Edge Side Includes (ESI) Overview

Abstract:
Edge Side Includes (ESI) accelerates dynamic Web-based applications by defining a simple markup language to describe cacheable and non-cacheable Web page components that can be aggregated, assembled and delivered at the network *edge*.

The Promise and Challenge of Dynamic Content

The World Wide Web has evolved beyond simply displaying static Web pages. Increasingly, e-businesses are delivering dynamically generated content -- such as product catalogs, auctions, exchanges, stock quotes, news, and weather -- to attract and retain customers. Web developers frequently use technologies like JavaServer Pages (JSP) and Active Server Pages (ASP) to design their applications, mainly because it is easier to develop and maintain feature-rich Web sites when information is stored in a database.

While dynamically generated content provides a more compelling experience for the end-user and an easier development model for the application designer, it presents e-businesses with a new challenge – delivering Web pages quickly. As traffic on Web sites increases, the computing overhead associated with building personalized pages on-the-fly for thousands of concurrent users can result in increasing delays and failures in data delivery.

Dynamic content creation places significant strain on traditional Web site architectures. This is because the same infrastructure used to *generate* the content is used to *deliver* the content (see Figure 1). Generating dynamic content typically incurs: 1) network overhead as user requests are dispatched to appropriate software modules that service these requests; 2) processing overhead as these modules determine which data to fetch and present; and 3) disk I/O as these modules query the back-end database. The assembled data then needs to be formatted and delivered to the browser. In short, building Web pages on-the-fly is computationally expensive. For this reason, e-businesses are often forced to spend millions of dollars on hardware, software and services to meet the demands of an ever-growing user population.

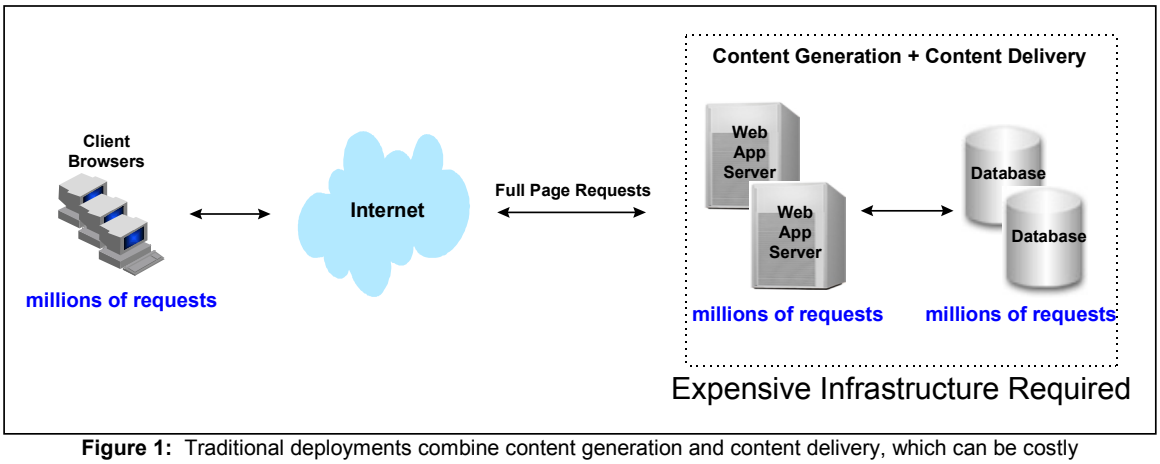

Figure 1: Traditional deployments combine content generation and content delivery, which can be costly

Given today's pressure to streamline IT budgets, e-businesses are demanding integrated solutions to solve two major issues:

1. **Site Experience and Effectiveness**, i.e. dynamic content, personalization, abandon rate, download speed, etc.

2. **Site Cost Structure**, i.e. investments to support scalability, reliability, performance, system management, application integration, etc.

To address these issues, e-businesses need a simple way to separate content delivery from content generation. In particular, Web developers need an industry-standard markup language to identify more fine-grained page elements called *content fragments* that can be automatically assembled into complete, personalized Web pages for faster delivery.

Introducing Edge Side Includes (ESI)

Edge Side Includes is a simple markup language that developers can use to identify content fragments for dynamic assembly at the network edge. ESI also specifies a content invalidation protocol for transparent content management across ESI-compliant solutions, such as application servers and content delivery networks. The ability to assemble dynamic pages from individual page fragments means that only non-cacheable or expired fragments need to be fetched from the origin Web site, thereby lowering the need to retrieve complete pages and decreasing the load on the Web site's content generation infrastructure (see Figure 2).

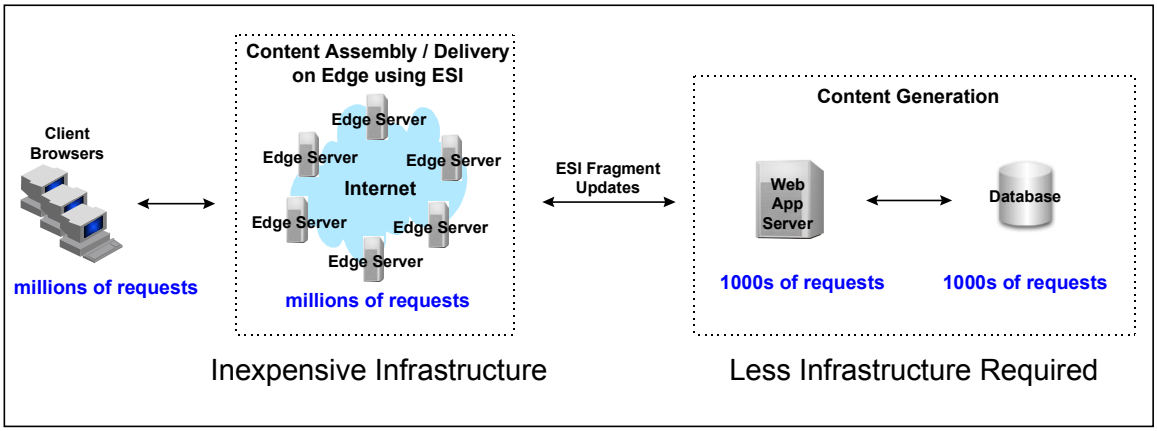

Figure 2: ESI separates content delivery from content generation for greater scalability and cost savings

With ESI,

- e-businesses can now develop highly dynamic Web-based applications that are assembled at the *edge* of the data center, or at the *edge* of the Internet, for improved performance [Site Experience and Effectiveness];

- the aggregation and assembly of content in edge servers dramatically reduces the cost of infrastructure required to deliver fast, scalable and fault-tolerant applications [Site Cost Structure].

Dynamic Caching and Content Assembly Using ESI

ESI enables Web pages to be broken down into fragments of differing cacheability profiles. These fragments are maintained as separate elements in the application server's local cache and/or on the content delivery network. ESI page fragments are assembled into HTML pages when requested by end users. This means that much more dynamically generated content can be cached, then assembled and delivered from the edge when requested. Furthermore, page assembly can be conditional, based on information provided in HTTP request headers or end-user cookies.

The ESI markup language (summarized in Table 1) includes the following key features:

- **Inclusion** – ESI provides the ability to fetch and include files to comprise a Web page, with each file subject to its own configuration and control, its own specified time-to-live in cache, revalidation instructions, and so forth. Included documents can include ESI markup for further ESI processing. Currently, ESI supports up to three levels of recursion.

- **Conditional inclusion** -- ESI supports conditional processing based on Boolean comparisons or environmental variables.

- **Environmental variables** -- ESI supports the use of a subset of standard CGI environment variables such as cookie information. These variables can be used inside ESI statements or outside of ESI blocks.

- **Exception and error handling** -- ESI allows developers to specify alternative pages and default behavior, such as serving a default HTML page in the event that an origin site or document is not available. Further, it provides an explicit exception-handling statement set. If a severe error is encountered while processing a document with ESI markup, the content returned to the end user can be specified in a "failure action" configuration option associated with the ESI document.

Tag	Purpose
`<esi:include>`	Include a separately cacheable fragment.
`<esi:choose>`	Conditional execution – choose among several different alternatives based on, for example, cookie value or user agent.
`<esi:try>`	Specify alternative processing when a request fails (e.g., the origin server is not accessible).
`<esi:vars>`	Permit variable substitution (for environment variables).
`<esi:remove>`	Specify alternative content to be stripped by ESI but displayed by the browser if ESI processing is not done.
`<!--esi … -->`	Specify content to be processed by ESI but hidden from the browser.
`<esi:inline>`	Include a separately cacheable fragment whose body is included in the template.

Table 1: Summary of ESI Tags

The basic structure a content provider uses to create dynamic content in ESI is a template page containing HTML fragments (see Figure 3). The template consists of common elements such as a logo, navigation bars, and other "look and feel" elements of the page. The HTML fragments represent dynamic subsections of the page.

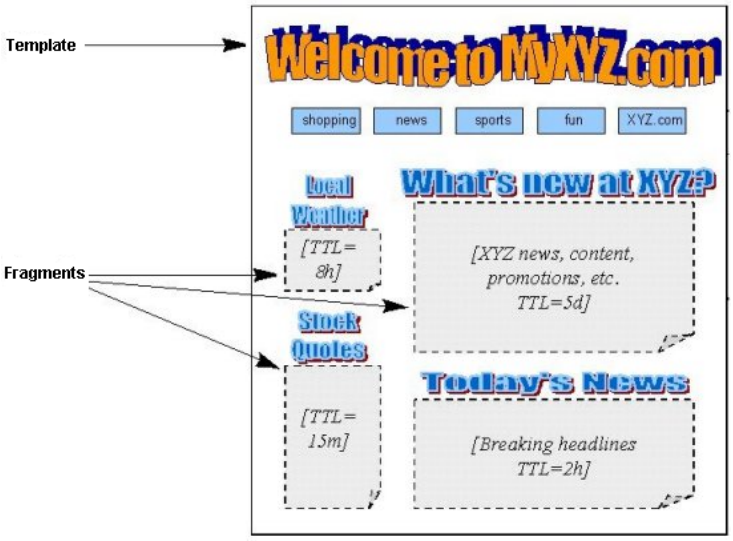

Figure 3: Example ESI template page containing ESI fragments and their expiration policies (TTL, or time-to-live)

The template is the file associated with the URL the end user requests. It is marked-up with ESI language (see Figure 4) that tells the cache server or delivery network to fetch and include the HTML fragments. The fragments themselves are HTML/ESI marked-up files containing discrete text or other objects.

```
<table>
<tr>
<td colspan="2">
<esi:try>
 <esi:attempt>
   <esi:include
   src="http://www.myxyz.com/news/top.html"
   onerror="continue"/>
 </esi:attempt>

 <esi:except>
 <!--esi
    This spot is reserved for your company's
   advertising. For more info <a
   href="www.myxyz.com"> click here </a>
  -->
 </esi:except>
</esi:try>
</td> </tr>
</table>
```

Figure 4: Example ESI markup within an HTML page template

Each fragment is treated as its own separate object. Each fragment has its own cache and access profile which are set in HTTP headers or configuration files. For example, content providers may want to cache the template for several days, but only cache a particular fragment (such as an advertisement or stock quote) for a matter of seconds or minutes. Other fragments (such as a user's bank account total) may be declared entirely non-cacheable.

Cached templates and fragments may be shared among multiple users. This means that for a large percentage of requests, the entire page can be assembled using shared components and delivered from the edge. ESI obviates the need for full-page updates when individual page fragments change. For example, when a single user requests the stock quote for General Electric Corp. (GE), and the fragment representing the GE stock quote has expired or changed, the revalidation of that fragment applies to all users' pages that reference GE.

Content Management Using ESI

ESI provides a number of ways to manage the consistency (freshness) of cached objects, including the ability to define expiration policies. Due to the unpredictable nature of frequently changing content, ESI also includes a content invalidation specification. The invalidation specification is used by syndication servers, content management systems, databases, custom scripts, application logic, etc. to send HTTP-based invalidation messages to the local cache and/or delivery network. The invalidation message tells the cache or delivery network to overwrite the metadata associated with particular objects residing on edge servers. In this way, e-businesses are able to control the validity of volatile content just as if it were residing on static Web servers, and at the same time, leverage the price/performance benefits of local caches and distributed content delivery networks.

Integrated into an application server, the invalidation specification allows for seamless purging of out of date or undesired content from the local cache or remote delivery network. This can be achieved with a database trigger, script, or another method that can then be integrated into the site administration process used by the application server. To the customer, the invalidation process is transparent. When a catalog item is flagged in the database, it is no longer viewable on the Web site, even though that site might now be served from thousands of content delivery servers spread across the hundreds of networks that form the Internet.

Edge Side Includes for Java (JESI)

Edge Side Includes for Java provides extensions to Java that make it easy to program JSPs using ESI. JSPs are server-side software modules that produce final user interface by linking dynamic content and static HTML through tags.

Tag	Purpose
`<jesi:include>`	Used in a "template" page to indicate to the ESI processor how the fragments are to be assembled (the tag generates the `<esi:include>` tag.
`<jesi:control>`	Assign an attribute (e.g., expiration) to templates and fragments.
`<jesi:template>`	Used to contain the entire content of a JSP container page within its body.
`<jesi:fragment>`	Encapsulate individual content fragments within a JSP page.
`<jesi:codeblock>`	Specify that a particular piece of code needs to be executed before any other fragment is executed (a database connection established, user id computed, etc.).
`<jesi:invalidate>`	Explicitly remove and/or expire selected objects cached in an ESI processor.
`<jesi:personalize>`	Insert personalized content into a page where the content is placed in cookies and inserted into the page by the ESI processor.

Table 2: Summary of JESI Tags

JESI is a specification (summarized in Table 2) and custom JSP tag library that developers can use to automatically generate ESI code. For JSP developers, JESI represents an easy way to express the modularity of pages and the cacheability of those modules, without requiring developers to learn a new programming syntax.

ESI and JESI as Open Standards

As ESI evolves into an open standard, e-businesses will be able to leverage the content management tools, application servers, packaged applications, syndication services, and content delivery networks that

are expected to adopt ESI as the common way to create and deliver fast, cost-effective applications on the Web.

Summary: Key Benefits of ESI

Proposed as an open standard, ESI extends the performance and cost-saving benefits of Web caching and content delivery services to support the most dynamic Web-based applications. Specifically, ESI:

- speeds up delivery of highly dynamic Web-based applications;
- saves e-businesses money by reducing the infrastructure required to deliver fast, scalable and fault-tolerant applications;
- reduces the complexity of developing dynamic Web-based applications for edge deployment;
- provides e-businesses with a choice in how to extend their Web infrastructure to support increasing traffic, from managed data centers to managed content delivery networks;
- enables e-businesses to deploy ESI-compliant applications seamlessly on both application servers and content delivery networks without re-writing the applications.

****