Version 1.2 - 15/08/12

Sage ID (Single Sign-On) Developer Guide

Sage (UK) Central R&D SSDP Team

Revision History

Revision	Author	Notes
Beta 1	GD	• Initial version.
1.0	DJM	• Addition of customisation kit tutorial and walkthrough.
1.1	DJM	• Initial changes for Sage ID v1.1
1.2	DJM	• Documented new fields in SignOnSuccess and SignOnFailed result

Table of Contents

1. Introduction

1.1. Introducing Sage Single Sign-On

Content

2. Understanding the Session Lifecycle

2.1. Introduction

When a user signs into a web application via Sage SSO, the Sage SSO server establishes an **SSO session** for that user. It is the SSO session which allows the user to sign into multiple web applications securely without having to re-enter their credentials. This is known as "single sign-on".

This section describes how SSO sessions work and their relationship to your web application sessions. In order for single sign-on to work most effectively, your web application needs to follow some simple patterns which govern the way that it participates in SSO sessions. This section describes these patterns and recommends best practice for implementing them in your web application.

2.2. How SSO sessions work

An SSO session is established when the user first signs into a web application which uses Sage SSO. There are two separate parts which form an SSO session:

- The server session state
- This is some data which is held on the Sage SSO server. It has a unique ID and keeps track of the web application which the user has signed into.
- The SSO session cookie
- This is established on the user's browser when an SSO session begins and is used by Sage SSO to determine the SSO session which belongs to the user.

Note that it *is* possible for a user to have multiple SSO sessions open across different machines or different browsers on a single machine. If your application needs to restrict the user to a single application session, it should take appropriate action after the sign-in process completes.

Your web application should not read the SSO session cookie nor should it make any assumptions about the contents or lifetime of the cookie.

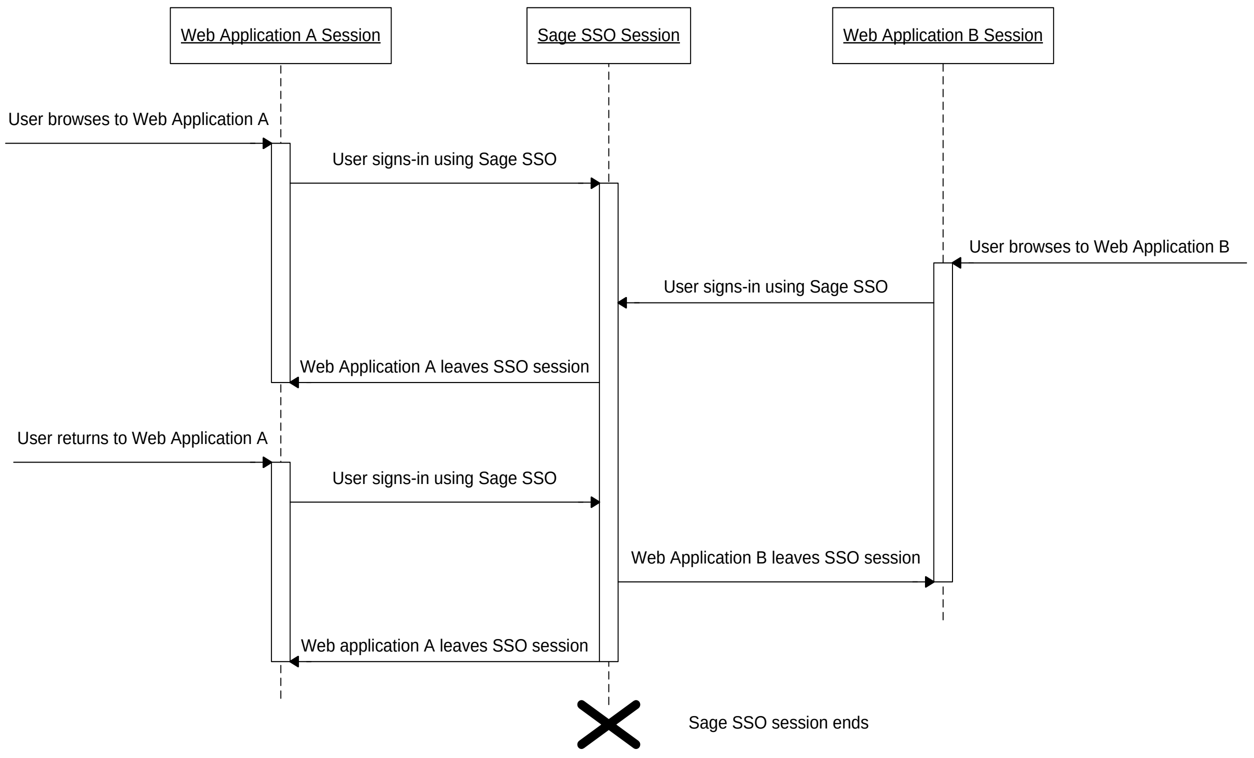
An SSO session holds the following data:

- A unique non-repeating session ID.
- The user identity originally authenticated for the session.
- The time at which the session started.
- The time beyond which the session may not be extended (known as the *hard-timeout*).
- A list of the web applications which are currently participants in the session.
- For each participant web application, the time at which that web application's participation in the SSO session is due to expire.

An SSO session ends when the last participant web application leaves the session, either because its participation has expired or because the web application programmatically left the session by calling the `WebSSOService/RemoveSessionParticipant` API.

There are a number of other reasons why a Sage SSO session may end. These are described later in this section.

It is possible for a web application to leave an SSO session and later become a participant in the same SSO session, as the following timeline illustrates:


Web Application A Session
Sage SSO Session
Web Application B Session
User browses to Web Application A
User signs-in using Sage SSO
User browses to Web Application B
User signs-in using Sage SSO
Web Application A leaves SSO session
User returns to Web Application A
User signs-in using Sage SSO
Web Application B leaves SSO session
Web application A leaves SSO session
Sage SSO session ends

2.3. *Signing into multiple web applications*

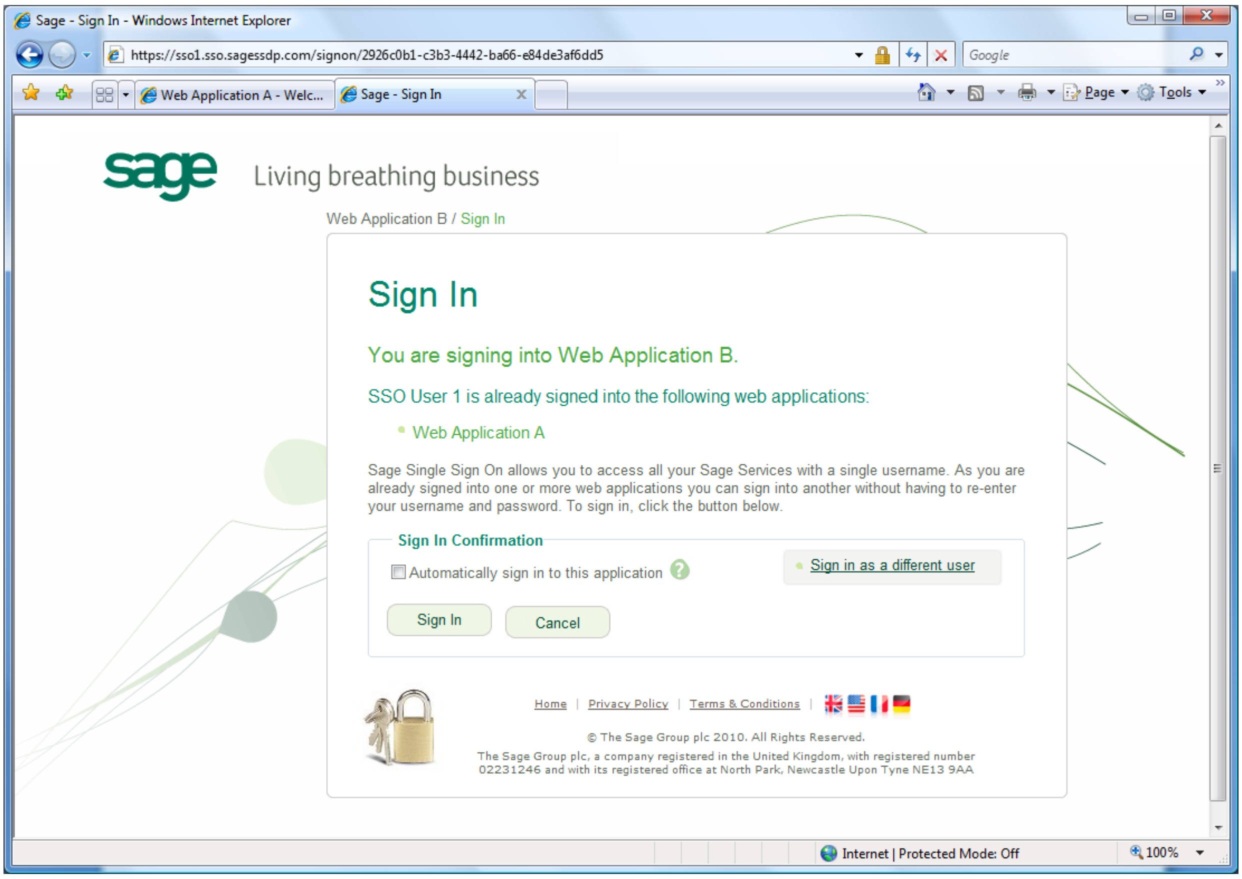
While there is at least one unexpired participant web application in the Sage SSO session, if the user signs into any other Sage SSO-enabled web application, they will be signed-on with the behaviour configured for that web application. This behaviour is configured in Sage SSO and can be one of the following:

- Transparent sign-on

  The user is not asked for credentials or confirmation and is signed into the web application transparently. This behaviour is recommended only for applications with minimal security requirements.

- Express sign-on

  The user sees a confirmation screen like the one below:


Sage - Sign In - Windows Internet Explorer

https://sso1.sso.sagessdp.com/signon/2926c0b1-c3b3-4442-ba66-e84de3af6dd5

Living breathing business

Web Application B / Sign In

Sign In

You are signing into Web Application B.

SSO User 1 is already signed into the following web applications:

- Web Application A

Sage Single Sign On allows you to access all your Sage Services with a single username. As you are already signed into one or more web applications you can sign into another without having to re-enter your username and password. To sign in, click the button below.

Sign In Confirmation

☐ Automatically sign in to this application

Sign in as a different user

Sign In | Cancel

Home | Privacy Policy | Terms & Conditions

© The Sage Group plc 2010. All Rights Reserved.

The Sage Group plc, a company registered in the United Kingdom, with registered number 02231246 and with its registered office at North Park, Newcastle Upon Tyne NE13 9AA

Internet | Protected Mode: Off 100%

The user is told which applications they are already signed into and they may continue or cancel the sign-in or, optionally, sign-in as a different user.

This behaviour is recommended for most applications. It provides a balance between security and convenience. It gives the user visibility of the sign-in and the option not to sign-in if that is not what they intended. It also provides a degree of protection from certain types of attack (in particular cross-site request forgery – CSRF).

The check box labelled "automatically sign-in to this application" is optional. If configured for the web application, it allows the user to choose to use transparent sign-on for this application in the future.

- Always request credentials

  The user is required to re-enter their password. The user's email address is pre-filled and cannot be edited. However, the user may sign-in as another user by clicking the "sign-in as a different user" link.

  This behaviour is recommended for those applications which require a high-degree of security.

2.4. The SSO session and the application session

The SSO session is independent of your web application session. This is convenient from a development perspective since it means that the degree of coupling between your web application and Sage SSO is minimal and you can take advantage of whatever application session mechanism is provided by your web framework of choice.

There are points, however, where the application session should be synchronised with the SSO session, in order to achieve the best single sign-on experience for the user.

The goal for single sign-on is to make using multiple web applications from the same vendor with the same identity as transparent and easy as possible. In particular, single sign-on tries to avoid situations where the user is required to continually enter their credentials when switching between web applications unless the security profile of a particular application truly requires it.

By synchronising your application session correctly with the SSO session, you help ensure that the user sees the right sign-on UI at the right time across all participating applications and that the user can definitively end their Sage SSO session when they choose to do so such that any subsequent user of the browser cannot accidentally or maliciously sign-in as that user.

To achieve this, your application must follow the three golden rules:

1) Sign-on to Sage SSO correctly and observe the session expiry time which is returned.
2) Receive session notifications from Sage SSO and act upon them correctly.
3) Sign-off from Sage SSO correctly, using the sign-off function that is appropriate for your application and the user's request.

2.5. When and how to sign-in a user

Your web application should sign-in the user before they enter the part of your web application which requires the user to prove their identity before use. This may be part of your web application, or all of it. In the following discussion, we'll refer to them as "protected pages". We recommend the following approach:

- Check for the existence of an application session on every protected page

  Every time the user's browser requests a protected page, your application should check to see whether an application session exists. If no application session exists your application should redirect the browser to a home page.

- Check for the existence of Sage SSO session information in the application session state on every protected page

  Assuming there is an application session when the user requests a protected page, your application should additionally check to see whether the session state contains the information that is supplied by Sage SSO when the user signs in. If this is not the case your application should redirect the browser to a home page.

- Check that the Sage SSO session information in the application session state is still valid on every protected page.

  If the session state contains Sage SSO information, the application should check that the information hasn't expired (i.e. the SSO session expiry time has not passed). If it has expired, your application should end the application session and then redirect the browser to a home page.

If your application exposes AJAX services to the browser which should only be called by signed-in users, these services can be protected in a similar way by checking for the existence of an application session with Sage SSO session *information in session state. If either of these is not true, the services should return a 403 Unauthorised error to the AJAX application running on the browser.*

To sign-in a user using Sage SSO, your application must make a web service call to `WebSSOService/StartSignOnAttempt`.

The following parameters are required:

- `SuccessUri`

  The URI to which the user's browser will be redirected following a successful sign-on. The URI should contain a placeholder **{0}** for a unique, non-repeating GUID which will be used in the call to `WebSSOService/EndSignOnAttempt` to retrieve the result of the sign-on.

  For example, if the `SuccessUri` is:

```
http://myapp.com/signon/{0}
```

  The browser will be redirected to (for example):

```
http://myapp.com/signon/d4286fe9-d5f0-4dbc-8dcc-1d935e7337bd
```

- `FailureUri`

  The URI to which the user's browser will be redirected following an unsuccessful sign-on. The URI should contain a placeholder as above. It's OK to have a matching `SuccessUri` and `FailureUri`. The call to `WebSSOService/EndSignOnAttempt` returns enough information to distinguish between a successful and unsuccessful sign-on.

- `SessionLengthMinutes`

  This should match the normal inactivity timeout for your application session. The minimum acceptable value is ten minutes. The maximum acceptable value is sixty minutes. If you pass in a value which violates these limits, the nearest limit will apply. If you don't specify a value, the default is sixty minutes.

There are a number of other parameters which can be provided to `WebSSOService/StartSignOnAttempt`*. These are described elsewhere in this document and in the API reference guide.*

The response from Sage SSO has the following parameters:

- `SignOnAttemptId`

  The ID of this sign-on attempt. If your application already has an application session, this ID can be stored in session state to help prevent unnecessary calls to Sage SSO later on.

- `RedirectUri`

  The URI to which the user's browser should now be redirected.

Your web application should now redirect the user's browser to the URI returned in `RedirectUri`. The user will be signed into Sage SSO using the appropriate behaviour (described earlier in this section).

After the user has been signed in successfully, the browser will be redirected to `SuccessUri`. If the user was not signed in successfully, the browser will be redirected to `FailureUri`. In either case, your web application should do the following:

- Check for the existence of the `SignOnAttemptId` in session state

  If your web application stored the `SignOnAttemptId` in session state, it should check for the existence of this state item. If it does not exist, this URL has been reached in error and the browser should be redirected accordingly.

  Note that the check is for the existence of the `SignOnAttemptId` in session state, not for a match with the `ResultId` which is part of the URI which the browser requested (which will always be different). Also note that this check is optional, but doing it will prevent having to make an unnecessary web service call if your web application's `SuccessUri` is requested accidentally (e.g. if the user uses refresh or the back button) or deliberately (by an attacker).

- Extract the `ResultId` from the request URI

  Your application should extract the GUID from the request URI. If the GUID isn't present or isn't a properly formatted GUID this is an error and your web application should redirect the browser accordingly.

- Make a call to `WebSSOService/EndSignOnAttempt`

  Your application must pass the `ResultId` extracted above as a parameter.

  If Sage SSO returns the error code `SSOAttemptUriUnknown` the result was already retrieved or has expired. If the user already has an authenticated SSO session, this is probably the result of the user using the back button or pressing refresh and the user should be redirected appropriately. If the application session is not authenticated or no application session exists the browser should be redirected to a home page.

  If Sage SSO returns any other error code at this point, this may be the result of a programming error or a problem in Sage SSO. The error should be logged and the browser should be redirected appropriately.

- If the user was signed in successfully, the `Result` parameter in the response from `WebSSOService/EndSignOnAttempt` will contain a `SuccessResult` with the following data.

  `SessionId`: The ID of the SSO session in which your web application is now a participant.

  `AuthenticationToken`: Base-64 encoded signed XML which can be used to make calls to other Sage SSO enabled applications on behalf of the user.

  `SessionExpiry`: The UTC time at which your application's participation in the SSO session will end.

  `SessionExpiry` *will usually match to within a few seconds the inactivity timeout of your session at the time that* `WebSSOService/EndSignOnAttempt` *is called and no further adjustment will be necessary. If, however, the difference is more than a few seconds, it is likely that the SSO session is nearing its hard-timeout. Your web application should, in this case, adjust the application session timeout to* *match, as near as possible, the* `SessionExpiry` *returned.*

`EmailAddress`: The email address associated with the authenticated identity.

`DisplayName`: The display name associated with the authenticated identity.

`IdentityId`: The Sage ID unique identity ID for the authenticated identity (as a GUID). Before version 1.2 of Sage ID, this was only available after decoding the AuthenticationToken, but is now available directly in the sign on SuccessResult response.

Most web frameworks establish an application session automatically when a page is requested. If an application session is not already established, however, it should be established now.

- This information should be stored in session state. The application session is now authenticated and will allow access to protected pages.

`SessionId` *and* `AuthenticationToken` *should* ***never*** *be passed to the browser in any form (even if encrypted).*

- If the user was not signed in successfully, the `Result` parameter in the response from `WebSSOService/EndSignOnAttempt` will contain a `FailedResult` with the following data:

`FailureReason`: This is the reason for the failure and will be one of the following codes

Code	**Meaning**	**Suggested action**
ProtocolViolation	An invalid sign-on protocol state was reached.	This is an internal error in Sage SSO. Log the error and start a new sign-on attempt. If the error persists, please contact SSDP Developer Support (**ssdpdevelopersupport@sage.com**).
`UnknownAccount`	The user attempted to sign-on with an unknown email address more times than is allowed on your web application.	**Do not inform the user they are using an unknown email address.** Log the error and redirect to an error page or restart the sign-on attempt.
`AccountNotActivated`	The user's account is not activated and your application is configured to not allow unactivated accounts to sign-in.	Inform the user that their account is not yet activated. Consider providing a link which re-sends the activation email.

`AccountSoftLocked`	The user's account has been temporarily locked due to the number of failed sign-in attempts.	Inform the user that their account has been temporarily locked and to try signing in again later. Also, if the user has forgotten their password a successful recovery will immediately clear the soft lock.
`AccountHardLocked`	The user's account has been locked due to the number of failed sign-in attempts or because their account was not activated before the activation grace period expired.	Inform the user that their account has been locked. Only an administrator can unlock an account in this state, so this will require a support call.
`AccountExpired`	The user's account has expired because their account was not activated before the activation grace period expired or the following expiry grace period expired.	The user's account has been removed from the system and cannot be recovered.
`AccountNotRegistered`	The user's account is not registered for your web application.	Redirect the browser to your registration page.
`SessionExpired`	The user's session on the sign-on page has expired, or they attempted to back-click through the sign-on pages.	If the user has an authenticated application session, the browser can be safely redirected to a suitable protected page. Otherwise, redirect the browser to a home page.
`SignOnCancelled`	The user ended the sign-on attempt by clicking the cancel button.	Redirect the browser to a home page.
`ValidationFailed`	The user used the "forgotten password" facility but was not successfully validated.	Inform the user of the failure. Redirect the browser appropriately.
`AccountPasswordReset`	The user used the "forgotten password" facility and the	Inform the user that they must respond to the password activation email for the new password to take

	password was successfully changed.	effect. Begin a new sign-on attempt.
`AccountBlocked`	The user's account has been blocked from signing onto your web application.	Redirect the browser appropriately. The sign-on block can be cleared by calling `UserAccountManagementService/RemoveSignOnBlock`.

- `ActivationLinkUri`: If the user successfully used the forgotten password facility, this will be the URI the user can use to activate their password. **This URI must not be displayed to the user**. If your application sends its own activation emails it may embed this URI in an email to the user. If Sage SSO is sending activation emails on behalf of your application this URI can be ignored.

As of Sage ID version 1.2, the FailureResult response now contains IdentityId, EmailAddress and DisplayName (if these are available) for the identity resolved during a failed sign on attempt. This is useful to a calling web application in situations such as an account being blocked, or not registered for the web application. Previously the calling web application was unable to know which account had attempted to sign in.

Identity information is available with the following failure responses:

`AccountNotActivated, AccountSoftLocked, AccountHardLocked, AccountExpired, AccountNotRegistered, AccountBlocked`

2.6. How to receive and validate session notifications

Sage SSO sends your application two types of notification:

- `Session.ExpiryDue`

  This notification is sent three minutes before your application's participation in an SSO session expires. It provides an opportunity for your application to extend an SSO session to match your application session.

- `Session.Ended`

  This notification is sent after your application's participation in an SSO session has expired. Your application should, in response, immediately close the application session associated with the SSO session.

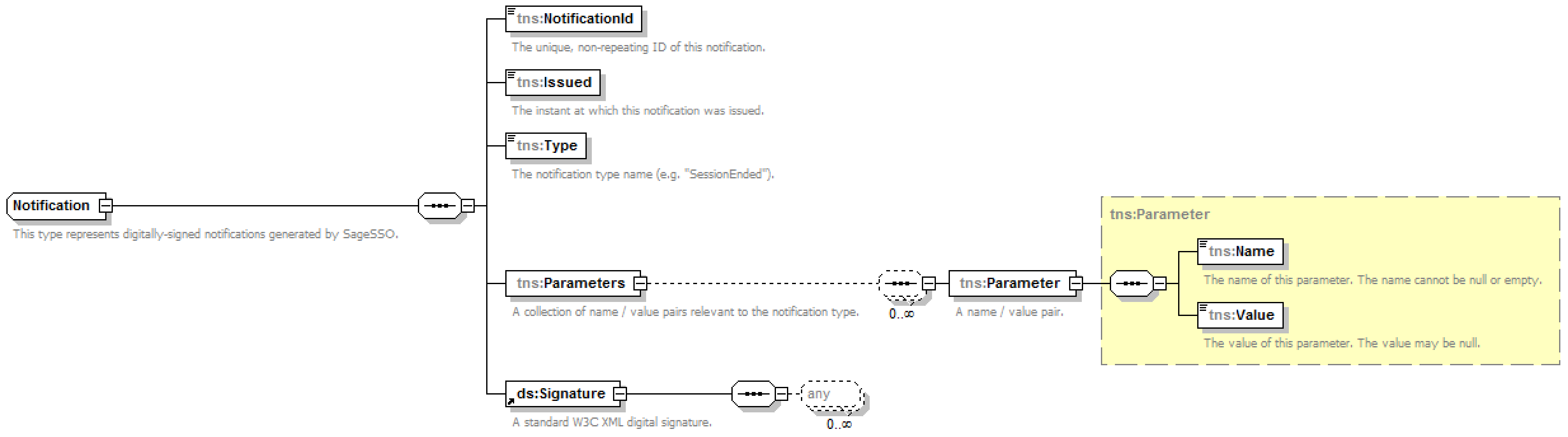
Sage SSO sends notifications using HTTP POST to a URL which you nominate for your application and which is configured in Sage SSO. There are a number of options for transport security and authentication which are dependent on the nature of the link between your application and Sage SSO, as summarised below:

- If the link between your application and Sage SSO is over a private, secure network or VPN we recommend that your firewall is configured such that only Sage SSO may send notifications to your nominated URL. We also recommend that SSL is used if at all possible. With this type of link, the probability of an external application sending rogue notifications to your application is very low and, in particular, the digital signature and replay checks (see below) are not strictly necessary.

- If the link between your application and Sage SSO is over a public link (for example, the Internet) or other unsecured network, we recommend that your firewall is configured such that only Sage SSO may send notifications to your nominated URL and we recommend the use of mutual SSL authentication. Sage SSO will authenticate with your application using a client certificate before sending a notification. The notification will be protected in transit using SSL. In this case, again, the probability of an external application sending rogue notifications to your application is very low and the digital signature and replay checks are not necessary.
- If the link between your application and Sage SSO is over a public link *and* you are unable to use mutual SSL authentication and/or set an appropriate firewall rule, your application must use SSL and must also check the digital signature of each notification received and also check that the notification has not been replayed in order to provide an appropriate level of protected against rogue notifications.

Under no circumstances should a production application using an unsecured link to Sage SSO receive notifications on a non-SSL endpoint.

The body of each notification message is a digitally-signed XML message encoded in Base 64 format which conforms to the following schema diagram:



Notification
This type represents digitally-signed notifications generated by SageSSO.
tns:NotificationId
The unique, non-repeating ID of this notification.
tns:Issued
The instant at which this notification was issued.
tns:Type
The notification type name (e.g. "SessionEnded").
tns:Parameters
A collection of name / value pairs relevant to the notification type.
0..∞
tns:Parameter
A name / value pair.
tns:Parameter
tns:Name
The name of this parameter. The name cannot be null or empty.
tns:Value
The value of this parameter. The value may be null.
ds:Signature
A standard W3C XML digital signature.
any
0..∞

Notification Schema Diagram

The XSD schema for notifications is available in the **Sage SSO SDK**.

Be aware of the namespacing used in this schema (and in others used by Sage SSO) as this can affect the evaluation of XPath queries and other operations.

In order to validate an incoming SSO notification, the following steps should be followed:

The ordering of these steps is significant. Steps which may be omitted if SSL mutual authentication and/or a secured link is in use are clearly marked.

- Decode the XML from its Base 64 representation
- Base 64 is used to avoid serialization errors which some web service frameworks introduce when creating proxies or service endpoints.
- Confirm that the Base 64 is valid XML
- This can be achieved in the simplest way by loading the XML into a DOM. Additionally, you may validate the incoming notification against the XSD schema.
- Check that the notification signature is valid
- This proves that the notification came from Sage SSO and that it was not altered in transit. Notifications have a standard W3C XML digital signature created by the **Sage SSO Identity Root** certificate which is supplied as part of the **Sage SSO SDK**.

If the link between your application and Sage SSO is via a secured network or VPN, or if the link uses mutual SSL authentication then this check is optional.

- The following .NET code sample shows how to validate the XML digital signature:

```
private void CheckNotificationValidSignature(XmlDocument notification)
{
    // Check the signature of the notification using the SSO Root certificate
    // (which should be installed
    // into the LocalMachine / Root certificate store). If the notification wasn't
    // signed by the SSO Root
    // certificate, it didn't come from Sage SSO so in that case we'll ignore it.
    SignedXml signedXml = new SignedXml(notification);

    XmlNodeList nodeList =
        notification.GetElementsByTagName("Signature",
                                          "http://www.w3.org/2000/09/xmldsig#");
    if (nodeList.Count != 1) throw new Exception("Notification was not signed.");

    signedXml.LoadXml((XmlElement)nodeList[0]);

    if (!signedXml.CheckSignature(SSORootCertificate, true))
        throw new Exception("Invalid notification signature.");
}
```

The Sage SSO SDK contains a functionally equivalent sample application written in PHP which illustrates how to integrate with Sage SSO.

- Check that the notification has not expired

  The following .NET code shows how to do this:

```
private XmlNode CheckNotificationNotExpired(XmlDocument notification, XmlNamespaceManager nsmgr)
{
    // We can now check that the Notification has an Issued timestamp, that it is in a valid
    // format and that it is recent.
    XmlNode issuedNode = notification.SelectSingleNode("/sso:Notification/sso:Issued", nsmgr);

    if (issuedNode == null) throw new Exception("Notification contains no Issued element.");

    DateTime issued = DateTime.MinValue;

    if (!DateTime.TryParse(issuedNode.InnerText, out issued))
        throw new Exception("Notification issue time is invalid.");

    // Notifications should be delivered within a few seconds of issue,
    // depending on the overall load of
    // your application and Sage SSO. If the notification is more than
    // ten seconds old, discard it.
    if (issued > DateTime.Now + TimeSpan.FromSeconds(10))
        throw new Exception("Notification has expired.");

    return issuedNode;
}
```

This validation check and several other parts of the session lifecycle require that your application server clock is synchronised with real-time to within a few seconds. For best results, we recommend a tolerance of no more than +/- two seconds.

If the link between your application and Sage SSO is via a secured network or VPN, or if the link uses mutual SSL authentication then this check is optional.

- Check that the notification has not been replayed

  Sage SSO notifications have a `NotificationId` field which contains a unique, non-repeating ID. In order to check that a notification has not been replayed, your application needs to keep a list of notification IDs which have been received. The IDs in the list need be kept only as long as the validity of the notification to which they relate (perhaps slightly longer to allow for minor clock sync differences). They may then be discarded because the preceding expiration check will reject replayed notifications after this time.

  We recommend the use of an external caching server such as Memcached for this purpose, especially for multi-node applications. The following .NET code illustrates the check using the built-in ASP.NET caching mechanism:

```
private void CheckNotificationNotReplayed(XmlDocument notification, XmlNamespaceManager nsmgr)
{
    // Even given the expiry check, there's a small window of opportunity for a replay attack.
    // It's best to hold a cache of
    // received notification IDs so, if a notification is replayed before it's considered
    // expired, we can detect
    // it. On a load-balanced, multi-node application, it's best to use a caching service such as
    // memcached or velocity
    // to manage this. For the purposes of this application, however, we'll just use
    // the ASP.NET cache.
    XmlNode idNode =
        notification.SelectSingleNode("/sso:Notification/sso:NotificationId", nsmgr);

    if (idNode == null) throw new Exception("Notification contains no NotificationId element.");

    Guid id = Guid.Empty;
    try
    {
        id = new Guid(idNode.InnerText);
    }
    catch (Exception)
    {
        throw new Exception("NotificationId is invalid.");
    }

    if (IsReplayedId(id)) throw new Exception("Notification has been replayed.");
}

private bool IsReplayedId(Guid id)
{
    bool replayed = false;
    string idKey = string.Format(CachedIdFormat, id.ToString());
    if (Cache.Get(idKey) == null)
    {
             Cache.Add(idKey,        // The Notification ID
                       string.Empty,  // Something that isn't null for the above if statement
                        null,
                        // Hold for one minute - the notification expiry check will deal with
                        // replayed notifications after that time
                        DateTime.Now + TimeSpan.FromMinutes(1),
                        Cache.NoSlidingExpiration,
                        CacheItemPriority.Default,
                        null);
    }
    else { replayed = true; }

    return replayed;
}
```

If the link between your application and Sage SSO is via a secured network or VPN, or if the link uses mutual SSL authentication then this check is optional. However, during debugging you may find this check helpful because Sage SSO will try to re-send notifications to your application if no response is received within a few seconds. Implementing this check will prevent multiple notifications from being processed in your application if one thread is paused in a debugger.

2.7. How to process the Session.ExpiryDue notification

The `Type` element of notifications of this type contains the text `Session.ExpiryDue`.

The `Session.ExpiryDue` notification has the following parameters:

- `SessionId`: The ID of the SSO session to which this relates.
- `EmailAddress`: The email address of the user identity signed into the session.
- `Timestamp`: The time at which your application's participation in the SSO session will end if no further action is taken.

Upon receiving and validating a `Session.ExpiryDue` notification, your application should take the following action:

- Extract the SSO session ID and expiry time from the notification.
- Find the associated application session

  The notification (if received directly by your web application and not, for example, by a supporting service) will not be received on the same application session as the one associated with the SSO session ID when the user signed-in. Depending on your web application framework you may need to keep a map or table of SSO session IDs and application session IDs in order to determine which application session relates to the SSO session ID specified in the notification.

  If there is no associated application session, the notification should be ignored.
- Determine if the SSO session should be extended

In order to do this, get the last activity time from the associated application session. If the last activity time + application inactivity timeout is greater than the SSO session expiry time, SSO will need to be called to extend the SSO session to match your application session.

- If the SSO session should be extended, determine the SSO session expiry to request

  The SSO session expiry to request is last activity time + application inactivity timeout.

  The following .NET code shows the last two steps together (`MapEntry` in this code is a collection item which correlates an SSO session with an application SSO – the **Sage SSO SDK** contains the complete code):

```
public bool ShouldExtendSSOSession(Guid ssoSessionId, DateTime ssoSessionExpiryDue,
                                   out DateTime newSSOSessionExpiry)
{
    string cacheId = MapEntry.GetCacheId(ssoSessionId);

    MapEntry mapEntry = (MapEntry)_cache.Get(cacheId);

    if (mapEntry != null)
    {
        // Calculate the new SSO session expiry time to request by adding our default application
        // session timeout to the last time we had any activity
        newSSOSessionExpiry = mapEntry.LastActivity + DefaultSessionTimeout;

        // Return true if the application session extends past the SSO session expiry
        return newSSOSessionExpiry > ssoSessionExpiryDue;
    }
    else
    {
        throw new Exception("No SSO session with specified ID in the cache.");
    }
}
```

- If the SSO session should be extended, call Sage SSO to extend the session

  Make a call to `WebSSOService/SessionExtend` passing in the SSO session ID and new SSO session expiry time as parameters.

  The response contains the following data items:

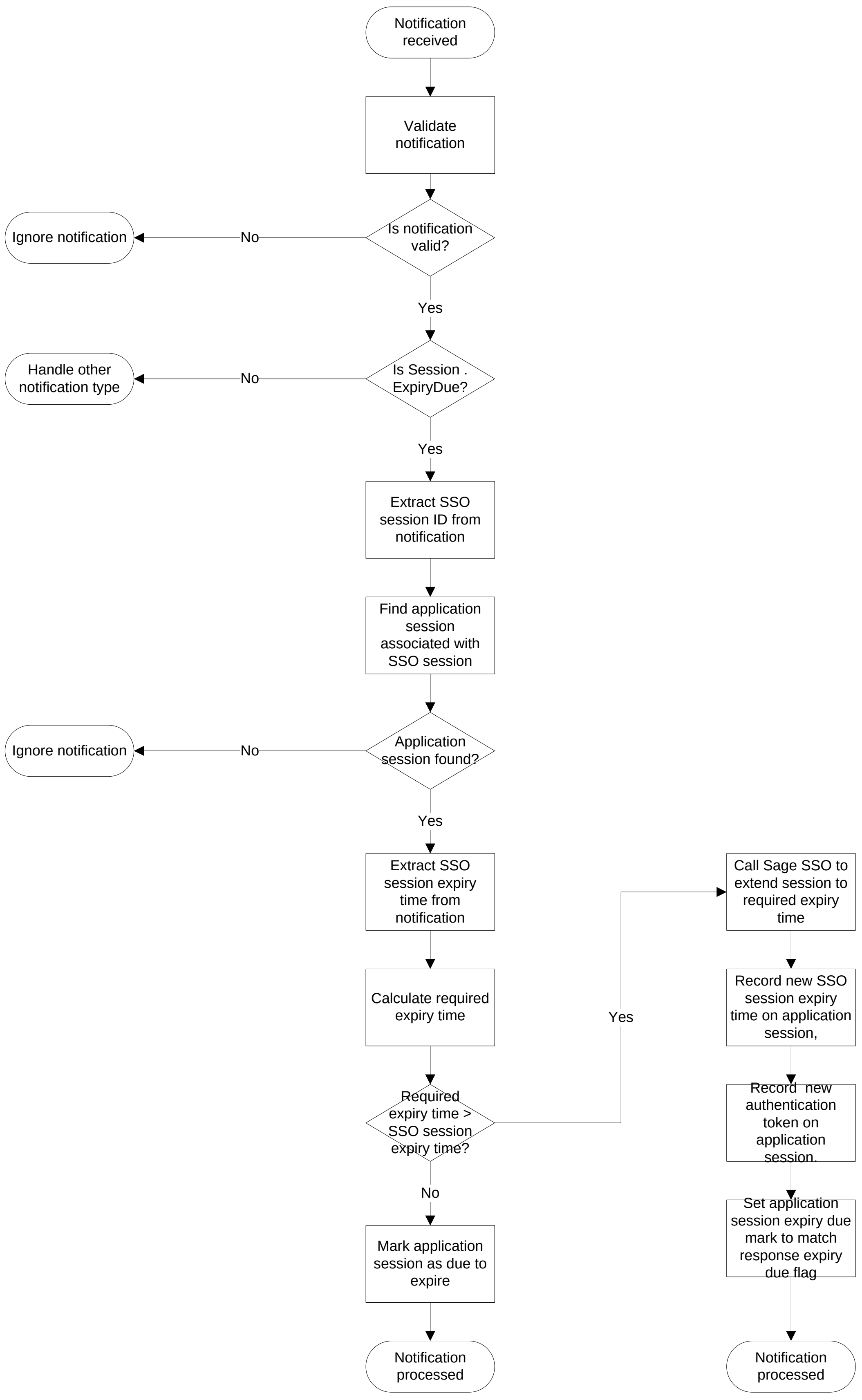
  `SessionExpiry`: The new expiry time for the SSO session. This should be stored on the application session associated with the SSO session ID.

  `AuthenticationToken`: A replacement authentication token for the user which is valid from the time of issue until the SSO session expiry time. This should replace the one which is already stored on the application session associated with the SSO session ID.

  `ExpiryDue`: If this flag is true, the application session associated with the SSO session ID should be marked as being due to expire. If it is false, any existing such mark should be removed.

- If the SSO session shouldn't be extended, mark the associated application session as being due to expire.

The following flowchart summarises these steps:


Notification received
Validate notification
Is notification valid?
No
Ignore notification
Yes
Is Session . ExpiryDue?
No
Handle other notification type
Yes
Extract SSO session ID from notification
Find application session associated with SSO session
Application session found?
No
Ignore notification
Yes
Extract SSO session expiry time from notification
Calculate required expiry time
Required expiry time > SSO session expiry time?
Yes
Call Sage SSO to extend session to required expiry time
Record new SSO session expiry time on application session,
Record new authentication token on application session.
Set application session expiry due mark to match response expiry due flag
Notification processed
No
Mark application session as due to expire
Notification processed

2.8. How to use the "expiry due" mark

After a `Session.ExpiryDue` notification has been successfully processed, the application session associated with the specified SSO session will have had its "expiry due" mark updated.

The expiry due mark tells your application whether the SSO session needs to be extended in response to further user activity on the session. Consider the following scenario for an application with an inactivity timeout of thirty minutes and a session start time *T*:

T+0: User signed in. SSO session established. Application session established.

T+15: Last recorded user activity on the application session.

T+27: `Session.ExpiryDue` notification received. Application extends SSO session to last activity time + application session inactivity timeout = T + 45. Sage SSO returns `ExpiryDue` = false in response.

...no further user activity is recorded...

T+42: `Session.ExpiryDue` notification received. Last activity time + application session inactivity time <= SSO session expiry time so no call is made the Sage SSO to extend the session and the application session is marked expiry due.

T+44: User activity is recorded on the session. **As the expiry due mark is set, the application calls Sage SSO to extend the SSO session to current time + application session inactivity timeout and clears the expiry due mark from the application session.**

Failure to call Sage SSO at this point will result in the SSO session ending before the application session.

...no further user activity is recorded...

T+71: `Session.ExpiryDue` notification received. Last activity time + application session inactivity time <= SSO session expiry time so no call is made the Sage SSO to extend the session and the application session is marked expiry due.

T+74: `Session.Ended` notification received. Application session is closed in response.

Therefore, when a protected page is requested, your application must:

- Check that the application session is authenticated

  This check is described in the section "When and how to sign-in a user" above.

- If the session is marked as expiry due, call Sage SSO to extend the session

  Record the new SSO session expiry time and user authentication token and clear the expiry due mark on the application session so that Sage SSO is not called again until another `Session.ExpiryDue` notification has been received.

If Sage SSO is called to extend an SSO session before the `Session.ExpiryDue` *notification has been sent it will return an error and the SSO session will not be extended.*

The following .NET code illustrates how a request for a protected page should be handled:

```
protected override void OnLoad(EventArgs e)
{
    if (IsSSOAuthenticatedSession)
    {
        Guid ssoSessionId = SSOSession.SSOSessionId;

        if (SSOSessionMap.ShouldExtendSSOSessionOnUserActivity(ssoSessionId))
        {
            // The Sage SSO session associated with this application session has
            // been marked expiry due since the last time there was activity.
            // Call Sage SSO to extend the session and then clear the mark so that we don't call
            // SessionExtend again until we receive another Session.ExpiryDue notification.

            using (WebSSOServiceSoapClient client =
                                                new WebSSOServiceSoapClient("WebSSOServiceSoapClient"))
            {
                client.Open();

                SessionExtendRequest request = new SessionExtendRequest()
                {
                    SessionId = ssoSessionId,
                    SessionExpiry = DateTime.UtcNow + TimeSpan.FromMinutes(Session.Timeout),
                    SessionExpirySpecified = true
                };

                try
                {
                    // Call Sage SSO to extend the session to match our application session.
                    SessionExtendResponse response = client.SessionExtend(request);
                    SSOSession.AuthenticationToken = response.UserAuthenticationToken;

                    // Extend the SSO session in the session map to synchronise with what Sage SSO
                    // tells us. It's important to observe the session expiry that's returned
                    // because the Sage SSO session may be nearing its hard timeout and we
                    // don't want to extend the application session beyond that.
                    SSOSessionMap.ExtendSSOSession(ssoSessionId, response.SessionExpiry);

                    // Unlike when we handle the Session.ExpiryDue notification, we always clear
                    // the expiry due flag here to avoid making continual calls to Sage SSO
                    // if the SSO session is nearing the hard timeout.
                    SSOSessionMap.SetShouldExtendSSOSessionOnUserActivity(ssoSessionId, false);
                }
                catch (Exception)
                {
                    // There was a problem extending the session on Sage SSO, or some other
                    // local problem with the SSO session map or application session state.
                    // The safest thing to do at this point is to end the application session.
                    EndSSOAuthenticatedSession();
                }
            }
        }
        else
        {
            try
            {
                // Record the activity in the SSO session map.
                SSOSessionMap.RefreshSSOSession(ssoSessionId);
            }
            catch (Exception)
            {
                // There was a problem refreshing the SSO session in the SSO session map. The
                // safest thing to do at this point is to end the application session.
                EndSSOAuthenticatedSession();
            }
        }
    }

    base.OnLoad(e);
}
```

2.9. How to process the Session.Ended notification

The `Type` element of notifications of this type contains the text `Session.Ended`.

The `Session.Ended` notification has the following parameters:

- `SessionId`: The ID of the SSO session to which this relates.
- `EmailAddress`: The email address of the user identity signed into the session.
- `Timestamp`: The time at which your application's participation in the SSO session ended.
- `Reason`: A reason code indicating why the session was ended, which will be one of the following

Code	Meaning
`ApplicationControlled`	Another application or an administrator closed the session.
`Timeout`	The session timed out.
`SwitchUser`	The user signed out of the session in order to sign-in to another application using a different account.

`SignOnAttemptFailure`	A sign-on attempt for another application, or a re-authentication for your application failed (for example because the account was locked) causing the session to be closed as a security precaution.
`RegistrationAttemptFailure`	An existing SSO user who is signed into your application tried to register as an existing user for another application which requires credentials and failed to authenticate, causing the session to be closed as a security precaution

Upon receiving and validating a `Session.Ended` notification, your application should take the following action:

- Extract the SSO session ID from the notification.
- Find the associated application session

  The notification (if received directly by your web application and not, for example, by a supporting service) will not be received on the same application session as the one associated with the SSO session ID when the user signed-in. Depending on your web application framework you may need to keep a map or table of SSO session IDs and application session IDs in order to determine which application session relates to the SSO session ID specified in the notification.

  If there is no associated application session, the notification should be ignored.
- End the application session

  Depending on your web application framework you may be able to end the application session directly, or you may need to indirectly end the session by removing the association between the SSO session and application session from a map or table which is checked each time a protected page is requested (the ASP.NET sample illustrates this approach).

For usability reasons, when a session ends your application should avoid simply dumping the user at a default page with no indication of what happened. Your application should, as a minimum, tell the user why the session ended.

2.10. How to sign-out a user

Often a user will not actively sign-out of a web application, instead closing the browser window. If the user re-opens the browser and the application session has not yet expired (and assuming that the user's browser is not, for example, set to clear cookies on closing) the user will usually be able to recommence use of the application without having to sign-in again.

This behaviour is convenient for the user and is, in general, perfectly acceptable for private machines. Sage SSO supports this behaviour by design. If a user who has previously signed-into your web application using Sage SSO returns and the application session is still "authenticated" there's no need to call Sage SSO again until the session is about to expire.

On public machines, however, the user needs a way to definitively end their SSO session. This is particularly important because Sage SSO allows the user to be transparently signed-on to web applications which are so configured. Failing to provide a sign-out function in your application, or not calling the appropriate Sage SSO sign-out API could, therefore, jeopardise the security of other web applications.

Sage SSO provides two APIs which are related to sign-out:

- `WebSSOService/SessionRemoveParticipant`

  This API removes your web application from a session in which it is a participant.

  If your web application is the last participant, the SSO session will immediately end and the next user of the same browser to sign-in to your web application or another web application which uses Sage SSO will be required to enter their credentials.

  If your web application is not the last participant, the SSO session will remain open and the next sign-in to your web application or another web application which uses Sage SSO will proceed according to the behaviour configured for that web application (see the above discussion of multiple web application sign-ins).

  If your web application uses this API, you must make it clear to the user that their session in any other participant web application in the same browser will remain open.

  If your web application is configured for transparent sign-in to existing SSO sessions, we recommend that you do not use this API. A user with an existing SSO session returning to your application in the same browser will be signed back in without any confirmation prompt or request for credentials. This can be confusing for the user and serves little purpose for your application.

- `WebSSOService/SessionSignOff`

  This API closes the specified session. All participant web applications will receive a `Session.Ended` notification. The next user of the same browser to sign-in to your web application or another web application which uses Sage SSO will be required to enter their credentials.

  We recommend that all web applications use this API. You must make it clear to the user that their session in any other participant web application in the same browser will be closed.

Both these APIs will only let your web application end or leave a session in which it is already a participant.

3. Using the SSO Customisation Kit

3.1. Introduction

Sage SSO allows web application developers to provide their own templates to replace the default pages that SSO serves to end users. These pages are hosted by the Sage SSO system, on the Sage SSO domain, and are vetted prior to deployment.

Customisation for each of the following flows is provided:

- First sign on (accepting full credentials)
- Additional sign on (when already signed in to one or more web applications)
- Express sign on (when already signed in to one or more web applications)
- Registration
- Password recovery
- Password change

To aid in the development of your customised templates, we provide a "customisation kit" which is located within the SDK at *\Samples\Customisation.* The following sections demonstrate using the customisation kit to develop a simple alternative view for your SSO application.

Customisation is provided for by an SSDP custom ASP.NET MVC 3.0 view engine, that performs efficient templating based on user provided "Views", our own parameterised SSO "controls" and locale specific resource strings. In all cases, default values are provided for your convenience.

3.2. Prerequisites

The SSO customisation kit consists of a special **Microsoft Visual Web Developer 2010 project.** This is a free download available which is available from Microsoft at the following location:

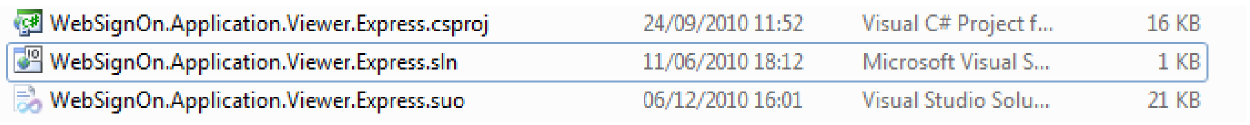
http://www.microsoft.com/express/Downloads/

The **Microsoft Web Platform Installer** will detect, download and install any pre-requisites.

By following the above link, it is possible to download the application as an ISO, or via the Web Platform Installer (recommended). Note, the full VS2010 Express is not required, only Web Developer Express 2010.

3.3. Examining the Customisation Kit Solution

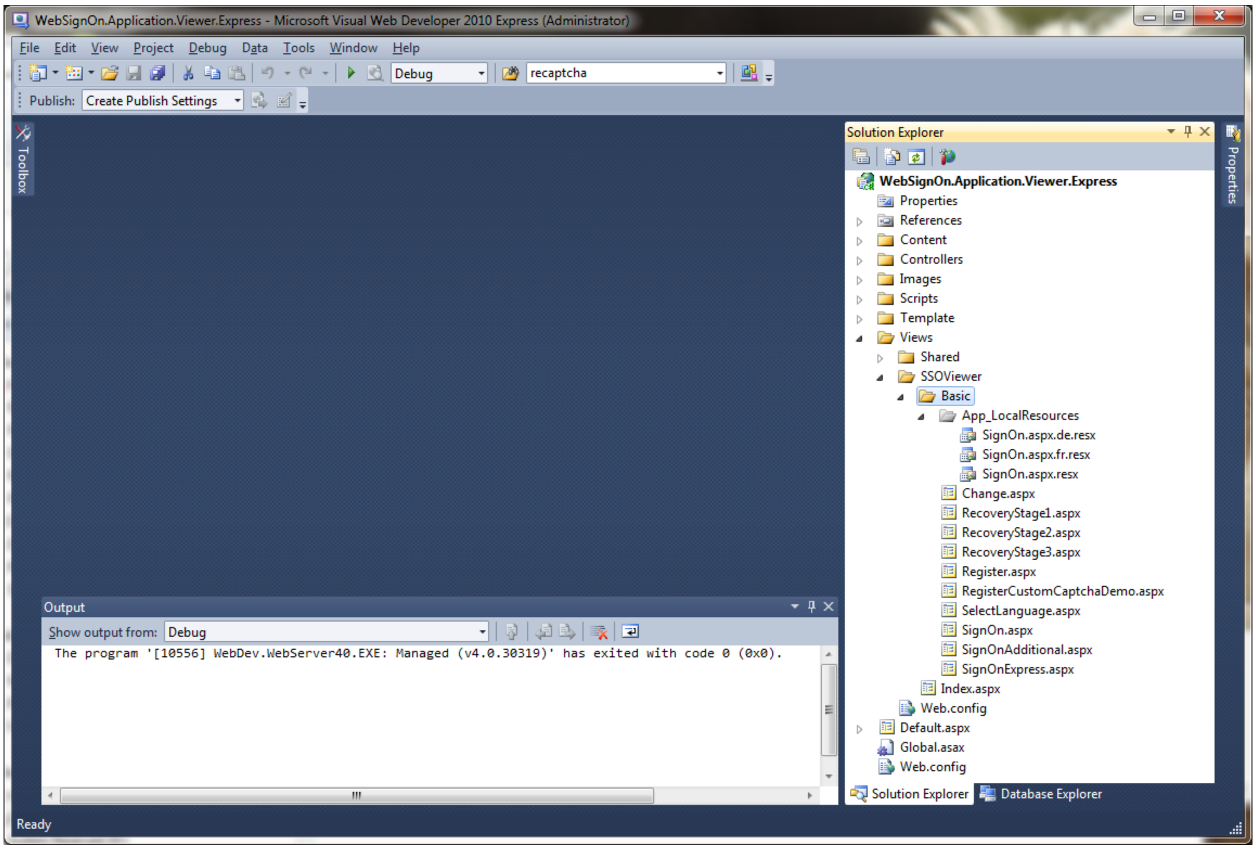
Once you have Visual Web Developer Express 2010 installed, you can extract the customisation kit from its .zip file, e.g. to C:\SSO\Customisation, and open the solution. The zip file is named WebSignOn.Application.ViewKit_XXXX.zip.



WebSignOn.Application.Viewer.Express.csproj	24/09/2010 11:52	Visual C# Project f...	16 KB
WebSignOn.Application.Viewer.Express.sln	11/06/2010 18:12	Microsoft Visual S...	1 KB
WebSignOn.Application.Viewer.Express.suo	06/12/2010 16:01	Visual Studio Solu...	21 KB

To open the Solution, double click the *WebSignOn.Application.Viewer.Express.sln* file.

The following screenshot shows the layout of the solution once it has been opened:



```
The program '[10556] WebDev.WebServer40.EXE: Managed (v4.0.30319)' has exited with code 0 (0x0).
```

On the right hand side, the tree shows the files in the solution. For the purpose of customisation, we are only interested in certain files in this list, and some files are out of bounds regarding the customisation of their contents.

Each folder in the tree holds files for a specific purpose. These are as follows:

- **Content** – This holds support files for the SDK. No files MUST be added or edited.
- **Controllers** – This holds the ***SSOViewerController.cs*** file which may be edited to parameterise the current views you are working on, e.g. to display the Captcha or not. This is only for the purpose of designing your template views to work with each of the various states that each page can be in.
- **Images –** This hold support files for the SDK. No files MUST be added or edited. (Note, your own images should be placed in the Template/XXXX folder as described later.)
- **Scripts** – This hold support files for the SDK. No files MUST added or edited.

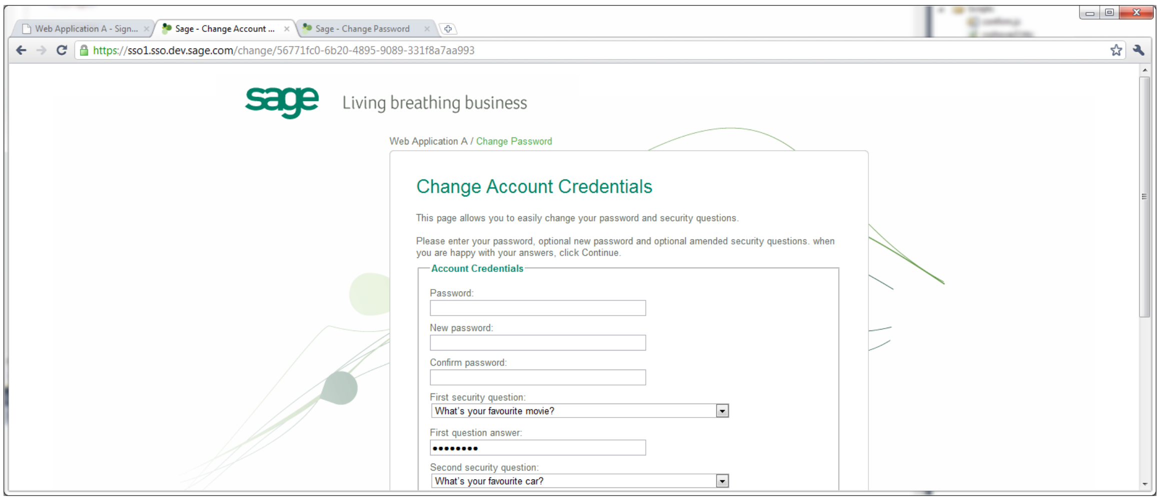
Please note that we do not support custom JavaScript functions or links to external script files in the custom templates. Any templates we receive containing client script will be rejected.

Contact ssdpdevelopersupport@sage.com for more information.

- **Template** – The holds various sub folders which correlate with your nominated and as yet undecided template name. These sub folders contain all the web assets your customised templates reference, e.g. images and cascading style sheet (CSS) files.

- **Views –** As with the Template folder, this contains sub folders that correlate with your nominated template name. These sub folders contain customised views and localised string resources as .resx and .aspx files.
- The views are specified as ASP.NET MVC mark-up, using our custom SSO partial views. There is IntelliSense support to aid the developer.
- Included in the customisation kit is a "Basic" template which contains all the items needed to demonstrate a skeleton template in the format that Sage SSO supports. You can see that in the Views\Basic folder, there are various .aspx files. These are as follows:

1. **Change.aspx** – The view for the password change page



Web Application A - Sign... | Sage - Change Account ... | Sage - Change Password

https://sso1.sso.dev.sage.com/change/56771fc0-6b20-4895-9089-331f8a7aa993

Living breathing business

Web Application A / Change Password

Change Account Credentials

This page allows you to easily change your password and security questions.

Please enter your password, optional new password and optional amended security questions. when you are happy with your answers, click Continue.

Account Credentials

Password:

New password:

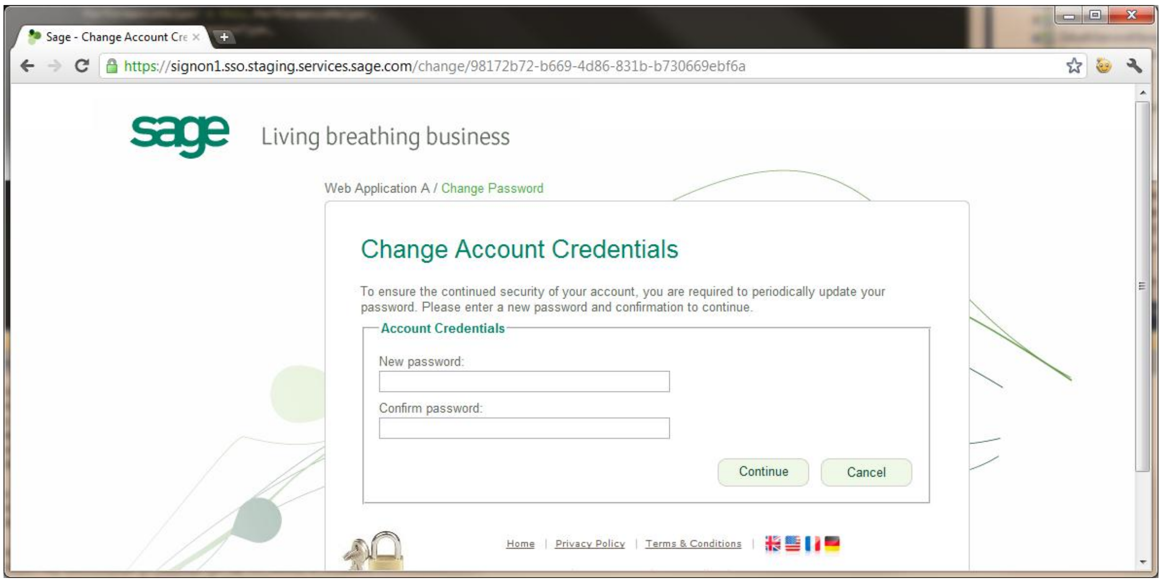
Confirm password:

First security question: What's your favourite movie?

First question answer: ••••••••

Second security question: What's your favourite car?

2. **ChangePasswordOnly.aspx (since v.1.1)** – The view used for administrative forced password changes or when a user's password has expired and needs to be changed. This is displayed as part of the sign on flow.



Sage - Change Account Cre

https://signon1.sso.staging.services.sage.com/change/98172b72-b669-4d86-831b-b730669ebf6a

Living breathing business

Web Application A / Change Password

Change Account Credentials

To ensure the continued security of your account, you are required to periodically update your password. Please enter a new password and confirmation to continue.

Account Credentials

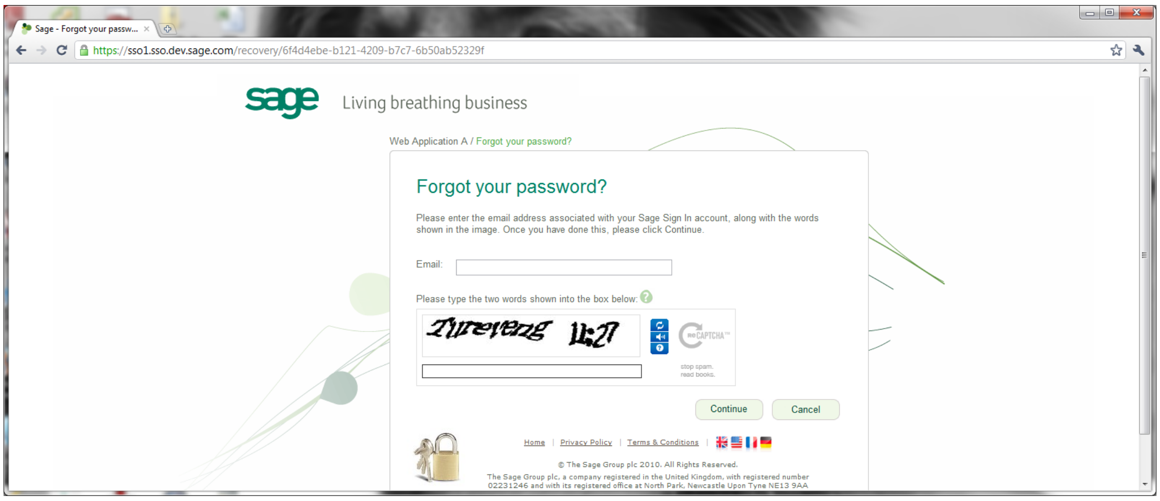
New password:

Confirm password:

Continue | Cancel

Home | Privacy Policy | Terms & Conditions

3. **RecoveryStage1.aspx** – The view for the first step of the password recovery flow. This step includes a box for the user to enter their email address and a Captcha to validate the user is a human being.


Sage - Forgot your passw...

https://sso1.sso.dev.sage.com/recovery/6f4d4ebe-b121-4209-b7c7-6b50ab52329f

Living breathing business

Web Application A / Forgot your password?

Forgot your password?

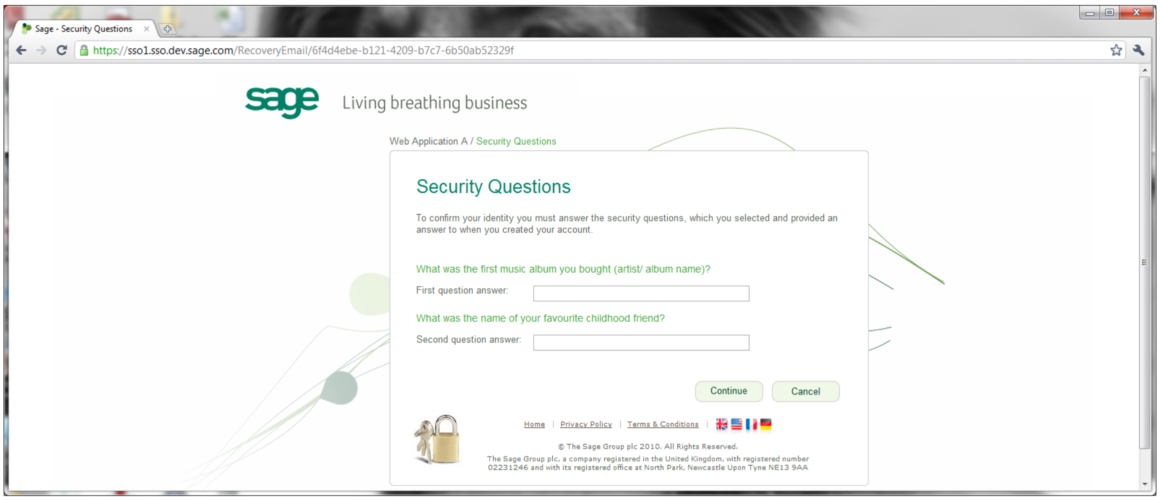
Please enter the email address associated with your Sage Sign In account, along with the words shown in the image. Once you have done this, please click Continue.

Email:

Please type the two words shown into the box below:

© The Sage Group plc 2010. All Rights Reserved.
The Sage Group plc, a company registered in the United Kingdom, with registered number 02231246 and with its registered office at North Park, Newcastle Upon Tyne NE13 9AA

4. **RecoveryStage2.aspx** – The view for the second step of the password recovery flow. This step displays two security questions to the user and has boxes for the user to enter their answers.


Sage - Security Questions

https://sso1.sso.dev.sage.com/RecoveryEmail/6f4d4ebe-b121-4209-b7c7-6b50ab52329f

Living breathing business

Web Application A / Security Questions

Security Questions

To confirm your identity you must answer the security questions, which you selected and provided an answer to when you created your account.

What was the first music album you bought (artist/ album name)?

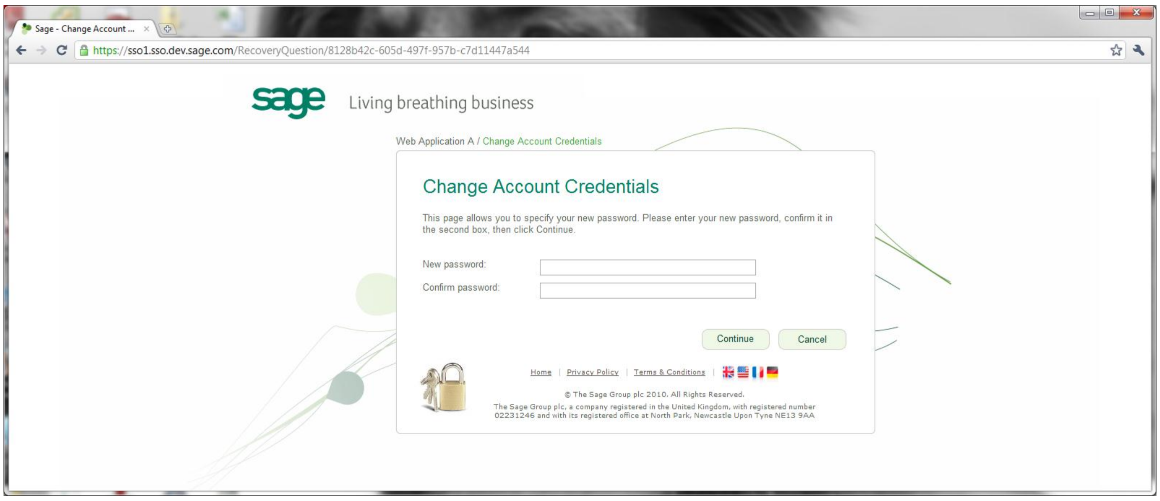
First question answer:

What was the name of your favourite childhood friend?

Second question answer:

© The Sage Group plc 2010. All Rights Reserved.
The Sage Group plc, a company registered in the United Kingdom, with registered number 02231246 and with its registered office at North Park, Newcastle Upon Tyne NE13 9AA

5. **RecoveryStage3.aspx** – The view for the third step of the password recovery flow. This allows the user to enter a new password, and to confirm it.


Sage - Change Account ...

https://sso1.sso.dev.sage.com/RecoveryQuestion/8128b42c-605d-497f-957b-c7d11447a544

Living breathing business

Web Application A / Change Account Credentials

Change Account Credentials

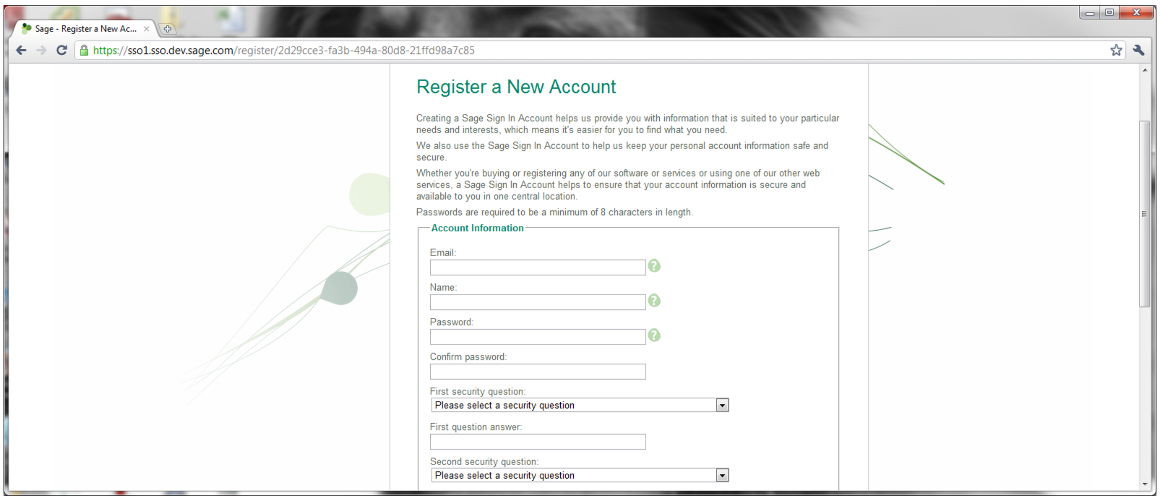
This page allows you to specify your new password. Please enter your new password, confirm it in the second box, then click Continue.

New password:

Confirm password:

© The Sage Group plc 2010. All Rights Reserved.
The Sage Group plc, a company registered in the United Kingdom, with registered number 02231246 and with its registered office at North Park, Newcastle Upon Tyne NE13 9AA

6. **Register.aspx** – This view contains the main registration fields for a new user and a Captcha to stop automated registrations by a third party.


Sage - Register a New Ac...
https://sso1.sso.dev.sage.com/register/2d29cce3-fa3b-494a-80d8-21ffd98a7c85

Register a New Account

Creating a Sage Sign In Account helps us provide you with information that is suited to your particular needs and interests, which means it's easier for you to find what you need.

We also use the Sage Sign In Account to help us keep your personal account information safe and secure.

Whether you're buying or registering any of our software or services or using one of our other web services, a Sage Sign In Account helps to ensure that your account information is secure and available to you in one central location.

Passwords are required to be a minimum of 8 characters in length.

Account Information

Email:

Name:

Password:

Confirm password:

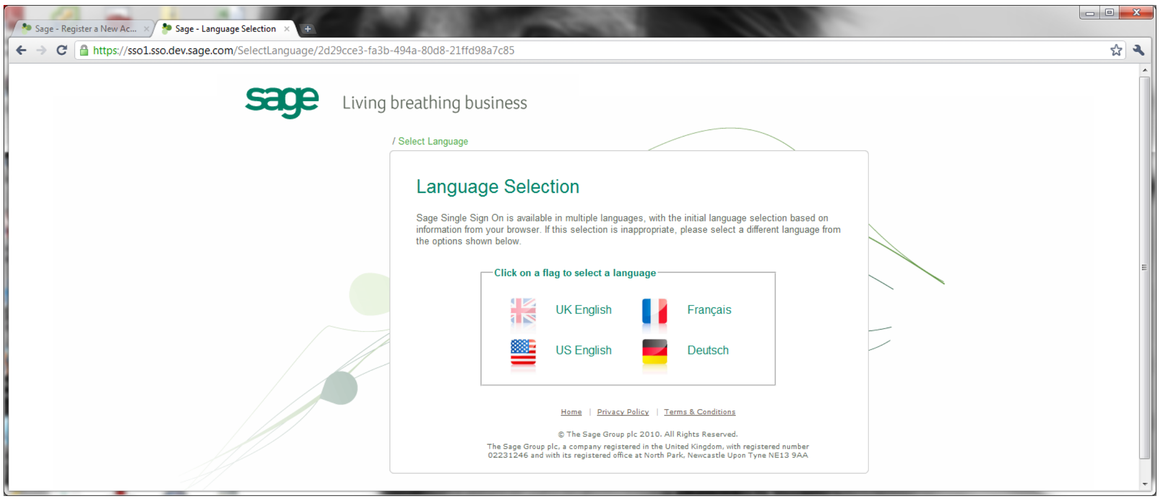
First security question: Please select a security question

First question answer:

Second security question: Please select a security question

7. **RegisterCustomCaptchaDemo.aspx** – This view is the same as register.aspx but is included in the SDK as an example of how to style your own Captcha if you do not wish to use the default themes provided by ReCaptcha. It is purely an example to demonstrate the custom <DIV> required and is not a valid filename for an SSO template.

8. **SelectLanguage.aspx** – This view is a page which allows the selection of a different language for the SSO pages by the end user. It is only displayed if the client does not have script enabled, otherwise a CSS based dialog box will appear rather than a redirect to this view (if the CultureSwitcher widget is implemented).


Sage - Register a New Ac...
Sage - Language Selection
https://sso1.sso.dev.sage.com/SelectLanguage/2d29cce3-fa3b-494a-80d8-21ffd98a7c85
Living breathing business
/ Select Language

Language Selection

Sage Single Sign On is available in multiple languages, with the initial language selection based on information from your browser. If this selection is inappropriate, please select a different language from the options shown below.

Click on a flag to select a language

UK English Français

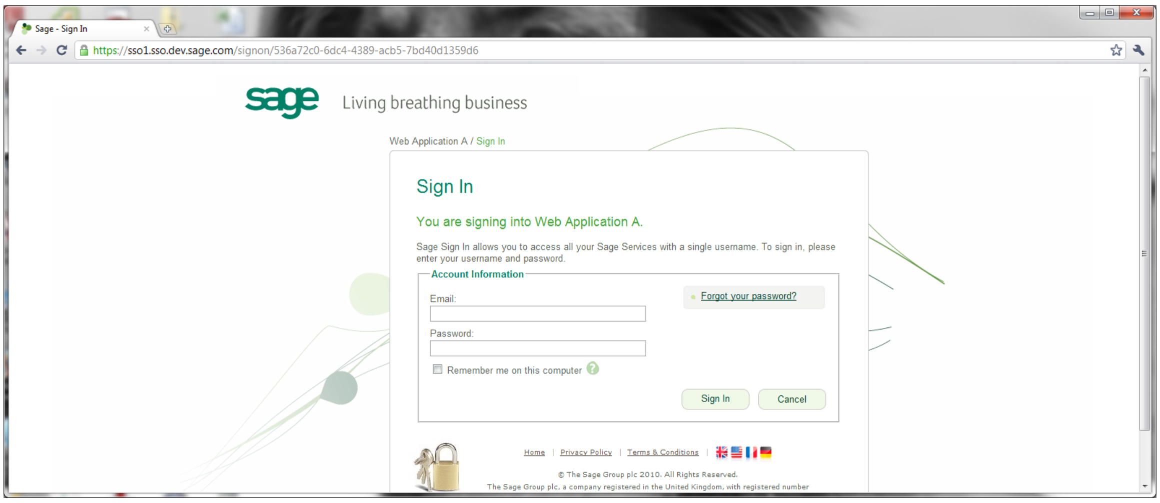
US English Deutsch

Home | Privacy Policy | Terms & Conditions

© The Sage Group plc 2010. All Rights Reserved.

The Sage Group plc, a company registered in the United Kingdom, with registered number 02231246 and with its registered office at North Park, Newcastle Upon Tyne NE13 9AA

9. **SignOn.aspx –** The view for the sign on page, requesting an email address and password, for when the user is not currently signed into a web application served by Sage SSO.


Sage - Sign In

https://sso1.sso.dev.sage.com/signon/536a72c0-6dc4-4389-acb5-7bd40d1359d6

Living breathing business

Web Application A / Sign In

Sign In

You are signing into Web Application A.

Sage Sign In allows you to access all your Sage Services with a single username. To sign in, please enter your username and password.

Account Information

Email:

Password:

Remember me on this computer

Forgot your password?

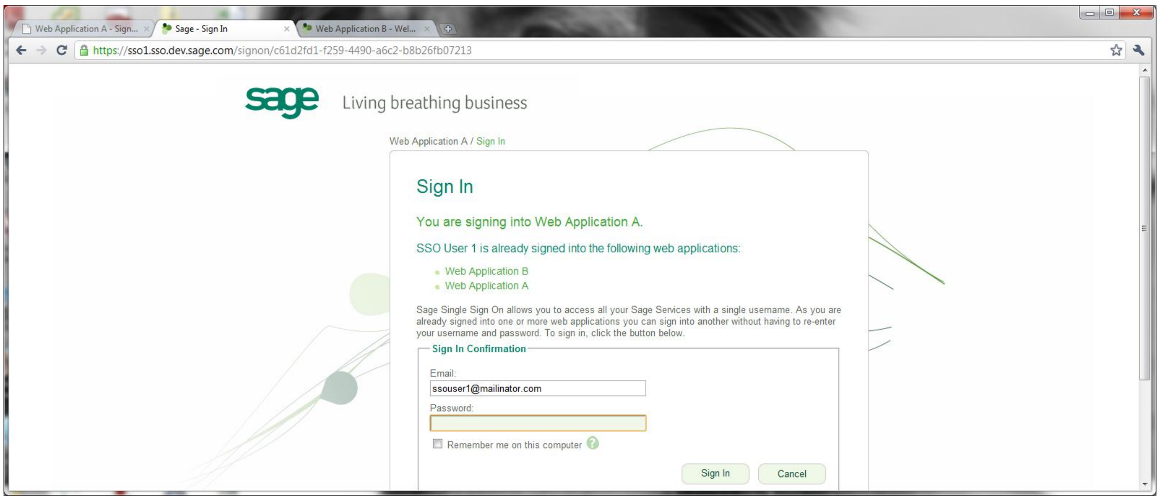
Sign In | Cancel

Home | Privacy Policy | Terms & Conditions

© The Sage Group plc 2010. All Rights Reserved.

The Sage Group plc, a company registered in the United Kingdom, with registered number

10. **SignOnAdditional.aspx –** The view for the sign on page, requesting an email address and password, for when the user is already signed into a web application served by Sage SSO. This view also displays a list of the web applications the user is currently signed into, and the name of the user.


Web Application A - Sign... | Sage - Sign In | Web Application B - Wel...

https://sso1.sso.dev.sage.com/signon/c61d2fd1-f259-4490-a6c2-b8b26fb07213

Living breathing business

Web Application A / Sign In

Sign In

You are signing into Web Application A.

SSO User 1 is already signed into the following web applications:

- Web Application B
- Web Application A

Sage Single Sign On allows you to access all your Sage Services with a single username. As you are already signed into one or more web applications you can sign into another without having to re-enter your username and password. To sign in, click the button below.

Sign In Confirmation

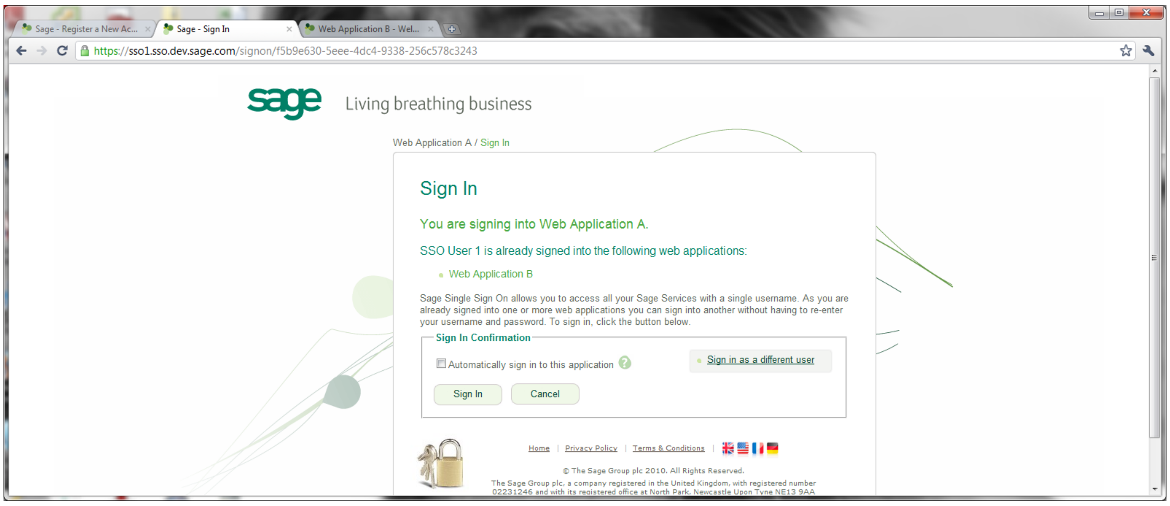
Email: ssouser1@mailinator.com

Password:

Remember me on this computer

Sign In | Cancel

11. **SignOnExpress.aspx** – The view for the sign on page, containing a sign in confirmation button, for when the user is already signed into a web application served by Sage SSO. This will only appear if your web application is not configured to *AlwaysRequestCredentials*.


Sage - Register a New Ac... | Sage - Sign In | Web Application B - Wel...

https://sso1.sso.dev.sage.com/signon/f5b9e630-5eee-4dc4-9338-256c578c3243

Living breathing business

Web Application A / Sign In

Sign In

You are signing into Web Application A.

SSO User 1 is already signed into the following web applications:

- Web Application B

Sage Single Sign On allows you to access all your Sage Services with a single username. As you are already signed into one or more web applications you can sign into another without having to re-enter your username and password. To sign in, click the button below.

Sign In Confirmation

Automatically sign in to this application

Sign in as a different user

Sign In | Cancel

Home | Privacy Policy | Terms & Conditions

© The Sage Group plc 2010. All Rights Reserved.

The Sage Group plc, a company registered in the United Kingdom, with registered number 02231246 and with its registered office at North Park, Newcastle Upon Tyne NE13 9AA

The customisation kit follows the MVC pattern, meaning that there is a separation of concerns between the view, the model and the controller. The purpose of each component is as follows:

- The **M**odel contains a set of data
- The **V**iews display the model data in the required format
- The **C**ontroller populates the model and decides which view to use to display it

As a developer of a new template for SSO, you are responsible for providing us with customised views that fit the SSO models. The Sage SSO system controls the models and the responsibilities of the controller, in reference to the above pattern. The model and the controller cannot be changed.

The intention of the customisation kit solution is to provide a basic framework to set the states of the model so that you can perform WYSIWYG editing of your views. ***The kit does not provide logic to emulate the SSO flows.***

In summary, the files and folders you can edit or add to as are follows:

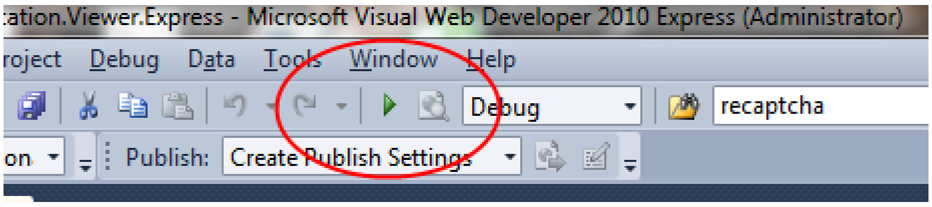
- Any views and resources below the ***Views\SSOViewer\XXX*** folder
- Any files in the ***Template\XXX***

  (where XXX is the name of your new template)

3.4. Anatomy of a Basic View and Trying the Sample

Once you have finished examining the customisation kit solution it is time to run the main project which will display the SSO view selection screen.

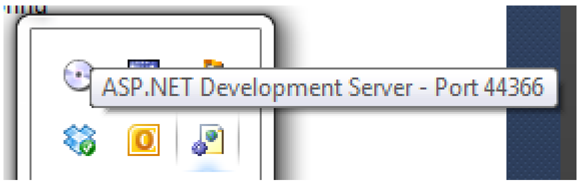
To run the solution, press the green play button on the toolbar (or press F5).



:ation.Viewer.Express - Microsoft Visual Web Developer 2010 Express (Administrator)
roject
Debug
Data
Tools
Window
Help
Debug
recaptcha
on
Publish:
Create Publish Settings

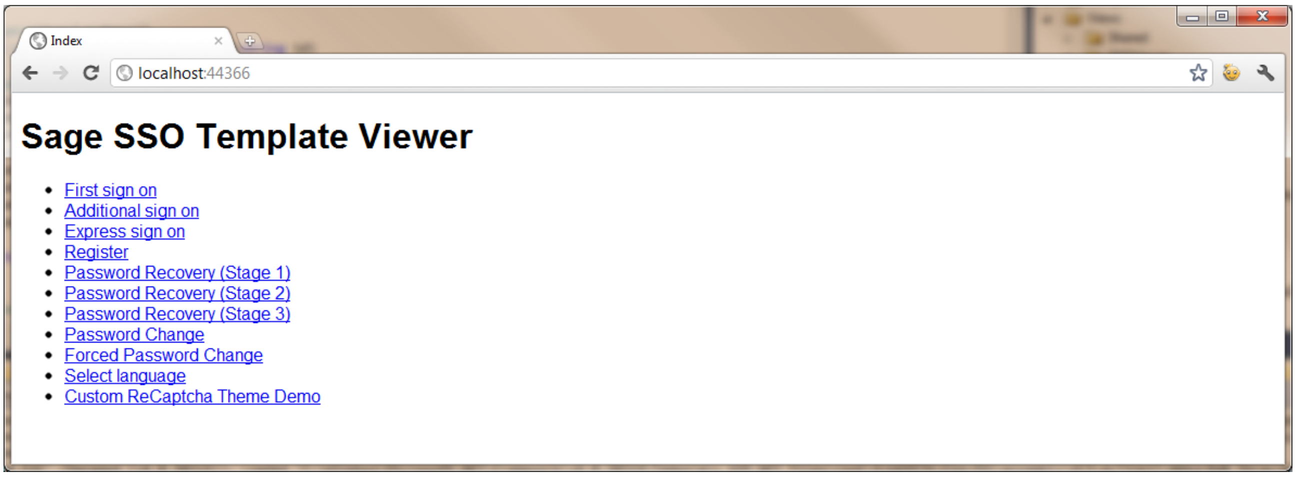
When you press the button the following happens:

- Visual Web Developer launches the ASP.NET Development Web Server, which you can see located in the system tray. It uses a random unassigned port.



Please note, the ASP.NET Development Web Server works exclusively with the ***localhost*** *loopback address. You will not be able to access hosted content from another machine or across a network.*

- Your default web browser is launched and redirected to the SSO view selection screen, hosted by the ASP.NET Development Web Server.



Sage SSO Template Viewer

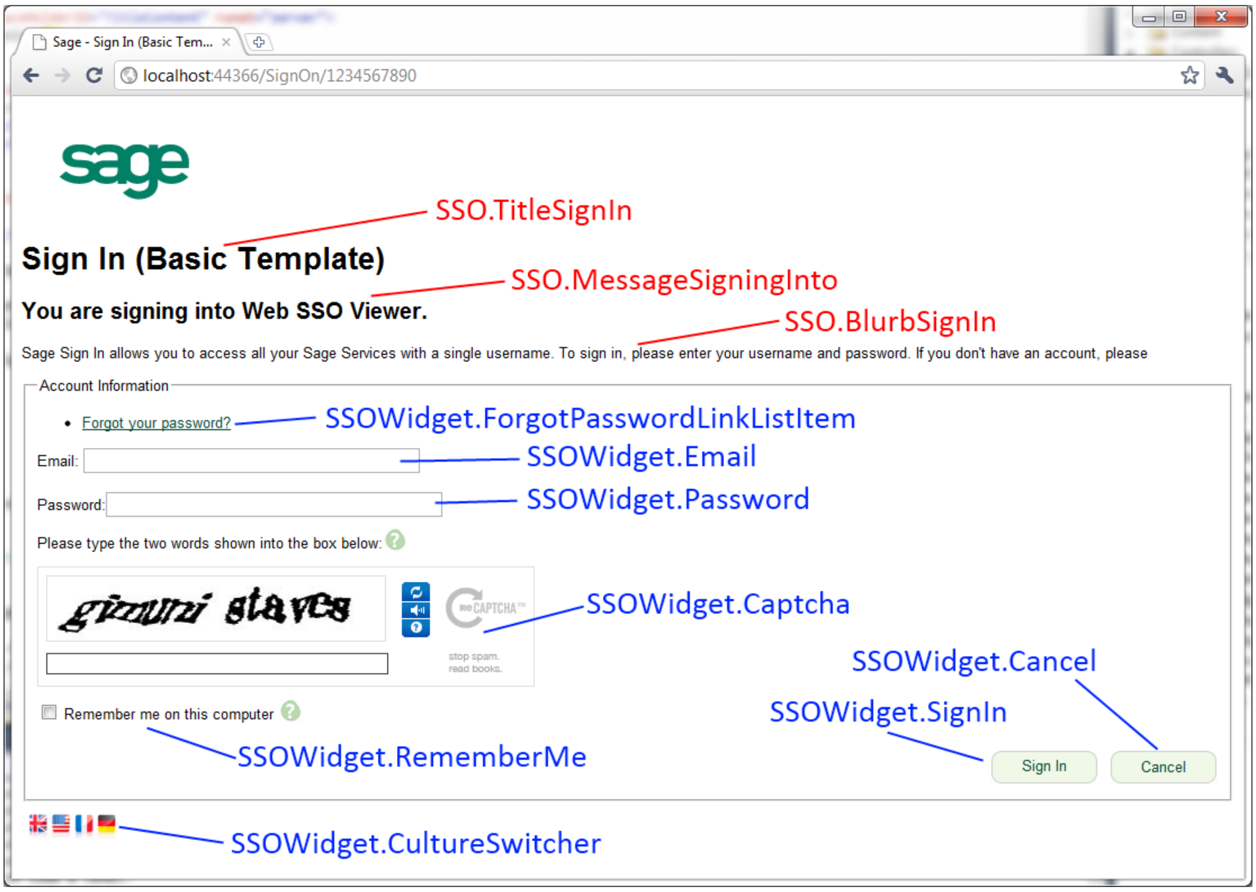
- First sign on
- Additional sign on
- Express sign on
- Register
- Password Recovery (Stage 1)
- Password Recovery (Stage 2)
- Password Recovery (Stage 3)
- Password Change
- Forced Password Change
- Select language
- Custom ReCaptcha Theme Demo

If you do not see the above screen, ensure that the correct URL exists in the address bar. It should be **http://localhost:XXXX/** where **XXXX** is the port the ASP.NET Development Web Server is using.

The purpose of this page is to allow the selection of the various views you have created so you can view the changes you have made. Any changes made to the views are displayed in real time by either refreshing the browser window (F5), or by re-launching the solution as before. The only caveat is that you must remember to save your files.

Let's examine the SignOn.aspx view. To do this, we click on the link for "First Sign On".

The following is an annotated view of the page which should be displayed then the link is clicked:


Sage - Sign In (Basic Tem...

localhost:44366/SignOn/1234567890

Sign In (Basic Template)

You are signing into Web SSO Viewer.

Sage Sign In allows you to access all your Sage Services with a single username. To sign in, please enter your username and password. If you don't have an account, please

Account Information

- Forgot your password?

Email:

Password:

Please type the two words shown into the box below:

Remember me on this computer

Sign In Cancel

The items in red are references to one of the standard SSO resource strings that are provided by Sage SSO out of the box. These messages can be changed by overriding the relevant resource key in a view specific local resource file. This process will be described in detail in a later section of this document.

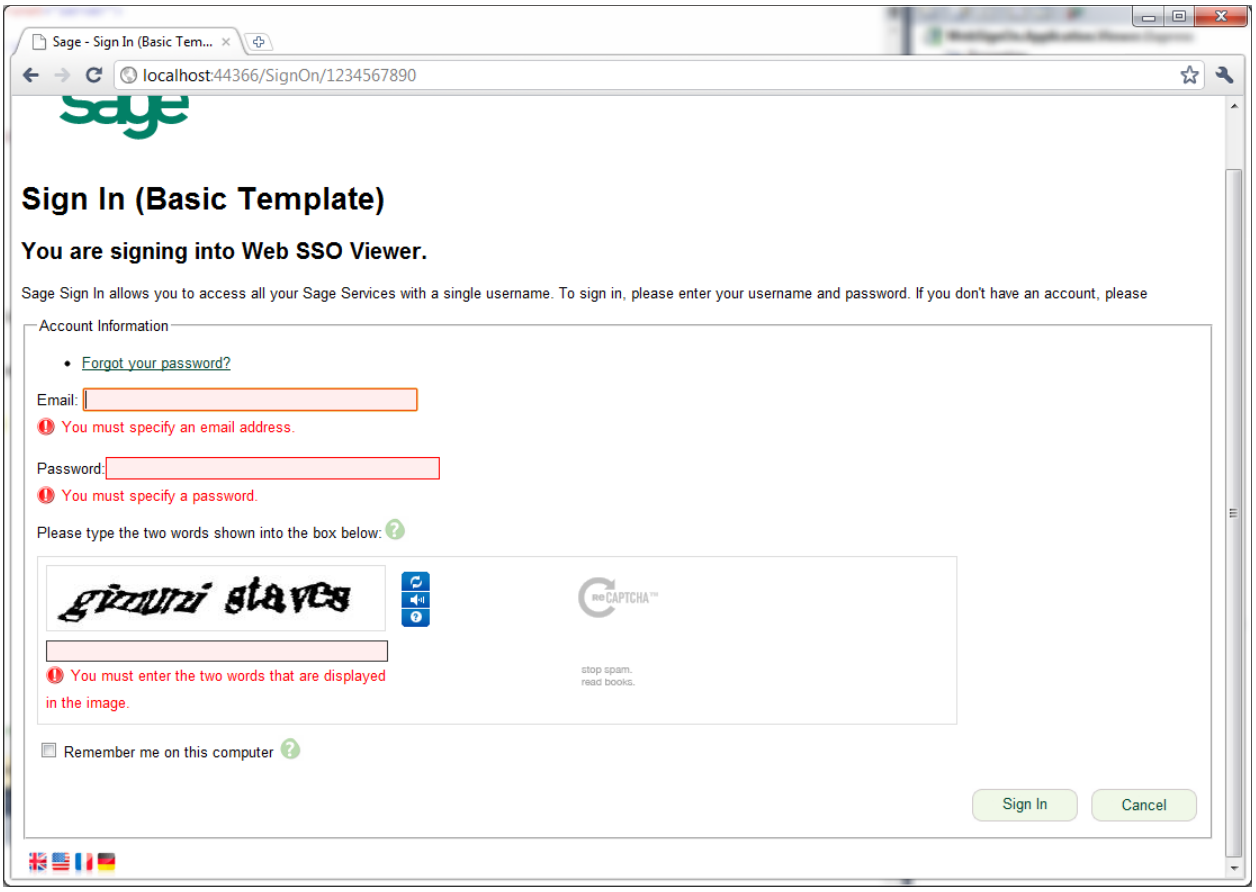
`<%= Html.Resource(SSO.XXX) %>` *(where XXX is the resource key)*

The items in blue denote parts of the rendered view which come from built in **SSO widgets.** An SSO widget is basically a ***partial view*** which is placed within the layout using a special piece of HTML mark-up. There are many SSO widgets which can be laid out in your views, in a specific place, by using the following tag:

`<% Html.SSO(SSOWidget.XXX); %>` *(where XXX is the name of the widget)*

SSO widgets perform the task of abstracting away the display logic of each of the key components on an SSO view. For example, the registration view has fields for email and display name. These fields can be provided by a web application before a registration attempt is started. In this case, SSO will make these two fields read-only so that the user cannot edit them during registration. This is performed automatically by the SSO controller and widget logic.

Returning to the browser window, if you click the “Sign In” button, you will see that the client side field validation is executed. This is key advantage to the customisation kit, allowing you to style your page layout, but also view changes made to the style of the validation messages in a realistic manner:


localhost:44366/SignOn/1234567890

Sign In (Basic Template)

You are signing into Web SSO Viewer.

Sage Sign In allows you to access all your Sage Services with a single username. To sign in, please enter your username and password. If you don't have an account, please

Account Information

- Forgot your password?

Email:

You must specify an email address.

Password:

You must specify a password.

Please type the two words shown into the box below:

You must enter the two words that are displayed in the image.

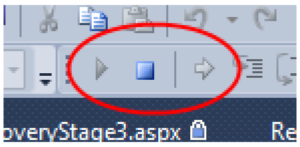
Remember me on this computer

Views are laid out in four sections as follows:

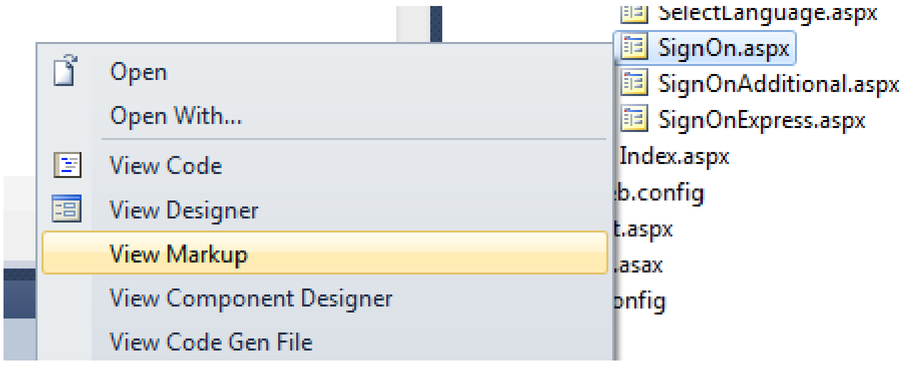
- **TitleContent** – Content declared in this section will be placed within the <title></title> HTML tags for the page. As such, this content should only be a plain string / tag which resolves to a plain string.
- **HeadContent** – Content declared in this section will be placed within the <head></head> HTML tags for the page. Examples of markup you may which to include here include CSS file references, e.g *<link rel="stylesheet" type="text/css" href="/template/xxxx/xxxx.css">*
- **MainContent –** Content declared in this section will be placed within the <body></body> HTML tags for the page. This is where the majority of your HTML should go.
- **FooterContent** – Content declared in this section will appear at the bottom of the page, e.g. the culture switcher widget

Now it is time to examine the HTML / ASP.NET MVC mark-up which creates the Basic template SSO Sign On View.

- Return to Visual Web Developer and press the square stop button in the toolbar.



- Next, right click on the SignOn.aspx View in the "Solution Explorer", then select "View Markup"



The following code listing is displayed, detailing the special mark-up required for the page. Each of the four sections described above is denoted by an *<asp:Content>* tag:

```
<%@ Page Language="C#" MasterPageFile="~/Views/Shared/Template.Master"
Inherits="System.Web.Mvc.ViewPage<WebSignOn.Application.SignOnModel>" %>
<%@ Import Namespace="WebSignOn.Application" %>
<%@ Import Namespace="Resources" %>

<asp:Content ID="titleContent" ContentPlaceHolderID="TitleContent" runat="server">
    Sage - <%=Html.Resource(SSO.TitleSignIn) %>
</asp:Content>

<asp:Content ID="headContent" ContentPlaceHolderID="HeadContent" runat="server">
<!-- This appears within the HEAD section. Link to stylesheets here via
/Template/<TemplateName>/file.ext -->
    <link rel="stylesheet" type="text/css" href="/Template/Basic/style.css" />
    <link rel="stylesheet" type="text/css" href="/Template/Basic/confirm.css" />
</asp:Content>

<asp:Content ID="loginContent" ContentPlaceHolderID="MainContent" runat="server">
    <div>
        <img src="/Images/sageclearzone.gif" title="Sage Group plc Logo" alt="Sage Logo" />
        <h1>
            <%= Html.Resource(SSO.TitleSignIn) %></h1>
        <h2>
            <%= Html.Resource(SSO.MessageSigningInto)%>
            <%= Html.Encode(Model.WebApplicationDisplayName)%>.</h2>
        <p>
            <%= Html.Resource(SSO.BlurbSignIn) %>
        </p>
        <% using (Html.BeginForm())
           { %>
        <%= Html.SSOAntiForgeryToken() %>
        <div>
            <fieldset>

                <legend><%= Html.Resource(SSO.TitleAccountInformation) %></legend>

                <ul>
```

```
                    <% Html.SSO(SSOWidget.ForgotPasswordLinkListItem); %>
                </ul>

                <% Html.SSO(SSOWidget.SignInValidationSummary); %>

                <p><% Html.SSO(SSOWidget.Email); %></p>
                <p><% Html.SSO(SSOWidget.Password); %></p>

                <% Html.SSO(SSOWidget.Captcha); %>

                <p><% Html.SSO(SSOWidget.RememberMe); %></p>

                <p>
                    <% Html.SSO(SSOWidget.Cancel); %>
                    <% Html.SSO(SSOWidget.SignIn); %>
                </p>
            </fieldset>
        </div>
        <% } %>

        <!-- Validation script needs to be positioned after the form -->
        <% Html.SSO(SSOWidget.SignInValidationScript); %>

    </div>

    <script type="text/javascript">
    <!--
        document.getElementById('sso_Email').setAttribute("autocomplete","off");
        document.getElementById('sso_Password').setAttribute("autocomplete","off");
    //-->
    </script>

</asp:Content>

<asp:Content ID="footerContent" ContentPlaceHolderID="FooterContent" runat="server">
    <!-- This appears at the bottom of the page -->
    <% Html.SSO(SSOWidget.CultureSwitcher); %>
</asp:Content>
```

You must only edit the content between the <asp:Content> and corresponding </asp:Content> tags. The Page and Import directives should remain untouched.

The top line of the .aspx mark-up includes a declaration of the Model type which the view uses. In the above example, we can see that it refers to the ***SignOnModel*** type. When we look at the controller later in the document, the purpose of this will become clear.

Feel free to run the solution again and select different views. It may also be helpful to view the source of the HTML pages to get a feel for how the above source relates to the HTML emitted to the browser.

3.5. Creating Your Own Template

To begin creating your own template, which will consist of a new set of customised "Views" and optionally, new resources, CSS and images, the following process should be followed:

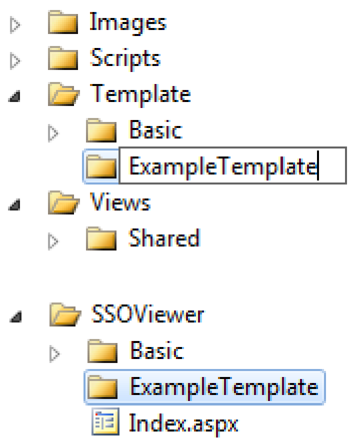
- Agree a template name with the SSDP team, by emailing your request to **ssdpdevelopersupport@sage.com**

The template name helps define the structure of the files on our servers, and uniquely identifies the set of files required when your web application requests an SSO sign-on, or other flow. It is the same as the *TemplateName* parameter that your web application developers can provide when initiating an SSO flow.

As the name is part of a directory path, it should not contain any special characters or spaces. Examples of valid template names include *UKSageForum, UKSageCustomerExtranet* and *UKSageLine50.*

- Create working directories for your template in the customisation kit project

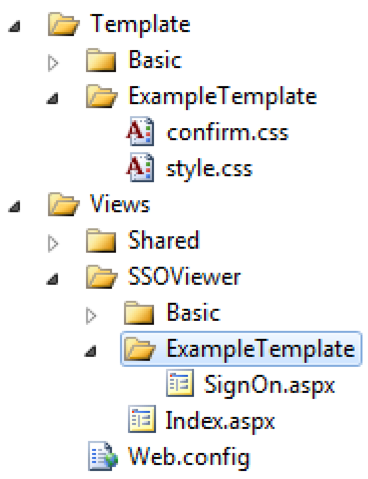
Once you have decided on the name of your template you must create two directories, which are each named the same as the template name, in the customisation kit project. One should be located within the **Templates** folder, and one should be located within the **Views\SSOViewer** folder.



1. Right click on the *Template* folder and select *Add*, then *New Folder.* Enter "ExampleTemplate" (or your template name).
2. Right click on the SSOViewer folder and select Add, then New Folder. Enter "ExampleTemplate" (of your template name).
3. Copy the *Basic* template files (from the *Template\Basic* folder) to use as a basis for your new template to your new *Template\ExampleTemplate* folder. There should be three files, named *confirm.css, favicon.ico* and *style.css*
4. Copy the Basic views (from the Views\SSOViewer\Basic folder) to use as a basic for your new template to your new Views\SSOViewer\ExampleTemplate folder. To start with, for the purpose of this tutorial, you can just copy the SignOn.aspx file. You can copy the rest when you come to customise these views later on.

For the purpose of this example, we will use a template name of "*ExampleTemplate"*.

Once complete, you should have the following layout:



5. Edit the SignOn.aspx view so that it refers to the correct CSS files in the new template folder. To do this, right click on the *SignOn.aspx* file in the *ExampleTemplate* folder and select "View Markup".

   About 10 lines from the top of the file, within the HeadContent, there are the links to the CSS files we copied earlier:

```
<asp:Content ID="headContent" ContentPlaceHolderID="HeadContent" runat="server">
    <!-- This appears within the HEAD section. Link to stylesheets here via
/Template/<TemplateName>/file.ext -->
    <link rel="stylesheet" type="text/css" href="/Template/Basic/style.css" />
    <link rel="stylesheet" type="text/css" href="/Template/Basic/confirm.css" />
</asp:Content>
```

   Change the references to the Basic template to refer to the new ExampleTemplate as follows, and then Save the file:

```
<asp:Content ID="headContent" ContentPlaceHolderID="HeadContent" runat="server">
    <!-- This appears within the HEAD section. Link to stylesheets here via
/Template/<TemplateName>/file.ext -->
    <link rel="stylesheet" type="text/css" href="/Template/ExampleTemplate/style.css" />
    <link rel="stylesheet" type="text/css" href="/Template/ExampleTemplate/confirm.css" />
</asp:Content>
```

6. Edit the page to include some customisation. E.g. Add a new <p> element below the BlurbSignIn resource, as follows:

```
<p>
<%= Html.Resource(SSO.BlurbSignIn) %>
</p>
<p class="example">
    This is rendered using the ExampleTemplate.
</p>
```

Notice that the <p> tag in this has the class "example" specified. Let's go and declare this in the correct place.

7. Go to the *Template\ExampleTemplate\style.css* file and double click it. Add the following CSS class declaration to the top of the file, ***but after the body{}*** declaration so that you have the following:

```
/*
      BASIC SSO template style sheet
      The classes below are an example and can be modified / added to as
      individual teams see fit.
*/
body
{
      background: #ffffff;
      font-family: Arial, sans-serif;
      font-size: 10pt;
}

.example
{
    color: #00dd00; /* green */
    font-style: italic;
}
```

Save style.css

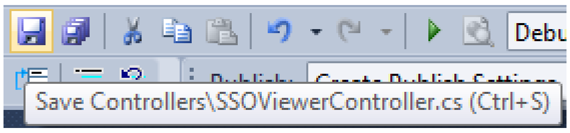
- Change the SSOViewerController to use your new template

  As mentioned previously, the SSOViewerController is responsible to orchestrating the model and view components. You will need to change a line of code in the controller so that the view selection page returns your new template.

  1. Double click the *SSOViewerController.cs* file in the *\Controllers* folder.
  2. Near the top, there is a notice comment and a line like the following:

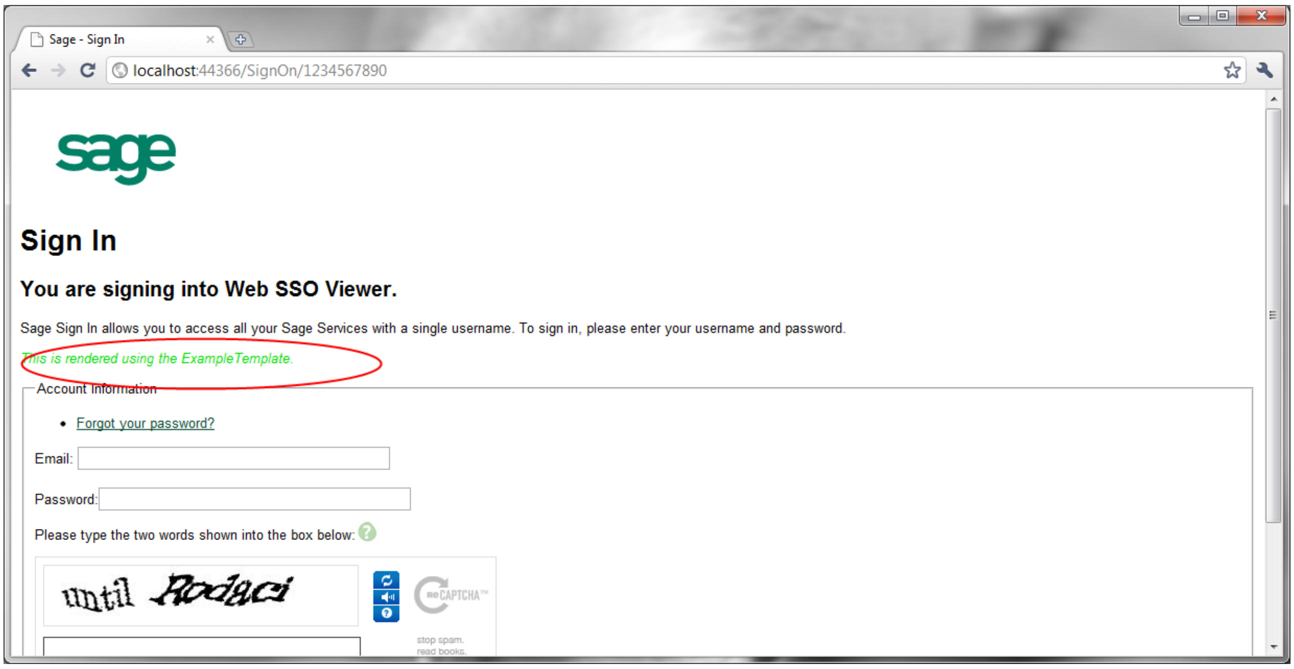
     ```
     private const string TemplateName = "Basic";
     ```

     Change "*Basic*" to "*ExampleTemplate*" and Save the file. If prompted that the file is readonly, click "Overwrite".

- Check your new template is working as expected

  1. Press F5 (or click the Play button) to start the project and bring up your web browser as before.
  2. Click the "First sign on" link. Your version of the basic template should be displayed. View the page source to confirm that the resources are being loaded from the correct location (by examining the references in the *<link>* tags within the *<head>* section).


Sage - Sign In
localhost:44366/SignOn/1234567890

Sign In

You are signing into Web SSO Viewer.

Sage Sign In allows you to access all your Sage Services with a single username. To sign in, please enter your username and password.

This is rendered using the ExampleTemplate.

Account Information

- Forgot your password?

Email:

Password:

Please type the two words shown into the box below:

Congratulations, you've now completed a walkthrough of the basics of the customisation process. More detail is available in the next sections.

3.6. SSO Widget Requirements

The SSO widgets are ASP.NET MVC 3.0 partial views which encapsulate business rule based presentation logic away from the template designer. Many SSO widgets can accept parameters to allow them to behave in slightly different ways to ease integration with your template design.

Each SSO view requires a certain set of SSO widgets to function properly. These are displayed in the following list, and also be definition, by looking at the Basic template sample views.

You should tick each one off, once you've ensured your views contain them.

- SignOn.aspx
  - ☐ **ForgotPasswordLink / ForgotPasswordLinkListItem** – Dynamic link to start a forgotten password flow, formatted as a list item (<li></li>)
  - ☐ **SignInValidationSummary** – Placeholder for a bulleted list that displays server side errors.
  - ☐ **Email** – Input box for an email address.
  - ☐ **Password** – Input box for a password.
  - ☐ **Captcha** – Placeholder for a ReCaptcha based Captcha that is displayed only if necessary (SSO server decides).
  - ☐ **RememberMe** – The checkbox that the user can tick to remember allow the computer to remember their email address

- ☐ **Cancel** – The cancel button, to cancel a sign in attempt.
- ☐ **SignIn** – The actual sign in button.
- ☐ **SignInValidationScript** – Client side validation script placeholder.
- ☐ **CultureSwitcher** – A widget that provides a dynamic link to the culture selection screen (if script not enabled) and a link to pop up a CSS based culture selection dialog otherwise.
- ☐ **Html.SSOAntiForgeryToken()** – This is required to mitigate CSRF attacks on Sage ID. Please note, you should use the Sage ID variant of this tag and MUST NOT use Html.AntiforgeryToken().

- SignOnAdditional.aspx
  - ☐ **ConfirmSignOutDialog** – A widget that writes HTML to the page that represents a dialog box that allows a user to confirm a sign out.
  - ☐ **ApplicationList** – Displays the applications that a user is currently signed into as a bulleted list.
  - ☐ **SwitchUserLink / SwitchUserLinkListItem** – Displays a dynamic link to allow the currently signed in user to sign out and let a different user sign in. The widget with the ListItem suffix emits HTML that is surrounded by the <li></li> tags (for placing the link within a <ul></ul> list).
  - ☐ **SignInValidationSummary** – Placeholder for a bulleted list that displays server side errors.
  - ☐ **Email** – Input box for an email address (**MUST** have **readonly** parameter set true – see the basic template sample view for an example of setting this parameter).
  - ☐ **Password** – Input box for a password.
  - ☐ **Captcha** – Placeholder for a ReCaptcha based Captcha that is displayed only if necessary (SSO server decides)
  - ☐ **RememberMe** – The checkbox that the user can tick to remember allow the computer to remember their email address
  - ☐ **AutoSignOn** – The checkout that the user can tick to specify that they should be automatically logged on in the future (if they have a valid SSO session). If this widget is specified, the web application must provide a way of allowing the user to unset this flag using the SSO API.
  - ☐ **Cancel** – The cancel button, to cancel a sign in attempt.

- ☐ **SignInAdditional** – The actual sign in button (for the SignOnAdditional view)
- ☐ **SignInValidationScript** – Client side validation script placeholder.
- ☐ **CultureSwitcher** – A widget that provides a dynamic link to the culture selection screen (if script not enabled) and a link to pop up a CSS based culture selection dialog otherwise.
- ☐ **Html.SSOAntiForgeryToken()** – This is required to mitigate CSRF attacks on Sage ID. Please note, you should use the Sage ID variant of this tag and MUST NOT use Html.AntiforgeryToken().

- SignOnExpress.aspx

  - ☐ **ConfirmSignOutDialog** – A widget that writes HTML to the page that represents a dialog box that allows a user to confirm a sign out.
  - ☐ **ApplicationList** – Displays the applications that a user is currently signed into as a bulleted list.
  - ☐ **SwitchUserLink / SwitchUserLinkListItem** – Displays a dynamic link to allow the currently signed in user to sign out and let a different user sign in. The widget with the ListItem suffix emits HTML that is surrounded by the <li></li> tags (for placing the link within a <ul></ul> list).
  - ☐ **SignInValidationSummary** – Placeholder for a bulleted list that displays server side errors.
  - ☐ **AutoSignOn** – The checkout that the user can tick to specify that they should be automatically logged on in the future (if they have a valid SSO session). If this widget is specified, the web application must provide a way of allowing the user to unset this flag using the SSO API.
  - ☐ **Cancel** – The cancel button, to cancel a sign in attempt – with the class parameter set to @class = "cancel notopmargin"
  - ☐ **SignInAdditional** – The actual sign in button (for the SignOnAdditional view) – *with the class parameter set to @class = "submit notopmargin"*
  - ☐ **SignInValidationScript** – Client side validation script placeholder.
  - ☐ **CultureSwitcher** – A widget that provides a dynamic link to the culture selection screen (if script not enabled) and a link to pop up a CSS based culture selection dialog otherwise.

- ☐ **Html.SSOAntiForgeryToken()** – This is required to mitigate CSRF attacks on Sage ID. Please note, you should use the Sage ID variant of this tag and MUST NOT use Html.AntiforgeryToken().

- Register.aspx
  - ☐ **RegisterValidationSummary** – Placeholder for a bulleted list that displays server side errors for registration form submissions.
  - ☐ **Email** – Input box for an email address (optional if provided when StartNewUserRegistrationAttempt is called by your web application)
  - ☐ **DisplayName** – Input box for the users name (optional if provided when StartNewUserRegistrationAttempt is called by your web application)
  - ☐ **Password** – Input box for a password.
  - ☐ **ConfirmPassword** – Input box for a password, which must match the first password.
  - ☐ **SecurityQuestion1** – Dropdown list from which the user can select their first security question.
  - ☐ **SecurityAnswer1** - Input box for the user to enter their answer to the first security question.
  - ☐ **SecurityQuestion2** – Dropdown list from which the user can select their second security question.
  - ☐ **SecurityAnswer2** - Input box for the user to enter their answer to the second security question.
  - ☐ **SecurityQuestion3** – Dropdown list from which the user can select their third security question.
  - ☐ **SecurityAnswer3** - Input box for the user to enter their answer to the third security question.
  - ☐ **Captcha** – Placeholder for a ReCaptcha based Captcha that is displayed only if necessary (SSO server decides). ***It will always be displayed on the registration page.***
  - ☐ **Cancel** – The cancel button, to cancel a registration in attempt.
  - ☐ **Register** – The submit registration button, to submit to the form and register the user.

- ☐ **RegisterValidationScript** – Client side validation script placeholder.
- ☐ **CultureSwitcher** – A widget that provides a dynamic link to the culture selection screen (if script not enabled) and a link to pop up a CSS based culture selection dialog otherwise.
- ☐ **Html.SSOAntiForgeryToken()** – This is required to mitigate CSRF attacks on Sage ID. Please note, you should use the Sage ID variant of this tag and MUST NOT use Html.AntiforgeryToken().

- Change.aspx

  - ☐ **ChangeValidationSummary** – Placeholder for a bulleted list that displays server side errors for change password form submissions.
  - ☐ **Password** – Input box for a password (the old password in the context of Change.aspx)
  - ☐ **NewPassword** – Input box for a new password.
  - ☐ **ConfirmPassword** – Input box for a password, which must match the password entered in NewPassword.
  - ☐ **SecurityQuestion1** – Dropdown list from which the user can select their first security question.
  - ☐ **SecurityAnswer1** - Input box for the user to enter their answer to the first security question.
  - ☐ **SecurityQuestion2** – Dropdown list from which the user can select their second security question.
  - ☐ **SecurityAnswer2** - Input box for the user to enter their answer to the second security question.
  - ☐ **SecurityQuestion3** – Dropdown list from which the user can select their third security question.
  - ☐ **SecurityAnswer3** - Input box for the user to enter their answer to the third security question.
  - ☐ **Cancel** – The cancel button, to cancel a password change attempt.
  - ☐ **Continue** – The submit password change button, to submit to the form and change the users password.
  - ☐ **ChangeValidationScript** – Client side validation script placeholder.

- ☐ **CultureSwitcher** – A widget that provides a dynamic link to the culture selection screen (if script not enabled) and a link to pop up a CSS based culture selection dialog otherwise.
- ☐ **Html.SSOAntiForgeryToken()** – This is required to mitigate CSRF attacks on Sage ID. Please note, you should use the Sage ID variant of this tag and MUST NOT use Html.AntiforgeryToken().

IMPORTANT
Please note, the ChangePasswordOnly.aspx view has a slightly different opening form tag. It should begin with `<% using (Html.BeginPasswordChangeForm()) { %>` *rather than* `<% using (Html.BeginForm()) { %>`

- ChangePasswordOnly.aspx (new in v.1.1)

- ☐ **ChangeValidationSummary** – Placeholder for a bulleted list that displays server side errors for change password form submissions.
- ☐ **Password** – Input box for a password (the old password in the context of Change.aspx)
- ☐ **NewPassword** – Input box for a new password.
- ☐ **ConfirmPassword** – Input box for a password, which must match the password entered in NewPassword.
- ☐ **Cancel** – The cancel button, to cancel a password change attempt.
- ☐ **Continue** – The submit password change button, to submit to the form and change the users password.
- ☐ **ChangeValidationScript** – Client side validation script placeholder.
- ☐ **CultureSwitcher** – A widget that provides a dynamic link to the culture selection screen (if script not enabled) and a link to pop up a CSS based culture selection dialog otherwise.
- ☐ **Html.SSOAntiForgeryToken()** – This is required to mitigate CSRF attacks on Sage ID. Please note, you should use the Sage ID variant of this tag and MUST NOT use Html.AntiforgeryToken().

IMPORTANT
Please note, RecoveryStage1/2/3.aspx views have a slightly different opening form tag. They should begin with `<% using (Html.BeginRecoveryForm()) { %>` *rather than* `<% using (Html.BeginForm()) { %>`

- RecoveryStage1.aspx

  - ☐ **RecoveryStage1ValidationSummary** – Placeholder for a bulleted list that displays server side errors for form submissions.
  - ☐ **Email** – Input box for an email address.
  - ☐ **Captcha** – Placeholder for a ReCaptcha based Captcha that is displayed only if necessary (SSO server decides). ***It will always be displayed on the password recovery page.***
  - ☐ **Cancel** – The cancel button, to cancel a password change attempt.
  - ☐ **Continue** – The submit password change button, to submit to the form and go to the next stage of password recovery.
  - ☐ **RecoveryStage1ValidationScript** – Client side validation script placeholder.
  - ☐ **CultureSwitcher** – A widget that provides a dynamic link to the culture selection screen (if script not enabled) and a link to pop up a CSS based culture selection dialog otherwise.
  - ☐ **Html.SSOAntiForgeryToken()** – This is required to mitigate CSRF attacks on Sage ID. Please note, you should use the Sage ID variant of this tag and MUST NOT use Html.AntiforgeryToken().

- RecoveryStage2.aspx

  - ☐ **RecoveryStage2ValidationSummary** – Placeholder for a bulleted list that displays server side errors for form submissions.
  - ☐ **RecoveryAnswer1** – Input box for users' answer to first security question.
  - ☐ **RecoveryAnswer2** – Input box for users' answer to second security question.
  - ☐ **Captcha** – Placeholder for a ReCaptcha based Captcha that is displayed only if necessary (SSO server decides). ***It will always be displayed on the password recovery page.***
  - ☐ **Cancel** – The cancel button, to cancel a password recovery attempt.

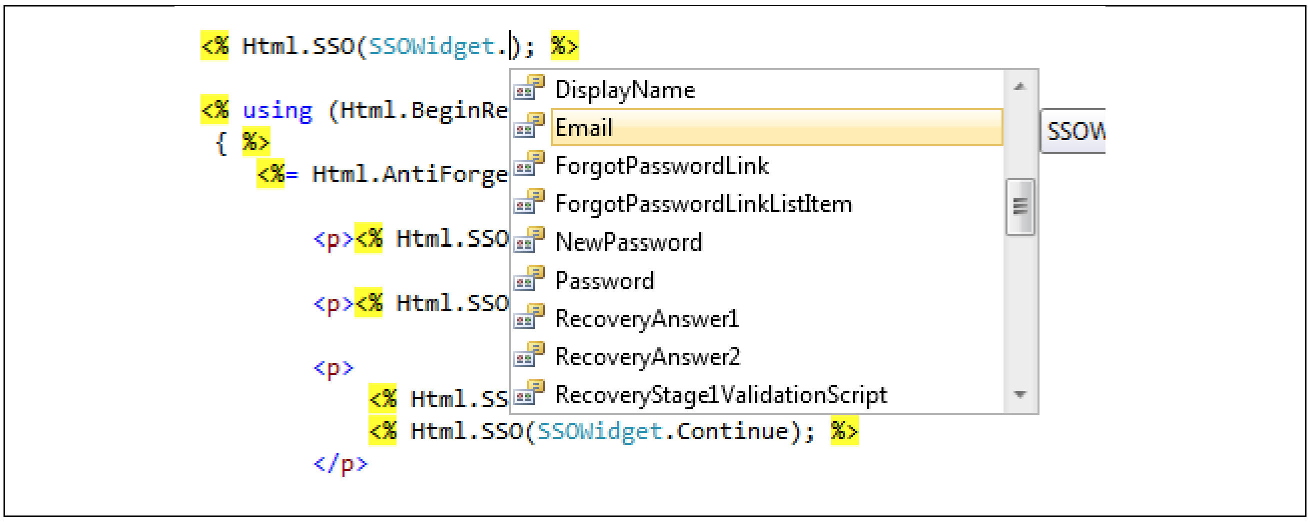
- ☐ **Continue** – The submit password change button, to submit to the form and go to the next stage of password recovery.
- ☐ **RecoveryStage2ValidationScript** – Client side validation script placeholder.
- ☐ **CultureSwitcher** – As above.
- ☐ **Html.SSOAntiForgeryToken()** – This is required to mitigate CSRF attacks on Sage ID. Please note, you should use the Sage ID variant of this tag and MUST NOT use Html.AntiforgeryToken().

Although not SSO widgets, you must ensure HTML tags are in place to present the two security questions to the user, by binding them from the model. E.g
`<h4><%= Html.Encode(Model.RecoveryQuestion1) %></h4>` *and*
`<h4><%= Html.Encode(Model.RecoveryQuestion2) %></h4>`

- RecoveryStage3.aspx

- ☐ **RecoveryStage3ValidationSummary** – Placeholder for a bulleted list that displays server side errors for form submissions.
- ☐ **Password** – Input box for the new password
- ☐ **ConfirmPassword** – Input box for a password, which must match the password entered in the Password box.
- ☐ **Cancel** – The cancel button, to cancel a password change attempt.
- ☐ **Continue** – The submit password change button, to submit to the form and go to the next stage of password recovery.
- ☐ **RecoveryStage3ValidationScript** – Client side validation script placeholder.
- ☐ **CultureSwitcher** – As above.
- ☐ **Html.SSOAntiForgeryToken()** – This is required to mitigate CSRF attacks on Sage ID. Please note, you should use the Sage ID variant of this tag and MUST NOT use Html.AntiforgeryToken().

Microsoft Visual Web Developer 2010 includes IntelliSense for the SSO Widget names, and also Global Resources as follows:



```
<% Html.SSO(SSOWidget.|); %>

<% using (Html.BeginRe
{ %>
    <%= Html.AntiForge

        <p><% Html.SSO

        <p><% Html.SSO

        <p>
            <% Html.SS
            <% Html.SSO(SSOWidget.Continue); %>
        </p>
```

3.7. Global Resource Strings

Sage SSO provides a set of global resources that are already localised into English, German and French. The SSO widgets get their text from these global resources, but you are free to override these using local resource files and shown in the next section.

Resources can be included in your View using the following tag:

```
<%= Html.Resource(SSO.XXX, yy) %>
```

(where XXX is the resource key and yy is an optional array of formatter parameters)

The following table shows all the global resources that Sage SSO is aware of by default:

Name	**Value**
BlurbChangeForgottenPassword	This page allows you to specify your new password. Please enter your new password, confirm it in the second box, then click Continue.
BlurbChangePassword	This page allows you to easily change your password and security questions.<br/><br/>Please enter your password, optional new password and optional amended security questions. when you are happy with your answers, click Continue.
BlurbChangePassword2	New passwords are required to be a minimum of {0} characters in length.
BlurbChangePasswordSuccess	Your password has been changed successfully.
BlurbConfirmSignout	By signing in as a different user, you will be signed out of all the web applications you are currently signed into.<br /><br />You are currently signed into:
BlurbConfirmSignout2	Are you sure you want sign out of all web applications and sign in as a different user?
BlurbCustomerServices	<p>Please contact customer services to have your

	password reset.</p> <p>Our telephone lines are open Monday to Friday, 9am to 5pm and if you send us an email, we aim to get back to you by the end of the next working day.<br /><br /></p>
BlurbLanguagePopup	Sage Single Sign On is available in multiple languages, with the initial language selection based on information from your browser. If this selection is inappropriate, please select a different language from the options shown below.
BlurbRecovery	Please enter the email address associated with your Sage Sign In account, along with the words shown in the image. Once you have done this, please click Continue.
BlurbRegisterAccount	<p>Creating a Sage Sign In Account helps us provide you with information that is suited to your particular needs and interests, which means it's easier for you to find what you need.</p> <p>We also use the Sage Sign In Account to help us keep your personal account information safe and secure.</p> <p>Whether you're buying or registering any of our software or services or using one of our other web services, a Sage Sign In Account helps to ensure that your account information is secure and available to you in one central location.</p>
BlurbSecurityQuestion	To confirm your identity you must answer the security questions, which you selected and provided an answer to when you created your account.
BlurbSignIn	Sage Sign In allows you to access all your Sage Services with a single username. To sign in, please enter your username and password.
BlurbSignInIPhone	To sign in, please enter your username and password.
BlurbSignOnAdditional	Sage Single Sign On allows you to access all your Sage Services with a single username. As you are already signed into one or more web applications you can sign into another without having to re-enter your username and password. To sign in, click the button below.
ButtonCancel	Cancel
ButtonChangePassword	Change Password
ButtonContinue	Continue
ButtonNo	No
ButtonRegister	Register
ButtonResetPassword	Reset Password
ButtonSignIn	Sign In
ButtonSignIntoAdditional	Sign In

ButtonSubmitted	Please wait
ButtonYes	Yes
CustomerServicesEmail	Email: <a href="mailto:support@sage.com?subject=Password Reset">support@sage.com</a>
CustomerServicesFax	Fax: 0845 111 55 11
CustomerServicesTelephone	Telephone: 0845 111 55 55
DemoRandomPassword	For this demo only, the password has been set to <b>p@ssword1</b>, but you would normally receive a new <i>random</i> password in an email.
FileTitleImage	sageLogo80.png
FileTitleImageIPhone	sageLogo80iphone.png
FooterCompanyDisclaimer	The Sage Group plc, a company registered in the United Kingdom, with registered number 02231246 and with its registered office at North Park, Newcastle Upon Tyne NE13 9AA
FooterCopyright	&copy; The Sage Group plc 2010. All Rights Reserved.
FormAutoSignIn	Automatically sign in to this application
FormCaptcha	Please enter the two words displayed above:
FormCaptchaNoScript	Please type the two words shown into the box below:
FormConfirmNewPassword	Confirm new password:
FormConfirmPassword	Confirm password:
FormCurrentPassword	Current password:
FormEmail	Email:
FormNewPassword	New password:
FormPassword	Password:
FormRecoveryAnswer1	First question answer:
FormRecoveryAnswer2	Second question answer:
FormRememberMe	Remember me on this computer
FormRememberMeIPhone	Remember me on this phone
FormSecurityDropdownDefault	Please select a security question
FormSecurityQuestion1	First security question:
FormSecurityQuestion2	Second security question:
FormSecurityQuestion3	Third security question:
FormSecurityQuestionAnswer1	First question answer:
FormSecurityQuestionAnswer2	Second question answer:
FormSecurityQuestionAnswer3	Third question answer:
FormSecurityQuestionAnswerShort	Answer:
FormUseEmailForUsername	Use email address for your username
FormUsername	Name:
FormYourSecurityQuestion	Your security question:
LinkChangePassword	Change Password
LinkClickToRegister	click to register
LinkForgotPassword	Forgot your password?
LinkHome	Home

LinkPrivacyPolicy	Privacy Policy
LinkRegisterAccount	Register an account
LinkRegisterAccountShort	Register
LinkReturnToWebApplication	Please click here to return to the web application.
LinkSelectLanguage	Select Language
LinkSignInDifferentUser	Sign in as a different user
LinkSignInShort	Sign In
LinkSignOut	Sign Out
LinkTermsConditions	Terms &amp; Conditions
MessageAccountCreated	Your account has been created successfully.
MessageAlreadySignedInto	{0} is already signed into the following web applications:
MessagePasswordResetSuccess	Your password has been successfully reset.
MessageSignedOutOf	You have successfully signed out of
MessageSigningInto	You are signing into
MessageStillSignedInto	{0} is still signed into the following web applications:
RegisterAccountPassword	Passwords are required to be a minimum of 8 characters in length.
TitleAccountCredentials	Account Credentials
TitleAccountInformation	Account Information
TitleChangePassword	Change Account Credentials
TitleChangePasswordSuccess	Change Password Success
TitleConfirmSignout	Are you sure you want to sign out?
TitleLanguageFlagClick	Click on a flag to select a language
TitleLanguageSelection	Language Selection
TitleRecovery	Forgot your password?
TitleRegister	Register a New Account
TitleSecurityQuestion	Security Questions
TitleSignIn	Sign In
TitleSignInConfirmation	Sign In Confirmation
ToolTipAutoSignIn	<strong>Automatically sign in to this application</strong> allows you to sign in without having to enter your credentials in the future.<br /><br />It is not recommended to use this feature on a shared computer.
ToolTipCaptcha	This step has been included to ensure your account is safe from machine based attacks. Please enter the two words which are displayed so the web application knows you are human.
ToolTipRecoveryUsername	Your <strong>user name</strong> is what uniquely identifies you to Sage applications. e.g. john.smith, my.company etc.<br /><br />Enter it here if you have forgotten your password.
ToolTipRegisterCaptcha	This step has been included to ensure only humans and not machines can register. Please enter the two words which are displayed.
ToolTipRegisterEmail	Your email address uniquely identifies you to Sage applications.<br/><br/>A valid <strong>email

	address</strong> should be provided so that you can activate your account and receive account emails from the system.
ToolTipRegisterPassword	Please enter a <strong>password</strong> to protect your account. It must be at least {0} characters long and should contain a mixture of upper, lower and special characters to help ensure maximum security.
ToolTipRegisterUsername	The name provides a friendlier way for the system to address you.<br /><br />For instance, applications will use the name on welcome screens e.g. Hello John Smith.
ToolTipRememberMe	<strong>Remember me on this computer</strong> saves your username on your computer so that you don't have to retype it the next time you sign in.<br /><br />It is not recommended to use this feature on a shared computer.
ToolTipSecurityAnswer	When your account was created you selected, and provided an answer to, a security question. This question is currently displayed above the answer box.<br /><br />To reset your password you need to provide the answer to this security question.
ToolTipUseEmailForUsername	<strong>Use email address for your username</strong> sets your username to be the same as your email address.<br /><br />This may make your username easier to remember.
UnknownError	An unknown error occurred. Please verify your entry and try again. If the problem persists, please contact support.
ValidationCaptchaIncorrect	The words that were entered were not correct.
ValidationCaptchaMissing	You must enter the two words that are displayed in the image.
ValidationChangeNothingToDo	The password or security questions have not been changed.
ValidationChangePasswordFail	The current password is incorrect or the new password is invalid.
ValidationConfirmPasswordInvalid	The new password and confirmation password do not match.
ValidationConfirmPasswordMissing	You must confirm your password.
ValidationDuplicateEmail	A username for that email address already exists. Please enter a different email address.
ValidationEmailInvalid	You must specify a valid email address.
ValidationEmailMissing	You must specify an email address.
ValidationIncorrectCredentials	The email or password provided is incorrect.
ValidationMaxLength	The field is too long. Use less characters.
ValidationNewPasswordGuessable	The new password must not be easily guessable.
ValidationNewPasswordMinLength	The new password must be at least 8 characters long.
ValidationNewPasswordMissing	You must specify a new password.
ValidationPasswordComplexity	The password you have entered is not complex

	enough. It is recommended to use upper and lower case letters, numbers and special characters to ensure a strong password.
ValidationPasswordGuessable	The password must not be easily guessable.
ValidationPasswordMinLength	The password must be at least 8 characters long.
ValidationPasswordMissing	You must specify a password.
ValidationSecurityAnswerIncorrect	The answer to the security question is incorrect.
ValidationSecurityAnswerMinLength	The answer to the security question must be at least 5 characters long.
ValidationSecurityAnswerMissing	You must provide an answer to the security question.
ValidationSecurityAnswersDifferent	Please select three different security questions.
ValidationSecurityQuestionMinLength	You must provide your own security question. It must be longer than 5 characters.
ValidationSecurityQuestionMissing	You must select a security question from the list.
ValidationSummaryNewPassword	Please correct the errors and try again.
ValidationSummaryPasswordChange	Password change was unsuccessful. Please correct the errors and try again.
ValidationSummaryRecovery	Please correct the errors and try again.
ValidationSummaryRegisterAccount	Account creation was unsuccessful. Please correct the errors and try again.
ValidationSummarySecurityQuestion	Please correct the errors and try again.
ValidationSummarySignIn	Your sign in was unsuccessful. Please correct the errors.
ValidationUsernameExists	The email address is already in use. Please enter a different email address.
ValidationUsernameMinLength	The name must be at least 3 characters long.
ValidationUsernameMissing	You must specify a name.
ValidationUsernameNotFound	The username entered is not registered with the system.

3.8. Overriding Global Resource Strings via Local Resource Files

The customisation kit allows the override of any of the above resources. It also allows the addition of your own resources in a manner that is culture aware. This means that if the SSO user changed the culture during an SSO flow, the SSO web application would select the appropriate version of you resource automatically. The implication of this is you will have to provide English, German and French versions of your resource strings.

Once you have declared your resources in a local resource file you can access them using the *Html.Resource* tag as shown previously. For new resources, with new keys, you have to explicitly include the key in the tag as a string. e.g:

```
<%= Html.Resource("YourNewResourceKey") %>
```

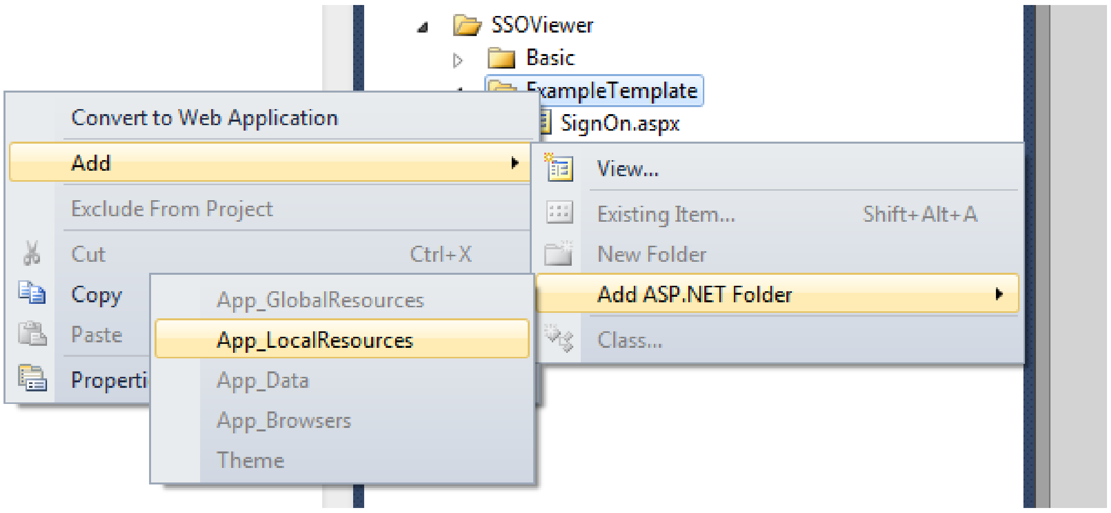
If you use any place holders in the string, e.g. {0}, you can provide the parameter (most probably from the model). The following shows how to do this with the DisplayName of the signed in user:

```
<%= Html.Resource("YourNewResourceKey" , new object[] {
Html.Encode(Model.DisplayName) }) %>
```

- To override a global resource, follow the following steps:

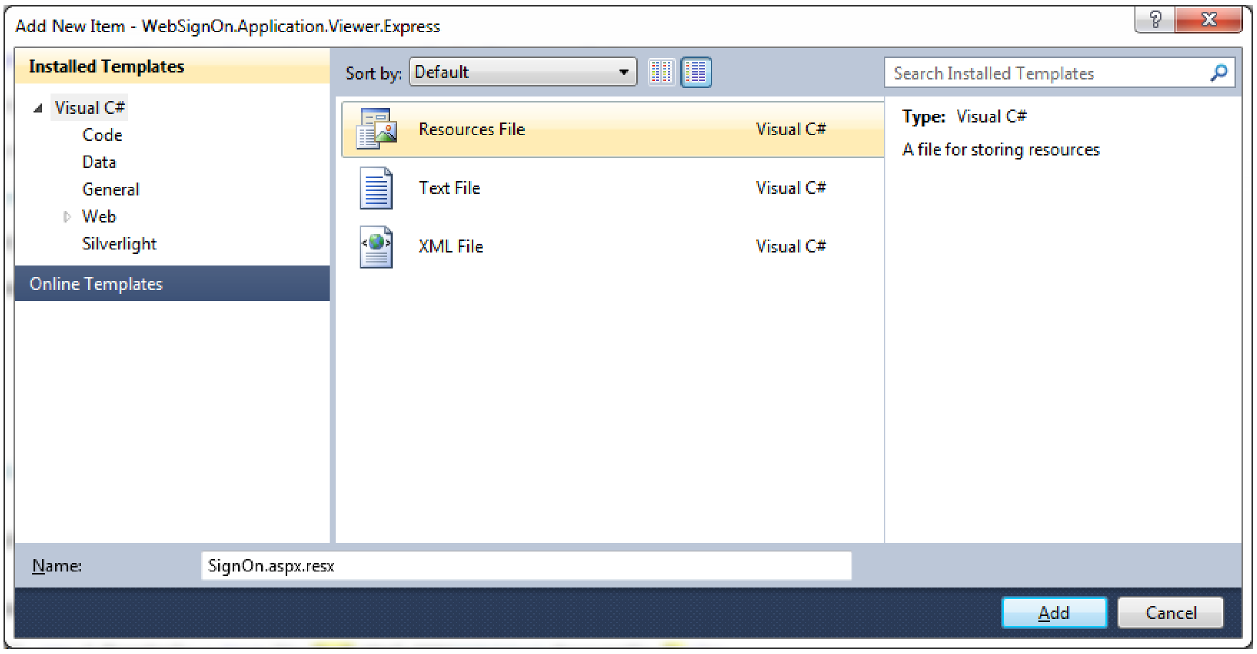
1. Local resources should exist in your *Views\SSOViewer\{TemplateName}\App_LocalResources* folder. We have not created this yet but it is very easy to do. Return to the customisation kit project and locate the *Views\SSOViewer\ExampleTemplate* folder, which should now contain your customised SignOn.aspx view from following the previous sections. **Ensure the project is stopped.**

   Right click the folder and select Add → Add ASP.NET Folder → App_LocalResources



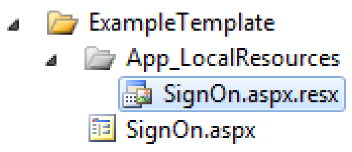
2. The next step is to create a local resources file for the SignOn.aspx view. Right click on the newly created App_LocalResources folder and select Add → New Item. A dialog appears. Select "General" on the left, and then select "Resources File" on the right.

Before clicking the "Add" button, set the Name (at the bottom of the dialog) to *SignOn.aspx.resx*



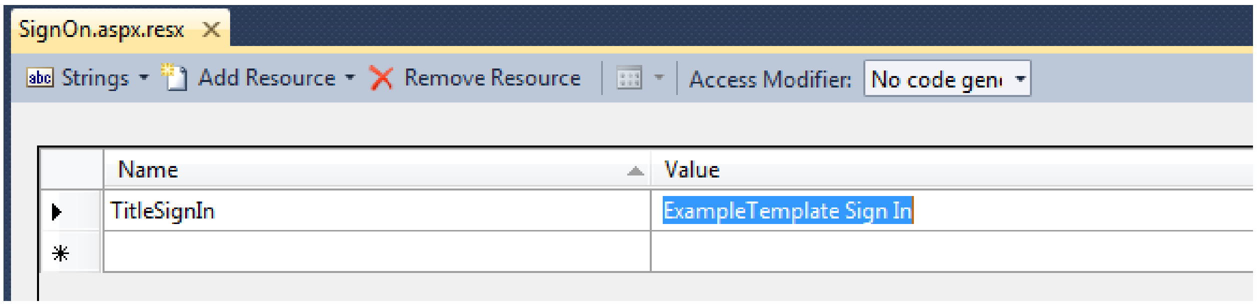
Click "Add".

3. The resource editing table should appear. If not, double click the newly added *SignOn.aspx.resx* file. This screen allows you to specify resource keys (in the Name column) and the corresponding string values (in the Value column).



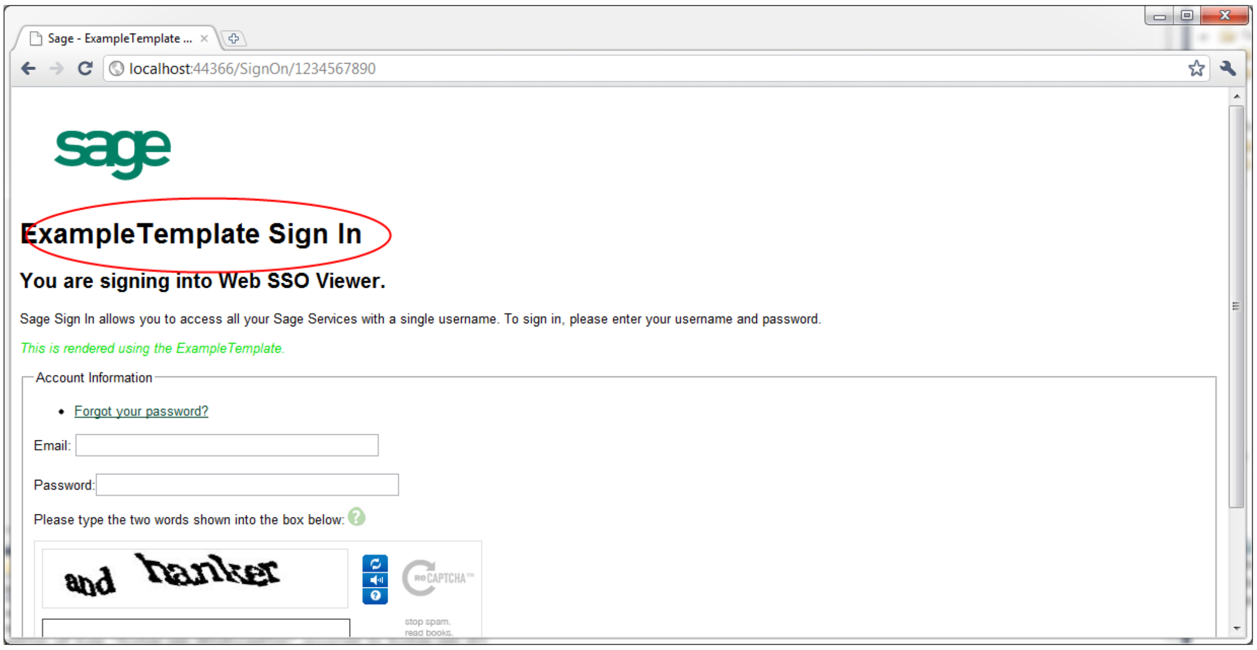
As an example, let's override the title on the sign in page. To do this, we add the resource key to the .resx file and enter a new value.

Add an entry for "*TitleSignIn*" and give it a value of *"ExampleTemplate Sign In"* as shown in the following screenshot:



Name	Value
TitleSignIn	ExampleTemplate Sign In

4. Save the SignOn.aspx.resx file.
5. Press F5 (or the play button) to start up your web browser and check the changes.
6. Click "First sign on"
7. If everything is working correctly, the following should be displayed. Notice the new resource string, highlighted by the red ellipse.


Sage - ExampleTemplate ...
localhost:44366/SignOn/1234567890

ExampleTemplate Sign In

You are signing into Web SSO Viewer.

Sage Sign In allows you to access all your Sage Services with a single username. To sign in, please enter your username and password.

This is rendered using the ExampleTemplate.

Account Information

- Forgot your password?

Email:

Password:

Please type the two words shown into the box below:

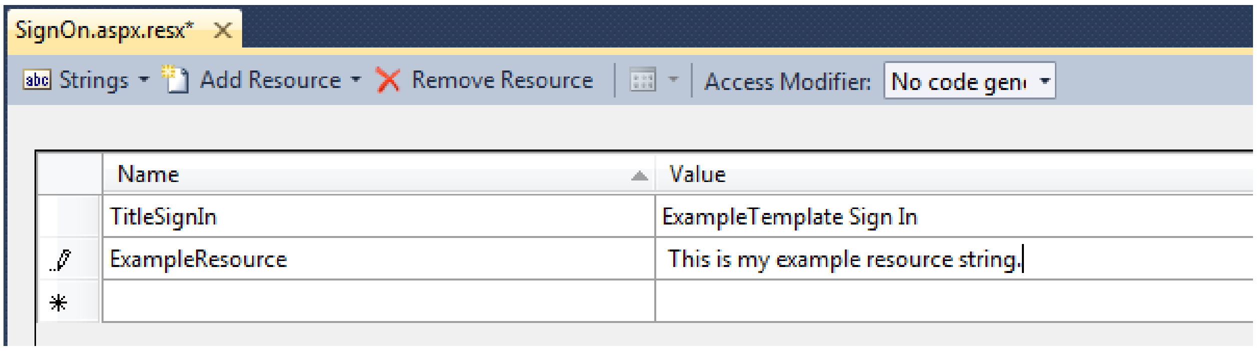
Remember for the culture selector to work, and for your template to be compatible across languages, you should provide .resx files for as many cultures as required.

*The convention is **ViewName.aspx.LanguageCode.resx***

See the files located in Views\SSOViewer\Basic\App_LocalResources for examples.

- To add a new local resource:

1. Ensure the project is stopped.
2. Return to the SignOn.aspx.resx file and add a new unique key for your resource, e.g. "ExampleResource". Give this entry a value of "This is my example resource string." as follows:



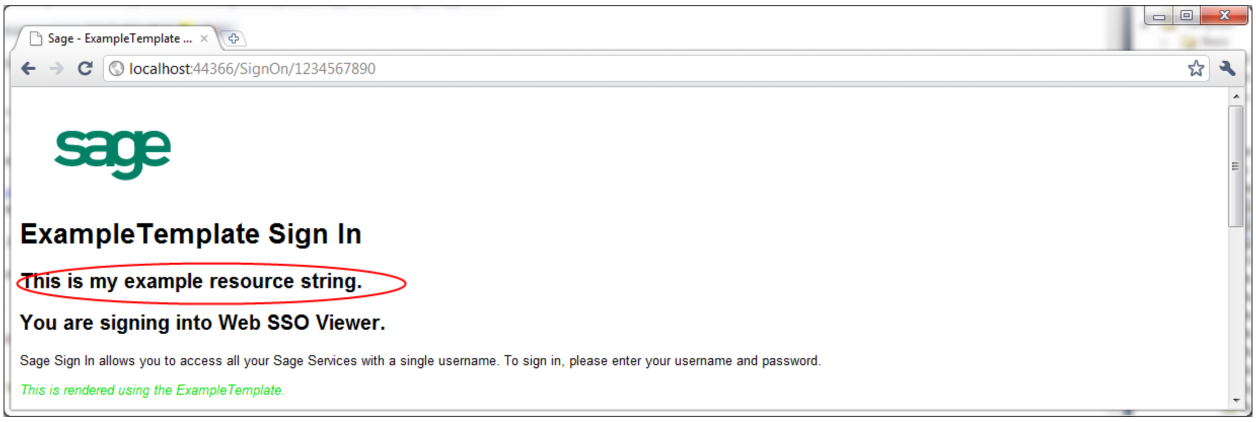
Name	Value
TitleSignIn	ExampleTemplate Sign In
ExampleResource	This is my example resource string.

3. Save the file, then view the mark-up for *SignOn.aspx*.
4. Add a HTML helper tag to read the resource and display it after the title:

```
<%= Html.Resource("ExampleResource") %>
```

```
        <img src="/Images/sageclearzone.gif" title="Sage Group plc Logo" alt="Sage Logo" />
        <h1>
        <%= Html.Resource(SSO.TitleSignIn) %></h1>
    <h2>
        <%= Html.Resource("ExampleResource") %>
    </h2>
```

5. Press F5 to load the project in the browser and example the "First sign on" view:



ExampleTemplate Sign In

This is my example resource string.

You are signing into Web SSO Viewer.

Sage Sign In allows you to access all your Sage Services with a single username. To sign in, please enter your username and password.

This is rendered using the ExampleTemplate.

3.9. Mocking Situations Using the SSOViewerController

The *SSOViewerController* class, located in the *Controllers* folder, is intended to be edited by the template designer so that they can mock situations and test their template layouts in the situations that the SSO server can place them in.

SSO widgets respond directly to the data within the model. The controller sets the model up based on the information from the SSO systems. By editing the code in the *SSOViewerController* class, you can change the properties in the model.

There are various sections (methods) in the *SSOviewerController.cs* file, each of which corresponds to an action in the system. As an example, when you click the "*First sign on*" link, when you run the project, the web browser causes the customisation kit to execute the code in the "*SignOn*" action, which is as follows:

```
        [Templated(TemplateName)]
        public ActionResult SignOn(string id)
        {
            SignOnModel model = new SignOnModel();
            model.WebApplicationDisplayName = "Web SSO Viewer";
            model.SignOnAttemptId = id;
            model.CancelAllowed = true;
            model.CaptchaRequired = true;
            model.CaptchaTheme = CaptchaThemes.clean.ToString();

            return View(model);
        }
```

This code creates a ***SignOnModel*** (which the *SignOn.aspx* is written to use). This model has various properties, and the above code statically sets these up to sensible defaults so that the

SignOn view can be displayed as a preview. In the real system, the SSO processes populate the model dynamically.

When designing your new template views, the most likely properties that you may wish to change are those related to the Captcha, such as ***CaptchaRequired*** and ***CaptchaTheme***. If you make any changes, save the *SSOViewerController.cs* file and press F5 to rebuild the project and launch your web browser.

3.10. Template Images and CSS

- **Resource Location**

Most likely your new template will require CSS and image resources. As the SSO templates are served from the SSO web servers, and via SSL, we must host these for you in a secure environment. As a result, there are conventions which must be followed whenever you reference an external resource in your template view HTML, or CSS files.

- All resources are to be stored in the *Template\XXXX* folder, where *XXXX* is your template name.
- All references to resources should be absolute and follow the format of */Template/XXXX/YourResource.ext*

The following is an example of referencing an image called *exampleimage.png*, that is hypothetically located in the *\Template\ExampleTemplate* folder.

```
<img src="/Template/ExampleTemplate/exampleimage.png" width="200" height="200" alt="Example" />
```

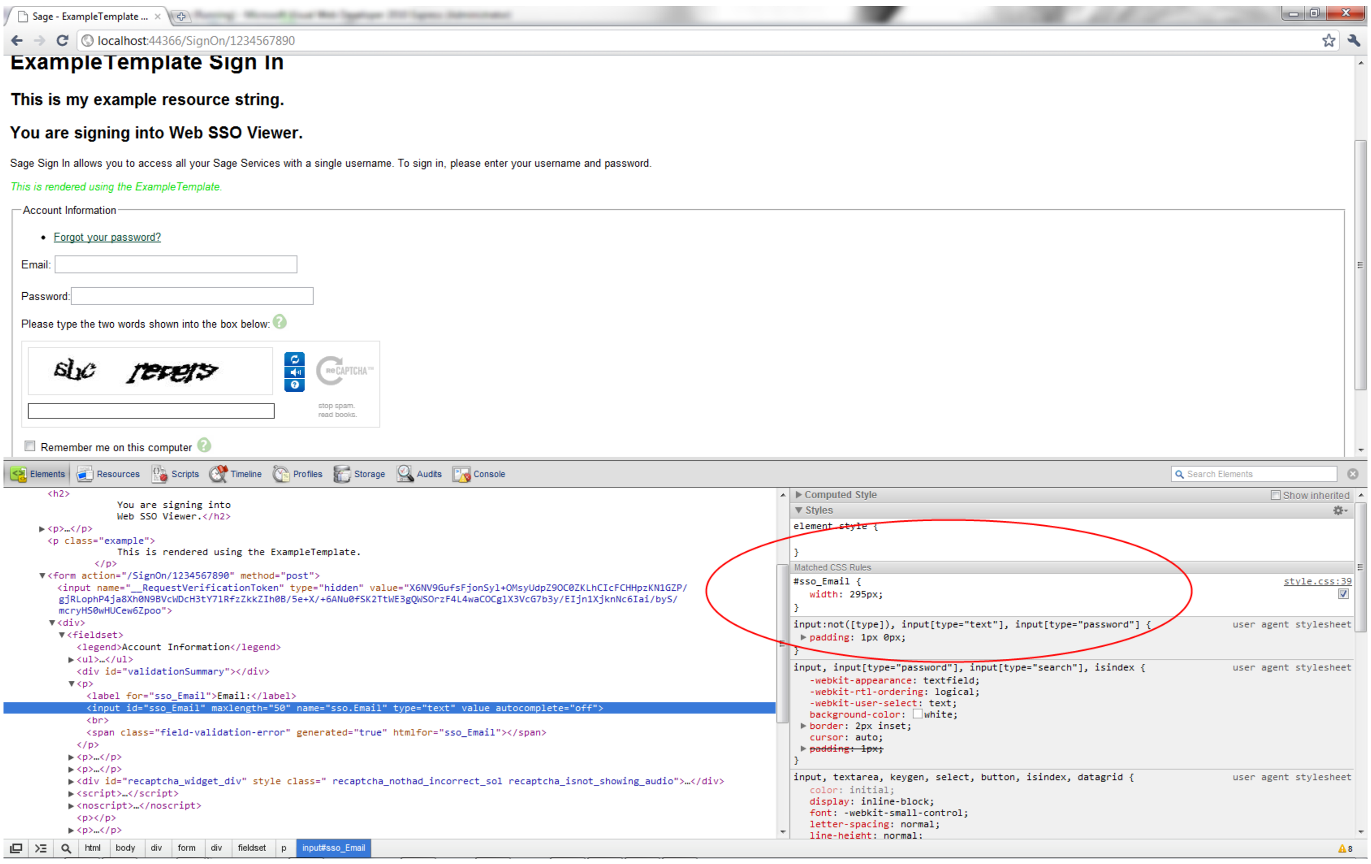
- **CSS Customisation**

The majority of the customisation of your template will probably involve working with cascading style sheets. The recommended approach is to get a skeleton templates working in the customisation kit, then examine the HTML source code, element IDs and CSS classes in the browser. There are many utilities, available for the common browsers, which support this examination of the DOM. Some browsers have this functionality built in. Here is a list of the methods recommended:

  - **Firefox** - Firebug Firefox Extension
  - **Google Chrome** – Right click → Inspect Element
  - **Internet Explorer** – Developer Tools (F12, or accessible from the IE tools menu)

Changes to the CSS styles can be included in your *\Template\ExampleTemplate\styles.css* file.

The following screenshot shows the CSS debugging capabilities of the Google Chrome. The matches CSS rule that applies to the Email Textbox is highlighted with the red ellipse:


Sage - ExampleTemplate ...

localhost:44366/SignOn/1234567890

ExampleTemplate Sign In

This is my example resource string.

You are signing into Web SSO Viewer.

Sage Sign In allows you to access all your Sage Services with a single username. To sign in, please enter your username and password.

This is rendered using the ExampleTemplate.

Account Information

- Forgot your password?

Email:

Password:

Please type the two words shown into the box below:

Remember me on this computer

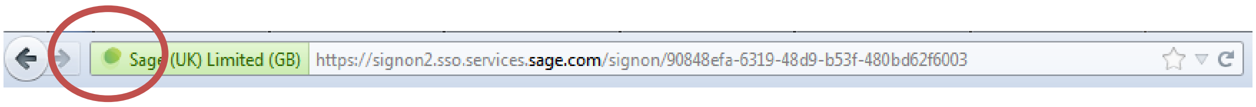
```
<h2>
            You are signing into
            Web SSO Viewer.</h2>
▶<p>…</p>
 <p class="example">
            This is rendered using the ExampleTemplate.
        </p>
▼<form action="/SignOn/1234567890" method="post">
  <input name="__RequestVerificationToken" type="hidden" value="X6NV9GufsFjonSyl+OMsyUdpZ9OC0ZKLhCIcFCHHpzKN1GZP/
  gjRLophP4ja8Xh0N9BVcWDcH3tY71RfzZkkZIh0B/5e+X/+6ANu0fSK2TtWE3gQWSOrzF4L4waCOCglX3VcG7b3y/EIjn1XjknNc6Iai/byS/
  mcryHS0wHUCew6Zpoo">
 ▼<div>
  ▼<fieldset>
     <legend>Account Information</legend>
    ▶<ul>…</ul>
     <div id="validationSummary"></div>
    ▼<p>
       <label for="sso_Email">Email:</label>
       <input id="sso_Email" maxlength="50" name="sso.Email" type="text" value autocomplete="off">
       <br>
       <span class="field-validation-error" generated="true" htmlfor="sso_Email"></span>
     </p>
    ▶<p>…</p>
    ▶<p>…</p>
    ▶<div id="recaptcha_widget_div" style class=" recaptcha_nothad_incorrect_sol recaptcha_isnot_showing_audio">…</div>
    ▶<script>…</script>
    ▶<noscript>…</noscript>
     <p></p>
    ▶<p>…</p>
```

```
element.style {

}
Matched CSS Rules
#sso_Email {                                                       style.css:39
   width: 295px;
}
input:not([type]), input[type="text"], input[type="password"] {   user agent stylesheet
 ▶ padding: 1px 0px;
}
input, input[type="password"], input[type="search"], isindex {    user agent stylesheet
   -webkit-appearance: textfield;
   -webkit-rtl-ordering: logical;
   -webkit-user-select: text;
   background-color: white;
 ▶ border: 2px inset;
   cursor: auto;
 ▶ ~~padding: 1px;~~
}
input, textarea, keygen, select, button, isindex, datagrid {      user agent stylesheet
   color: initial;
   display: inline-block;
   font: -webkit-small-control;
   letter-spacing: normal;
   line-height: normal;
```

3.11. Favourites Icon (favicon.ico) (new in v1.1)

Many web sites now display a custom icon to help identify the displayed page and also to brand any favourites / bookmarks of pages in the site. Sage ID allows you to specify a "favicon" for your Sage ID views. The image below shows the favicon used by SageOne:



Sage (UK) Limited (GB)
https://signon2.sso.services.sage.com/signon/90848efa-6319-48d9-b53f-480bd62f6003

We recommend using the .ico format as this is supported by all major browsers, including Internet Explorer. To include a favicon, you must place it in your /Template/<templateName>/ directory, along with the other resources that are referenced from your views.

You can then refer to the favicon by inserting the following code in the header section of your views:

```
<link rel="SHORTCUT ICON" href="/Template/<templateName>/favicon.ico"/>
```

3.12. Submit Your Template for Review

All custom templates are subjected to a review process before they will be accepted for use on the pre-prod and production SSO systems.

To submit your template, you should "zip up" both template folders and send them to ssdpdevelopersupport@sage.com, including the agreed name of the template in the email.

For example, if the agreed name of your template was "*ExampleTemplate*", you should zip up the following folders, including sub-folders and files:

- *\Template\ExampleTemplate*
- *\Views\SSOViewer\ExampleTemplate*

3.13. Advanced Widget Parameters

These can be used, if required, to allow tighter integration with your HTML / CSS layout.

Widget Name	**Parameters**
SSOWidget.AutoSignOn	ToolTip – true / false Paragraph – true / false HasLabel – true / false
SSOWidget.AutoSignOnLabel	Class – string (CSS class name)
SSOWidget.Cancel	Class – string (CSS class name)
SSOWidget.Captcha	ToolTip – true / false HasLabel – true / false LabelClass – string (CSS class name)
SSOWidget.ConfirmPassword	HasLabel – true / false Plain – true *(hides validation span)* / false
SSOWidget.Continue	Class – string (CSS class name)
SSOWidget.CultureSwitcher	Image – string (URI to image) Width – image width Height – image height
SSOWidget.DisplayName	ReadOnly – true / false HasLabel – true / false ToolTip – true / false Plain – true *(hides validation span)* / false
SSOWidget.Email	ReadOnly – true / false HasLabel – true / false ToolTip – true / false Plain – true *(hides validation span)* / false
SSOWidget.ForgotPasswordLink	Class – string (CSS class name)

SSOWidget.ForgotPasswordLinkListItem	Class – string (CSS class name)
SSOWidget.NewPassword	HasLabel – true / false Plain – true *(hides validation span)* / false
SSOWidget.Password	HasLabel – true / false ToolTip – true / false Plain – true *(hides validation span)* / false
SSOWidget.RecoveryAnswer1 SSOWidget.RecoveryAnswer2	HasLabel – true / false ToolTip – true / false Plain – true *(hides validation span)* / false
SSOWidget.Register	Class – string (CSS class name)
SSOWidget.RememberMe	HasLabel – true / false ToolTip – true / false
SSOWidget.SecurityAnswer1 SSOWidget.SecurityAnswer2 SSOWidget.SecurityAnswer3	HasLabel – true / false Plain – true *(hides validation span)* / false
SSOWidget.SecurityQuestion1 SSOWidget.SecurityQuestion2 SSOWidget.SecurityQuestion3	HasLabel – true / false Plain – true *(hides validation span)* / false
SSOWidget.SignIn	Class – string (CSS class name)
SSOWidget.SignInAdditional	Class – string (CSS class name)

The following code snippet demonstrates how to declare some of the optional parameters in the mark-up:

Email and password boxes with user label and no <p> tag wrapping:

```
<label for="sso_Email">Email<abbr title="required">*</abbr></label><%
Html.SSO(SSOWidget.Email, new { Plain = true, HasLabel = false }); %>

<label for="sso_Password">Password<abbr title="required">*</abbr></label><%
Html.SSO(SSOWidget.Password, new { Plain = true, HasLabel = false }); %>
```

Captcha with no tooltip icon and no integrated label:

```
<% Html.SSO(SSOWidget.Captcha, new { ToolTip = false, HasLabel = false }); %>
```

3.14. Alternative Language Selection

There may be times when you do not wish to use the provided culture selector and associated CSS modal dialog box for selecting the current language. It is possible to add links to you page that, when clicked, will change the current language to the culture code specified in the link URI.

The following shows how to create two links that allow the switching between English and French:

```
<a href="<%= Url.Action("SetCulture", new { culture = "en-gb", id = ViewContext.RouteData.Values["id"] })
%>">EN</a>
<br />
<a href="<%= Url.Action("SetCulture", new { culture = "fr-fr", id = ViewContext.RouteData.Values["id"] })
%>">FR</a>
```