Search Engine Optimization for Silverlight Applications

Ashish Shetty
Microsoft Corporation
October 2008

Applies to:
Microsoft® Silverlight™

Summary: This document describes some best practices for search engine optimization of Silverlight applications. These practices are designed to help developers make their Silverlight content discoverable on a search engine results page and to provide an acceptable experience for users who do not have Silverlight enabled.

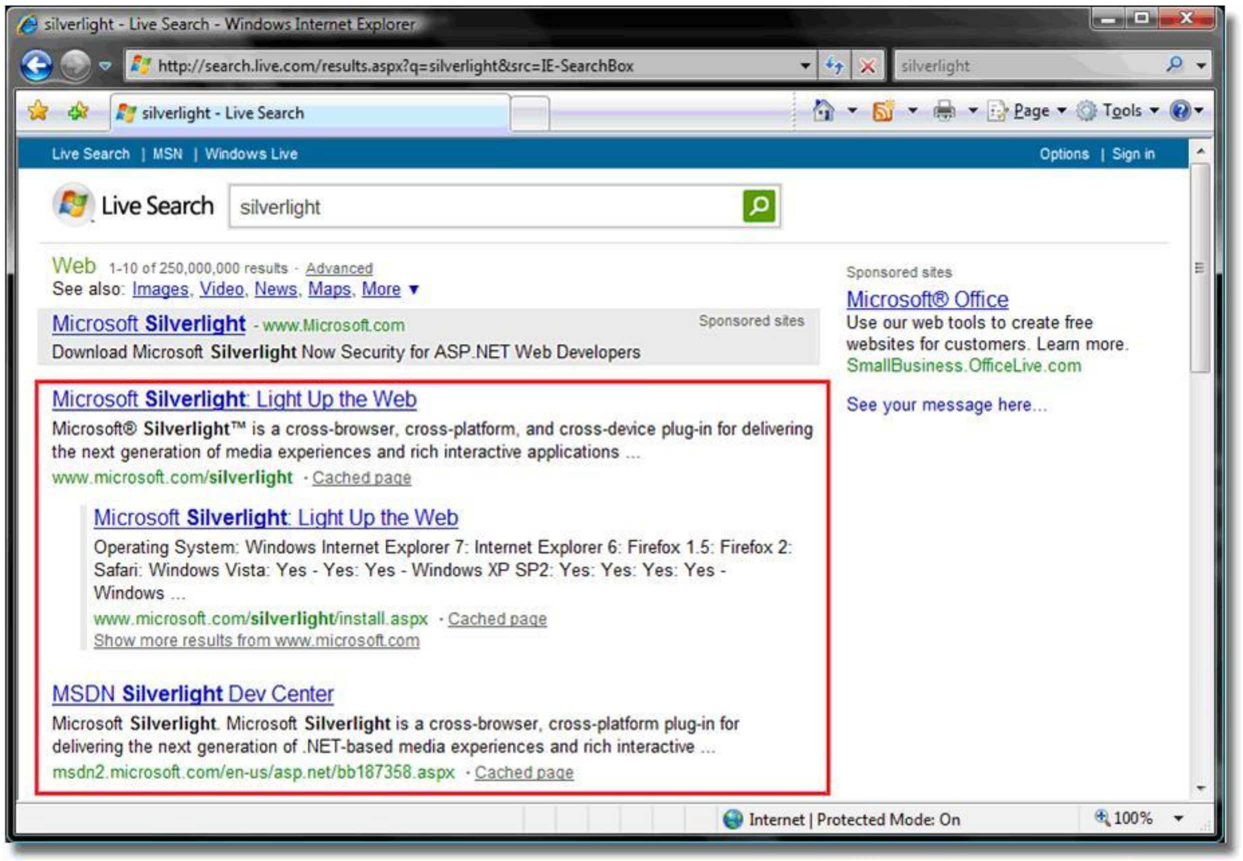
Legal Notice

Contents

Introduction

Search engines are geared toward recognizing HTML content. At this time, they do not recognize Silverlight content natively. This is similar to the search engine support for other objects on the HTML page, such as script blocks, CSS blocks, media files, and ActiveX controls. To make Silverlight content indexable by search engines, you can use approaches that search engines are already familiar with, such as combining the islands of Silverlight content with HTML metadata about that content.

The goal of search engine optimization (SEO) is to increase the chances that your page will appear in the main section of the search engine results page (outlined by the red rectangle in Figure 1), not the paid or sponsored results.


silverlight - Live Search - Windows Internet Explorer
http://search.live.com/results.aspx?q=silverlight&src=IE-SearchBox
silverlight
silverlight - Live Search
Page
Tools
Live Search | MSN | Windows Live
Options | Sign in
Live Search
silverlight
Internet | Protected Mode: On
100%

Web 1-10 of 250,000,000 results · Advanced
See also: Images, Video, News, Maps, More

Microsoft **Silverlight** - www.Microsoft.com
Download Microsoft **Silverlight** Now Security for ASP.NET Web Developers
Sponsored sites

Microsoft **Silverlight**: Light Up the Web
Microsoft® **Silverlight**™ is a cross-browser, cross-platform, and cross-device plug-in for delivering the next generation of media experiences and rich interactive applications ...
www.microsoft.com/**silverlight** · Cached page

Microsoft **Silverlight**: Light Up the Web
Operating System: Windows Internet Explorer 7: Internet Explorer 6: Firefox 1.5: Firefox 2: Safari: Windows Vista: Yes - Yes: Yes - Windows XP SP2: Yes: Yes: Yes: Yes - Windows ...
www.microsoft.com/**silverlight**/install.aspx · Cached page
Show more results from www.microsoft.com

MSDN **Silverlight** Dev Center
Microsoft **Silverlight**. Microsoft **Silverlight** is a cross-browser, cross-platform plug-in for delivering the next generation of .NET-based media experiences and rich interactive ...
msdn2.microsoft.com/en-us/asp.net/bb187358.aspx · Cached page

Sponsored sites
Microsoft® Office
Use our web tools to create free websites for customers. Learn more.
SmallBusiness.OfficeLive.com

See your message here...

Figure 1 - Search engine results page

Even though the search landscape is rapidly changing, with multiple competitors continuously improving and evolving how they implement search, SEO relies on some fundamental similarities among search engine algorithms.

How Search Engines Work

Search engines crawl, weight, and index Web page content. Crawling is done by a search robot that traverses the links in a Web site and captures the content. Search engines then use algorithms and heuristics to assign weights to Web pages. This information is used to build the search index, which is used to build a results page based on your query.

The main reason for a Web page to be highly ranked in search engine results is that the words on the page match the keywords that are used to search. The presence of dynamic and nonstandard elements such as script, style, object, and embed tags in the page is a challenge to search engines, and this is an area where they have traditionally not done well.

In this situation, search engines have to do the following:

- Download linked content and associate it with the source page.
- Parse, convert, execute, or render elements to obtain the same experience as the user viewing the page in a browser.

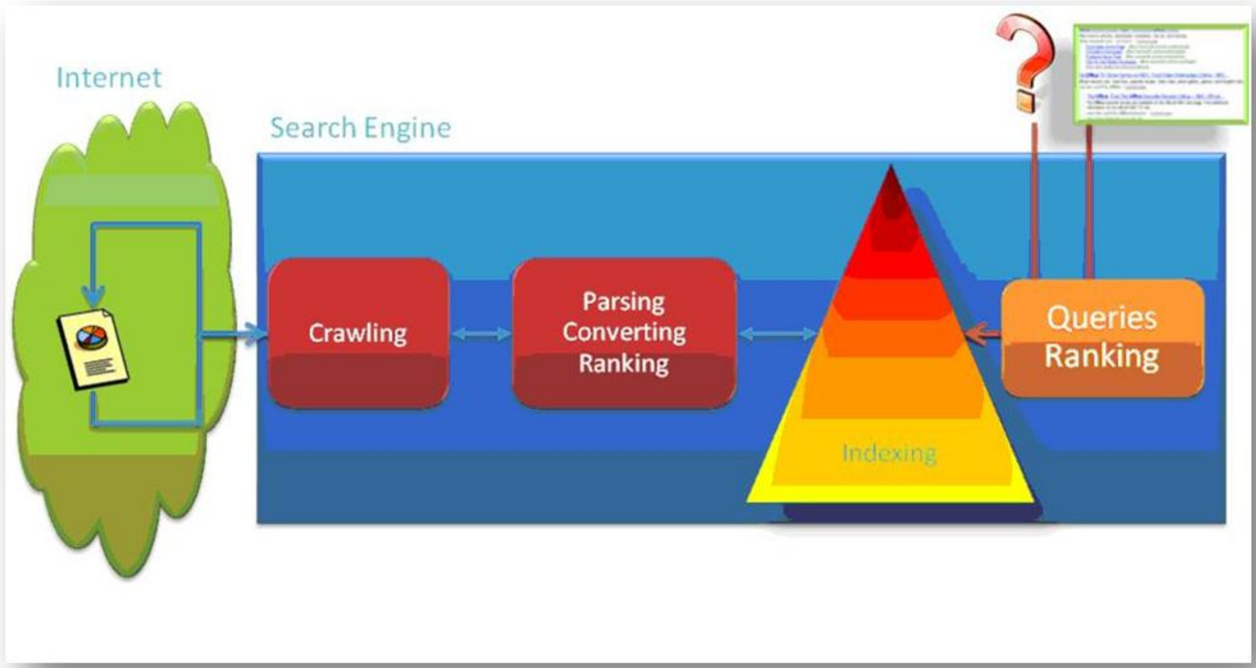


Figure 2 is a simplified view of how a search engine works.

Figure 2 – A simplified view of a search engine

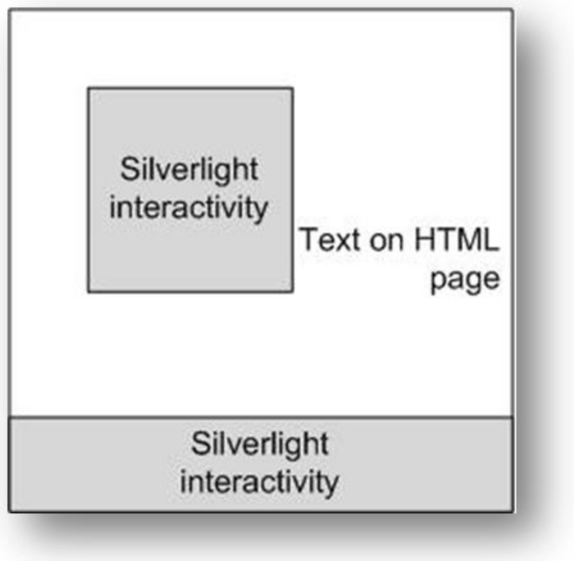
Approaches for Developing a Silverlight Application

If you are planning to build a Silverlight application, there are things you can do to make sure that your application is discovered and returned by search engines.

The following are some of the patterns that you can adopt for how your application coexists with the HTML content.

Mix HTML with Silverlight Content

This pattern involves mixing HTML text with Silverlight content in the same page so that it delivers richness in functionality and the native HTML content is consumable by search engines. To do this, consider designing your Silverlight content in such a way that it fits within, or around, a block of text. This looks like a grid of interacting components that fit around HTML text. An advantage of



this approach is that it ensures that your Silverlight interactivity is truly supporting the text, rather than hiding otherwise searchable text from search engines. Figure 3 shows a Web page with this approach.

Figure 3 – Islands of text and Silverlight interactivity

Use HTML Bridge to Generate Silverlight Content Dynamically

This approach is slightly harder to achieve and can be limiting to the Silverlight experience. You will have most success with this approach if you have existing XHTML content and want to enhance the experience with Silverlight. In this approach, the XHTML content is the base experience for search robots and down-level clients, while the Silverlight experience is reserved for consumers on client platforms capable of running Silverlight.

In this approach, the XHTML content has the full-fidelity experience for its target clients. It is still declared as nested alternate content within the **object** tag for the Silverlight plug-in. The one attribute that differentiates this pattern from the graceful degradation pattern is that in this pattern, the Silverlight application's UI is driven by the nested alternate content. In other words, application logic will use the DOM Bridge to get the nested alternate content from the **object** tag and use it to construct the Silverlight UI. This can be as simple as using XSLT to transform the XHTML to XAML, or perhaps using data binding to bind XAML UI properties to an object representing content from the extracted markup.

Graceful Degradation

In this document, we focus primarily on search engine optimization using the graceful degradation approach. In such a scenario, the Silverlight content is the primary experience for consumers, and the use of nested alternate content within the **object** tag serves as the down-level experience.

Search Engine Optimization Techniques for Silverlight Applications

The key consideration for making Silverlight content indexable by search engines is to use the approaches that are used for systems and users for which Silverlight is not enabled. Considerations include the following:

How the Web page with Silverlight content behaves in client/browser configurations such as Opera or Windows 98, which are not currently supported by Silverlight.

How the Web page behaves for customers who use accessibility programs such as screen readers and narrators.

How the Web page behaves for customers who use from a text browser such as Lynx, where no scripts can execute.

Presenting contextual metadata and alternate content that would make Silverlight content friendly to down-level users will also make it friendly to search engines.

When creating your Silverlight application, do not assume that all users will have Silverlight installed or have computers with the ability to install Silverlight. Prepare for how you would describe your application to these users.

Know your audience.

Plan on how you would describe your application to them.

Identify the keywords that you would use to connect with searchers.

The words that you use in your titles, page and section headers, body content, and alternate content play an important role in how the search engines find and index your content, and also how a user finds your content.

The following are some of the techniques you can use to optimize your search engine results and improve the experience for all users:

- Use a Descriptive Page Title
- Add Description Metadata
- Use a Meaningful Application Name
- Use the object Tag
- Specify Alternate Content for Silverlight
- Use createObject When Using Silverlight.js
- Test Down-Level Experiences

Use a Descriptive Page Title

Give your page a good title. Web page authors must update and customize the markup in the pages generated by Silverlight project templates in Visual Studio or Expression Blend. For the purposes of discussion, let’s assume you have a Silverlight application that provides interactive traffic maps for the Seattle area. Figure 4 shows bad (default) and good Web page titles.

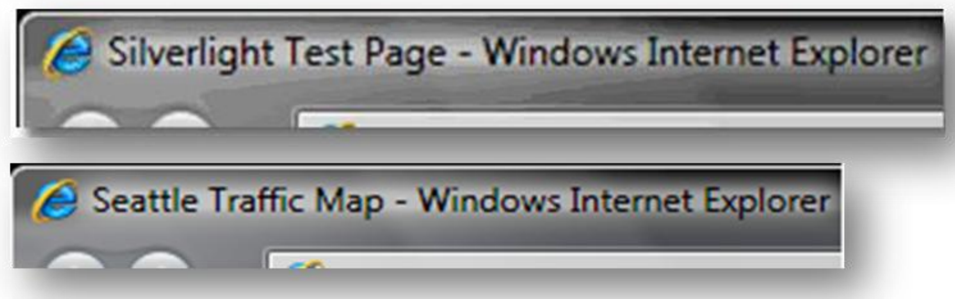

Silverlight Test Page - Windows Internet Explorer
Seattle Traffic Map - Windows Internet Explorer

Figure 4 – Bad and good page titles

Add Description Metadata

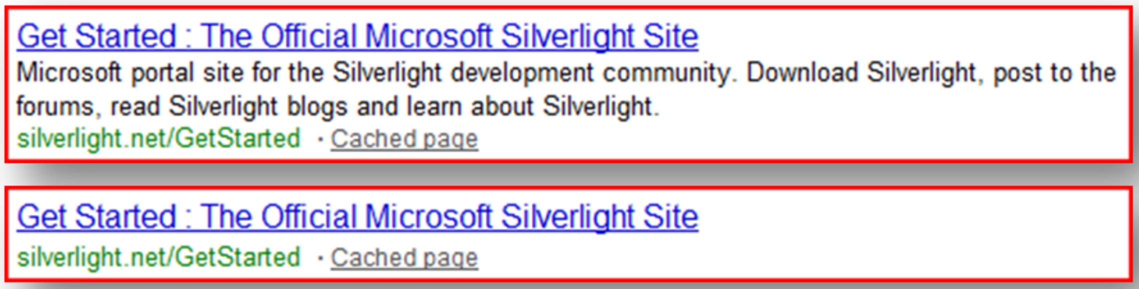
Keywords in your page's meta tag are not very useful for search engines to determine your page's rank. However, a page title and meta description tag (that is, a **meta** tag whose **name** attribute is set to "description") are extremely useful in ensuring that searchers who view your page on a results page associate it with content that they are looking for.

If you have a top-level Silverlight application that occupies the full extent of the browser's client area, or one that exists on your landing page, then you must have a meta description tag on your HTML page. The following code shows the format of the meta description tag.

```
<head>
  <meta name="description"
    content="Microsoft portal site for the Silverlight development community..." >
...
</head>
```

If you have a page with a lot of text content that contains relevant keywords, then you can omit the meta description. The search engine will show a portion of your page content on the results page, and any stub description may actually prove counterproductive.

The following figure shows a sample search page entry with a description, and how it would appear if the meta description tag were not present.


Get Started : The Official Microsoft Silverlight Site
Microsoft portal site for the Silverlight development community. Download Silverlight, post to the forums, read Silverlight blogs and learn about Silverlight.
silverlight.net/GetStarted · Cached page

Get Started : The Official Microsoft Silverlight Site
silverlight.net/GetStarted · Cached page

Figure 5 – Search results for a page with and without a meta description tag

Use a Meaningful Application Name

Giving a useful name to your application is another way to help the search engines find your Web page. For example, an application that provides traffic maps for the Seattle area could be named SeattleTrafficMaps.xap.

Even if your application was built using a different name, it is easy to change the name, for example, from MyTestSLApp.xap to SeattleTrafficMaps.xap. Changing the name back at a later time is a simple operation if that name is not referred to elsewhere in your code. By default, there are no dependencies of this kind in the Silverlight templates.

Use the object Tag

The **object** tag in HTML is designed so that if the main object cannot be loaded to display content, then browser clients will continue to look for alternative content within the **object** tag.

Silverlight content publishers must use the **object** tag (not the **embed** tag) to instantiate Silverlight. The following code shows how you can add the **object** tag.

```
<object type="application/x-silverlight-2"
        data="data:application/x-silverlight,"
        width="..." height="...">
  <param name="source" value="SeattleTrafficMaps.xap" />
  <!-- Other parameters, if any -->
  ...
  <!-- The “Get Silverlight” messages and badge -->
  <p>This content requires Microsoft Silverlight.
    <a href="http://go.microsoft.com/fwlink/?LinkID=124807"
       style="text-decoration: none;">
      <img src="http://go.microsoft.com/fwlink/?LinkId=108181"
           alt="Get Microsoft Silverlight"
           style="border-style: none"/>
    </a>
  </p>
</object>
```

Specify Alternate Content for Silverlight

The **object** tag for the Silverlight application must be supplemented with nested alternate content, namely the inner HTML that is displayed on systems where Silverlight is absent. The following code shows how you can do that.

```
<object type="application/x-silverlight-2"
        data="data:application/x-silverlight,"
        width="..." height="...">
  <param name="source" value="SeattleTrafficMaps.xap" />
  <!-- Other parameters, if any -->
  ...
  <!-- Nested alternate HTML content for search -->
  <h3>Traffic map of the Seattle-Puget Sound area</h3>
  <p>Up-to-the-minute traffic situation overlaid on the map of
     the Seattle-Puget Sound area, powered by
     <a href="http://maps.live.com">Live Maps</a>
  </p>

  <!-- Canned image representing the application contents -->
  <img src="SeattleTraffic_RushHour.jpg"
       alt="Seattle traffic at 5:30pm (evening rush-hour)" />

  <!-- The “Get Silverlight” message and badge -->
  <p>This content requires Microsoft Silverlight.
    <a href="http://go.microsoft.com/fwlink/?LinkID=124807"
       style="text-decoration: none;">
     <img src="http://go.microsoft.com/fwlink/?LinkId=108181"
          alt="Get Microsoft Silverlight"
          style="border-style: none"/>
    </a>
  </p>
</object>
```

Use createObject When Using Silverlight.js

The primary function of Silverlight.js is to provide a cross-browser, cross-platform means of constructing the right markup to get Silverlight content hooked up to the HTML DOM. Typically this involves working around browser quirks to generate the **object** tag with the right set of parameters. The **createObject** function also takes the **id** of a parent element, for example a **div** or **span**, within which the Silverlight **object** will be hooked up as child element. This approach uses the following logic.

```
if (slParentElement != null) {
    slParentElement.innerHTML = slPluginHTML;
}
```

For example, assume that your markup consisted of the following code.

```
<div id="divWithinWhichSLObjectExists">
  <script type="text/javascript">
    Silverlight.createObject("slObjectId",
                             "divWithinWhichSLObjectExists",
                             ...);
  </script>
</div>
```

The effective DOM would be like the following code when executed on the browser.

```
<div id="divWithinWhichSLObjectExists">
<object type="application/x-silverlight-2"
        data="data:application/x-silverlight,"
        width="..." height="...">
  <param name="source" value="SeattleTrafficMaps.xap" />
  <!-- Other Parameters, if any -->
  ...
  <!-- The “Get Silverlight” message and badge -->
  <p>This content requires Microsoft Silverlight.
    <a href="http://go.microsoft.com/fwlink/?LinkID=124807"
       style="text-decoration: none;">
      <img src="http://go.microsoft.com/fwlink/?LinkId=108181"
           alt="Get Microsoft Silverlight"
           style="border-style: none"/>
    </a>
  </p>
</object>
</div>
```

This has a beneficial side effect for the following reasons:

Replacing the inner HTML of the parent **div** removes any other existing child elements in that **div**.

Search engines often parse and index markup as it is served—before any scripts execute and change the DOM.

In other words, you can add detailed contextual metadata as native HTML within the parent **div** element of the Silverlight **object**. Search engines will process the metadata, but the metadata will not show up in the browser.

If your parent **div** has nested contextual content, the following code is what the search robot sees.

```
<div id="divWithinWhichSLObjectExists">
  <!-- Nested alternate HTML content for search -->
  <div>
    <h3>Traffic map of the Seattle-Puget Sound area</h3>
    <p>Up-to-the-minute traffic situation overlaid on the map of
       the Seattle-Puget Sound area, powered by
       <a href="http://maps.live.com">Live Maps</a>
    </p>
    <!-- canned image representing app contents -->
    <img src="SeattleTraffic_RushHour.jpg"
         alt="Seattle traffic at 5:30pm (evening rush-hour)" />
  </div>
  <!-- Invocation of the createObject function in Silverlight.js -->
  <script type="text/javascript">
    Silverlight.createObject("slObjectId", //SL plug-in id
                             "divWithinWhichSLObjectExists", //parent id
                             ...);
  </script>
</div>
```

Test Down-Level Experiences

Regardless of the relative importance of Silverlight and HTML in your content, it is important to test the page as a user who does not have Silverlight installed.

To access the page as a down-level user, perform the following steps:

1. Close all instances of Internet Explorer, and then start a new instance.
2. On the **Tools** menu, point to **Manage Add-ons**, and then click **Enable or Disable Add-ons**. The **Manage Add-ons** dialog box is displayed.
3. Select **Microsoft Silverlight**, select the **Disable** button, and then click **OK**.
4. Restart Internet Explorer and navigate to the Web page that has your Silverlight content.

To revert, return to the **Manage Add-ons** dialog box, select **Microsoft Silverlight**, and select **Enable**.

Note: The procedure for temporarily disabling the Silverlight plug-in differs depending on the browser you are using.

After you have tested the down-level experience, it is worthwhile validating your page with a static analysis tool such as SEO Browser that understands the impact of markup and content on search engine optimization.

Conclusion

The ability of search engines to index content that is not native HTML is very limited. However, if you structure and present your Silverlight application in certain ways, they have a better chance of appearing on a search engine results page in response to a search query.