



Transitioning to Tempo: Demonstration with Tips and Tricks

Michael Chirlin
21/08/2014



Tip #1: Using Legacy Forms Designer to SAIL Form Plugin

Legacy Application Form

Application Details

* Name Contact Type

* Description Email

Addresses

Address 1	Address 2	City	State	Postal Code
<input type="radio"/> 264 George St		Sydney	NSW	2000

- Use the Portal to SAIL plugin freely available on the *Shared Components* record on Appian Forum

- Benefits
 - Quick win
 - Creates good examples for learning SAIL
 - Creates mobile enabled forms

Appian

News Tasks (9) Records Reports Actions

Michael Chirlin - Appian

Application Form - Tip #1: Using Legacy Forms Designer to SAIL Plugin

Application Details

Name * Contact Type

Description * Email

Phone

Addresses

Address 1	Address 2	City	State	Postal Code
<input type="radio"/> 264 George St		Sydney	NSW	2000



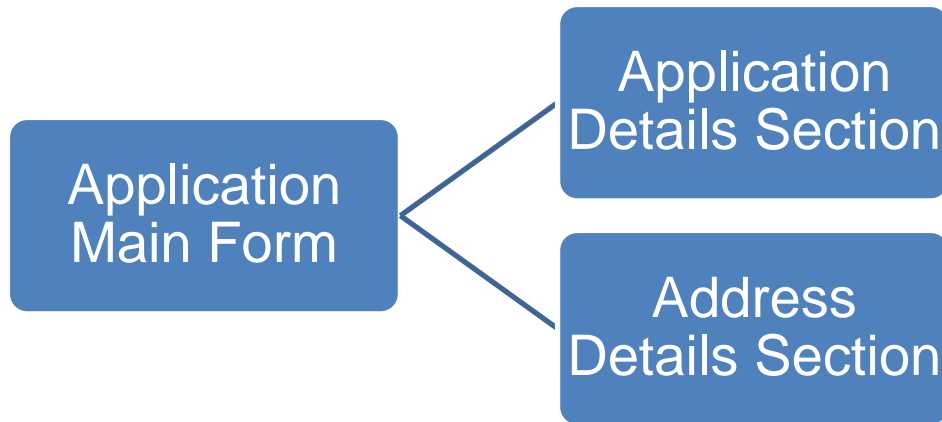
Tip #1: Using Legacy Forms Designer to SAIL Form Plugin

Converts	Does not Convert
All tempo enable fields	All non tempo enabled fields except editable grid, eg. tabbed section or message field
Default value and Save Into value for fields	Javascript (click/load/change events and validations)
Conditional show/hide expressions	CSS
Sections with one or two columns	Two column form format



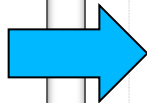
Tip #2: Modularize your SAIL rules

- Define a SAIL rule for each form section
 - Use descriptive names, like `ABC_customerContactInfoSection`
- Use a single SAIL rule in the record/report creation interface
 - This rule combines all the section rules
- Benefits
 - Easier to understand
 - Allows for parallel development
 - Isolate defects
 - Simplifies refactoring
 - Creates reusable sections



Tip #2: Modularize your SAIL rules

```
alformLayout(  
  label: "Application Form - Tip #1: Using Legacy Forms Designer to SAIL P",  
  firstColumnContents: {  
    a!sectionLayout(  
      label: "Application Details",  
      firstColumnContents: {  
        a!textField(  
          label: "Name",  
          required: true,  
          value: ri!application.name,  
          saveInto: {ri!application.name},  
          readOnly: false  
        ),  
        a!textField(  
          label: "Description",  
          required: true,  
          value: ri!application.description,  
          saveInto: {ri!application.description},  
          readOnly: false  
        ),  
      },  
      secondColumnContents: {  
        a!dropdownField(  
          label: "Contact Type",  
          required: false,  
          value: "null",  
          choiceLabels: {  
            "-Select Type-",  
            "Email",  
            "Phone"  
          },  
          choiceValues: {  
            "null",  
            "email",  
            "phone"  
          },  
        ),  
        a!textField(  
          label: "Email",  
          required: false,  
          value: ri!application.contactEmail,  
          saveInto: {ri!application.contactEmail},  
          readOnly: false  
        ),  
        a!textField(  
          label: "Phone",  
          required: false,  
          value: ri!application.contactPhone,  
          saveInto: {ri!application.contactPhone},  
          readOnly: false  
        ),  
      },  
    ),  
  },  
)  
if(  
  ruleUTIL_isBlank(  
    ri!application.addresses  
  ),  
  ...  
)
```



```
a!formLayout(  
  label: "Application Form - Tip #2: Modularize your SAIL rules",  
  firstColumnContents: {  
    rule!UG_applicationDetailsSectionTip2(application: ri!application),  
    rule!UG_applicationAddressDetailsSectionTip2(application: ri!application),  
  },  
  buttons: {  
    a!buttonLayout(  
      primaryButtons: {  
        a!buttonWidget(  
          label: "Add Address",  
          required: false,  
          value: "add",  
          saveInto: {ri!decision},  
          style: "NORMAL",  
          validate: false,  
          submit: true  
        ),  
        a!buttonWidget(  
          label: "Submit",  
          required: false,  
          value: "submit",  
          saveInto: {ri!decision},  
          style: "NORMAL",  
          validate: true,  
          submit: true  
        ),  
      },  
    ),  
  },  
)
```



```
a!sectionLayout(  
  label: "Application Details",  
  firstColumnContents: {  
    a!textField(  
      label: "Name",  
      required: true,  
      value: ri!application.name,  
      saveInto: {ri!application.name},  
      readOnly: false  
    ),  
    a!textField(  
      label: "Description",  
      required: true,  
      value: ri!application.description,  
      saveInto: {ri!application.description},  
      readOnly: false  
    ),  
  },  
  secondColumnContents: {  
    a!dropdownField(  
      label: "Contact Type",  
      required: false,  
      value: "null",  
      choiceLabels: {  
        "-Select Type-",  
        "Email",  
        "Phone"  
      },  
    ),  
  },  
)
```



Tip #3: Reusable SAIL Forms

- Define a single rule that can be used for both data entry and confirmation screens
 - Use readOnly value to hide action links or buttons
 - Use rule inputs to set readOnly or required parameters
- Benefits
 - Rules can be reused on dashboards and approval tasks
 - Utility form can be used throughout system (eg. multiple doc upload form)
 - Simplifies refactoring (changes will be uniform across system)

Application Form - Tip #3: Reusable Sail Forms

Application Details

Name *	Contact Type
<input type="text" value="Michael Chirlin"/>	<input type="text" value="-Select Type-"/>
Description *	Email
<input type="text" value="Description"/>	<input type="text" value="michael.chirlin@gmail.com"/>
	Phone
	<input type="text"/>

Application Form - Tip #3: Reusable Sail Forms

Application Details

Name	Michael Chirlin	Contact Type	<input type="text" value="-Select Type-"/>
Description	Description	Email	michael.chirlin@gmail.com
		Phone	



Tip #4: Creating Dynamic Forms

- Functionality earlier using javascript, off-form validation, or sequential user tasks can be built into a single form
 - Validation for text fields (eg. email address validation, length validation)
 - Section or form level validation for errors independent of specific input fields
 - Create new sections on the fly for unlimited data entry
- Benefits
 - More work can be done on a single form
 - No javascript necessary
 - Declutters process models
 - Unlimited possibilities

Application Form - Tip #4: Dynamic Forms

Application Details

Name *	Contact Type
<input type="text" value="Michael"/>	<input type="text" value="Email"/>
<small>Please enter a valid first and last name</small>	
Description *	Email
<input type="text" value="This description is just too long"/>	<input type="text" value="michael.chirlin"/>
<small>This description is just too long</small>	<small>Please enter a valid email address</small>
<small>The max length of this field is 20 characters.</small>	



Tip #5: Using Style Guide

- Guide covers
 - When to use buttons vs links
 - When to use button stylings, primary vs destructive
 - Which label position to use when, and why
 - Can be found on forum under documentation
- Benefits
 - Consistent look and feel across applications
 - Consistent and intuitive user interface

Application Details

Name *

Description *

Contact Type

Email

Phone

Application Details

Name *

Description *

Contact Type

Email

Addresses

Address 1	Address 2	City	State	Postal Code
No items available				



Tip #6: Creating Editable Grids

- Creating a dynamic editable grid
 - Combine the use of `applyComponents` with a custom rule created to display a single row
 - When using dynamic links to update rows from within the table remember to use and update `arrayVariable` parameter for `applyComponents`
- Benefits
 - Reduces user clicks
 - Less forms to design/manage
 - Simplifies process models
 - Reduces need for off form validation

The screenshot shows a web form titled "Application Details". It contains several input fields and a dynamic grid.

Form Fields:

- Name ***: Text input with value "Michael Chirlin".
- Description ***: Text input with value "Test Description".
- Contact Type**: Dropdown menu with value "-Select Type-".
- Email**: Text input with value "michael.chirlin@appian.com".

Addresses Section:

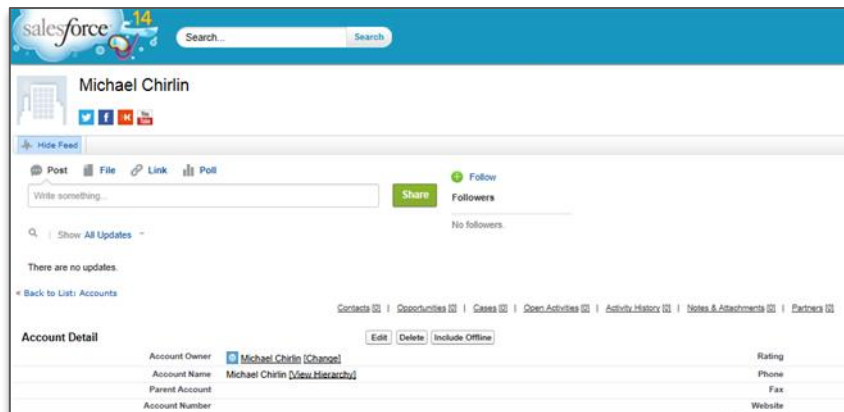
There is an "Add Address" button above a table. The table has columns: Address 1, Address 2, City, State, Postal Code, and a delete button (X).

Address 1	Address 2	City	State	Postal Code	
264 George St		Sydney	NSW	2000	X
1875 Explorer St		Reston	VA	20190	X

At the bottom right of the form is a green "Submit" button.

Tip #7: Using Connector Functions

- Connector functions facilitate
 - Querying data from external systems
 - Writing to external systems
 - Deleting from external systems
- Benefits
 - Incredibly quick implementation
 - Exposes external systems data to Appian



Application Form - Tip #7: Using Connector Functions

Application Details

Name *	Contact Type
Michael Chirlin	-Select Type- <input type="checkbox"/>
Description *	Email
Test Description	michael.chirlin@appian.com

Addresses

Address 1	Address 2	City	State	Postal Code
14/186 Sutherland St		Paddington	NSW	2021



Tip #7: Using Connector Functions



Tip #8: Using Linkable Images

- Can be used to
 - Create links to external sites, eg. linking to google maps
 - Create linkable logos
 - Combine with new `a!iconIndicators` to create dynamic grids

Applications		
Name	Description	Status
Application 1	Description 1	✓
Application 2	Description 2	!
Application 3	Description 3	⊘

Application Details

Name*
Michael Chirlin

Contact Type
-Select Type-

Description*
Test Description

Email
michael.chirlin@appian.com

Addresses

Add Address

Address 1
264 George St

Address 2

City
Sydney


State
NSW

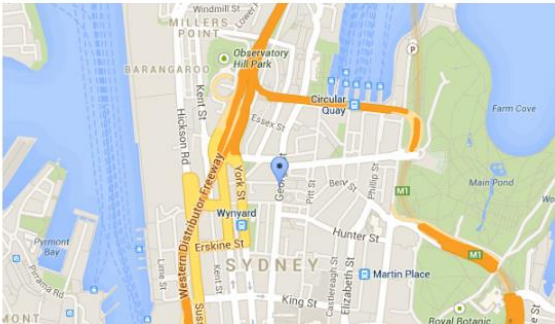
Postal Code
2000

[View Map](#)

×

Address # 1





Tip #9: Using Report Builder

- Can be used to
 - Create reports based on data store entities or records
 - Create pie chart, line chart, column chart, and grid reports
 - Configure default sort fields
- Benefits
 - Quick and easy
 - Simplifies coding
 - Teaches good coding practice for further report creation

The screenshot shows the 'Report Builder' window. Under 'Select Data', the 'Source Constant' is set to 'cons!UG_ADDRESSES_ENTITY'. Below this, there are two columns: 'city' and 'postalCode'. The 'city' column has a 'Count' aggregation and a 'Group By' checkbox checked. The 'postalCode' column has a 'Postal Code' label. Under 'Choose Visualization', the 'Pie Chart' option is selected. The 'Chart Labels' section shows 'Postal Code' and the 'Chart Data Series' section shows 'Field Name' with 'city' selected. A 'Preview' section at the bottom shows a partial pie chart with a purple segment labeled '2000'.

```
load(
  local!pagingInfo: alpageInfo(
    startIndex: 1,
    batchSize: -1,
    sort: alsortInfo(
      field: "city",
      ascending: true
    )
  ),
  with(
    local!datasubset: alqueryEntity(
      entity: cons!UG_ADDRESSES_ENTITY,
      query: alquery(
        aggregation: alqueryAggregation(aggregationColumns: {
          alqueryAggregationColumn(field: "city", aggregationFunction: "COUNT"),
          alqueryAggregationColumn(field: "postalCode", isGrouping: true),
        })
      ),
      pagingInfo: local!pagingInfo
    ),
    alpieChartField(
      series: {
        apply(
          alchartSeries(label: _, data: _),
          merge(
            index(local!datasubset.data, "postalCode", null),
            index(local!datasubset.data, "city", null)
          )
        )
      }
    )
  )
)
```



Tip #10: Tempo UI Utilities and Portal to SAIL Plugin

- Contains helper functions for
 - Default Record Views
 - Tempo Report Helpers
 - Process Optimization Template
 - Other Tempo-Enabled UI Components
- Benefits
 - Reuse of portal artifacts.
 - Translate design concepts that were commonly used on legacy portal dashboards.
 - Enable rapid development of default dashboards and reports.

Records / Applications
Sail Form Tip 9

In Progress Complete

Details

Status: Completed

Start Time Aug 21, 2014 6:07 AM GMT+10:00

Deadline

End Time Aug 21, 2014 6:07 AM GMT+10:00

On Time?

Task Checklist and Metrics

Submit

Application Details

Total Processing Time
0 Days

Processing Time Per Task (Days)

No data available

Active Tasks and Audit Trail

Status All

Task Name	Started	Assigned To	Owner	Task Status	Priority	Deadline	Completed	Performance
Submit Application Details	8/20/2014	Michael Chirlin	mchirlin				8/20/2014	
Submit Application Details	8/20/2014	Michael Chirlin	mchirlin				8/20/2014	

