress Binary 0.9.3.0](#) 22nd Jan 2016

Required: MySQL 6.9.6+ Net Connector

Copy all Files in Zip to Dependency Folder of Application

[Release Notes](#)

<http://apppress.tech/AppPress/AppPress-0.9.3.0.zip>

[About AppPress](#)

[AppPress Technology](#)

[AppPress Getting Started](#)

[Prerequisite](#)

[Download](#)

[Create Database](#)

[Edit web.config](#)

[Edit AppInit.cs](#)

[Compile and Run](#)

[Change Logos Names etc](#)

[Remove Demo Stuff](#)

[Remove Plug-in](#)

[Rename AppPress.aspx](#)

[Add a new Menu Item](#)

[Debugging in AppPress](#)

[Database in AppPress](#)

[AppPress Query Execution Functions](#)

[AppPress Layer](#)

[DAOBasic Layer](#)

[Calling Stored Procedures from Any Layer](#)

[AppPress Form Save Functions](#)

[Caching in AppPress](#)

[Audit in AppPress](#)

[Reusable Forms](#)

[MergedForm](#)

[EmbeddedForm](#)

[Plugin Forms](#)

[Remote Forms](#)

[RemotePopup](#)

[RemoteRedirect](#)

[Using Skins in AppPress](#)

[How Skins work in AppPress](#)

[Create Skin](#)

[Modifying Skin](#)

[Using Scripting in Skin](#)

[How Refresh works in Skin](#)

[How Hide Show works in Skin](#)

[Using Skins from Plugins](#)

[AppPress Reference](#)

[Form XML](#)

[Name](#)

[MasterFormName\\*](#)

[NonSecure](#)

[TableName](#)

[Fields](#)

[Form Code Behind](#)

[Init](#)

[BeforeSave](#)

[AfterSave](#)

[Field XML](#)

[Name](#)

[Type](#)

[Required](#)

[Static](#)

[DoNotSaveInDB](#)

[Encryption](#)

[Shortcut](#)

[ContainerStyle](#)

[<LabelStyle><!\[CDATA\[float:left\]\]></LabelStyle>](#)

[LabelStyle](#)

[ControlStyle](#)

[PartStyle](#)

[Code Behind of Field](#)

[Event](#)

[Properties](#)

[Methods](#)

[Text, TextArea Field](#)

[Password Field](#)

[Number Field](#)

[Datetime Field](#)

- [DateRange Field](#)
- [FileUpload Field](#)
- [MultiFileUpload Field](#)
- [HTML Field](#)
- [Button Field](#)
- [Redirect Field](#)
- [Pickone Field](#)
- [PickMultiple Field](#)
- [Checkbox Field](#)
- [FormContainerDynamic Field](#)
- [FormContainerGrid Field](#)
- [SelectRow Field](#)
- [ForeignKey Field](#)

## [AppPress API](#)

### [Overview](#)

#### [AppPress](#)

- [Alert\(string message\)](#)
- [PromptClient](#)
- [ExecuteJSScript](#)
- [SetPageDirty](#)
- [RefreshPage](#)
- [ClosePopup](#)
- [ReadFormData](#)
- [RedirectToUrl](#)
- [GetSecureUrl](#)

#### [URL Options](#)

#### [Settings](#)

## [Transforming TT Files](#)

## [Frequently Asked Questions](#)

- [How to create a Dynamic Menu](#)
- [How to Invoke click of Button from JavaScript](#)
- [How to hide a Column in FormContainerGrid](#)
- [How to Create a Scalar User Control](#)
- [How to Create a Merge Form](#)
- [How to Create a Plugin Form](#)

[How to Create an Embedded Form](#)

[How to Add Forms Dynamically](#)

[After some changes in Form XML I am getting too many errors](#)

[How to override functionality of Add, Modify, Delete and Save buttons generated in FormContainerGrid](#)

[How to Send Email using AppPress](#)

[How to Prompt User for confirmation while executing Code Behind](#)

[How to hide Sidebar Menu on Page load](#)

[How to move FormContainer Caption Buttons to Left on entire Page](#)

[How to set column width in Grid](#)

[How to show button as icon](#)

[Using Field Groups](#)

[How to change height width or other attributes of text fields](#)

[How to organize checkbox options for PickMultiple as a Grid](#)

[How to change font size for a grid](#)

[How to add reordering functionality to a Grid](#)

[Best Practices](#)

[Using LoginUserId from SessionData versus using FormDataId](#)

[Recommended Settings for AppPress based application in Production for IIS](#)

[Appendix 1](#)

[Formats of Shortcut Key](#)

[Appendix 2](#)

[Release Notes](#)

[Appendix 3](#)

# About AppPress

AppPress is a .net based framework suitable for developing Business Applications like Admin, Workflows, Reports etc. Starting with User defined forms AppPress creates the **Application Architecture** where backend engineers add business logic to complete the functionality. Additionally Web Designers can edit the generated skins for improved UI wherever required.

Key benefits of using AppPress are

- Rapid Prototyping of application functionality
- Factory like application development with distinct phases
  - Prototyping
  - Database development
  - Business Logic
  - UI Customization
- Better resource utilization with skilled programmers working mostly on business logic
- Single Enforced Development Pattern for creating UI and related database with Framework creating Application Architecture
- Agile development with shorter cycles
- Much reduced code size compared to ASP or MVC
- Eliminate need to do client side programming
- UI names are used in middle layer and Database, reducing need for creating intermediate names
- Business logic is added in AppPress predefined initialization and event functions
- Rich library of functions to manage UI from backend

# AppPress Technology

**AppPress** is a Form based web development framework, designed for rapid prototyping and development of functionality rich business applications and workflows. Further AppPress allows to build Applications which can be customized using Plug-ins for individual installations.

AppPress defines a Page as a tree of forms. Each form being a set of fields. A Page can be a Browser Page or a Popup Page. Forms can be nested and aggregated to any level with FormContainer fields. Forms are defined in schema based XML.

At Build time Form XML is augmented with Database Properties like nullable, size and Foreign Keys. MergeForm and UserControlScalar forms are merged and the final form list transformed to C# Class Definitions and HTML Page Skins. The Classes are nested in same way as in the Form XML and have backward and forward pointers to allow navigation within the data tree. The Skin contains Markers in the structure of Form XML

At Application Initialization Dynamic Forms defined in Code are added to Form list. Dynamic forms can only use AppPress builtin logic.

At Page load **Form Data Tree** is created for the page. Starting with root form the form data is initialized using the binding specified in the form definition and any backend initialization functions defined in the Business Logic. The Page Skin is combined with Form Data Tree to generate the initial HTML Page. User actions on the page invoke backend predefined event functions for the field. The event functions receive the Form Data Tree pivoted by pointer to the action field. The event functions use a set of predefined functions to manipulate the Page. The manipulations can be Show Hide, Enable Disable, Refresh, Redirect, Popup, Alert Prompt etc. Further AppPress also provides functions to Save Page or Form to database.

Table below illustrates Relationship between AppPress components. **TDB complete the XML and related.**

XML	Classes	Skin	FormData	Database
<code>&lt;Form Name="Employee"&gt; ... &lt;/Form&gt;</code>	<code>class EmployeeClass { } }</code>	<code>&lt;!-- AppPress.Employee.Begin --&gt; &lt;!-- AppPress.Employee.End --&gt;</code>	<code>EmployeeClass Employee.val</code>	<code>Employee Id,</code>

To reduce the development effort, AppPress senses the if backend logic for initialization of a field uses value of other fields in the page, the field is auto refreshed when the values of source fields are modified by the user. This is similar to the way formulas in spreadsheet work.

AppPress defines some high level fields which use the core fields and logic to create commonly used UI elements like Nestable Grid with Add/Modify/ Delete, Date Range, FormContainer with single form of different type, embedded form etc.

## TDB Examples of inbuilt Composite Controls. FormContainerGrid, Date Range, FormContainer with single form of different type

Developers can use the generated skin and edit it like HTML to create custom Skins.

AppPress has a inbuilt security layer which uses application logic for verification. Additionally fields on UI have a built in mechanism to check for HTML injection and manipulation.

AppPress streamlines the development process by segregating UX, UI and business logic development. Using AppPress for core functionality development client side programming is not needed as that is handled by AppPress runtime.

AppPress allows creation of User Defined Controls. The Controls can have there own logic.

Another significant advantage of AppPress is in building multi deployment applications, Here AppPress provides build in plugin architecture by which you can extend, reduce functionality of core application based on deployment requirement. The plugin is capable of horizontally (insert/append/remove columns of grids) and vertically (insert/append/remove fields in forms) modifying the core UI allowing easier change management and user training. The UI additions from plugin can have their business logic.

AppPress allows sharing of Forms across AppPress instances within a defined group. This allows workflows to extend to other AppPress instances.

Some key technologies which enable this

- **UI Definition:** XML Based UI definition.  
UI has simple fields and complex fields like Form Containers.  
FormContainer hold instances of single types of forms or multiple types of forms.  
Forms in FormContainer can further have FormContainer Fields allowing nesting to any depth  
This maps well to common XML Data definition.
- **Class Generation:** UI Definition is transformed to C# classes
- **UI Generation:** AppPress uses the Form Definition to generate UI skins on application load. UI skin contain markers for form and field definitions. On Page load UI skin is Compiled with Form instances (FormData) to generate the actual UI.
- **UI Refresh:** For part refresh the part to be refreshed is extracted from UI skin and compiled with corresponding FormDatas and the resulting HTML is injected into the UI.
- **Backend Logic** is build with few reserved method names (Init, Calc, Options, OnClick and OnChange) with signature of field class name as method identifier. This removes the complexity of giving suitable unique names to methods. This standardizes the code and lets



programmers concentrate on business logic than finding good names for methods. Fields has build in properties to allow common actions like validation, hiding, readonly, title and coloring.

- **Spreadsheet like programming** where by can attach backen calculators functions to fields. Calculators calculate values of simple fields and also calculate the forms to be shown in FormContainer. The Calculators can use values of other fields in page. If value of used field is changed the field is auto refreshed.
- This can be used with little effort in interesting ways to do things like
  - Country State linked dropdowns where options in State changes based on country
  - A grid which shows data based in text search field and refreshed automatically
  - A Gender Pickone field controlling the type of form shown in a FormContainer

Spreadsheet functionality in a programming environment.

- **Client side to backend data transfer:** On Action call like OnClick all PageData on UI is smartly submitted to backend in form of navigable tree. So backend functions always have full data of UI. This removes Client side programming completely for functionality implementation.
- **UI Management:** Backend logic can prompt user for Yes No like decisions to continue with a action. For example Delete a Record confirmation prompt is triggered from C# code. In AppPress the prompt can be more intelligent as while prompting all UI and database is available.

The backend logic can trigger a refresh of parts of UI in response to action call like onclick from UI.

The skin used for initial page load is also used for Refresh part of page. This reduces effort and standardises the application code.

AppPress can show a Form on Page or in a Popup

AppPress maintains page/popup Un-Saved Status on UI and prompts user when user tries to navigate away from unsaved page.

- **Database:** XML UI Definition Maps to Database Tables and column names

Functionality provided to Save Forms to database and retrieve forms from Database. Database Transactions have been enhanced in AppPress to roll back middle layer changes along with database changes.

Build in Audit feature logs all AppPress performed changes to database in Audit log table. Interface is also provided for users to log audit entries for database changes done directly in backend logic.

- **Navigation:** Backend logic to perform redirect to forms and show forms in popup
- **Security:** Inbuilt layer of security, whereby all elements on UI are secured with SHA Encryption to disallow html/JS tampering. AppPress restricts users from opening a unauthorized URL by checking it against previous redirection to that URL from Backend
- **Plugin based extensibility:** In Core product define Extension field type in XML UI Definition. In plugin assembly define Form with same name as Extension Field. At runtime AppPress will inject the Extension form fields in the Core Form. Similarly the skin from plugin assembly will be injected into the Core Form skin. The plugin can add its own business logic to the fields it added. This allows for plugin to extend existing forms or grids in the core product.



# AppPress Getting Started

## Prerequisite

Visual Studio 2013 onwards  
MS SQL Server

## Download

AppPress Template: Use link from the Top page.

Unzip to any temporary folder.

Zip the files again and unzip to the Project Folder. *This step is needed to unblock all the files in the folder*

## Create Database

Restore Database using AppPressApplication.bak in `Database` folder on MS SQL Server. This will create the Database AppPressApplication along with tables required for AppPress and Tables needed for AppPress Demo

*Note:*

- *If you have existing database then Copy Application\_\* tables to existing database*
- *You can change the name of Database as needed or your Application*

## Edit web.config

Open web.config and setup connectionStrings and mailSettings sections.

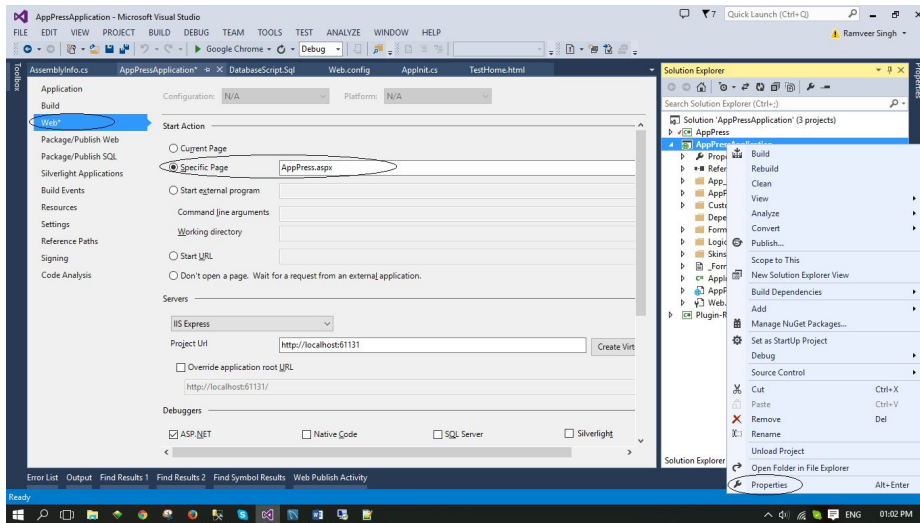
## Edit AppInit.cs

Open AppInit.cs file

Look for `Settings.databaseType` and change as per comments. Normally you need to change only till `Settings.DebugEmail`

## Compile and Run

Update *Web Properties* and set *Specific Page* as *AppPress.aspx*.



Right click on `_Forms.tt` and choose option `Run Custom Tool`. This will build `_Forms.cs` file from xmls in Forms folder  
Rebuild solution and Run. You will see following



## Change Logos Names etc

Change AppPress logo and small logo by replacing files `Logo.png` and `Logo_Small.png`

To change name of Application search for `Application Name Here` in `Skins` folder. Replace with name of your Application.

Change Footer. Search for `Footer Message Here` and Change.

Remove the Login User Name and Password by searching for the message in `Login.html`

## Remove Demo Stuff

You can keep the demo menu around in initial phase of your development. The Demo Menu provides all type of Controls and their usage along with logic functions. Once you are ready you can remove by using following steps

1. In Database remove all tables starting with AppPressDemo\_
2. Remove Following Files
  - a. Logic\\_AppPressDemoLogic.cs
  - b. Forms\\_AppPressDemo.xml
3. In file Skins\ApplicationMaster.html  
Search for <!--Demo Skin Start--> and remove till <!--Demo Skin End-->

Rebuild TT and Run application

## Remove Plug-in

If you are not going to use plugins in your application

1. Remove the Plugin-Reporting Project from solution
2. In AppInit.cs remove following code

```
Settings.pluginAssemblyNames = // List of plugins for this application
    new List<string> { "Plugin-Reporting.dll" };
```

## Rename AppPress.aspx

Just change the name of AppPress.aspx in solution to what you want.

## Add a new Menu Item

We will add a new Menu Item “EmployeeManagement” under Admin Menus

Open Forms\Masters.xml and Add a new Field in AdminMenusExtension MergedForm

```
<MergedForm Name="AdminMenusExtension">
  <Fields>
    <Button Name="UserManagement">
```

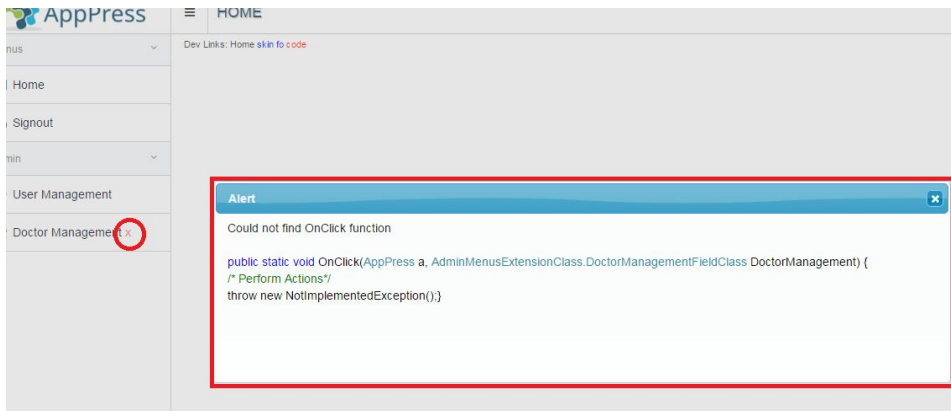
```

        <NoSubmit/>
    </Button>
    <Button Name="EmployeeManagement">
        <NoSubmit/>
    </Button>
</Fields>
</MergedForm>

```

Copy Forms\UserManagement.xml and rename it to EmployeeManagement.xml. Delete all <Form>..</Form> from this file. Copy Logic\UserManagement.cs and rename it to EmployeeManagement.cs. Delete all functions from this file. Run Custom Tool on \_Forms.TT and run the application.

You will see Menu item Employee Management with a red cross mark on side. The red cross means that logic to handle the click on menu item is missing. Click on Red cross and copy the code from Alert.

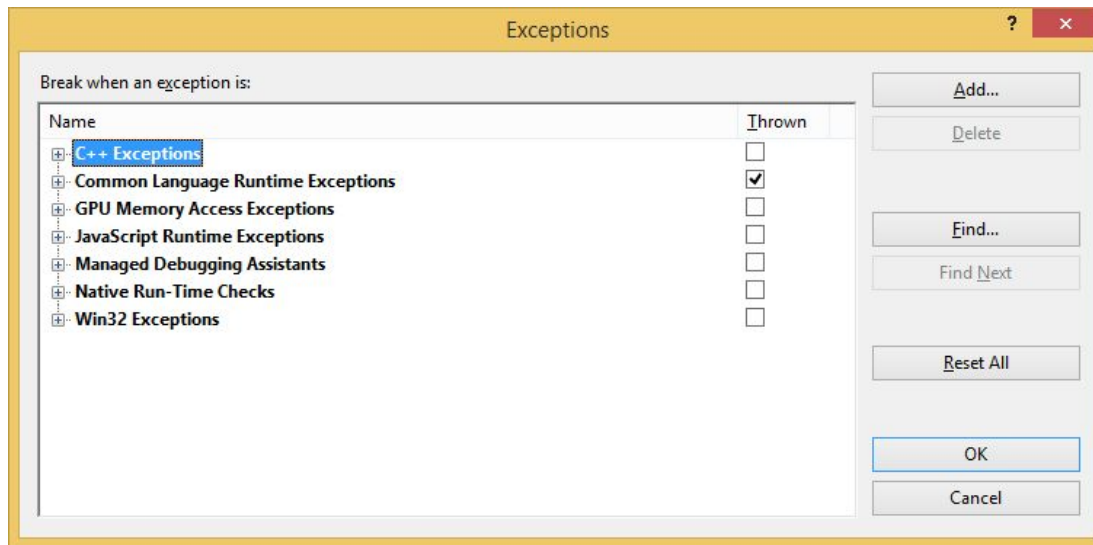


Paste the code in EmployeeManagement.cs and change it as following

# Debugging in AppPress

Debugging in AppPress is same as you would debug any .net C# application in Visual Studio. Some points to be aware of

AppPress internally uses some Exception types for control flow. It's recommended that in normal course of development you keep Break When Exception is thrown off. To do that open Debug->Exceptions and turn off Common Language Runtime Exceptions. When you encounter an Error Message in Application and you need to catch that Exception then turn the option on.



When FormContainer, Pickone and PickMultiple fields are bound using SQL Query, The query is executed in AppPress code. AppPress will throw a AlertMessage for any SQL Exception generated from Query Execution. The message includes the Domain or Options function which returned the Query, The SQL Exception message and the Query which generated the error.



# Database in AppPress

One of the main features of AppPress is ability to bind *Forms* with Database

If a Form is bound to a table following are done at time of Application Initialization

- If table has Primary Key the name of primary is Mapped to Id field of Application Class
- If field in form does not have Required Validation set it is set if the column in table does not allow Null Values
- if a Text or TextArea field in form does not have MaxChars set and field is not encrypted then MaxChars of the field is set using the max length of the column in the table
- For Pickone and PickMultiple fields if no Options function is defined, the options are loaded from the Table referenced in foreignkey from tableName of the Form and FieldName of the Field.  
If the referenced table is modified using AppPress functions, the options are loaded again.
- For FormContainer Fields if the RowFields have field of type ForeignKey and no Domain Function is Defined, then AppPress gets forms for the FormContainer using the ForeignKey column.
- At Application Initialization AppPress throws an Exception if Tables used by AppPress (starting with Application\_) are structurally incorrect.

## AppPress Query Execution Functions

### AppPress Layer

When AppPress object is constructed a default Database connected is created using the name of Connection String in web.config specified in AppInit.

This layer provides following functionality

- DML statement like (Insert, Update, Delete) should be executed using ExecuteNonQuery. Exception will be thrown if not executed within a Transaction.

Example:

```
try
{
    a.BeginTrans();
    a.ExecuteNonQuery("Update Users Set FirstName = 'Test' Where Id = 1");
}
```

```

        a.CommitTrans();
    }
    catch
    {
        a.RollbackTrans();
    }
}

```

- Rollback will rollback the database transaction along with any changes made to Id field of ApplicationClasses object. So if a new form was saved, which generated a Id for the form, in rollback the id will be reverted to original id.
- bool DBTableExists(string tableName)
- bool DBRoutineExists(string routineName)

## DAOBasic Layer

This is lower layer of Database execution function matching closely with C# database functions.

## Calling Stored Procedures from Any Layer

```
a.ExecuteScaler("CALL <SPName>(<Arguments>)");
```

Example:

```
a.ExecuteScaler("CALL CreateUser('FirstName', 'Lastname')");
```

If there is a result expected from Stored Procedure, then result are returned in IDataReader.

Example

```
IDataReader dr = p.ExecuteQuery("CALL SearchUser('Test')");
```

## AppPress Form Save Functions

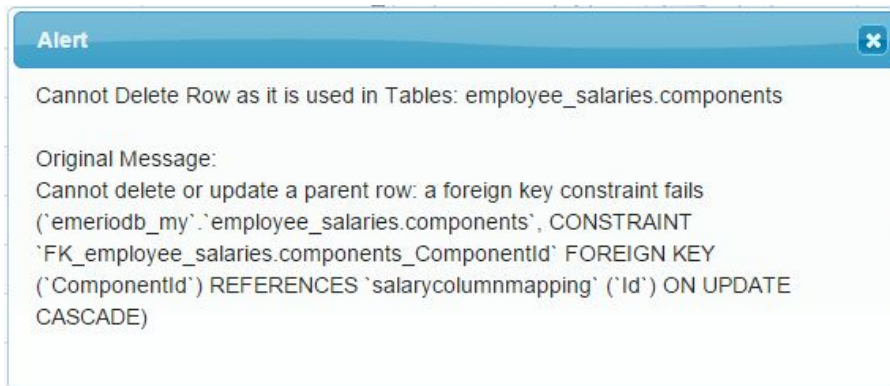
```
<ClassName>.Save();
```

Saves the form along with all the nested forms. Application\_Audit table is updated for insert, modify and delete.

### User Friendly Message for Database Exceptions

AppPress changes Message for following exceptions to make it more user friendly. This way in coding you do not need to check or the condition and provide your custom message

- Insert null in column not allowing null value  
Show "Required Error"
- Duplicate value in Unique Columns  
Shows "Duplicate: Column Names"
- Delete Row in FormContainer Grid. ForeignKey constraint error



## Encryption of Data

Index.aspx?EncryptionType=DES&EncryptTableColumn=employee:email|employee:personalEmail

## Caching in AppPress

### Server side Caching

Pickone and PickMultiple options are cached in Application launch if

- Options are generated using ForeignKey

- ReCaching is done if the option table is updated using AppPress
- If you are modifying the Options table directly then you will need to restart the Application for updated options to reflect on UI
- If Pickone style is Dropdown the cache is done in JS. So that the options values are not transmitted to client on every page load. The JS cache is ReChached when option table is modified using AppPress.
- If the options table is modified outside AppPress you will have to restart AppPress.

## Client side Caching

Pickone options are cached using AppPressCache.js?t=<number> on client if

- Options are generated using ForeignKey and type is Pickone
- The JS cache is ReChached when option table is modified using AppPress. This is achieved by modifying the <number> in . AppPressCache.js reference on the page.
- If the options table is modified outside AppPress you will have to restart AppPress.

# Audit in AppPress

Changes made by AppPress to Database are saved into Application\_Audit table.

Id	AuditType	LoginUserId	UserName	Page	PageId	TableName	RowId	Change	Time
51135	100	1	admin@email.com	Controls		AppPressDemo_Manager s	11	Name: Test Manager; DateOfBirth: 02-Dec-1996; Address: 301, Sapphire Court, Essel Towers, M G Road; Country: India; State: Haryana; City: Gurgaon; PinCode: 122001; Responsibilites: Administrator   HR;	2015-12-14 15:57:14
51139	102	4	admin@email.com	Controls		AppPressDemo_Manager s	11	Responsibilites: Administrator   HR => n/a;	2015-12-14 16:19:02
51140	101	4	admin@email.com	Controls		AppPressDemo_Manager s	11	Responsibilites: ; Name: Test Manager; DateOfBirth: 02-Dec-1996; Address: 301, Sapphire Court, Essel Towers, M G Road; Country: India; State: Haryana; City: Gurgaon; PinCode: 122001; InActive: Active;	2015-12-14 16:19:53

Id: Primary key of Application\_Audit table

AuditType:

Id	AuditType
1	Login
2	Logout
3	UserDefined
100	InsertRow
101	DeleteRow
102	UpdateRow
103	View

LoginUserId: Id of user who is logged in

UserName: Name of Login User

Page: Form where the change happened

PageId: Id of the form where the change happened

TableName: Table which was modified

RowId: Primary Key value of the row which was modified

Change: Description of Change

Time: Time of change in UTC

## API

```
public void SaveDBAudit(string TableName, int TableRowId, string FormName, string PageId, string Change)
```

Add a Audit Entry to Application\_Audit table. If you are changing database in Business Logic, use this to add audit entry for the change

## Reusable Forms

### MergedForm

ReplaceFieldNameInMergedForm

**EmbeddedForm**

# Plugin Forms

If the containing form of plugin is created directly then add a new row to Embedded form table and use that id in the column of containing form table.



# Remote Forms

**RemotePopup**

**RemoteRedirect**

# Using Skins in AppPress

## How Skins work in AppPress

### Create Skin

Get Skin From DevLinks

Get Skin From Url

<http://localhost:52801/AppPress.aspx?GetSkin=<Form Definition Id>>

To get form definition Id of a form look at the class generated for the form

### Modifying Skin

### Using Scripting in Skin

ScriptBegin and ScriptEnd markers

### How Refresh works in Skin

### How Hide Show works in Skin

If a field is Hidden AppPress will replace content between Begin and End markers of the field in following way

if the field skin has HiddenBegin and HiddenEnd markers the content between these markers will be removed.

```
<!--|AppPress.HiddenBegin|--><!--|AppPress.HiddenEnd|-->
```

If the field is a column in grid the content will be replaced with `<td id='AppPressId'></td>`

otherwise content is replaced with `<span id='AppPressId'></span>`

### Using Skins from Plugins



# AppPress Reference

AppPress application UI and events are defined by a collection of Forms. The forms are defined in .xml files in Forms folder.

Each form has a list of fields.

By convention each xml file contains forms for a functionality. For example forms for Login Functionality can be added to Login.xml.

\_Forms.tt [TextTransformation](#) file converts xml to C# classes.

Each form is converted to a class with name as Name of Form appended by Class and each field is converted to a class with name of field appended by FieldClass. The fields are generated within the form class.

FormContainer field types are collection of forms within a form. FormContainerGrid and FormContainerDynamic types implicitly generate forms with names FieldName+RowClass and FieldName+PopupClass. RowClass form defines the form to be used to show a row in the FieldContainer and PopupClass form is used to show a Popup to allow addition and Modification of forms in the FormContainer

Following documents describes the XML format for AppPress Forms, **Nodes** and **XML Attributes**. Fields generated in Classes **Class Fields**, what Action Functions **Code Events** can be written for the field in Backend logic and what methods can be called from the Field **Code Methods**

## Form XML

FormDef node defines a Form. FormDef has Name, Properties and list of fields.

```
<Form Name="Home">
  <MasterFormName>ApplicationMaster</MasterFormName>
  <TableName>ApplicationMaster</TableName>
  <Fields />
</Form>
```

### Name

Name of the form. Name should be unique across the application.

### MasterFormName\*

```
<MasterFormName>ApplicationMaster</MasterFormName>
```

Name of the Master Form.

Master form contains template including Header, Menus , Footer and Content Area. Form will be shown in Content Area.

The screenshot displays the AppPress application interface. On the left is a sidebar menu with options: Home, Signout, Admin (selected), User Management, AppPressDemo, Controls, Control Validations, Control Events, Form Containers, Composit Controls, Skins, and Examples. The main content area is titled 'APP PRESS FORM CONTAINER GRIDS' and 'Header'. It contains two tables. The first table, 'FORM CONTAINER GRID', has columns: Name, Date Of Birth, Address, Country, State, City, and Pin Code. It lists five entries. The second table, 'CLIENT ACCOUNTS', has columns: Id, Name, Address, Country, Phone, Email, and Licenses. It lists one entry. Below the tables is an 'Employee Status' dropdown and a 'Save' button. The footer contains 'Footer Message here.' and 'Version 2.2.0'.

Name	Date Of Birth	Address	Country	State	City	Pin Code
r ddsdf sd sdsdf	09-Sep-2015	retdf sdsdf fsdf sdsdf	India	Andhra Pradesh	Hyderabad	110044kkj
aseqe	01-Sep-2015	weqwe	India	Haryana	Gurgaon	weqw
qweqr	09-Sep-2015	qweqwe	India	Haryana	Gurgaon	eqwqw
qwe	01-Sep-2015	qewqwe	China	Hebei	Langfang	qweqw
DGGD	08-Sep-2015	FGD	Brazil	Rio de Janeiro	São Gonçalo	w

Id	Name	Address	Country	Phone	Email	Licenses										
1	sdsdfdsf	sdsdf	Brazil	dsfsdf	sdsdf	<table border="1"><thead><tr><th>Id</th><th>Name</th><th>Purchase Date</th><th>Expiry Date</th><th>Payment</th></tr></thead><tbody><tr><td>1</td><td>ddas</td><td>03-Sep-2015</td><td>02-Sep-2015</td><td>0</td></tr></tbody></table>	Id	Name	Purchase Date	Expiry Date	Payment	1	ddas	03-Sep-2015	02-Sep-2015	0
Id	Name	Purchase Date	Expiry Date	Payment												
1	ddas	03-Sep-2015	02-Sep-2015	0												

By Default AppPress comes with following MasterForms

ApplicationMaster: This Master is used form forms which needs a menu like Home Page.

BlankMaster: This Master is used for the form do not need a menu like Login.

NonSecure

```
<NonSecure />
```

This is used to open the Form without login. For example Login Form.

TableName

```
<TableName>Application_Users</TableName>
```

Name of Database table associated with the Form. The table must have a primary key with column name as Id.

## Fields

```
<Fields [Group="GroupName"]>
  <FileUpload Name="FileUpload">...</FileUpload>
  <Button Name="LoadFile">...</Button>
</Fields>
```

Contains list of Fields in the form You can create multiple Fields node in the Form and optionally assign GroupName in the Group attribute. The Group is generated as Enum Values in code. GroupShow, GroupHide and GroupReadonly functions use the Enum Values to Hide or make Readonly all fields in group.

## Form Code Behind

### Init

```
public static void Init(PageData p, HomeClass Home)
{
}
```

### BeforeSave

```
public static void BeforeSave(PageData p, HomeClass Home)
{
}
```

### AfterSave

```
public static void AfterSave(PageData p, HomeClass Home)
{
}
```

### Note:

Any FieldError, FormError or AlertMessage generated during BeforeSave and AfterSave cause the Save to be aborted.

When XML is compiled, a class is generated with the name <formName>Class.

Init function having the class name as second parameter is called before page load in browser. In this function you can change value and properties (Hidden, Readonly etc) of fields.

## Field XML

```
<Text Name="Text">  
  <Required />  
  <DoNotSaveInDB />  
  <Static />  
  <Shortcut>Ctrl+B</Shortcut>  
</Text>
```

### Name

If table is associated with the form containing this field, Use table column name as Name to associate with field.

### Type

Following field types are available

Checkbox, Text, TextArea, Date .... TBD

### Required

```
<Required/>
```

In UI field is marked as Required and user will need to enter a valid value. if this node is not present and the field is bound to a data column in database and data column is set to not null, the Required property will set to true

### Static

```
<Static RetainValue="true"/>
```

Static fields are shown as label and cannot be edited. By default Static field value is not send to code behind unless attribute RetainValue is set to true.

## DoNotSaveInDB

<DoNotSaveInDB/>

If table is associated with the form and this field is not available as column in table, this node is used to avoid AppPress error while saving the form.

## Encryption

<Encryption Type="AES"></Encryption>

Column type should be Varchar in database table.

Key is stored in Web.config. Key on Production Environment and Development Environment should be different.

In Debug mode, Decryption with wrong key will return junk value instead of exception. This allows use of production database in development.

Helper function are available to Encrypt / Decrypt values for use in Logic functions.

Applicable to:

Text, TextArea and Number, DateTime, DateRange, FileUpload, MultiFileUpload.

Following type are available -

AES: Encrypted value will be different for same source value on each encryption.

DES: Encrypted value will be same every time on each encryption.

## Shortcut

```
<Shortcut>Enter</Shortcut>
```

Applicable to:

Triggers OnChange Event : Text, TextArea, Number, DateTime, Pickone, PickMultiple.

Triggers OnClick Event: Button.

Formats of Shortcut Key: See [Appendix 1](#)



## ContainerStyle

```
<LabelStyle><![CDATA[float:left]]></LabelStyle>
```

## LabelStyle

```
<LabelStyle><![CDATA[float:left]]></LabelStyle>
```

## ControlStyle

```
<LabelStyle><![CDATA[float:left]]></LabelStyle>
```

## PartStyle

```
<LabelStyle><![CDATA[float:left]]></LabelStyle>
```

## Code Behind of Field

On XML compilation, a class is generated for every field informat of <fieldName>FieldClass. This class is generated nested in Form class.

### Event

```
public static void Calc(PageData p, AppPressControlsClass.NumberFieldClass Number) {}  
public static void OnChange(PageData p, AppPressControlsClass.FileUploadFieldClass FileUpload) {}  
public static void OnClick(PageData p, AppPressControlsClass.SubmitFieldClass Submit) {}  
BeforeDelete ???
```

## Properties

Hidden, Readonly, Title, FormData, IsStatic

### Hidden

Field is hidden on UI. AppPress will still call the Initialization functions Init, Domain, Calc for hidden fields. For performance reasons you can check the Hidden Property in the Initialization function and return.

## Methods

Refresh()

# Text, TextArea Field

## Description

Text Field in which user can enter single line of text.

TextArea Field where user can enter multiple lines of text.

## XML

```
<Text Name="FirstName">
    <EmailValidation/>
    <RegexValidation><![CDATA[^[a-zA-Z0-9]*$]]></RegexValidation>
    <MaxChars>100</MaxChars>
    <Style>TitleCase</Style>
</Text>

<TextArea Name="Comment">
    <MaxChars>100</MaxChars>
    <Style>RichTextCKEditorBasic</Style>
</TextArea>
```

### EmailValidation

Validates the content of text to be valid email.

OnBlur trims the content.

If no Case style is provided OnBlur changes content to lower case

### RegexValidation

Validates the content of text to be match the specified Regex expression.

### MaxChars

Maximum number of characters allowed in the field.

If this is not specified and the field binds to a column in TableName of form and the field is not Encrypted, then the value will be taken from size of the field in Database.

Style: UpperCase, LowerCase, TitleCase

OnBlur changes the text to UPPER, lower or Title case

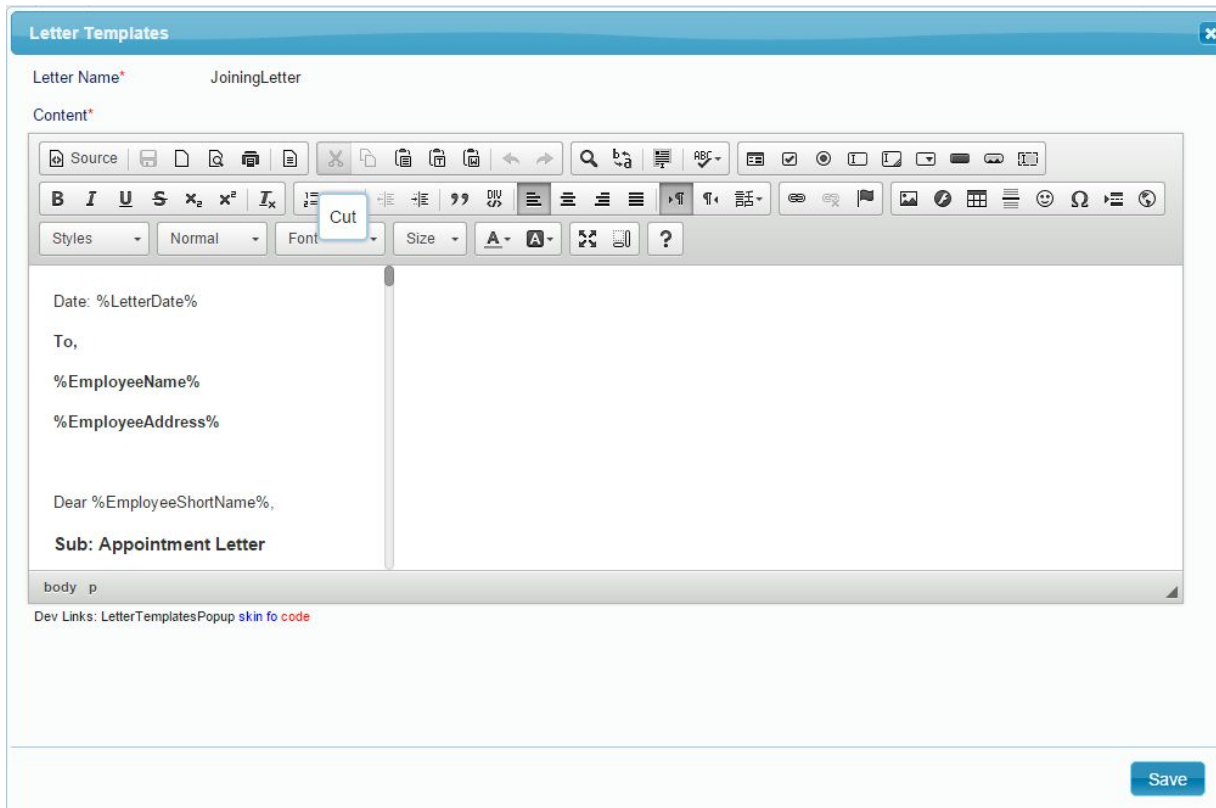
Style: RichTextCKEditorBasic, RichTextCKEditorStandard, RichTextCKEditorFull

Style allows editing and formatting of text. Look at CKEditor.com

RichTextCKEditorBasic

The screenshot shows a 'Letter Templates' dialog box with a blue header bar containing a close button. The dialog is divided into two main sections: 'Letter Name\*' and 'Content\*'. The 'Letter Name\*' field contains the text 'JoiningLetter'. The 'Content\*' section features a rich text editor with a toolbar containing icons for undo, redo, bold, italic, strikethrough, link, unlink, list, and other formatting options. The editor area contains the following text: 'Welcome to SysMates', 'Authorized Signatory', '%AuthorizedSignatory%', and 'Date: %LetterDate%'. At the bottom of the dialog, there is a 'Save' button and a small text link that reads 'Dev Links: LetterTemplatesPopup skin fo code'.

Advanced



## Other Properties

Required, DoNotSaveInDB, Shortcut, Encryption, LabelStyle, ControlStyle, Label, Changeable, CSSClass, Hidden

## Code Generation

```
public class FirstNameFieldClass : FieldValue
{
    public string val { get { ... } set { ... } }
    // members fieldDefId, FormData, Hidden, Readonly as described in Field Code Generation
}
```

```
public class CommentFieldClass : FieldValue
{
    public string val { get { ... } set { ... } }
    // members fieldDefId, FormData, Hidden, Readonly as described in Field Code Generation
}
```

```
}
```

val of type `string`

## Initialization Functions


`Init (of Containing Form), Calc`

## Events

`OnChange`

## Example

Look at AppPressApplication -> Control -> Text, Text Area  
Type in field and press Tab will show a Alert



≡

CONTROLS

Menus

Home

Signout

Admin

User Management

AppPressDemo

Controls

Validations

Properties

Actions

Auto Refresh

Text Controls

Text

Number\*

Text Area

Selection Controls

Checkbox

Pickone\*

India

Brazil

China

France

USA

Pickone Radio Style

Pickone Auto Complete\*

Pick Multiple

India

Brazil

## Example XML

```

<Form Name="Controls">
  <MasterFormName>ApplicationMaster</MasterFormName>
  <Fields>
    ...
    <Text Name="Text"></Text>
    ...
    <TextArea Name="TextArea"></TextArea>
    ...
  </Fields>

</Fields>
</Form>

```

## Code Generation

```
public class ControlsClass : FormData
{
    ...
    public class TextFieldClass : FieldValue
    {
        public string val { get { ... } set { ... } }
        public new FieldHiddenType Hidden { get { return base.Hidden; } set { base.Hidden = value; } }
        public new ControlsClass FormData {...}
        public TextFieldClass() : base() { }
        public TextFieldClass(FieldValue fieldValue) : base(fieldValue) { }
        public const long fieldDefId = -596688720000190;
    }
    public TextFieldClass Text { get { ...} }
    ...
}
```

## OnChange Logic

```
public static void OnChange(AppPress a, ControlsClass.TextFieldClass Text)
{
    a.AlertMessage("Text OnChange Value: " + string.Join(",", Text.val));
}
```



# Password Field

## Description

Field in which user can enter password.

The Encryption is managed by the Application Logic. AppPress treats Password field as DoNotSaveInDB and does not save it when form containing the field is saved.

## XML

```
<Password Name="Password">
    <MaxChars>100</MaxChars>
</Text>
```

MaxChars

Maximum number of characters allowed in the field.

## Other Properties

Required, LabelStyle, ControlStyle, Label, Changeable, CSSClass, Hidden

## Code Generation

```
public class FirstNameFieldClass : FieldValue
{
    public string val { get { ... } set { ... } }
    // members fieldDefId, FormData, Hidden, Readonly as described in Field Code Generation
}
```

```
public class CommentFieldClass : FieldValue
{
    public string val { get { ... } set { ... } }
    // members fieldDefId, FormData, Hidden, Readonly as described in Field Code Generation
}
```

val of type `string`

## Initialization Functions

Init (of Containing Form), Calc

## Example

Look at AppPressApplication -> UserManagement-> Change Password

## Example XML

```
<Form Name="ChangePassword">
  <MasterFormName>BlankMaster</MasterFormName>
  <TableName>Application_Users</TableName>
  <Fields>
    ...
    <Password Name="NewPassword">
      <Required></Required>
      <MaxChars>20</MaxChars>
    </Password>
    <Password Name="ConfirmPassword">
      <Required></Required>
      <MaxChars>20</MaxChars>
    </Password>
    ...
  </Fields>
</Form>
```

## Code Generation

```
public class ChangePasswordClass : FormData {
  public class NewPasswordFieldClass : FieldValue {
    public string val {get { return GetFieldString(); }set { SetFieldString(value); }}
    public new FieldHiddenType Hidden { get { return base.Hidden; } set { base.Hidden = value; }}
    public new ChangePasswordClass FormData {get {...}}
    public ChangePasswordFieldClass(ButtonFieldValue fieldValue): base(fieldValue){}
```

```
    public const long fieldDefId = -596688720000190;  
}  
public ChangePasswordFieldClass ChangePassword { get {...} }  
}
```

# Number Field

## Description

Text Field in which user can enter a number with or without decimal digits.

## XML

```
<Number Name="Marks" >
    <ZeroAsBlank/>
    <Decimals>2</Decimals>
    <MinimumValue>0.00</MinimumValue>
    <MaximumValue>100.00</MaximumValue>
    <RegexValidation>100.00</RegexValidation>
</Number>
```

### ZeroAsBlank

If the value of the field and field is Static, it will be shown as blank. Useful for reports.

### Decimals

Number of Decimals allowed for the field. Default is 0.

### MinimumValue

Minimum decimal value that can be entered in the field.

### MaximumValue

Maximum decimal value that can be entered in the field.

### RegexValidation

Validates the content of field to be match the specified Regex expression.

## Other Properties

Required, DoNotSaveInDB, Shortcut, Encryption, LabelStyle, ControlStyle, Label, Changeable, CSSClass, Hidden

## Code Generation

```
public class MarksFieldClass : NumberFieldValue
{
    public decimal? val { get { ... } set { ... } }
    // members fieldDefId, FormData, Hidden, Readonly as described in Field Code Generation
}
```

val of type [decimal](#)?

## Initialization Functions

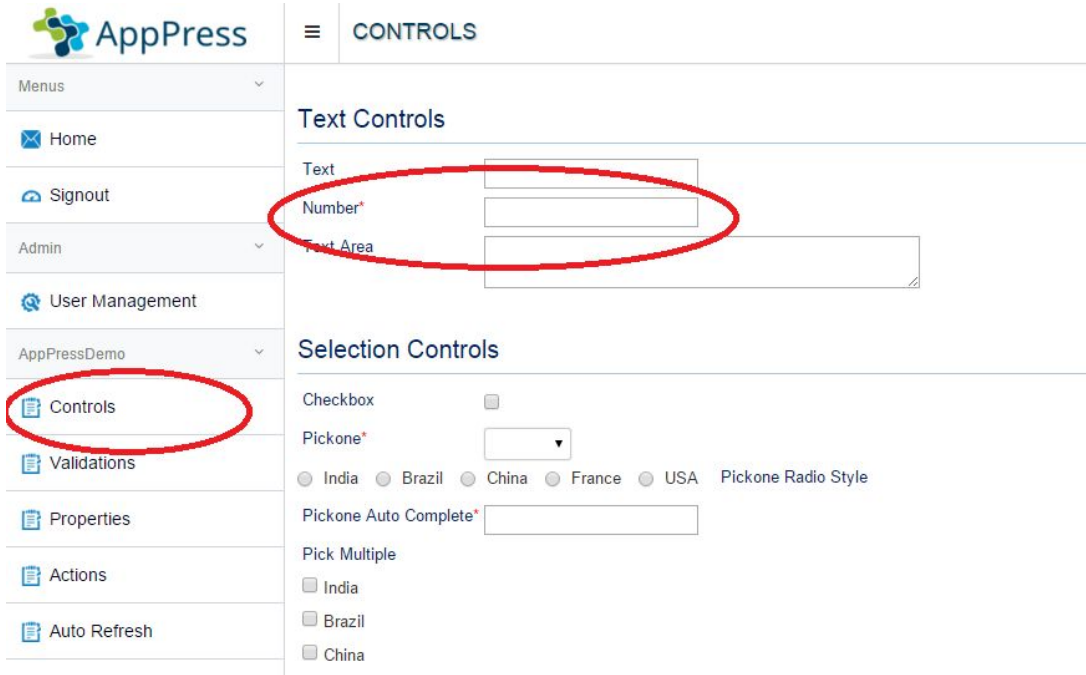
Init (of Containing Form), Calc

## Events

OnChange

## Example

Look at AppPressApplication -> Control -> Number



AppPress

≡ CONTROLS

Menus

- Home
- Signout
- Admin
  - User Management
- AppPressDemo
  - Controls**
  - Validations
  - Properties
  - Actions
  - Auto Refresh

### Text Controls

Text

Number\*

Text Area

### Selection Controls

Checkbox

Pickone\*

India Brazil China France USA Pickone Radio Style

Pickone Auto Complete\*

Pick Multiple

India

Brazil

China

## Example XML

```
<Form Name="Controls">
  <MasterFormName>ApplicationMaster</MasterFormName>
  <Fields>
    ...
    <Number Name="Number" >
      <Required/>
      <Decimals>2</Decimals>
      <RegexValidation><![CDATA[[0-9]*$]]></RegexValidation>
    </Number>
    ...
  </Fields>
</Form>
```

## Code Generation

```

public class ControlsClass : FormData
{
    ...
    public class TextFieldClass : FieldValue
    {
        public string val { get { ... } set { ... } }
        public new FieldHiddenType Hidden { get { return base.Hidden; } set { base.Hidden = value; } }
        public new ControlsClass FormData
        {
            ...
        }
        public TextFieldClass() : base() { }
        public TextFieldClass(FieldValue fieldValue) : base(fieldValue) { }
        public const long fieldDefId = -596688720000190;
    }
    public TextFieldClass Text { get { ... } }
    ...
}

```

## OnChange Logic

```

public static void OnChange(AppPress a, ControlsClass.TextFieldClass Text)
{
    a.AlertMessage("Text OnChange Value: " + string.Join(",", Text.val));
}

```

# Datetime Field

## Description

Field where user can enter Date, Date and Time or Time.

## XML

```
<DateTime Name="DateOfBirth" >
    <DateFormat>dd/mm/yyyy</DateFormat>
    <Style TimeFormat="Hours">Date</Style>
</DateTime>
```

Style (Default Date)

Date  
Time

TimeFormat  
Hours  
Minutes  
Seconds

DateFormat

Format of the Date to be used when date is displayed as Static. The format is in [.net](#) date format  
When Date is editable AppPress uses Date Formats specified in Settings parameter of InitAppPress.

## Other Properties

Required, DoNotSaveInDB, Shortcut, Encryption, LabelStyle, ControlStyle, Label, Changeable, CSSClass, Hidden, Filter

## Code Generation

```
public class DateOfBirthFieldClass : DateTimeFieldValue
{
    public DateTime? val { get { ... } set { ... } }
    // members fieldDefId, FormData, Hidden, Readonly as described in Field Code Generation
```



```
}
```

val of type `DateTime?`

```
public class DateTimeFieldValue : FieldValue
{
    public DateTime? BaseDate = null;
    public DateTime? MinDateTime = null;
    public DateTime? MaxDateTime = null;
}
```

BaseDate

If this is set this on UI the field displays time relative to this date.

BaseDate	val	TimeFormat	Display
25th July 2015	25th July 2015 10:00		10:00
	26th July 2015 18:00	Hours	+1 18
	24th July 2015 14:19:56	Seconds	-1 14:19:56

MinDateTime

Minimum Date Time allowed for the field. If null no minimum date time.

MaxDateTime

Maximum Date Time allowed for the field. Allowed value includes MaxDateTime. if null no maximum date time

## Initialization Functions

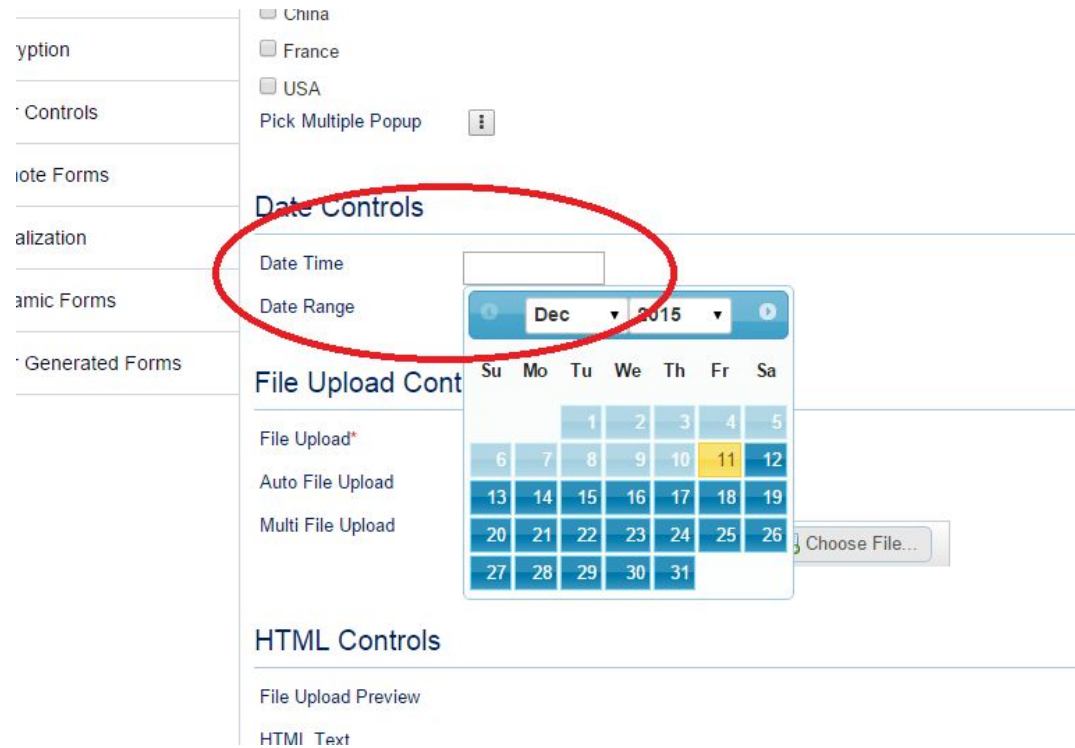
Init (of Containing Form), Calc

# Events

OnChange

## Example

Look at AppPressApplication -> Control -> DateTime



## Example XML

```
<Form Name="Controls">
  <MasterFormName>ApplicationMaster</MasterFormName>
  <Fields>
    ...
    <DateTime Name="DateTime"></DateTime>
    ...
  </Fields>
</Form>
```

```
</Fields>
</Form>
```

## Code Generation

```
public class ControlsClass : FormData
{
    ...
    public class DateTimeFieldClass : DateTimeFieldValue
    {
        public DateTime? val { get { ... } set { ... } }
        public new FieldHiddenType Hidden { get { return base.Hidden; } set { base.Hidden = value; } }
        public new ControlsClass FormData { ... }
        public DateTimeFieldClass() : base() { }
        public DateTimeFieldClass(FieldValue fieldValue) : base(fieldValue) { }
        public const long fieldDefId = -596688720000190;
    }
    public DateTimeFieldClass DateTime { get { ... } }
    ...
}
```

## OnChange Logic

```
public static void OnChange(AppPress a, ControlsClass.DateTimeFieldClass DateTime)
{
    a.AlertMessage("DateTime OnChange Value: " + DateTime.val);
}
```

# DateRange Field

## Description

Set of 2 DateTime fields where user can enter a Range of Date or Date and time.

## XML

```
<DateRange Name="EmployeeDesignation" >  
    <DateFromRequired/>  
    <DateToRequired/>  
    <Contiguous/>  
    <NonOverlapping/>  
</DateRange>
```

DateFromRequired, DateToRequired

Makes Corresponding field as required

NonOverlapping

This property is valid when the field is defined as part of RowFields in FormContainerGrid. This implies that in the grid Date Ranges will not overlap amongst rows

Contiguous

This property is valid when the field is defined as part of RowFields in FormContainerGrid. This implies that in the grid

- Dates Range will be nonoverlapping

- Date From will be less than equal to DateTo in a row

- Date From will be 1 day more than DateTo of previous row

- Last row DateTo will be null

## Properties from DateTime Field Type

Style, TimeFormat, DateFormat

## Other Properties

Required, DoNotSaveInDB, Shortcut, Encryption, LabelStyle, ControlStyle, Label, Changeable, CSSClass, Hidden, Filter

## Code Generation

```
public class EmployeeDesignationFromFieldClass : DateTimeFieldValue
{
    public DateTime? val { get { ... } set { ... } }
    // members fieldDefId, FormData, Hidden, Readonly as described in Field Code Generation
}

public class EmployeeDesignationToFieldClass : DateTimeFieldValue
{
    public DateTime? val { get { ... } set { ... } }
    // members fieldDefId, FormData, Hidden, Readonly as described in Field Code Generation
}
```

val of type `DateTime?`

## Initialization Functions

Init (of Containing Form), Calc

## Events

OnChange

## Example

Look at AppPressApplication -> Control -> DateRange

ption

Controls

ote Forms

ilization

mic Forms

Generated Forms

☐ China  
☐ France  
☐ USA  
Pick Multiple Popup

Date Controls

Date Time  
Date Range

File Upload Controls

File Upload\*  
Auto File Upload  
Multi File Upload

HTML Controls

File Upload Preview  
HTML Text

## Example XML

```

<Form Name="Controls">
  <MasterFormName>ApplicationMaster</MasterFormName>
  <Fields>
    ...
    <DateRange Name="DateRange"></DateRange>
    ...
  </Fields>
</Form>

```

## Code Generation

AppPress User Guide

53

The field is split into 2 fields of type DateTime. name of first field is name of the field suffixed with “From” and fieldName of 2nd field is name of field suffixed with “To”.

```
public class ControlsClass : FormData
{
    ...
    public class DateRangeFromFieldClass : DateTimeFieldValue
    {
        public DateTime? val { get { ... } set { ... } }
        public new FieldHiddenType Hidden { get { return base.Hidden; } set { base.Hidden = value; } }
        public new ControlsClass FormData { ... }
        public DateRangeFromFieldClass() : base() { }
        public DateRangeFromFieldClass(FieldValue fieldValue) : base(fieldValue) { }
        public const long fieldDefId = -596688720000190;
    }
    public DateRangeFromFieldClass DateTime { get { ... } }
    public class DateRangeToFieldClass : DateTimeFieldValue
    {
        public DateTime? val { get { ... } set { ... } }
        public new FieldHiddenType Hidden { get { return base.Hidden; } set { base.Hidden = value; } }
        public new ControlsClass FormData { ... }
        public DateRangeToFieldClass() : base() { }
        public DateRangeToFieldClass(FieldValue fieldValue) : base(fieldValue) { }
        public const long fieldDefId = -596688720000190;
    }
    public DateRangeToFieldClass DateTime { get { ... } }
    ...
}
```

## OnChange Logic

```
public static void OnChange(AppPress a, ControlsClass.DateRangeFromFieldClass DateRangeFrom)
{
    a.AlertMessage("DateRange From OnChange Value: " + DateRangeFrom.val);
}
```

# FileUpload Field

## Description

Control which allows files to be uploaded and used in backend logic.

If DoNotSaveInDB property is not specified, The file is saved in Application\_Files table and id of row is set in val field. If DoNotSaveInDB is specified then the File is saved as temp file and name of file is set in val field.

## XML

```
<FileUpload Name="FileUpload" >
  <Accept ValidateOnServer="true"><![CDATA[.jpg,.jpeg,.png]]></Accept>
  <AutoUpload/>
  <DoNotSaveInDB/>
  <MaxFileSizeInKB>1024</MaxFileSizeInKB>
  <Encryption>DES</Encryption>
  <NonSecure/>
  <Storage Type="Directory">
    <Directory><![CDATA[C:\Live]]></Directory>
  </Storage>
</FileUpload>
```

### Accept

Types of files that can be uploaded. The syntax is same as that specified for html input type File.

Example 1: .jpg,jpeg,.png

Example 2: audio/\*,video/\*,image/\*

if ValidateOnServer on server is true the file type will be validated on the server.

### AutoUpload

The upload will start as soon as file is selected. Otherwise upload will start on Click on Upload Button.

### DoNotSaveInDB

The uploaded file will be saved in temp folder. The full path will be set in Value. If DoNotSaveInDB is not used then uploaded file will be saved in Application\_File table and id of the row will be set in Value



MaxFileSizeInKB

The maximum file size that can be uploaded.

Encryption

Encrypts the file using the specified encryption algorithm

NonSecure

File can be accessed with url without logged in session. Useful for logo images etc.

Storage

Defines how the uploaded file will be saved.

Type

Directory

Files will be saved in a Directory on the server.

Database

Files will be saved in Application\_Files table as a blob

Directory

Full Path where of Directory on Server where files will be saved.

## Other Properties

Required, LabelStyle, ControlStyle, Label, Changeable, CSSClass, Hidden

## Code Generation

```
public class ControlsClass : FormData
{
    ...
    public class FileUploadFieldClass : FileUploadFieldValue
    {
        public const long fieldDefId = -618825790000219;
        public string val { get { return GetFieldString(); } set { SetFieldString(value); } }
        public new FieldHiddenType Hidden { get { return base.Hidden; } set { base.Hidden = value; } }
        public new ControlsClass FormData { get { ... } }
        public FileUploadFieldClass() : base() { }
        public FileUploadFieldClass(FileUploadFieldValue fieldValue) : base(fieldValue) { }
    }
    public FileUploadFieldClass FileUpload { get { ... } }
```

```
}  
...  
}
```

val of type `string`

## Initialization Functions

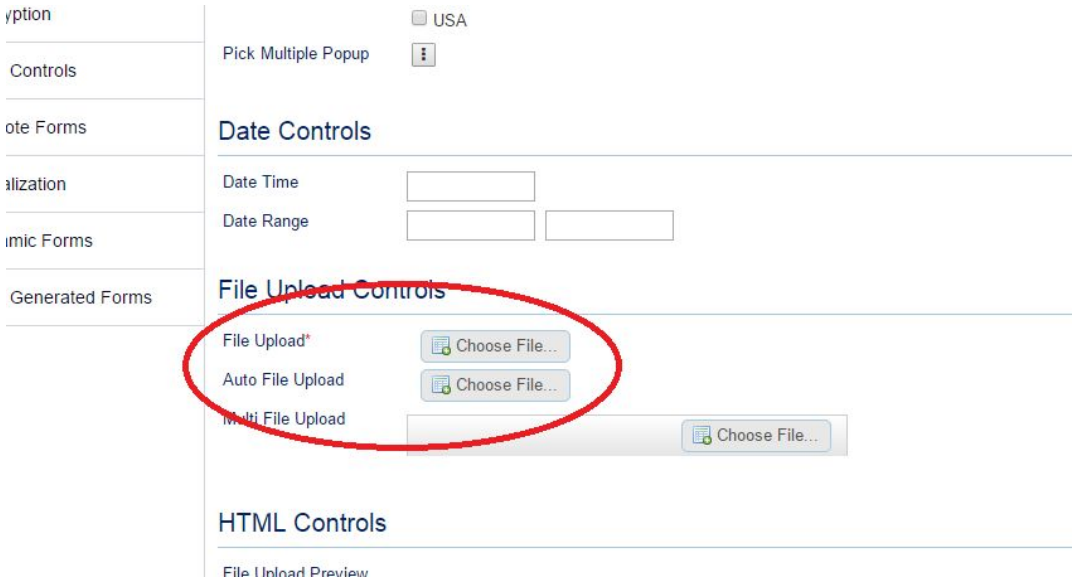
Init (of Containing Form), Calc

## Events

OnChange

## Example

Look at AppPressApplication -> Control -> FileUpload



## Example XML

```

<Form Name="Controls">
  <MasterFormName>ApplicationMaster</MasterFormName>
  <Fields>
    ...
    <FileUpload Name="FileUpload" >
      <Accept><![CDATA[.jpg,.jpeg,.png]]></Accept>
      <Required></Required>
      <DoNotSaveInDB/>
    </FileUpload>
    ...
  </Fields>
</Form>

```

## OnChange Logic

```

public static void OnChange(AppPress a, ControlsClass.FileUploadFieldClass FileUpload)
{
  a.AlertMessage("FileUpload OnChange Value: " + FileUpload.val);
}

```

# MultiFileUpload Field

## Description

Control to allow upload and management of Multiple files.

## XML

```
<MultiFileUpload Name="MultiFileUpload" >
  <TableName>MultiFileUpload_Table</TableName>
  <ForeignKey>ControlId</ForeignKey>
</MultiFileUpload>
```

TableName

Name of table where the files will be saved. The structure of the table is **To BE Checked**

Column	Description
Id	Primary Key of table
ControlId	ForeignKey to Parent
File	Blob to save the file

ForeignKey

ForeignKey in above table to the Table of the form containing the MultiFileUpload field

## Properties from FileUpload

Accept, AutoUpload, DoNotSaveInDB, MaxFileSizeInKB, Encryption, NonSecure, Storage

## Other Properties

Required, LabelStyle, ControlStyle, Label, Changeable, CSSClass, Hidden

## Code Generation

```

public class ControlsClass : FormData
{
    ...
    public class FileUploadFieldClass : FileUploadFieldValue
    {
        public const long fieldDefId = -618825790000219;
        public string val { get { return GetFieldString(); } set { SetFieldString(value); } }
        public new FieldHiddenType Hidden { get { return base.Hidden; } set { base.Hidden = value; } }
        public new ControlsClass FormData { get { ... } }
        public FileUploadFieldClass() : base() { }
        public FileUploadFieldClass(FileUploadFieldValue fieldValue) : base(fieldValue) { }
    }
    public FileUploadFieldClass FileUpload { get { ... } }
    ...
}

```

val of type `string`

## Initialization Functions

Init (of Containing Form), Calc

## Events

No Events

## Example

Look at AppPressApplication -> Control -> MultiFileUpload

Visualization
Dynamic Forms
User Generated Forms

Date Time
Date Range

### File Upload Controls

File Upload\*

Auto File Upload

Multi File Upload

1004.pdf	Download	Delete
1005.pdf	Download	Delete

### HTML Controls

File Upload Preview
HTML Text

## Example XML

```

<Form Name="Controls">
  <MasterFormName>ApplicationMaster</MasterFormName>
  <Fields>
    ...
    <MultiFileUpload Name="MultiFileUpload" >
      <Storage Type="Directory">
        <Directory><![CDATA[C:\Live]]></Directory>
      </Storage>
      <MaxFileSizeInKB>1000</MaxFileSizeInKB>
    </MultiFileUpload>
    ...
  </Fields>
</Form>

```

# HTML Field

## Description

Display Static Content on UI. The content is val of the field class. If val is null then Label. If Label is null then field name

## XML

```
<HTML Name="SectionHeading">
  <Label>Display this Text</Label>
  <LabelStyle><![CDATA[border-bottom: 1px solid #a6c9e2;font-size:20px;margin-bottom:12px;padding-top:25px;width:
100%;]]></LabelStyle>
  <CSSClass>sectionHeading</CSSClass>
</HTML>
```

Label

Text to shown. If not present field name is used after adding space before Capital Letters

LabelStyle

Inline style to be applied to the text

CSSClass

Class to be applied to the text

## Other Properties

None

## Code Generation

```
public class HTMLControlsFieldClass : FieldValue
{
    public const long fieldDefId = -618942620000230;
    public string val { get { ... } set { ... } }
    public new FieldHiddenType Hidden { get { ... } set { ... } }
    public new ControlsClass FormData { get {...}}
    public HTMLControlsFieldClass() : base() { }
    public HTMLControlsFieldClass(FieldValue fieldValue) : base(fieldValue) { }
```

```

    }
    public HTMLControlsFieldClass HTMLControls { get { ... } }

```

## Initialization Functions

Init (of Containing Form), Calc

## Events

None

## Example

Look at AppPressApplication -> Encryption -> Download

The screenshot shows the 'File Upload Controls' section of the AppPress application. It includes three upload options: 'File Upload\*' with a 'Choose File...' button, 'Auto File Upload' with a 'Choose File...' button, and 'Multi File Upload' which displays a table of uploaded files. Below this, the 'HTML Controls' section is circled in red, showing 'File Upload Preview' and 'HTML Text' options.

Multi File Upload	
1004.pdf	Download
1005.pdf	Download

## Example XML

```

<Form Name="Encryption">
  <MasterFormName>ApplicationMaster</MasterFormName>
  <TableName>AppPressDemo_Encryption</TableName>
  <Fields>

```



```
...
<HTML Name="HTMLControls">
  <LabelStyle><![CDATA[border-bottom: 1px solid #a6c9e2;font-size: 20px;margin-bottom: 12px;padding-left: 0;padding-top:
25px;width: 100%;]]></LabelStyle>
</HTML>
...
</Fields>
</Form>
```

# Button Field

## Description

Field in which user can click to perform an action

## XML

```
<Button Name="Save" >  
    <Style>Button</Style>  
    <NoSubmit/>  
</Button>
```

Style

Button, Link

NoSubmit

OnClick Page Data will not be submitted to OnClick function. Useful where you just have to redirect to another form without using page data.

## Other Properties

Shortcut, ControlStyle, Label, Changeable, CSSClass

## Code Generation

```
public class SaveFieldClass : ButtonFieldValue  
{  
    // members fieldDefId, FormData, Hidden, Readonly as described in Field Code Generation  
}
```

## Initialization Functions

Init (of Containing Form), Calc

## Events

OnClick

## Example

### OnClick Example

Look at AppPressApplication -> Encryption -> Download

Click on Download will download the file uploaded via the DESFile FileUpload field

The screenshot shows the AppPress application interface. On the left is a sidebar menu with the AppPress logo at the top. The menu items are: Home, Signout, Admin (with a dropdown arrow), User Management, AppPressDemo (with a dropdown arrow), Controls, Validations, Properties, Actions, Auto Refresh, Encryption (circled in red), and User Controls. The main content area is titled 'ENCRIPTION' (sic) and contains several input fields: 'AES Text' (a single-line text box), 'DES Text Area' (a multi-line text area), 'AES Number' (a single-line text box), 'DES Number' (a single-line text box), and 'DES File' (a file upload button labeled 'Choose File...'). Below these fields is a 'Download' button (circled in red) and a 'Save' button. At the bottom of the main area, it says 'Dev Links: Encryption skin fo code'.

### Example XML

```
<Form Name="Encryption">
  <MasterFormName>ApplicationMaster</MasterFormName>
  <TableName>AppPressDemo_Encryption</TableName>
```

```

<Fields>
    ...
    <FileUpload Name="DESFile" >
        <Encryption Type="DES"/>
    </FileUpload>
    <Button Name="Download" >
        <Style>Link</Style>
    </Button>
</Fields>
</Form>

```

### OnClick Logic

```

public static void OnClick(AppPress a, EncryptionClass.DownloadFieldClass Download)
{
    a.DownloadFile(long.Parse(Download.FormData.DESFile.val));
}

```

# Redirect Field

## Description

Shows a button, Click on the button Redirects to the Form given in the FormName property or Name attribute

## XML

```
<Redirect Name="Administration" >  
    <Style>Button</Style>  
</Redirect>
```

Style

Button, Link

## Other Properties

Shortcut, ControlStyle, Label, Changeable, CSSClass

## Code Generation

```
public class AdministrationFieldClass : ButtonFieldValue  
{  
    // members fieldDefId, FormData, Hidden, Readonly as described in Field Code Generation  
}
```

## Initialization Functions

Init (of Containing Form), Calc

## Events

None

## Example

### OnClick Example

AppPress User Guide

Look at AppPressApplication -> Controls  
Click on Controls redirects to Controls Form

The screenshot shows the AppPress application interface. On the left is a sidebar menu with the AppPress logo at the top. Below the logo, there are several menu items: 'Home', 'Signout', 'Admin', 'User Management', 'AppPressDemo', 'Controls' (highlighted with a red circle), 'Validations', 'Properties', 'Actions', 'Auto Refresh', 'Encryption', and 'User Controls'. The 'AppPressDemo' menu is expanded, showing 'Controls' as the selected item. The main content area on the right is titled 'CONTROLS' and contains two sections: 'Text Controls' and 'Selection Controls'. The 'Text Controls' section includes a 'Text' input field, a 'Number\*' input field, and a 'Text Area' input field. The 'Selection Controls' section includes a 'Checkbox' input field, a 'Pickone\*' dropdown menu, a 'Pickone Radio Style' section with radio buttons for 'India', 'Brazil', 'China', 'France', and 'USA', and a 'Pickone Auto Complete\*' input field. Below these is a 'Pick Multiple' section with checkboxes for 'India', 'Brazil', 'China', 'France', and 'USA'.

## Example XML

```
<MasterForm Name="ApplicationMaster">
  <Fields>
    ...
    <Redirect Name="Controls" >
    </Redirect>
    ...
  </Fields>
```

```
</Form>
```

# Pickone Field

## Description

Field in which user can choose one option from given set of options

## XML

```
<Pickone Name="AssetType">
  <Style>DropDown</Style>
  <TableName>Assets</TableName>
</Pickone>
```

Style

Radio, DropDown, AutoComplete

TableName

Table to get options for field

## Other Properties

Required, DoNotSaveInDB, Static, NonStatic, Hidden, LabelStyle, ControlStyle, Label, Changeable, CSSClass

## Code Generation

```
public class AssetTypeFieldClass : PickFieldValue
{
    public string val { ... }
    // other members as described in Field Code Generation
}
```

val of Type `string`

## Options Generation



AppPress needs list of Options for displaying in UI. Each option has Id and Value fields. AppPress gets options using one of following methods

- Options function in code which returns a query string and Id and Value columns in results. AppPress executes the query and populates the options in UI.
- Options function in code which returns a list of Option objects.
- Options from TableName property in the Field XML
- From Database. if the field is bound to a column in database table and a foreign key exists for the column, AppPress will get the options from the referenced table as follows
  - id in Option will map to Id column in Table
  - Value in Option will map to First Varchar column in reference table

Note:

- ForeignKey options are cached and generated only first time. If the option table is modified via AppPress Save function options are automatically re-cached. If you modify the table directly then you will need to restart AppPress.
- You can use IsStatic property of the field and return query or list containing only the option with id as the val of the field

```
public static string Options(AppPress a, GetActiveCountryOptionsClass.CountryFieldClass Country)
{
    var q = "Select Id, Country From Country ";
    if (Country.IsStatic)
        if (Country.val == null)
            return null;
        else
            q += " Where id=" + Country.val;
    q += " Order By Country";
    return q;
}
```

- For AutoComplete Style, AppPress will call The options logic every time user makes change to the text field. AppPress will limit the query execution to get top 7 options
- When using Options function the logic should ensure that option for field value is returned. A use case is as follows

Lookup table having Id, Designation, InActive column.

A Designation is marked InActive, which means new designations will not have this option but old designation should show. The options function for such a case should be written as

```
public static string Options(AppPress p, EmployeeProfileClass.DesignationPopupClass.DesignationFieldClass Designation)
{
```

```
        return "Select * From Designation Where InActive=0 or Id='" + (Designation.val??"-1") + "'";  
    }
```

## Code behind Functions

Options, Calc

# PickMultiple Field

## Description

Field in which user can choose multiple options from given set of options

## XML

```
<PickMultiple Name="AssetType" >
  <Style>PopupCheckboxes</Style>
  <SaveTableName>Assets.Managers.AssetType</SaveTableName>
  <SaveTableForeignKey>AssetManager</SaveTableForeignKey>
  <PartStyle><![CDATA[float:left;width:80px]]></PartStyle>
  <ShowSelectAll/>
</PickMultiple>
```

### Style

InlineCheckboxes, PopupCheckboxes

### SaveTableName

Name of Table to save PickMultiple Selection. This Table should have a column of type int with name as the fieldName. Required only if PickMultiple is to be saved.

### SaveTableForeignKey

Column Name in Table which is Foreign Key into TableName of Form containing the Field. Required is TableName property is defined.

### PartStyle

Style to be used on Checkboxes

### ShowSelectAll

Show checkbox to allow select all or unselect all

## Properties from Pickone

TableName

## Properties from Field

Required, DoNotSaveInDB, Static, NonStatic, Hidden, LabelStyle, Label, Changeable, CSSClass

## ControlStyle

ControlStyle is applied to div containing the option checkboxes.

## Code Generation

```
public class AssetTypeFieldClass : PickFieldValue
{
    public string val { ... }
    // other members fieldDefId, Hidden, FormData, Readonly as described in Field Code Generation
}
```

val of type List<int> containing ids of selected options

## Options Generation

AppPress needs list of Options for displaying in UI. Each option has Id and Value fields. AppPress gets options using one of following methods

- Options function in code which returns a query string and Id and Value columns in results. AppPress executes the query and populates the options in UI, AppPress will get the option
- From Database. if the field is bound to a TableName and ForeignKey, AppPress will get the options from the referenced table as follows
  - id in Option will map to Id column in Table
  - Value in Option will map to First Varchar column in reference table

## Initialization Functions

Init (of Containing Form), Options, Calc

## Events

OnChange

In case of `InlineCheckboxes` Style triggered when a checkbox is checked or unchecked. In case of `PopupCheckboxes` `OnChange` is triggered on close of popup.

## Examples

### OnChange, Options function Example

Look at AppPressApplication -> Controls -> Selection Controls

Clicking on Selection will display a Alert

**Selection Controls**

Checkbox ☐

Pickone\*

☐ India ☐ Brazil ☐ China ☐ France ☐ USA Pickone Radio Style

Pickone Auto Complete\*

Pick Multiple

☐ India

☐ Brazil

☐ China

☐ France

☐ USA

Pick Multiple Popup

### Bind to Database, Options from Foreign Key Example

Look at AppPressApplication -> Controls

HTML Controls

File Upload Preview

HTML Text

Form Container

Form Container Grid With Fixed Header

MANAGERS

<input type="checkbox"/>	Name	Date Of Birth
<input checked="" type="checkbox"/>	s1	13-Oct-
<input type="checkbox"/>	s1	01-Oct-
<input type="checkbox"/>	kjjbjb	07-Oct-
<input type="checkbox"/>	h gfh gfhgfh	06-Oct-

Managers

Name\*

s1

Date Of Birth\*

13-Oct-2015

Address\*

df fkj j b djknf kf rd fd hijb jbjjknknknj

Country\*

India

State\*

Haryana

City

Gurgaon

Pin Code\*

r33243

In Active

☒

Responsibilites

Administrator | HR | Accounts

Dev Links: ManagersPopup

skin fo code

Save

Add

Modify

Delete

Responsibilites

Administrator | HR | Accounts

FORM CONTAINER GRID INLINE

## Example XML

```

<Form>
...
<FormContainerGrid Name="Managers" >
  <TableName>AppPressDemo_Managers</TableName>
  <RowFields Height="150" Sortable="true">
    ...
    <PickMultiple Name="Responsibilites" >
      <Style>PopupCheckboxes</Style>
      <TableName>AppPressDemo_Responsibilites</TableName>
      <ForeignKey>ManagerId</ForeignKey>
    </PickMultiple>
  </RowFields>
  <PopupFields>
    ...
    <PickMultiple Name="Responsibilites" >
      <Style>PopupCheckboxes</Style>

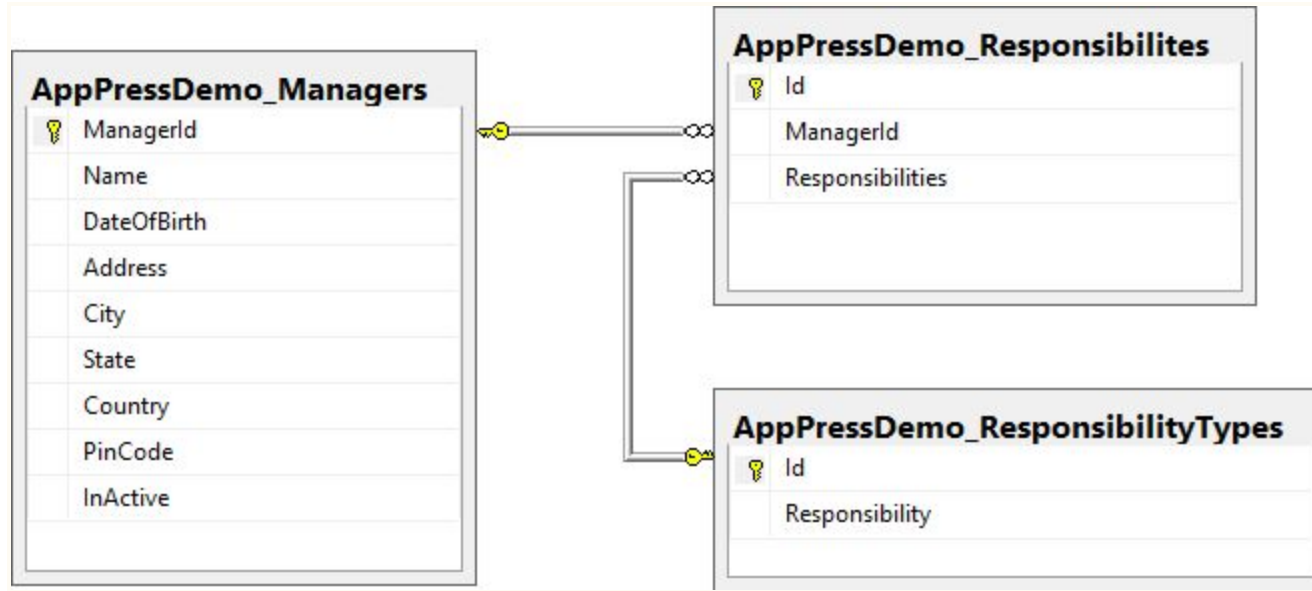
```

```

        <TableName>AppPressDemo_Responsibilites</TableName>
        <ForeignKey>ManagerId</ForeignKey>
    </PickMultiple>
</PopupFields>
</FormContainerGrid>
</Form>

```

## Example Database Tables and Relationships





# Checkbox Field

## Description

Field in which user can check uncheck a checkbox. If Static Property of field is set, the value is displayed from Options list for the field.

## XML

```
<Checkbox Name="Inactive" >  
</Checkbox>
```

## Other Properties

DoNotSaveInDB, Static, NonStatic, Hidden, LabelStyle, ControlStyle, Label, Changeable, CSSClass

## Code Generation

```
public class AssetTypeFieldClass : PickFieldValue  
{  
    public bool val { ... }  
    // other members as described in Field Code Generation  
}
```

val of Type `bool`

## Options Generation

AppPress needs list of Options for displaying in UI. Each option has Id and Value fields. AppPress gets options using one of following methods

- Options function in code which returns a query string and Id and Value columns in results. AppPress executes the query and populates the options in UI, AppPress will get the option
- From Database. if the field is bound to a column in database table and a foreign key exists for the column, AppPress will get the options from the referenced table as follows
  - id in Option will map to Id column in Table
  - Value in Option will map to First Varchar column in reference table
- If none of above exists, AppPress will display “Yes” for true and “No” for False

Code behind Functions

Options, Calc

# FormContainerDynamic Field

## Description

Container for Child Forms displayed inline.

## XML

```
<FormContainerDynamic Name="EmployeeStatusDetails">  
    <ControlStyle><![CDATA[border:1px solid black]]></ControlStyle>  
</FormContainerDynamic>
```

### ControlStyle

The style is applied to div containing the Forms

## Other Properties

LabelStyle, Label, Changeable, CSSClass

## Code Generation

```
public class EmployeeStatusDetailsFieldClass : FormContainerDynamicFieldValue  
{  
    public List<FormData> val { get {...} set {...} }  
    // other members fieldDefId, Hidden, FormData, Readonly as described in Field Code Generation  
}
```

val of type List<FormData> List of FormData in the field

FormContainerDynamicFieldValue member functions

```
public List<FormData> GetMasterContainer(AppPress a)
```

Used to Bind Form from Url to MasterContentArea in MasterForm

```
public List<FormData> BindSingleForm(AppPress a, long? formDefId)
```

Dynamically bind a single form to FormContainerDynamic control. Some use cases are

- Embed a form inside another form
- Embed a form depending on another pickone field

If the field has another type of form bound to it deletes them

If FormName is bound to a table returns forms bound to parent form with foreign Key field in formName

If no form is bound returns a new form of type formName

## Initialization Functions

Domain	return List<FormData>. FormData can be of different types. can use BindSingleForm or return List<FormData> from own logic
Calc	Set val to List<FormData>.

## Events

None

## Examples

Look at AppPressApplication -> Controls -> FormContainerDynamic

Changing Employee Status will Bind the FormContainerDynamic to Form linked to Employee Status field value.

Name*	Date Of Birth*	Address*	Country*	State*	City	Pin Code*
<input type="checkbox"/> John Smith	01-Dec-1997	769, Smith Street,	USA ▼	New York ▼	Long Island City ▼	23233
<input type="checkbox"/> Pankaj Singh	06-Dec-1996	1 Sapphire Court, Essel Tow	India ▼	Haryana ▼	Gurgaon ▼	122001

Form Container

Dynamic Control

Employee Status Joined ▼

Date Of Joining\*

Designation\*

Action Controls

Submit Link

## Example XML

```

<Form>
...
<Pickone Name="EmployeeStatus" ></Pickone>
<FormContainerDynamic Name="FormContainerDynamic">
    <ControlStyle><![CDATA[border:1px solid black]]></ControlStyle>
</FormContainerDynamic>
...
</Form>

<Form Name="Joined">
    <TableName>AppPressDemo_Joined</TableName>
    <Fields>
        <ForeignKey Name="Employee" ></ForeignKey>
        <DateTime Name="DateOfJoining" ></DateTime>
        <Text Name="Designation" ></Text>
    </Fields>
</Form>

<Form Name="Resigned">

```

```

<TableName>AppPressDemo_Resigned</TableName>
<Fields>
    <ForeignKey Name="Employee" ></ForeignKey>
    <Text Name="ResignationReason" ></Text>
    <DateTime Name="DateOfResignation" ></DateTime>
</Fields>
</Form>

```

## Logic

```

public static List<FormData> Domain(AppPress a, ControlsClass.FormContainerDynamicFieldClass FormContainerDynamic)
{
    switch (FormContainerDynamic.FormData.EmployeeStatus.val)
    {
        case null:
            return null;
        case "1":
            return FormContainerDynamic.BindSingleForm(a, JoinedClass.formDefId);
        default:
            return FormContainerDynamic.BindSingleForm(a, ResignedClass.formDefId);
    }
}

```

# FormContainerGrid Field

## Description

Container for Child Forms displayed as Grid. Each row in the grid is a child form.

## XML

```
<FormContainerGrid Name="Managers">
  <TableName>AppPressDemo_Managers</TableName>
  <PopupTitle><![CDATA[Changed Title]]></PopupTitle>
  <ControlStyle><![CDATA[border:1px solid black]]></ControlStyle>
  <Label><![CDATA[Managers (%Count%)]]></Label>
  <RowFields Height="150" Sortable="true">
    <SelectRow>
      <AllowMultiSelect/>
    </SelectRow>
    <Text Name="Name">
    </Text>
    <DateTime Name="DateOfBirth" ></DateTime>
    ...
  </RowFields>
  <PopupFields>
    <Text Name="Name" >
      <RegexValidation><![CDATA[]]></RegexValidation>
    </Text>
    <DateTime Name="DateOfBirth" ></DateTime>
    ...
  </PopupFields>
  <ContainerFields>
    <Button Name="InActivateManager">
    </ContainerFields>
  </FormContainerGrid>
```

### ControlStyle

The style is applied to div containing the Forms

TableName

The table to which the grid is bound. If Domain is not used for the form gets the rows in the grid from the Table. If ForeignKey is specified in RowFields then rows are filtered by the Id if containing form.

PopupTitle

Title of Popup for Add and Modify. If absent the field Name is split by adding space before Capital Letter and used as Title.

RowFields

Columns for the Grid. Added as fields. By default all fields except buttons are Static in RowFields. To make the field editable use NonStatic property. If SelectRow fieldType is not present then Modify and Delete buttons are hidden.

Use Filter attribute in Fields within RowFields to show a Filter button in Column Header.

Attributes

Height Height of the grid. If Height is provided a vertical scroll is added and title will remain fixed.

Sortable If true a sort option is added to column header.

PopupFields

Fields to be displayed in Popup on Click of Add or Modify buttons. If PopupFields is absent, Add and Modify buttons are hidden

ContainerFields

Fields to be displayed in Caption of the Grid

Label

Works like field Label except that %Count% is replaced by count of rows in the grid.

Other Properties

LabelStyle, Changeable, CSSClass

## Code Generation

```
public class ManagersFieldClass : FormContainerGridFieldValue
{
    public List<FormData> val { get {...} set {...} }
    // other members fieldDefId, Hidden, FormData, Readonly as described in Field Code Generation
}
```

val of type List<FormData> List of FormData in the field



## FormContainerGridFieldValue member functions

```
public void AddNewForm(AppPress a)
```

Add a New Form to the grid

```
public void AddNewForm(AppPress a, FormData formData)
```

Add formData in Parameter to the grid

```
public FormData TryGetSingleSelection()
```

Gets the only selected form. If no form is selected returns null. If more than one selected then returns null.

```
public FormData GetSingleSelection()
```

Gets the only selected form. If no form or more than 1 selected then throws Error

```
public FormData GetSingleSelection()
```

Add formData in Parameter to the grid

```
public List<FormData> GetSelection()
```

return List<FormData> selected by the user

```
public void DeleteSelectedSubForms()
```

Deletes the Selected forms

## Initialization Functions

List<FormData> Domain	return List<FormData>. FormData can be of different types. can use BindSingleForm or return List<FormData> from own logic. return null if grid is empty
string Domain	return a query, the result of query execution is used to generate the List<FormData> for the grid. return null if grid is empty
Calc	Set val to List<FormData>.

Events

None

Examples

Look at AppPressApplication -> Properties

Client Accounts

SEARCH

Add

Modify

Delete

	Id	Name	Address	Country	Phone	Email	Client Accounts										
<input type="checkbox"/>	1	John Smith	897 E House, Sam Street, Rio De Janerio	Brazil	7676769891	js@email.com	<table><thead><tr><th>Id</th><th>Name</th><th>Purchase Date</th><th>Expiry Date</th><th>Payment</th></tr></thead><tbody></tbody></table>	Id	Name	Purchase Date	Expiry Date	Payment					
Id	Name	Purchase Date	Expiry Date	Payment													
<input type="checkbox"/>	2	Ramesh Kumar	803 Banni, The Address	India	65424225	ramesh@email.com	<table><thead><tr><th>Id</th><th>Name</th><th>Purchase Date</th><th>Expiry Date</th><th>Payment</th></tr></thead><tbody></tbody></table>	Id	Name	Purchase Date	Expiry Date	Payment					
Id	Name	Purchase Date	Expiry Date	Payment													
<input checked="" type="checkbox"/>	5	Hun Chow	786, Mega Street, Shanghai	China	31212132	hun@email.com	<table><thead><tr><th>Id</th><th>Name</th><th>Purchase Date</th><th>Expiry Date</th><th>Payment</th></tr></thead><tbody><tr><td>6</td><td>Soi Chin</td><td>07-Oct-2015</td><td>23-Oct-2015</td><td>213</td></tr></tbody></table>	Id	Name	Purchase Date	Expiry Date	Payment	6	Soi Chin	07-Oct-2015	23-Oct-2015	213
Id	Name	Purchase Date	Expiry Date	Payment													
6	Soi Chin	07-Oct-2015	23-Oct-2015	213													

Employee Status\*

Save

Client Accounts

SEARCH

Add Modify Delete

	Id	Name	Address	Country	Phone	Email
<input type="checkbox"/>	1	John Smith	897 E House, Sam Street, Rio De Janerio	Brazil	7676769891	is@email.com
<input type="checkbox"/>	2	Ramesh Kumar	803 Banni, The Address	India	6542	com
<input checked="" type="checkbox"/>	5	Hun Chow	786, Mega Street, Shanghai	China	3121	

Employee Status\*

Save

Select Opti... x

☐ Brazil

☐ India

☐ China

Ok

Id	Name	Purchase Date	Expiry Date	Payment
6	Soi Chin	07-Oct-2015	23-Oct-2015	213

Client Accounts

SEARCH

Add Modify Delete

	Id	Name	Address	Country	Phone	Email
<input type="checkbox"/>	1	John Smith	897 E House, Sam Street	Brazil	7676769891	is@email.com
<input type="checkbox"/>	2	Ram Kumar				
<input checked="" type="checkbox"/>	5	Hun Chow				

Employee S

Save

Client Accounts x

Id 5

Name\* Hun Chow

Address\* 786, Mega Street, Shanghai

Country\* China

Phone\* 31212132

Email\* hun@email.com

Add Modify Delete

Id	Name	Purchase Date	Expiry Date	Payment
6	Soi Chin	07-Oct-2015	23-Oct-2015	213

Save

## Example XML

```
<Form Name="Properties">
  <TableName>AppPressDemo_Employee</TableName>
  <MasterFormName>ApplicationMaster</MasterFormName>
  <Fields>
    <FormContainerGrid Name="ClientAccounts" >
      <TableName>AppPressDemo_ClientAccounts</TableName>
      <RowFields Width="90%">
        <SelectRow></SelectRow>
        <Text Name="Id" ></Text>
        <Text Name="Name" ></Text>
        <TextArea Name="Address" ></TextArea>
        <Pickone Name="Country" >
          <Sortable/>
          <Filter/>
        </Pickone>
      </RowFields>
    </FormContainerGrid>
    ...
    <FormContainerGrid Name="Licenses" >
      <TableName>AppPressDemo_Licenses</TableName>
      <RowFields Width="500px">
        <ForeignKey Name="ClientAccount" ></ForeignKey>
        <Text Name="Id" ></Text>
        ...
      </RowFields>
    </FormContainerGrid>
  </RowFields>
  <PopupFields PopupWidth="800">
    <Text Name="Id" >
      <Static></Static>
    </Text>
    <Text Name="Name" ></Text>
    ...
  </PopupFields>
  <FormContainerGrid Name="Licenses" >
    <TableName>AppPressDemo_Licenses</TableName>
    <RowFields>
      <SelectRow></SelectRow>
      <ForeignKey Name="ClientAccount" ></ForeignKey>
      <Text Name="Id" ></Text>
      ...
    </RowFields>
  </FormContainerGrid>
</Form>
```

```

        </RowFields>
        <PopupFields>
            <ForeignKey Name="ClientAccount" ></ForeignKey>
            <Text Name="Id" >
                <Static></Static>
            </Text>
            ...
        </PopupFields>
    </FormContainerGrid>
</PopupFields>
<ContainerFields>
    <Text Name="Search" >
        <Shortcut>Enter</Shortcut>
        <DoNotSaveInDB/>
    </Text>
</ContainerFields>
</FormContainerGrid>
...
</Fields>
</Form>

```

## Logic

```

public static string Domain(AppPress a, PropertiesClass.ClientAccountsFieldClass ClientAccounts)
{
    var search = ClientAccounts.FormData.Search.val;
    var searchClause = search == null ? "" : @" and Name like '%" + search.ToSqlEncodedString() + "%' or Phone like '%" +
search.ToSqlEncodedString() + "%' or Email like '%" + search.ToSqlEncodedString() + @"%'";
    var query = @"
        Select
            AppPressDemo_ClientAccounts.*
        From
            AppPressDemo_ClientAccounts
        Where 1=1 " + searchClause + @"
        Order by Id Asc";

    return query;
}

```



# SelectRow Field

## Description

Shows a Checkbox on UI for selecting row in FormContainerGrid. This field is not saved in Database

## XML

```
<SelectRow>  
    <AllowMultiSelect/>  
</SelectRow>
```

## Other Properties

None

## Code Generation

```
public class SelectRowFieldClass : PickFieldValue  
{  
    public bool val { ... }  
    // other members as described in Field Code Generation  
}
```

val of Type `bool`

## Initialization

Init

Calc

## Example

Look at AppPressApplication -> Controls->FormContainerGrid

## HTML Controls

File Upload Preview

HTML Text

## Form Container

Form Container Grid  
With Fixed Header



MANAGERS			
<input type="checkbox"/>	Name 	Date Of Birth 	Address
<input type="checkbox"/>	John Smith	01-Dec-1997	769, Smith Street,
<input type="checkbox"/>	Pankaj Singh	06-Dec-1996	1 Sapphire Court, Esse

### Example XML

```
<MasterForm Name="ApplicationMaster">
  <Fields>
    ...
    <Redirect Name="Controls" >
  </Redirect>
    ...
  </Fields>
</Form>
```



# ForeignKey Field

## Description

Field defining ForeignKey from Table of Containing Form to Table of Form containing this field. If ForeignKey is defined and Domain Function is not defined, ForeignKey will be used to filter forms in FormContainer

## XML

```
<ForeignKey Name="ClientAccount" ></ForeignKey>
```

## Other Properties

None

## Code Generation

```
public class ClientAccountFieldClass : FieldValue
{
    public string val { ... }
    // other members as described in Field Code Generation
}
```

val of Type `string`

## Initialization

None

## Example

Look at AppPressApplication ->Properties->Client Accounts->Licenses

Here we want to filter record in Client Accounts sub grid

Client Accounts				
ID	Name	Purchase Date	Expiry Date	Payment
ID	Name	Purchase Date	Expiry Date	Payment
ID	Name	Purchase Date	Expiry Date	Payment
6	Soi Chin	07-Oct-2015	23-Oct-2015	213

```
<MasterForm Name="ApplicationMaster">
  <Fields>
    ...
    <Redirect Name="Controls" >
  </Redirect>
  ...
</Fields>
</Form>
```

# AppPress API

## Overview

### AppPress

Alert(string message)

PromptClient

ExecuteJSScript

Executed after other client actions

SetPageDirty

RefreshPage

ClosePopup

ReadFormData

RedirectToUrl

GetSecureUrl

- Form Class Functions
  - Redirect
    - RedirectParams
      - Add to Url “&HideRegions=<Name of Regions>” to hide Regions in Skin
- Form Data Functions
  - IsNew
  - IsDeleted

- IsPopup
- Error
- Validate
- Save
- Delete
- Redirect
- Popup
  - Trailing Buttons Moved to Popup Buttons
- PopupContainer
- RefreshContainer
- ErrorException
- Field Functions
  - Hidden
  - Readonly
  - Color
  - Title
  - Error
  - Refresh
    - Loads Data from Database, Merges it with Client Data and Refreshes the Field
    - Should be called After Save. Use Case
      - Readonly field value updated from logic
      - Should not call Refresh here
      - Value Saved in DB
      - Should call Refresh here
  - SetFocus
  - Validate
  - val
  - RequiredValidation
  - ErrorException
- FormContainer Field Functions
  - ReadFormDatas
- Grid Functions
  - GetSingleSelection
  - GetMultipleSelected
  - container

- DeleteForm
  - ModifyForm
  - AddNewForm
- Database Execution
  - ExecuteQuery
  - ExecuteNonQuery
  - IdentityInsert
  - Transactions
    - BeginTrans
    - Commit
    - Rollback
  - ExecuteStringList
  - ExecuteIntList
  - ExecuteScalar
  - ExecuteScalarList
  - ExecuteInt
  - ExecuteDateTime
- Download
  - DownloadPDF
  - DownloadCSV
  - DownloadHTML
  - DownloadFile
  - If More than once down is initilted AppPress automatically creates a zip and downloads the zip
- Email
  - SendEmail
- Exception
  - AppPressException

## URL Options

- IgnoreSkin

•

# Settings

## Overview

Before Calling InitAppPress() from your application startup, define following in AppData.Settings

```
AppData.Settings.Add("DateFormat", "dd-MMM-yyyy");  
    Format to Display and Input Date in UI  
AppData.Settings.Add("DateTimeFormat", "dd-MMM-yyyy HH:mm");  
    Format to Display Date and Time in UI  
AppData.Settings.Add("JQueryDateFormat", "dd-M-yy");  
    Format to use in JQuery to display Date in UI  
AppData.Settings.Add("SQLDateFormat", "%d-%b-%Y");  
    Format to use to format Date in SQL Queries  
AppData.Settings.Add("AdditionalInputDateFormats", "dd-MM-yyyy | dd/MM/yyyy");  
    Additional formats for users to enter date in UI  
AppData.Settings.Add("ApplicationKey", "kjndjkndjkndkj");  
    Key to use in Application level Encryption  
AppData.Settings.Add("DBConnection", "AppPressMySql");  
    Database connection string from web.config to use  
AppData.Settings.Add("SupportEmail", "support@sysmates.com");  
    Email where AppPress internal errors are sent  
AppData.Settings.Add("DebugEmail", "support@sysmates.com");  
    in Debug mode all emails are redirected to this email address
```

# Using Plugins

```
public static void PluginInit(AppPress a)
{
}
```



# Transforming TT Files

AppPress uses Microsoft Text Transform to generate Classes from From XML. Sometimes there is confusion when there are errors in this transformation. To resolve the errors

- In Visual Studio close all open files (In case of error in TT, visual studio shows error errors in all open files)
- In Solution Explorer Right click on `_Forms.tt` and choose option `Run Custom Tool`
- Open `_Forms.cs` under `_Forms.tt` and check if this is generated properly.
- If there is a error the error is shown in the `.cs` file

File: file:///C:/EmerioPortal/Empire/Forms/AppFormDefs.xml The 'MasterFormName1' start tag on line 20 position 6 does not match the end tag of 'MasterFormName'. Line 20, position 36.

- You can also look at Visual Studio Error list and there will be a error giving line number of XML file where error was encountered.

Error 26 Running transformation: System.Exception: File: file:///C:/EmerioPortal/Empire/Forms/AppFormDefs.xml The 'MasterFormName1' start tag on line 20 position 6 does not match the end tag of 'MasterFormName'. Line 20, position 36.

- Correct the error and `Run Custom Tool` again

Note:

Whenever you make changes to Forms XML or get a new version of AppPress.dll you need to build TT files again  
If you have multiple TT files in your project you can compile all using Menu Build=>Transform All T4 Templates

# Frequently Asked Questions

## How to create a Dynamic Menu

To Create a dynamic menu or Data driven menu follow following steps

- In Masters.xml add a FormContainerGrid field with Button field. For explanation we have used field name as Asset, you can use any field name of your requirement. Run Custom Tool on \_Forms.tt as you do after modifying XML

```
<FormContainerGrid Name="Asset">
  <RowFields>
    <Text Name="MenuLabel">
    </Text>
    <Button Name="Open">
      <NoSubmit />
    </Button>
  </RowFields>
</FormContainerGrid>
```

- Write a Domain function for the above field. The Domain function will return as many rows as number of Dynamic Menu items needed. Here we use inline query, you will build the query from some table or use the List<FormData> Domain option.

```
public static string Domain(AppPress p, ApplicationMasterClass.AssetFieldClass Assets)
{
  var query = @"
  Select 1 as id, 'Open 1st' as MenuLabel
  UNION
  Select 2 as id, 'Open 2nd' as MenuLabel
  ";
  return query;
}
```

- Write a OnClick for Button field and use the Id of the row form to perform a redirect. Here we have just displayed a Alert.

```
public static void OnClick(AppPress p, ApplicationMasterClass.AssetRowClass.OpenFieldClass AssetManager)
{
  p.AlertMessage("Clicked "+ AssetManager.FormData.id);
}
```

- Edit the Skin Master.html in Skins Folder and add following wherever you want the menu items to appear. You will need menu\_Open.png in the given path to display the menu icon.

```
<!--|AppPress.Asset.Begin|-->
<!--|AppPress.Asset.RowBegin|-->
<li prefix="AppPressDemoMenus">
    <!--|AppPress.Open.Begin|--><a id='AppPressId' onclick='AppPressOnClick' class="open"><!--|AppPress.Open.End|-->
    <i></i>
    <!--|AppPress.MenuLabel.Begin|--><span>AppPressValue</span> <!--|AppPress.MenuLabel.End|-->
    </a>
</li>
<!--|AppPress.Asset.RowEnd|-->
<!--|AppPress.Asset.End|-->
```

## How to Invoke click of Button from JavaScript

For example Click on bar in Bar Chart shows an alert containing the name of Bar.

GetOnClickJSScript

## How to hide a Column in FormContainerGrid

To hide a column make all fields for the column Hidden. This will hide the complete column automatically.

For Example Hide Column in FormContainer in Controls Page

## How to Create a Scalar User Control

Field in a form replaced by Single Field from UserControlScalar form

For example Create a Reusable Pickone of style AutoComplete to show Active Users of the System

## How to Create a Merge Form

Field Replaced by Multiple Fields with Insertion Point in Destination Form

For example create a form containing linked Country State City pickone fields and use it in other forms

## How to Create a Plugin Form

Field Replaced by Multiple Fields with Insertion Point in Source Form

For example extend Permission Page with Report Permissions

## How to Create an Embedded Form

A FormContainer containing single child form. The Table bound to child form has a primary key. The Form containing the Embedded form has a field of same name as Embedded form which is a foreign key to child form table.

For example create a Discussion Form reusable in other forms. In this example we

- Create a Reusable Discussion Form
- Add that to Controls Page in AppPress Demo Application
- Create a Popup to add a row to Controls Page without the discussion form

If any other form binds to the Form Table and does not have a EmbeddedForm field, then add a Hidden field of Type EmbeddedForm. This will add a default value for the column.

## How to Add Forms Dynamically

Forms can be added during Initialization. No classes are generated for such forms therefore backend logic cannot be written for the form. As an Example lets extend Controls Form in AppPressApplication dynamically by adding a field of Type Text Name DText.

in Applnit.cs before calling InitAppPress add following code

```
var formFields = new List<FormField>();
var formField = new FormField("DText", FormDefFieldType.Text);
// Set Field Properties here. The properties are same as in XML
formField.Required = true;
formFields.Add(formField);
var formDef = new FormDef(FormType.MergedForm, "DTextForm", formFields);
// Add Form Properties here. The Properties are same as in XML
formDef.MergeIntoFormName = "Controls";
Settings.ApplicationFormDefs.Add(formDef);
```

Now in Controls link from left menu you will see a DText field in the end

## After some changes in Form XML I am getting too many errors

This happens because the Text Transformation failed. Refer to [Transforming TT Files](#)

## How to override functionality of Add, Modify, Delete and Save buttons generated in FormContainerGrid

When Application is running in Developer mode click on code Dev link at bottom of form containing the FormContainerGrid. There you will see template of action functions. Copy the function and paste into your Logic code. Uncommenting the commented code will retain the functionality of the buttons. Make changes as needed.

```
public static void OnClick(AppPress a, ControlsClass.FormContainerGridWithPopupPopupClass.SaveFormContainerGridWithPopupFieldClass
SaveFormContainerGridWithPopup)
{
    //a.BeginTrans();
    //try {
    //SaveFormContainerGridWithPopup.FormData.Save(a);
    //SaveFormContainerGridWithPopup.FormData.RefreshContainer(a);
    //a.ClosePopup();
    //a.CommitTrans();
    //}
    //catch (Exception)
    //{
    //a.RollbackTrans();
    //throw;
    //}
    throw new NotImplementedException();
}

public static void OnClick(AppPress a, ControlsClass.AddFormContainerGridWithPopupFieldClass AddFormContainerGridWithPopup)
{
    //ControlsClass.FormContainerGridWithPopupPopupClass.Popup(a,null,null);
    throw new NotImplementedException();
}

public static void OnClick(AppPress a, ControlsClass.ModifyFormContainerGridWithPopupFieldClass ModifyFormContainerGridWithPopup)
{
    //ControlsClass.FormContainerGridWithPopupPopupClass.Popup(a,ModifyFormContainerGridWithPopup.FormData.FormContainerGridWithPopup.GetSingleSelect
ion().id,null);
    throw new NotImplementedException();
}

public static void OnClick(AppPress a, ControlsClass.DeleteFormContainerGridWithPopupFieldClass DeleteFormContainerGridWithPopup)
{
    //DeleteFormContainerGridWithPopup.FormData.Delete(a);
    throw new NotImplementedException();
}
```

## How to Send Email using AppPress

Use following functions to send email. Parameters to the functions are self explanatory

<code>a.SendEmail</code>	To be used when you have “a” of type AppPress
<code>AppPress.SendEmail</code>	To be used when you don't have “a”. Typically used from threads.

You will need to configure `mailSettings` in `web.config` for email server

## How to Prompt User for confirmation while executing Code Behind

use

```
a.PromptClient(string message)
```

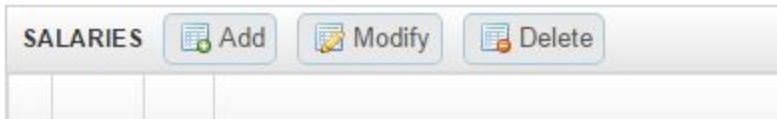
This will show a confirmation popup to user. If user presses cancel the popup will be removed execution will stop. If user presses ok popup will be removed and execution will continue from next statement after `PromptClient` call.

## How to hide Sidebar Menu on Page load

Use following code in Init of root form

```
p.ExecuteJSScript("setTimeout('$\\'.sidebar-toggle\\').click();',1);");
```

## How to move FormContainer Caption Buttons to Left on entire Page



Use following code in Calc of FormContainer field

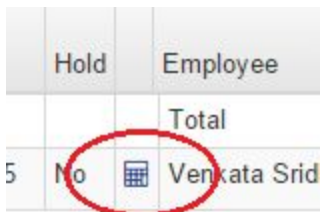
```
p.ExecuteJSScript("$.AppPressFormContainerFields').css('float','none');");
```

## How to set column width in Grid

Set the Width attribute in fields in RowFields

```
<FormContainerGrid Name="Salaries" >
  <RowFields Height="400">
    ...
    <Text Name="Month" Width="60">
  </Text>
    ...
  </RowFields>
</FormContainerGrid>
```

## How to show button as icon



Set style to link, Label as blank and class for the icon

```
<Button Name="Calculation">
  <CSSClass>fa fa-calculator</CSSClass>
```

```
<Label></Label>
<Style>Link</Style>
</Button>
```

## Using Field Groups

For long forms it is convenient to group fields.

```
<Form Name="FormWithFieldGroups">
  <Fields Group="Group1">
    <Field>
      ...
    </Field>
    ...
  </Fields>
  <Fields Group="Group2">
    <Field>
      ...
    </Field>
    ...
  </Fields>
  <Fields Group="Group3">
    <Field>
      ...
    </Field>
    ...
  </Fields>
</Form>
```

In TT build a Groups enum is generated in the form Class

```
public enum Groups {
    Group1,
    Group2,
    Group3,
}
```



## In Code use

```
public static void Init(AppPress a, FormWithFieldGroupsClass FormWithFieldGroups)
{
    FormWithFieldGroups.GroupHide(FormWithFieldGroups.Groups.Group1);
    FormWithFieldGroups.GroupShow(FormWithFieldGroups.Groups.Group1);
    FormWithFieldGroups.GroupReadOnly(FormWithFieldGroups.Groups.Group1);
}
```

## How to change height width or other attributes of text fields

Use ControlClass property in the field

### Example

```
<Text Name="Subject" >
    <ControlStyle><![CDATA[width:50%;]]></ControlStyle>
    <Required></Required>
    <MaxChars>200</MaxChars>
</Text>
```

## How to organize checkbox options for PickMultiple as a Grid

Use ControlStyle and PartStyle property in the field. This will work for both InlineCheckboxes and PopupCheckboxes styles

### Example

```
<PickMultiple Name="Subject" >
    <ControlStyle><![CDATA[width:1000px;]]></ControlStyle>
    <PartStyle><![CDATA[float:left;width:200px;]]></ControlStyle>
</PickMultiple>
```

Select Options

☐ Levi Strauss
☐ DBS
☐ NCS
☐ Accenture
☐ T Systems
☐ URA
☐ CB GFTS CT
☐ GFTS Asia
☐ East
☐ CIMB
☐ HP-GDBA
☐ Motorola
☐ Tatum (KL) Sdn Bhd
☐ NTT COM
☐ GIIS
☐ ALI
☐ Sungard
☐ Xtrategic
☐ EMA
☐ CITIBANK
☐ ZPAP

☐ VISA
☐ Oracle
☐ MPA
☐ International Baccalaureate (IB)
☐ OCBC
☐ NTT Data
☐ CB ICG
☐ NTT Pasir Panjang
☐ East ePDC
☐ HP-COM
☐ HP-PACA
☐ OCBC 2
☐ Dell
☐ PAS
☐ SINDA
☐ EDS
☐ Kellog
☐ HP Cathay Pacific
☐ Bridgestone
☐ IBO
☐ PRINGLES INT'L OPERATION SARL

☐ Starhub
☐ NLB
☐ JLT Asia
☐ GIC
☐ Hoya Lens
☐ Economic Dev. Board
☐ CB GFTS O&T
☐ PA
☐ Averis
☐ HP-DSMY
☐ HP-SHELL
☐ PS
☐ EMERIO
☐ Rotary Engineering
☐ NTT-Bridgestone
☐ Getronics
☐ HP Philippines
☐ Fuji Xerox
☐ NTT DATA Getronics Corporation
☐ PECTRA
☐ Bancnet
☐ VANLI

☐ UOB
☐ M1
☐ Infocomm Development Authority Singapore
☐ DBS (Indonesia)
☐ NKF
☐ Lenovo
☐ CB GCT
☐ Mapletree @ Alexandra
☐ Bank Negara Malaysia
☐ HP-ES
☐ Maybank
☐ SINGTEL
☐ UTAC
☐ Multiple-Client
☐ Mendaki
☐ Zuellig
☐ HP Hong Kong
☐ CGH/Keppel
☐ Pactera
☐ FrieslandCampina Service Centre Asia Pacific SD
☐ Ayala Land Inc
☐ QCCU

Ok

## How to change font size for a grid

Use control style as in following example

```

<Form Name="ViewCTCs">
  <Fields>
    <FormContainerGrid Name="MyCTCs">
      <ControlStyle><![CDATA[font-size:10px]]></ControlStyle>
      <Label>My CTCs</Label>
      <RowFields Height="600" Sortable="true">
        <Text Name="Employee" Width="150px"></Text>
        <Text Name="StaffId"></Text>
      </RowFields>
    </FormContainerGrid>
  </Fields>
</Form>

```

```

        <DateRange Name="Date" >
            <Contiguous></Contiguous>
        </DateRange>
    </RowFields>
</FormContainerGrid>
</Fields>
</Form>

```

## How to add reordering functionality to a Grid

In this example we have to allow movement of rows within a group. In Move up we exchange id of selected row with id of previous row and in down exchange with next row. After that field is refreshed to show the changes in UI.

COLUMNS	Group	CSV Column Name	Display Name	Hide If Zero Blank
<input type="checkbox"/>	1	Basic	Basic	No
<input type="checkbox"/>	1	HouseRentAllowance	HouseRentAllowance	No
<input checked="" type="checkbox"/>	1	ConveyanceAllowance	ConveyanceAllowance	No
<input type="checkbox"/>	1	SpecialAllowance	SpecialAllowance	No
<input type="checkbox"/>	1	FoodCoupons	FoodCoupons	No
<input type="checkbox"/>	1	MedicalAllowance	MedicalAllowance	No
<input type="checkbox"/>	1	MobileAllowance	MobileAllowance	No
<input type="checkbox"/>	1	OtherAllowance	OtherAllowance	No
<input type="checkbox"/>	1	LOPDays	LOPDays	No
<input type="checkbox"/>	1	WorkingDays	WorkingDays	No
<input type="checkbox"/>	1	DaysWorked	DaysWorked	No
<input type="checkbox"/>	1	GrossSalary	GrossSalary	No
<input type="checkbox"/>	1	ProvidentFund	ProvidentFund	No
<input type="checkbox"/>	1	TDS	TDS	No
<input type="checkbox"/>	1	ProfessionalTax	ProfessionalTax	No
<input type="checkbox"/>	1	OtherDeduction	OtherDeduction	No

```

<Form Name="SalaryColumnMapping">
    <Fields>
        <FormContainerGrid Name="Columns">
            <ContainerStyle><![CDATA[float:left]]></ContainerStyle>
            <TableName>SalaryColumnMapping</TableName>

```

```

<RowFields Height="500">
  <SelectRow></SelectRow>
  <Number Name="Group">
    <Static></Static>
    <SubmitIfStatic></SubmitIfStatic>
    <Decimals>0</Decimals>
  </Number>
  <Text Name="CSVColumnName"></Text>
  <Text Name="Label"></Text>
  <Checkbox Name="HideIfZeroBlank"></Checkbox>
</RowFields>
<PopupFields>
  ...
</PopupFields>
<ContainerFields>
  <Button Name="Up"></Button>
  <Button Name="Down"></Button>
</ContainerFields>
</FormContainerGrid>
</Fields>
</Form>

```

```

public static void Move(AppPress a, SalaryColumnMappingClass.ColumnsFieldClass columns, bool up)
{
  var selectedFormData = (SalaryColumnMappingClass.ColumnsRowClass)columns.GetSingleSelection();
  if (selectedFormData.Group.val == null)
    throw new AppPressException("Cannot Move Columns if Group is blank");
  var prevId = a.ExecuteInt32(@"Select Id From SalaryColumnMapping Where `Group`=" + selectedFormData.Group.val.Value + " and id "
+ (up ? "<" : ">") + selectedFormData.id + " Order by Id " + (up ? "desc" : "asc"));
  if (prevId == null)
    throw new AppPressException("Already on " + (up ? "top" : "bottom") + " of group");
  var id = int.Parse(selectedFormData.id);
  a.BeginTrans();
  try
  {
    // Exchange Ids
    a.ExecuteNonQuery(@"Update SalaryColumnMapping Set Id=" + (id + 10000) + " Where id=" + prevId.Value);
    a.ExecuteNonQuery(@"Update SalaryColumnMapping Set Id=" + prevId.Value + " Where id=" + id);
    a.ExecuteNonQuery(@"Update SalaryColumnMapping Set Id=" + id + " Where id=" + (id + 10000));
    // Exchange Ids in formData
    var prevFormData = columns.val.Find(t => t.id == prevId.ToString());

```

```

        selectedFormData.id = prevId.ToString();
        prevFormData.id = id.ToString();
        a.CommitTrans();
    }
    catch (Exception)
    {
        a.RollbackTrans();
        throw;
    }
    columns.Refresh(a);
}

public static void OnClick(AppPress a, SalaryColumnMappingClass.UpFieldClass Up)
{
    var columns = Up.FormData.Columns;
    Move(a, columns,true);
}

public static void OnClick(AppPress a, SalaryColumnMappingClass.DownFieldClass Down)
{
    var columns = Down.FormData.Columns;
    Move(a, columns,false);
}

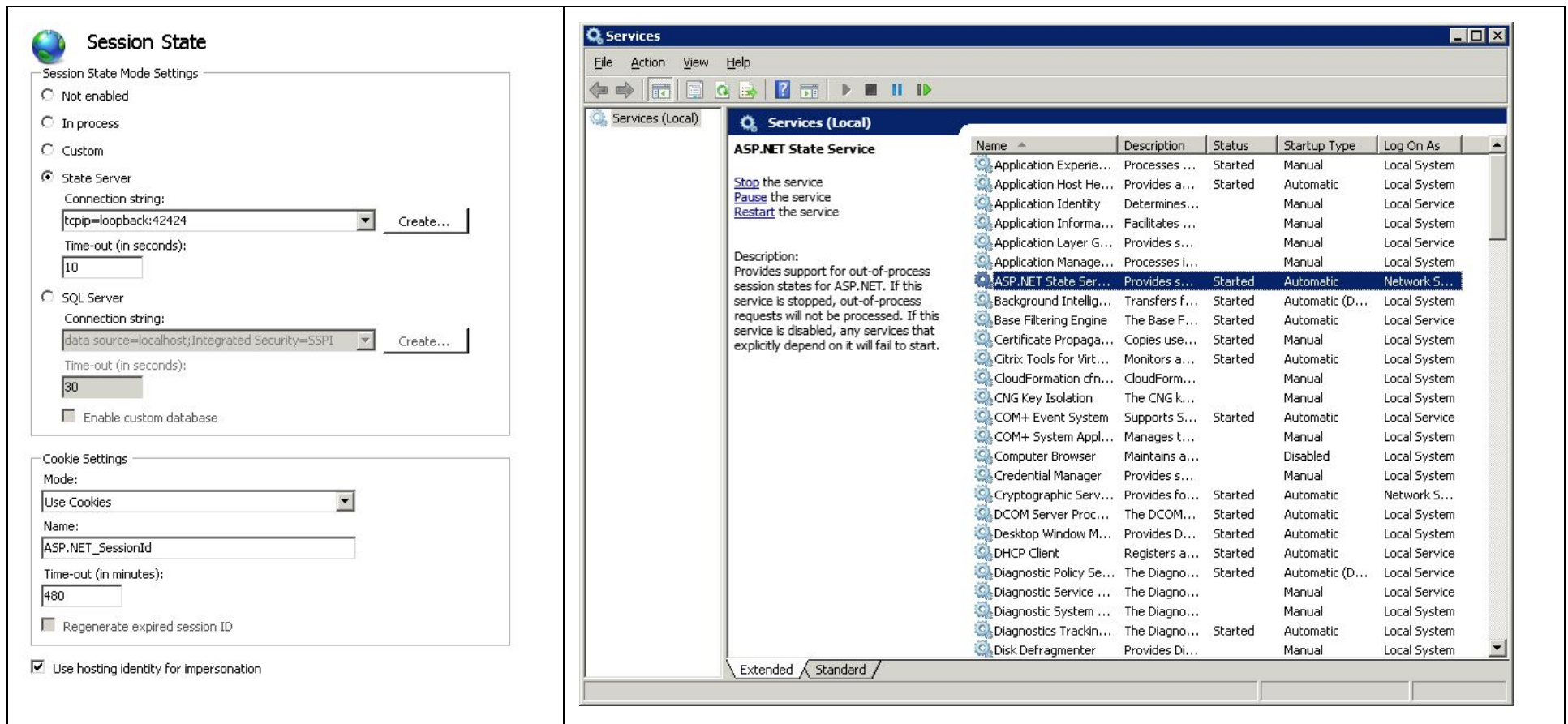
```

# Best Practices

## Using LoginUserId from SessionData versus using FormDataId

## Recommended Settings for AppPress based application in Production for IIS

- Minimum Configuration Windows Server 2008+, IIS 7.0+, NET4.0+
- Use different Application Pool for each Application
- Recommended Session State Settings  
You will need to start ASP .NET State Server in Services and set its startup to Automatic



## Appendix 1

### Formats of Shortcut Key

Modifier[+Modifier..]+Key

The valid modifiers are

- Ctrl
- Alt
- Shift
- Meta

You can specify a sequence without a modifier as well - like this...

X

The valid Keys are

- All alpha/numeric keys - abc...xyz,01..89
- Special Characters - Every special character on a standard keyboard can be accessed.
- Special Keys...
  - Tab
  - Space
  - Return
  - Enter
  - Backspace
  - Scroll\_lock
  - Caps\_lock
  - Num\_lock
  - Pause
  - Insert
  - Home
  - Delete
  - End
  - Page\_up
  - Page\_down
  - Left
  - Up
  - Right
  - Down
  - F1
  - F2
  - F3
  - F4
  - F5
  - F6
  - F7



- F8
- F9
- F10
- F11
- F12

The key is case insensitive. So “a” and “A” are same

**Examples:**

Ctrl+A

Ctrl+Shift+F

Ctrl+F10

Alt+=

# Appendix 2

## Release Notes

Removed UserControlScalar Column Hide function FormContainer HideColumn(fieldName)  
Added PopupHeight and PopupPosition  
Added UpperCase, LowerCase, TitleCase for Text AutoUpload in MultiFileUpload Auto Purge of Unused Files in Database  
CKEditor to PDF  
Added RichTextCKEditorFull,RichTextCKEditorStandard,RichTextCKEditorAdvanced Style for TextArea  
PickMultiple AutoRefresh  
Correction for Pickone options caching  
Pickone Drop Down Value Caching  
Correction for FormContainerModify if there are 2 forms of same name Added IsStatic property in Field Added %Label% in F...  
Changed Default Fileupload size 2MB Improved ForeignKey Error  
Redirect from RemotePopup  
Error if SelectRow used outside RowFields  
Removed \* in Label for Readonly Fields  
Added %Count% in Label for FormContainerGrid to Show Count of Rows in Grid Added Heading for Button column in Grid

## Change Log 0.9.2.0

AppPress XML Schema: Label -> Label  
Correction for Security Error for Readonly Fields  
User friendly error message for Database ForeignKey Exception  
Added missing /span in Skin Generation  
Refresh FormContainers if any FormData Id is changed because save of new Form  
Readonly Buttons made to work  
Added TableExists Generated Code BeforeSave, AfterSave  
AppPress XML Schema: Added ContainerStyle  
Reformatted MySQL Unique Exception. Show as Duplicate  
BeforeSave and AfterSave check for AlertMessage  
Moved FileUpload JS Code to AppPress\_JS.tt  
Split Base Class FormContainerFieldValue to FormContainerDynamicFieldValue and FormContainerGridFieldValue  
ControlStyle for FormContainerDynamic  
Correction for pickmultiple styling  
AppPress XML Schema: Added PartStyle  
User friendly error message for LoadedException for TT  
Added Height for Pickmultiple Checkboxes

Correction for Remotepopup logic  
Moved Connection String to Applnit  
Correction for hidden on domain  
Audit of PickMultiple Height Width of Remote Popup  
PopupTitle for Modify  
Cancel PickMultiple will undo changes in Checkboxes  
Added PopupTitle in FormContainerGrid  
Remote Form Corrections  
Number Sorting  
Added HideBegin and HideEnd Markers to retain the container tag like tr while hiding the field  
Added readonly for popup.

## Change Log 0.9.1.0

Removed Spurious un-saved error on page change in browser  
Added Field group Feature  
Added Embedded Forms  
If FieldValidation Error then AjaxCalls should not work.  
Added ReadOnly for FormContainer fields.  
Get PickMultiple options from ForeignKey Added PickMultiple Example with Save  
OnChange of PickMultiple  
Added Pickone style Options for Checkbox  
Correction for Delete Message when no row is selected  
Added Sorting and Filtering for Grids  
Removed HardCoding of Id Column in Tables. Get the Column Name from PrimaryKey of the Table

## Appendix 3

Resources included with the AppPressApplication project

img\_Female.jpg

img\_Male.jpg