Proxy File Cache

Platform-X uses a built-in proxy to serve externally hosted files to client machines. This proxy marshals the file from the remote machine and then performs any necessary post processing such as the injection of comments and highlights.

Repeated retrieval of the external content is expensive due to network latency. In order to avoid this problem, this document specifies a caching component that will be used to store external content closer to the web server it is being processed by. Repeated access of the external content will be reduced because disk IO is significantly faster than network data transfer, especially for large files.

# File Cache Specification

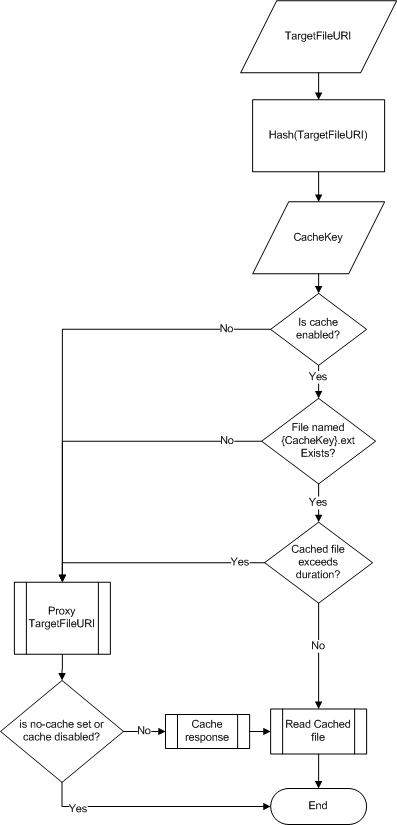
## Configuration

The following table describes the configuration options for the File Cache component.

|  |  |
| --- | --- |
| Key | Description |
| CacheLocation | Local path to where cached files are stored |
| CacheDuration | Duration, in minutes, that cached file is valid for |
| CacheMetaExtension | Extension used for metadata of cached files. Do NOT include the dot |
| CacheContentExtension | Extension used for content of cached files. Do NOT include the dot |
| CacheEnabled | True if caching should occur, false otherwise |

## File Caching Algorithm

The algorithm used to load external content from the cache is straightforward. The image below provides the details:



As depicted above, the URI of the external content (e.g. URL or file path) is first hashed. The resulting hash is assumed to be the name of the file in the File Cache's storage location. If a matching file is found in the location specified by CacheLocation, then the creation date and time of the file is checked against the CacheDuration. If the creation date indicates the file is older than allowed, then a new copy of the file is requested and the cache is updated. Otherwise, the cached file is loaded and passed up the stack for further processing.

## Cache Data Format

Responses from the external source are stored in two files. One file stores the metadata about the response such as the ContentType, CharacterSet, etc. The second file stores the verbatim response body.

The following listing shows the expected metadata file format:

<CachedResponse>

<CharacterSet></CharacterSet>

<ContentEncoding></ContentEncoding>

<ContentLength></ContentLength>

<ContentType></ContentType>

<Cookies></Cookies>

<Headers></Headers>

<IsFromCache><IsFromCache>

<IsMutuallyAuthenticated></IsMutuallyAuthenticated>

<LastModified></LastModified>

<Method></Method>

<ProtocolVersion></ProtocolVersion>

<ResponseUri></ResponseUri>

<Server></Server>

<StatusCode></StatusCode>

<StatusDescription></StatusDescription>

</CachedResponse>

## Implementation Details

There are only two classes necessary to implement the File Cache. The following sections detail these classes and the API they expose.

### CachedHttpWebResponse

*CachedHttpWebResponse : HttpWebResponse*

Derived from System.Net.HttpWebResponse, this class allows the entire state of a web response to be cached to disk using the format described in "Cache Data Format". All methods of HttpWebResponse should be overwritten, with the following members added:

|  |  |
| --- | --- |
| Member | Description |
| public CachedHttpWebResponse(XDocument meta, Stream content) | Constructor that initializes the object using the given metadata and content |
| public XDocument GetMetaData() | Returns the XDocument that conforms to the "Cache Data Format" |

### IResponseProxy

Interface that defines API for proxying requests to external resources.

|  |  |
| --- | --- |
| Member | Description |
| HttpWebResponse Proxy(Uri target, HttpWebRequest original) | Implementations should proxy the call to the target URI using all necessary information from the original request |

### CachingResponseProxy

*CachingResponseProxy : IResponseProxy*

Implements the IResponseProxy interface using the "File Caching Algorithm".

|  |  |
| --- | --- |
| Member | Description |
| HttpWebResponse Proxy(Uri target, HttpWebRequest original) | Implements the "File Caching Algorithm". Returns a CachedHttpWebResponse unless no-cache is set. If the Cache-Control header on the proxied response contains "no-cache", then do not cache the response regardless of CacheEnabled's value |

### CachingResponseProxySection

*CachingResponseProxy : ConfigurationSection*

Implements a custom configuration section that collects all the properties from "Configuration"

|  |  |
| --- | --- |
| Member | Description |
| String CacheLocation | Required. Local path to where cached files are stored |
| Int CacheDuration | Required, Default 60. Duration, in minutes, that cached file is valid for |
| String CacheMetaExtension | Required, Default "meta". Extension used for metadata of cached files. Do NOT include the dot |
| String CacheContentExtension | Required, Default "cont". Extension used for content of cached files. Do NOT include the dot |
| Boolean CacheEnabled | Required, Default "true". True if caching should occur, false otherwise |

## Cleaning up the cache

Every server that uses the cache should have a scheduled task that should run on an interval that is 15 minutes longer than the CacheDuration (e.g. if CacheDuration is 60 minutes, then task should run every 75 minutes). The task should delete any files whose creation date is older than the interval. This can most easily be done by creating a PowerShell script and using the built-in task scheduling of Windows Server.

## Integrating with Platform-X

There are several steps to integrate the response proxy with Platform-X.

1. Add necessary configuration to all Web.config files. This includes the Unity IoC container configuration for the IResponseProxy and CachingResponseProxy classes.
2. Add the IResponseProxy to the HighlightController constructor so that Unity injects the correct implementation.
3. Modify the HighlightController.ProxyPage action so that the IResponse.Proxy method is being called.
4. Update any action methods for content types PX controls (e.g. HtmlDocument) so that they use the Cache-Control header set to "no-cache". The only exception is for ExternalContent. This avoids cache coherency issues since HtmlDocument and other types can actually be edited whereas ExternalContent cannot.