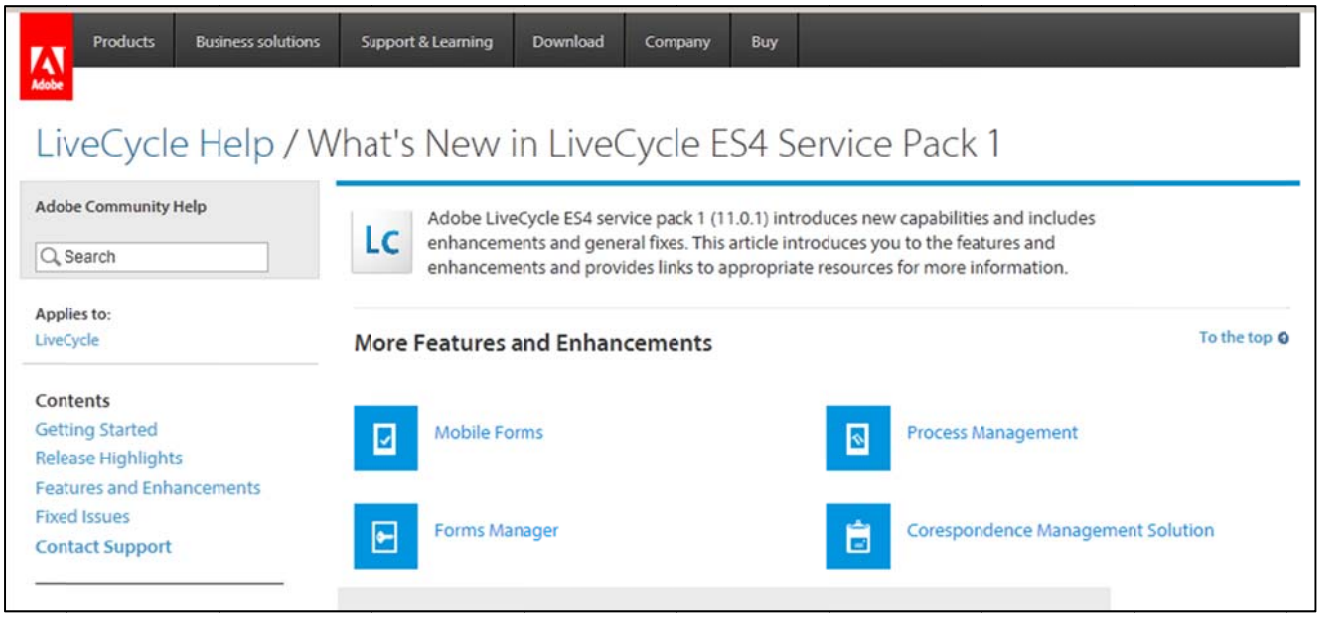
7 Best Practices for HTML Forms

Last Updated: 1-15-2014



The LiveCycle ES4 Service Pack 1 contains many updates for Mobile Forms. Some of the JavaScript techniques that we detail in Chapter 8 are no longer necessary. The JavaScript techniques in Chapter 8 will still work and are still valid, the updates that Adobe has made to Mobile Forms in Service Pack 1 now support more of the XFA constructs without modifying the JavaScript.

Adobe has more information on Service Pack 1 at this URL.

http://helpx.adobe.com/livecycle/help/whats-new-livecycle-es4-sp1.html#ID_MOBILE_WORKSPACE_SECTION

Targeting

As you saw previously, it's always a good idea to know the type of viewer your form fillers are using. Chapter 5, "Best Practices for PDFs," detailed the issues involved with different versions of Acrobat, Reader, and other PDF viewers. In this section, you'll explore the added dimension of web browser targeting and support.

Testing with Different Browsers

If you preview your forms only in Designer, you'll see only one possible rendering of your forms. You can preview your forms in various browsers with Adobe's Mobile Forms IVS application, which you saw in the last chapter. Since Designer's preview is based on WebKit, you should also view and test your forms in the following two browsers, which don't use WebKit:

- Microsoft Internet Explorer
- Firefox

Internet Explorer uses the Trident layout engine, and Firefox uses the Gecko engine.

Form Development Strategy

Adobe LiveCycle technology offers great power and flexibility for form and document generation. Before you start a form development project, you should consider your form development strategy. This section profiles two useful strategies that you'll see demonstrated in the exercises of Chapter 8, "Creating HTML Forms."

One Master File Strategy

Since LiveCycle Forms Pro enables you to render HTML and various types of PDFs, you can follow a One Master File Strategy (**Figure 7.1**).

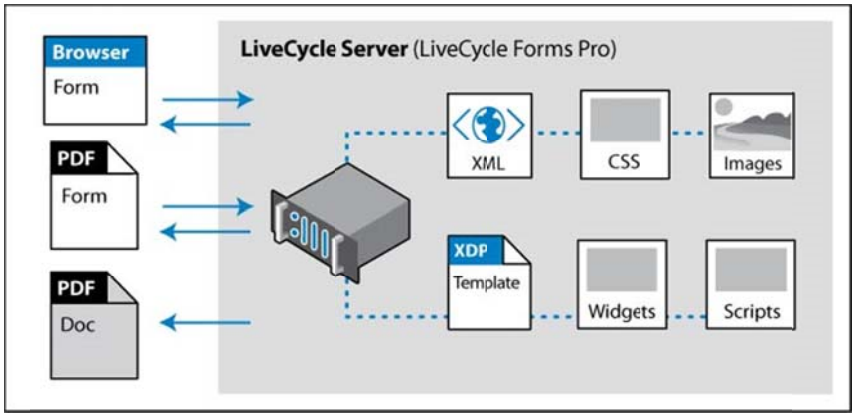


Figure 7.1 The One Master File Strategy makes form maintenance very easy.

This approach enables you to create one Designer XDP file and render it as HTML, interactive and dynamic PDF forms, and read-only PDF documents. A big advantage of this approach is that you need to maintain and update only one master template. To make this approach succeed, your master template needs to meet the following requirements:

- You must use data patterns and scripting events that work for both PDF and HTML.
- You must perform host detection and provide appropriate scripting for the different hosts.

You'll create a form that meets all these requirements in the next chapter. Although this strategy requires only one master XDP file, you can make the HTML form look different than the PDF form with a custom Mobile Forms profile.

Multiple Master File Strategy

If a One Master File Strategy is not appropriate, LiveCycle technology also supports a Multiple Master File Strategy (**Figure 7.2**).

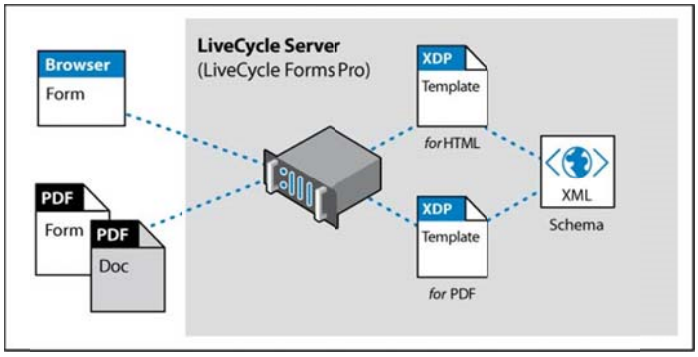


Figure 7.2 The Multiple Master File Strategy enables you to have two related documents that look very different from one another.

In this strategy, you create a set of two or more master Designer files. Each file can have its own look and feel, but they'll all be bound to the same XML Schema file. Since they all share the same data structure, it's easy to move XML data from one form to another. The Address Change form in the next chapter is part of a Multiple Master File Strategy.

Forms Graphics

This section will introduce you to some best practices for creating and maintaining your form graphics for HTML forms.

Form Fragments

As you learned in Part 2, "PDF Forms," fragments enable you to create a section of a form and reuse it across many different forms. When you update the fragment, all the forms that reference the fragment are also updated. The

combination of form fragments with LiveCycle's ability to generate HTML forms enables your organization to greatly reduce the costs associated with form creation and management.

For instance, an insurance company typically manages hundreds of forms in many different file formats. Each form may have an HTML, PDF, and Microsoft Word version. Adobe LiveCycle Forms Pro enables an organization to streamline the creation and management of its forms and documents at the enterprise level (**Figure 7.3**).

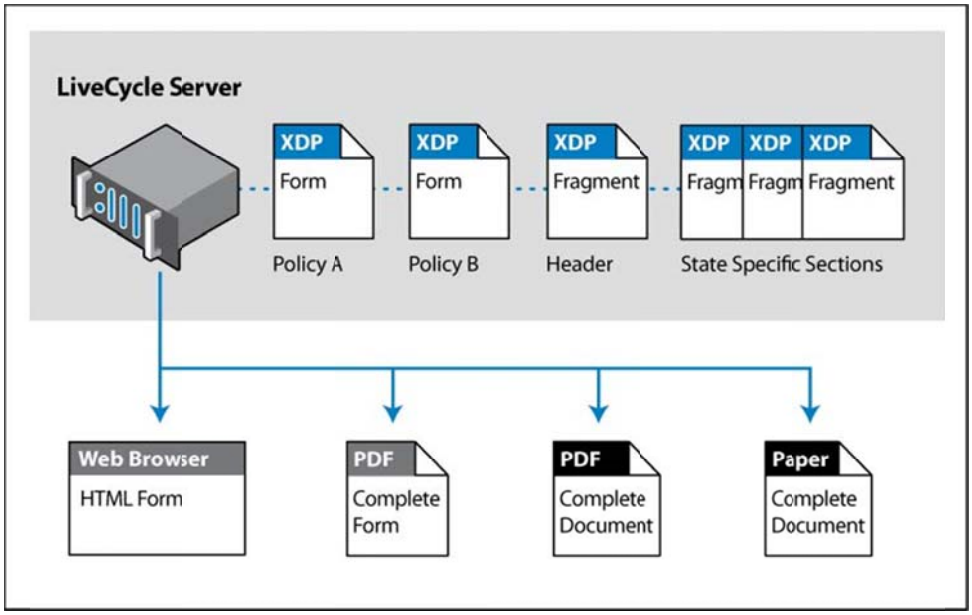


Figure 7.3 LiveCycle Server enables you to optimize your enterprise form management and develop master form components that can be used for many different business purposes.

Instead of creating and managing each form as a separate file, an organization can design a library of master form components. Figure 7.3 shows a LiveCycle Server system that meets all the following goals:

- Policy documents are saved as separate XDP files. LiveCycle Server can combine these policy documents into a final insurance package.
- The common header is saved as a separate form fragment that all policy documents reference.
- State specific sections are saved as separate fragments so they can be updated at one time and referenced in many different policy documents.
- These master LiveCycle forms can be used to generate any format your business requires, including HTML, PDF forms, and finalized PDF documents.

By organizing and rationalizing your form library and using LiveCycle Forms Pro, you'll realize the following business benefits:

- **Reduced costs:** Form development and management will be faster, easier, and cheaper because redundant work will be eliminated.

- **Greater consistency:** Your forms will look and function more consistently, which will improve the usability of your forms and the success of your form system.
- **Faster time to market:** You can quickly change a fragment based on a business update, and it will be deployed automatically to all forms once you publish it. You can change a fragment's content, layout, scripting, data bindings, and graphic style, and all forms that reference the fragment will be updated immediately.

Tables

If you're developing HTML forms in Designer, it's a good idea to keep your tables as simple as possible. Adobe advises that complex tables may have rendering issues in various web browsers.

In the first release of LiveCycle Forms Pro, there were a number of Designer's table features that were not supported in HTML forms. However, Adobe has been enhancing LiveCycle, and it now supports the following features in ES4 Service Pack 1:

- Adding sections to a table.
- Using the `colSpan` property in JavaScript.
- Using merged cells. Please note that even though merged cells will render in HTML, there are some inconsistencies in the thickness of the borders in tables with merged cells.

You should test your tables in various browsers with Adobe's Mobile Forms IVS application. In my testing, I noticed that row shading works for static forms (**Figure 7.4**) but doesn't work for dynamic forms.

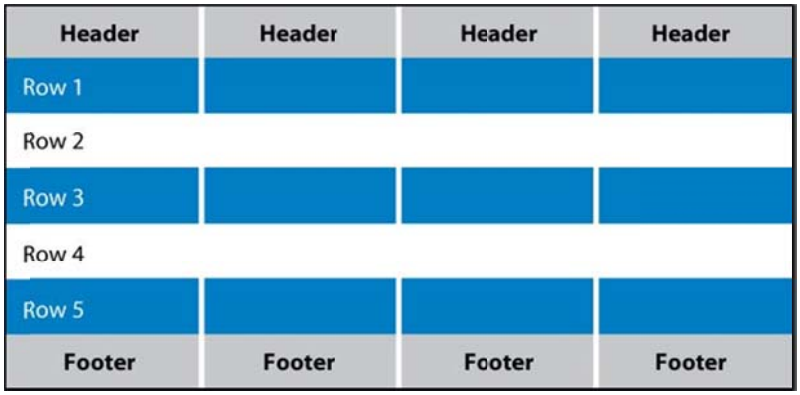


Figure 7.4 Row shading works on a static table but doesn't currently work on a dynamic table in an HTML form.

Signatures

Traditional PDF digital signatures are not supported in HTML forms. However, Adobe has added a new XFA object called the Scribble Signature. This new object enables users to sign a mobile form with their finger or a stylus. If the form is being accessed by a mobile device, the object will also gather a timestamp and

the geolocation of the user. The addition of this time and place information increases the validity of the signature. When a user taps on the Scribble Signature field, a signature dialog box will appear (**Figure 7.5**).

Figure 7.5 LiveCycle's new Signature Scribble object enables you to sign a form on a mobile tablet with your finger or a stylus.

Using Custom Profiles

As you saw in the last chapter, the Mobile Forms profile you use to render your HTML forms will affect how they look and function. In this section, you'll see a few examples of custom profiles. The first two custom profiles will illustrate how you can use CSS to change the visual aspects of your HTML forms. The last custom profile will show how you can use a custom widget to change the functional aspects of an HTML form field.

CSS

Chapter 5 showed how a standard Designer Text Field object will be rendered by LiveCycle Forms Pro as an HTML file that's similar to this structure:

```
<div class="field textfield myTextField">
	<div class="widget textfieldwidget textField">
		<input type="text">
	</div>
	<div class="caption">
		<svg>
			<text>myTextField</text>
		</svg>
	</div>
</div>
```

The first line includes three class selectors: `field`, `textfield`, and `myTextField`. You can apply Cascading Style Sheets to each one of these class selectors to set the visual style of your HTML forms. Of the three, the `textfield` class selector will be the one you use most in your form development. The following bullets will help you understand the hierarchy of these class selectors and how they relate to your form design:

- `field`: The `field` class selector will apply your CSS styles to all fields on the form, including check boxes, radio buttons, drop-down lists, and text fields. Because you'll likely want subtle differences in the styles you apply to these different form objects, you probably want to define only high-level attributes like a font setting to this class selector.
- `textfield`: The `textfield` class selector will apply your CSS styles to all text fields on the form so you can achieve a consistent form design and specify the unique graphic details of your text fields.
- `myTextField`: The `myTextField` class selector will apply your CSS styles to the specific text field named myTextField. You'll use this type of class selector only if you want a unique style applied to only one field on your form.

Because CSS affects the visual aspect of your forms, it's easiest to understand with a few visual examples applied to a simple form design.

Default profile

For this example, I created a form in Designer with two standard Text Field objects on a subform. I entered *value text* as each object's default text. If I render an HTML form with the default profile, LiveCycle Forms Pro will generate an HTML form that looks like **Figure 7.6**.

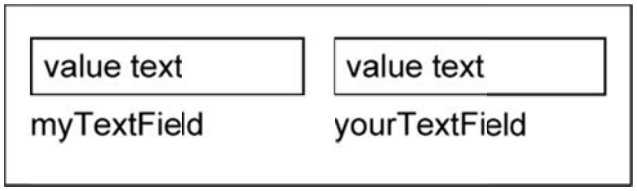


Figure 7.6 The example form rendered with the Adobe default profile.

The same Designer form can look very different if LiveCycle Forms Pro uses a custom profile.

Custom profile 1

The first custom profile includes a CSS file that specifically styles the first text field by referencing the `myTextField` class selector. This CSS file changes the font, font style, and font color of the text field **(Figure 7.7**).

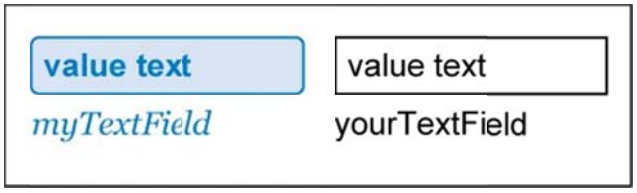


Figure 7.7 You can provide unique styles to a single form field by referencing that field's unique class selector.

```
.myTextField .widget {
        background-color: #d8e2f3;
        border-radius: 5px;
        border: 1px solid #007cc2;
}
```

```
.myTextField .widget input {
        color: #007cc2;
        font-weight: bold;
}
.myTextField .caption svg text {
        fill: #007cc2;
        font-style: italic;
        font-family: Georgia;
}
```

The first two blocks style the input field, and the last block styles the caption font.

Custom profile 2

The second custom profile includes a CSS file that references two other class selectors. The first class selector sets the background color of the subform, and the second class selector sets the visual properties of all the text fields (**Figure 7.8**).

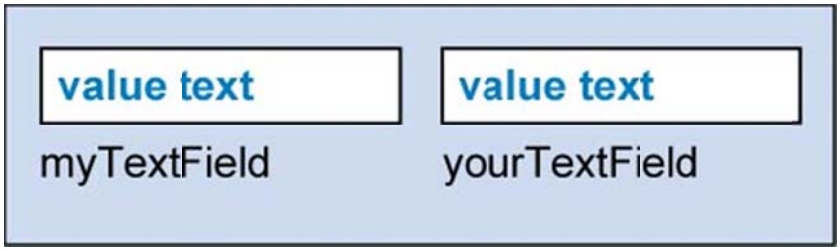


Figure 7.8 By using the `textfield` class selector, you can style all the text fields on a form.

```
.mySubform {
        background-color: #d8e2f3;
}
.textfield .widget {
        background-color: #ffffff;
}
.textfield .widget input {
        color: #007cc2;
        font-weight: bold;
}
```

You can use CSS files to maintain consistent fonts and font styles on your HTML forms even when the XDP file has different fonts and font styles.

Custom Widgets

Like style sheets, custom widgets also enable you to customize your HTML forms. Custom widgets are small applications that will override or enhance the default functionality of your form objects. You can create custom widgets in Adobe's CRX DE Lite interface using the framework provided by Adobe.

In this example, you'll see how a custom widget will improve the usability of your state drop-down lists for users of mobile tablets like the Apple iPad or Google Nexus. **Figure 7.9** shows a custom widget that changes the State drop-down list of the Address Change form.

Figure 7.9 Custom widgets enable a rich data capture experience on mobile tablet.

The State drop-down list can take up a great deal of screen real estate on a mobile tablet. In this example, the form field uses a custom widget that supports auto-completion. When a user types the first letter of a state, all the states that begin with that letter will appear. Custom widgets support JavaScript and the JQuery libraries. Adobe provides out-of-the box widgets that you can review and extend.

Data Submission

As you learned in Chapter 5, a LiveCycle Server can receive and process data submissions from a PDF form. You can also create a LiveCycle Server process that receives and processes data from an HTML form. There are many business benefits to processing form data submissions. **Figure 7.10** shows how a LiveCycle Server can process your data and use it to update databases and enterprise systems, create workflows, and render and archive documents.

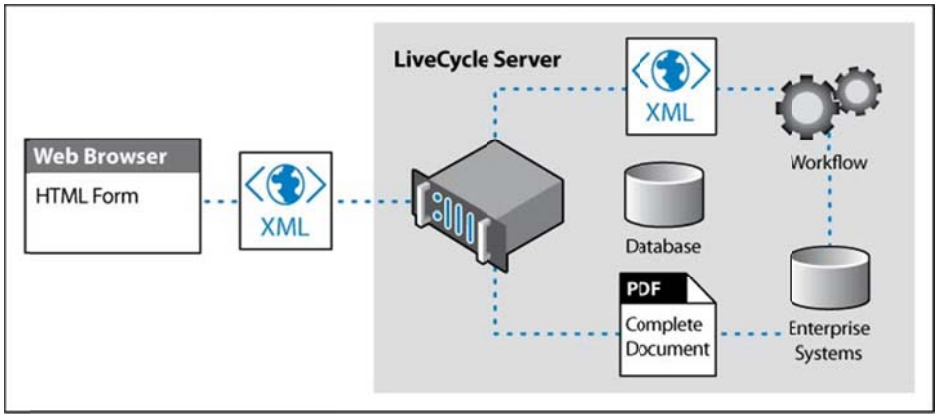


Figure 7.10 A LiveCycle Server will automate your business by processing form data and integrating it with your existing IT systems.

You create a LiveCycle process in LiveCycle WorkBench, which is part of LiveCycle Enterprise Suite. You'll learn more about this in Chapter 9, "LiveCycle Enterprise Suite." To call the process from an HTML form, you'll add a REST endpoint. REST stands for Representational State Transfer and is sometimes spelled ReST. It uses the HTTP protocol and has many similarities with web services, including the following:

- It's platform independent.
- It's language independent.
- It can easily be used across firewalls.

To call the LiveCycle process from your form, you'll need the exact and complete URL for your process. You can retrieve this in the LiveCycle Workbench application (**Figure 7.11**).

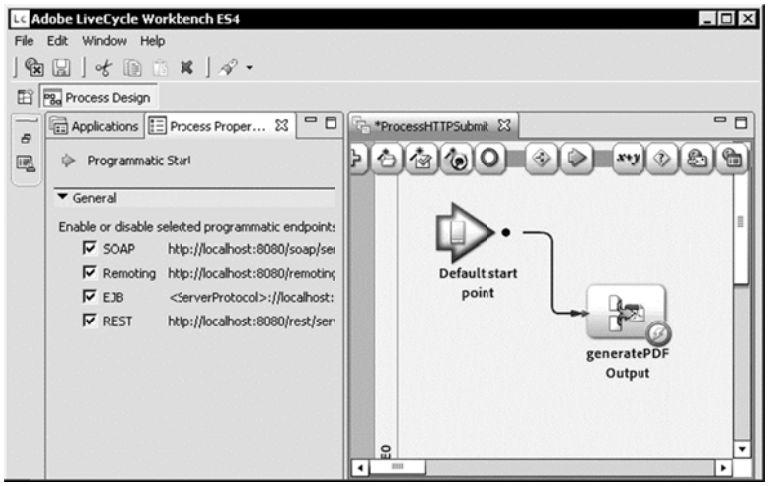


Figure 7.11 You can get the complete URL for a LiveCycle process by selecting the Default start point in LiveCycle Workbench.

The REST URL for your process will follow this format:

```
http://[server]:8080/rest/services/[ServiceName]/[Operation]:[version]
```

You can add a REST URL to two different button types in your Designer forms. Follow these steps to learn how to configure a standard Button object:

1. Open a new blank form in Designer by selecting File > New > Use A Blank Form.
2. Click Next in the New Form Assistant and click Finish to create your form.
3. Drag and drop a Button object from the Standard Object Library to your form.
4. Select the Field tab of the Object palette.
5. Set the Control Type to Submit. The Submit tab will appear.
6. Select the Submit tab and enter your REST URL in the Submit To URL field (**Figure 7.12**).
7. Click the Submit drop-down list and select XML Data (XML). This will result in your form data being submitted as XML when the button on your form is clicked.

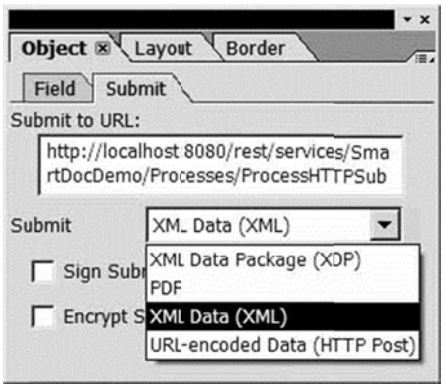


Figure 7.12 You can use a standard Button object as a submit button by setting its Control Type to Submit.

You can also use an HTTP Submit Button object to submit HTML form data to a REST URL. Follow these steps to learn how to configure this object:

8. Drag and drop an HTTP Submit Button object from the Standard Object Library to your form.
9. Select the Field tab of the Object palette.
10. Enter your REST URL in the URL field (**Figure 7.13**). Many other properties for this object already have default property settings, such as a caption, appearance, and highlighting, but you can change them if you want.

Figure 7.13 Designer includes an HTTP Submit Button that submits form data to a REST URL.

It's a best practice to validate your form data before you execute a form submission. You'll see an example of this in the next chapter.