Developers' Guide

Framework

Version 2.0.3.1

20th January 2012

6 Data Persistence & Performance

The Microsoft ASP.NET framework provides many methods for developers to persist data between page requests. Three of the most common methods are:

- **ViewState** – HTML hidden fields are used to store encrypted data securely within the web page for subsequent retrieval following a post back. An example of the markup generated by ViewState is shown below. For more details on ViewState read Understanding ASP.Net ViewState

  ```
  <input type="hidden" name="__VIEWSTATE" id="__VIEWSTATE" value="/wEPDwUJNzIwNzAy...
  ...U9szr+EL8hu+MLJH+l873A==" />
  ```

- **Session** – data is held within the server environment and indexed using a unique key. The developer selects the storage mechanism based on the characteristics of the web application. Storage options include using an SQL Server, a Session State Service, the worker process, or a custom implementation. A unique session key is stored as a cookie on the web browser or embedded into the URL. For more details on Session read ASP.Net Session State.

- **User Profile** – an authentication mechanism is required to identify an individual user uniquely and assign a user profile. Objects can then be added to the user profile for subsequent retrieval. An SQL database is typically used to store the data. For more details on UserProfile read Personalization and User Profiles in ASP.NET 2.0.

These methods are all available within mobile web applications developed using 51Degrees.mobi. The functionality and service is unaltered. However consideration should be given to their use when designing mobile web applications. The following specific issues should be considered:

1. The radio bottleneck can significantly impact the perceived performance of a mobile web application. For this reason it’s essential the web server performs as fast as possible and this may require changes to web server components.
2. ViewState can often generate large amounts of data increasing data costs for the end user and slowing page load time. Try using the browser to view the source HTML of ASP.NET generated web pages that contain a lot of controls and look at the __VIEWSTATE hidden field. 51Degrees.mobi can be configured to store ViewState information within a server side database significantly reducing the size and improving performance of pages that use ViewState. See the “**Error! Reference source not found.**” chapter and the persistViewState attribute to enable this feature. It is enabled by default when a new mobile web application is created.
3. Depending on the input methods available on the mobile device, authenticating an individual with a username and password can be cumbersome. Mobile web applications are often used whilst on the move and are “dipped” into for a few minutes at a time. Therefore entering a username and password will often limit the usability of a mobile web application. For this reason the use of UserProfile, or any mechanism requiring a username and password, should be considered first and foremost from a usability viewpoint.
4. Mobile web applications are often interacted with in short bursts over a long period of time using a number of different methods. For example; initial access to the application may be via a text message and only after a number of exchanges will the browser be used. Alternatively a search engine will be used to find the mobile web application that will then be bookmarked for subsequent quick retrieval when its functionality is required again. A compelling user experience can be generated if each interaction has some knowledge of previous interactions. Neither ViewState, Session nor UserProfiles provide a satisfactory solution.

6.1 Performance

Consider a traditional web site accessed via a desktop computer connected via a broadband internet connection. The time taken to transmit data between the browser and the web server is very short. Therefore the time the server takes to respond to a request will represent the most significant component of the response time. See Figure 1.

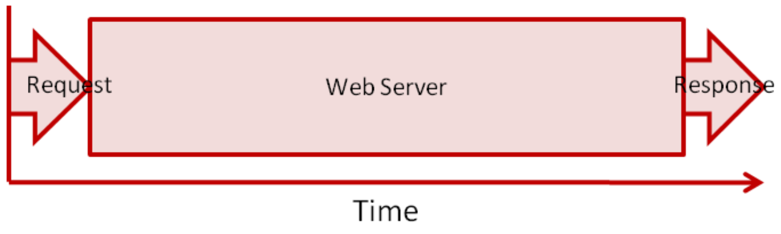

Figure 1 - Traditional web site accessed via broadband connection

If the same web server, unaltered, is now accessed via a mobile device the request and the response components of the process will be significantly longer compared to access via a broadband connection. See Figure 2.

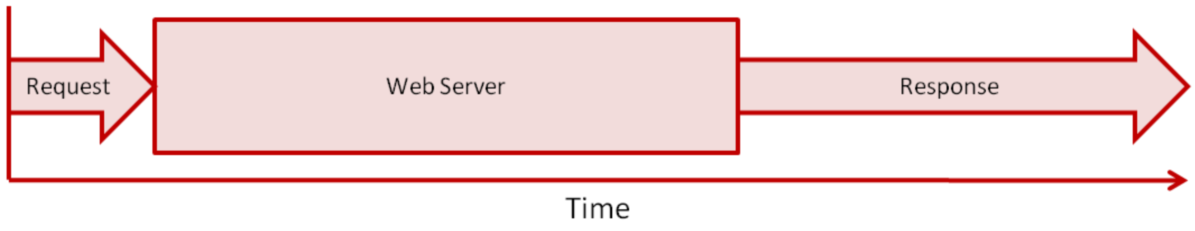

Figure 2 - Traditional web site accessed via a mobile connection

51Degrees.mobi will reduce the amount of data that needs to be sent between the web server and the mobile device reducing the amount of time taken to transmit request and response data. The precise improvements will depend on the original web site design and the number of 51Degrees.mobi features implemented. See Figure 3.

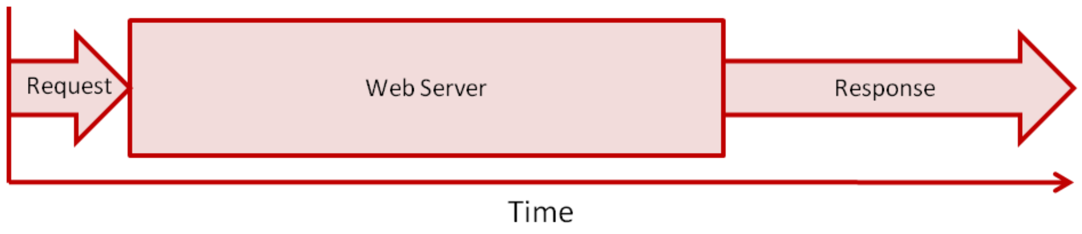

Figure 3 - Traditional web site optimised with 51Degrees.mobi accessed via a mobile connection

There is nothing more we can do to reduce the time associated with the request and response data transmission. The only other place to seek performance improvements will be the web server. This is likely to involve a change to the methods used to persist data, or access data consumption services. The precise changes will depend on the web server implementation. See Figure 4.

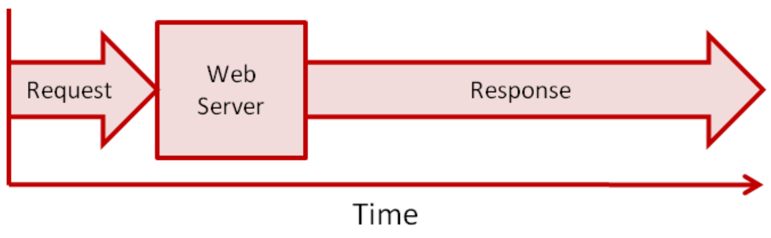

Figure 4 - Traditional web site fully optimised accessed via a mobile connection

The performance of the web server should be optimised to reduce the amount of time taken to respond to requests when implementing a mobile web application alongside a traditional web site where business rules and data persistence mechanisms will be shared. When designing mobile web applications that are entirely new emphasis should be placed on ensuring the web server performance is as fast as possible.

6.2 Mobile Profile

51Degrees.mobi includes a hybrid data persistence mechanism called Mobile Profile to enhance ASP.NET and provide persistence for courtesy data such as previous selections or user interface preferences. It is enabled by default when creating a mobile web application based on the 51Degrees.mobi templates and persists data to an SQL Server database much like ASP.NET SQL State Server. SQL Express 2005 or greater must be present on the server. A database will automatically be created if one is not already available for use within the Visual Studio environment. For large volume web sites a dedicated SQL Server running Standard or Enterprise edition should be considered for performance reasons. Depending on the configuration the amount data storage required could become quite large compared to a standard SQL State Server.

Each mobile device accessing the web site will have its own random 64bit unique mobile profile assigned to it. The mobile profile will either be created during the first request from the device, or by the developer when a text message or email URL is created for a mobile number. Data cannot be shared between different mobile profiles.

The length of time data persisted in the mobile profile should be stored is controlled in the web.config file. See **Error! Reference source not found.** and the retentionPeriod attribute. After the retentionPeriod has expired the data will automatically be removed from the mobile profiles' database. For this reason the value should be as high as possible. The larger the value the more data storage will be required.

If the browser does not support cookies the Mobile Profile will not be available. For this reason any application using Mobile Profile for data storage critical to the function of the application should check for the presence of cookies and advise the user if they're not enabled before proceeding. The MobileProfile is a fundamental component of the 51Degrees.mobi Framework. Understanding its operation and use is important when creating compelling mobile web applications.

Important Note: Data held in the MobileProfile should be given the same security consideration as cookies. If the data could be stored in a cookie then it's probably okay to store it in the mobile profile. Sensitive information such as financial details or personal information should not be stored in mobile profile. If a malicious party were able to obtain the Mobile Profile ID for a mobile device they would be able to retrieve the mobile profile data.

6.2.1 Data Types

MobileProfile facilitates persistence of 4 different types of data:

1. **User data**
   When the mobile profile is used from developers code to store application data controlled by the developer, this type of data is termed user data. It's similar to It is very similar to the Profiles feature of ASP.Net.The data stored in MobileProfile can be shared across multiple sessions of the same Mobile User. To know how to use the MobileProfile as custom data store see Using MobileProfile.
2. **Page Scope Control data**
   Controls that request data can be configured to store previous values within the mobile profile. For example; a drop down list will automatically default to the previously entered value when the page is displayed a subsequent time. All mobile profile aware controls enable this feature by default.
3. **Application Scope Control Data**
   Controls can be configured to share previous information via the mobile profile. For example; a search textbox may exist on many pages within the same application. It may be desirable for each instance of the search text box to have knowledge of the data entered in other text boxes. The control attribute dataKey would be set to common value for each text box instance.
4. **ViewState**
   If enabled the MobileProfile will be used to persist the view state.

6.2.2 Enabling Mobile Profile

New mobile web applications created using 51Degrees.mobi templates will not have Mobile Profile enabled by default as a new SQL Database is required. The following steps must be followed to enable Mobile Profile. The steps will be familiar to any used to configuring ASP for SQL sessions.

The following section must be added to the configuration file to tell Framework to use a Mobile Profile.

```
<fiftyOne>
      <profile retentionPeriod="10" applicationName="Demo"
      connectionStringName="profileConnectionString"
      persistViewState="true"/>
</fiftyOne>
```

If .NET v4 is being used the following modules do not need to be added to the configuration.

If versions of .NET prior to v4 are used one of the following module configurations will need to be added to the web.config.

```
<system.webServer>
      <modules>
            <add name="MobileProfile"
            type="FiftyOne.Framework.Mobile.Profile.ProfileModule,
            FiftyOne.Framework"/>
      </modules>
</system.webServer>
```

IIS7.X

```
<system.web>
      <httpModules>
```

```
            <add name="MobileProfile"
            type="FiftyOne.Framework.Mobile.Profile.ProfileModule,
            FiftyOne.Framework"/>
      </httpModules>
</system.web>
```

IIS6 or Visual Studio Web Server

Mobile Profile requires an SQL database is created. There are two ways to create the database and one of the following methods should be used.

6.2.2.1 User Instance

User instance database require SQL Express. The database is automatically created when the Framework process starts.

The following section must be added to the web.config file to enable a user instance database to be created automatically.

```
<connectionStrings>
      <add name="profileConnectionString" connectionString="Data
      Source=.\SQLEXPRESS;Integrated
      Security=True;Database=MobileProfile;AttachDBFilename=|DataDirectory|
      MobileProfile.mdf;User Instance=True;Asynchronous Processing=True;"/>
</connectionStrings>
```

If IIS 7.X is used as the web server the identity of the worker process must have read, write and modify access to the App_Data folder of the web site. If IIS6 is used the logged in user must have read, write and modify access to the App_Data folder.

When the Framework starts for the first time it will automatically create a MobileProfile database in the App_Data folder as specified in the connection string.

6.2.2.2 SQL Database

User Instance databases are not suitable for production deployments and may be undesirable for development and testing environments where more than one developer is working on the application. Use the **fiftyOne_regsql.exe** executable found in the installation folder to create a new mobile profile database. Documentation concerning this tool can be found in the **Error! Reference source not found.** of the Appendix.

Once the mobile profile database is created adde a connection string in the following form.

```
<connectionStrings>
      <add name="profileConnectionString" connectionString="Data
      Source=.\SQLEXPRESS;Integrated
      Security=True;Database=MobileProfile;Asynchronous Processing=True;"/>
</connectionStrings>
```

Ensure the identity used by the web server has dataread and datawrite access to the new database.

6.2.3 User Data

Similar to Session or ViewState data can be stored and read to and from MobileProfile using a single line of code. The following example placed in the page class shows how to store the string IT1098 in the Mobile Profile with the key ITEM_CODE.

C#

```
        MobileProfile["ITEM_CODE"] = "IT1098";
```

The current MobileProfile associated with the request can be retrieved using the MobileProfile property of the page as shown in the above example. The current MobileProfile associated with the request can be retrieved from any method during page processing using the `MobileProfiles.Current` static property.

6.2.4 Profile ID

MobileProfiles are normally randomly generated automatically by 51Degrees.mobi when a web browser first requests a page from the web site. In this situation developers do not need to be concerned with how the ProfileID is generated.

Another and extremely powerful feature of MobileProfile is the ability to generate MobileProfiles and manipulate their content outside independently of the page processing cycle. The following code example shows how to create, or retrieve an existing, mobile profile keyed on a mobile number, add some data, persist it and then use the MobileProfileID within a URL so that any subsequent requests will be able to retrieve the mobile number and associated data.

C#

```
        // Create, or get a previous, mobile profile using a mobile
        // number as the key field.
        ProfileBase mobileProfile;
        var mobileNumber = new MobileNumber("01234987654");
        if (MobileProfiles.Contains(mobileNumber))
            mobileProfile = MobileProfiles.GetProfile(mobileNumber);
        else
            mobileProfile = MobileProfiles.Create();

        // Add some data to the mobile profile and then save it.
        mobileProfile["DataKey"] = "HelloWorld";
        MobileProfiles.Save(mobileProfile);

        // Create an email body containing a link to the web site. Add the
        profile
        // id to the URL to enable the mobile number and data to be
        // retrieved when the user clicks on the link.
        string email = String.Format(
            "Click this link to open web site. http://example.com/{0}/",
            mobileProfile.ProfileID);
```

When the link above is selected the MobileProfile of the request will contain the mobile number and user data defined previously.

To simplify the creation of text messages containing MobileProfileID the Message class can be used to create a text message based on the message text, a URL and a mobile number. The following code example shows how a simple text message can be created containing a hyperlink which when selected will provide access to the mobile number.

C#

```
        string message = FiftyOne.Framework.SMS.Message.Create(
            "Click this link to open the web site. {0}",
            new Uri("http://example.com"),
            new MobileNumber("01234987654"));
```

This message could then be sent as the body of a text message to the mobile number. During the page load event of the mobile web application the following code can be used to retrieve the mobile number.

C#

```
        var mobileNumber = MobileProfile.MobileNumber;
```

The MobileProfileID will always be encoded using upper case Base32 to ensure that it is not mangled by mobile devices that manipulate the case of text messages or URLs. The encryption algorithm and keys are controls by the Crypto section of the configuration. See **Error! Reference source not found.** section for more details.

Examples of where this feature may be useful in an application include:

1. Retrieving the mobile number associated with the mobile device.
2. Tracking user activity across multiple and independent channels of interaction.
3. Pre-populating already known information such as username, or UI preferences avoiding the need to recapture the information via the mobile device which is often cumbersome and time consuming.

6.2.5 Performance Properties

Mobile Profile exposes two properties that enable near real time monitoring of the mobile web application.

- `Bandwidth` – provides an approximation of the round trip bandwidth available between the web server and mobile device.

- `ResponseTime` – provides the average time between a request being made from the mobile device and a response being received from the web server.

The following sections explain each of these properties.

6.2.5.1 Round Trip Bandwidth

In addition to persisting application data the Mobile Profile is also used to calculate an approximate round trip bandwidth value in kilo bits per second. The total amount of data sent and received between the mobile device and the web server will be stored. The time taken to receive and transmit the data is also stored. See Figure 5 – Round Bandwidth Calculation.

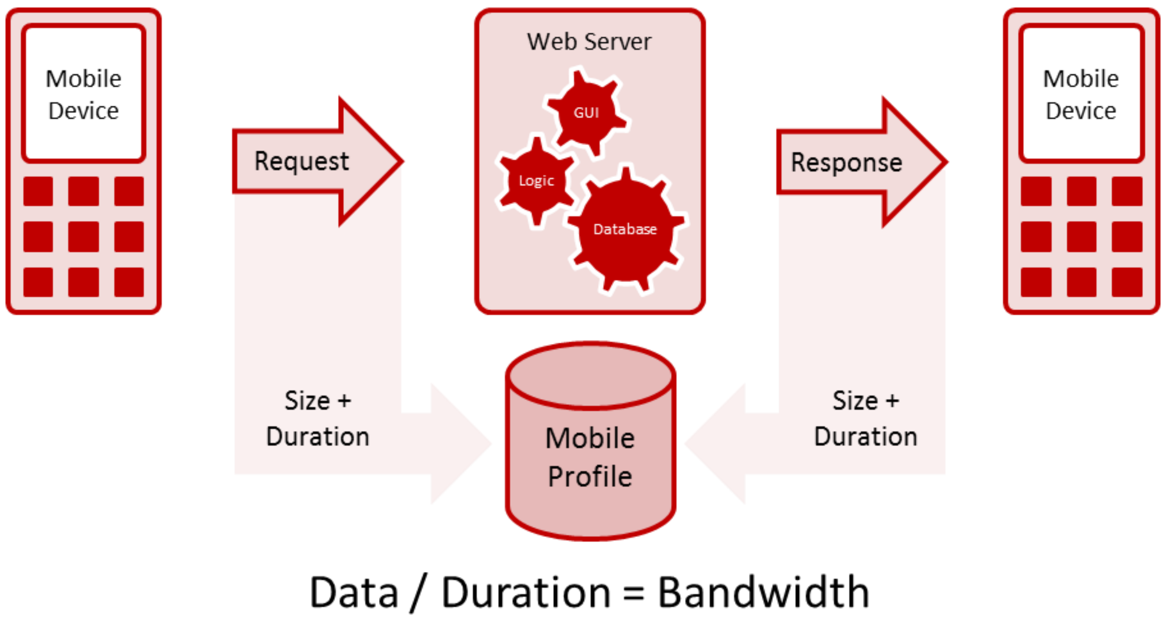

Figure 5 – Round Bandwidth Calculation

When the `Bandwidth` property is queried an average of the past 5 minutes activity is used to provide an average round trip bandwidth. This figure can be used to adapt the mobile web application to the current environment. For example; less images could be used, or a page presented informing the user their current experience is not satisfactory advising them to try again when they're in better mobile network coverage.

6.2.5.2 Response Time

In addition to round trip bandwidth the MobileProfile provides the average end to end response time experience by the end user via the `ResponseTime` property. Each request and response transaction is measured from the time the request is initiated to the time the response is received from the web server. See Figure 6 - End to End Response Time.

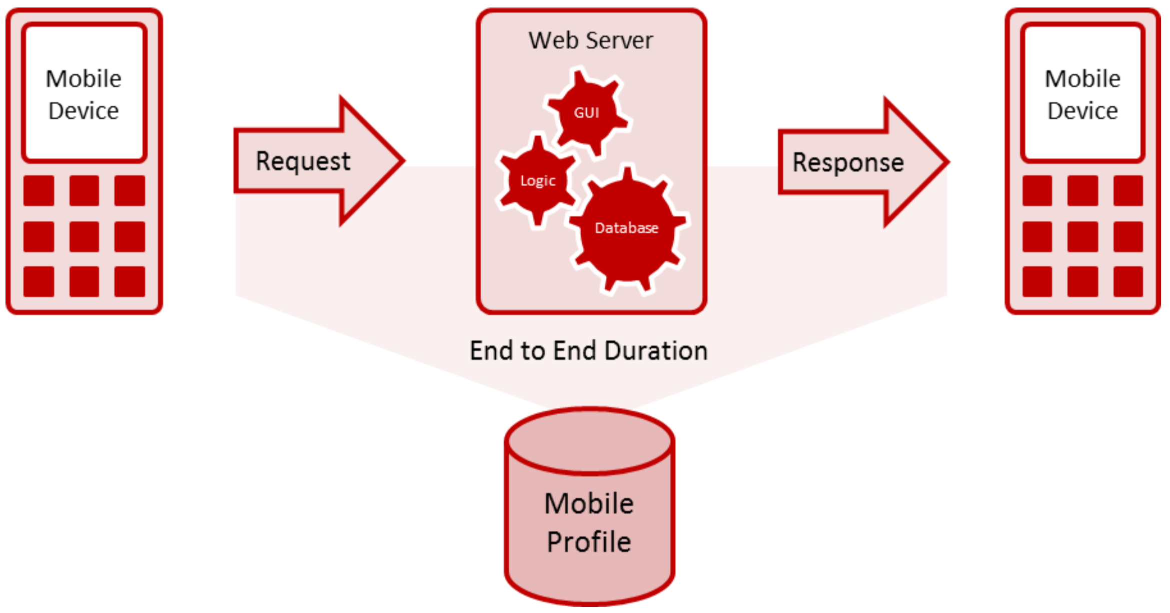

Figure 6 - End to End Response Time

When the `ResponseTime` property is queried the average response time over the past 5 minutes is returned in milliseconds. If insufficient data is held in the Mobile Profile to calculate a response time zero is returned.

The `ResponseTime` property of the mobile profile does not include the time taken to load images, JavaScript or CSS. If configured correctly these items will only be loaded from the web server the 1st time they are required.

6.2.5.3 Common Considerations

When using performance properties of the MobileProfile the following key points should be considered:

1. Performance calculations require some data points to be provided from the mobile device using JavaScript. If the mobile device does not support JavaScript the performance properties will always return zero.
2. Several transactions need to be completed by the same mobile device to populate sufficient information for the performance properties to return a value. If insufficient data is available zero will be returned.
3. Performance values are calculated the first time the property is queried during page processing and cached for subsequent instant retrieval avoiding subsequent database activity.

6.2.6 Mobile Controls

51Degrees.mobi controls use the mobile profile to store and repopulate data. The ability to recall data from previous interactions increases usability as it avoids the need for users to re-enter information they've already provided.

Controls which can support Mobile Profile include; TextBox, CheckBox, CheckBoxList, RadioButton, RadioButtonList, ListBox, Calendar and Login Control.

Mobile Profile aware controls can vary the scope of the data they remember using the `ProfileDataKey` property. Controls of the same type with the same `ProfileDataKey` value will share their default value across the application. Consider a textbox that is used for search on many different pages. Setting the `ProfileDataKey` to “Search” for all instances of the textbox will enable previously entered values to be displayed consistently for all search controls.

ASP.NET

```
<mob:TextBox runat="server" ID="TextBoxSearch" ProfileDataKey="Search"
MaxItems="5">
</mob:TextBox>
```

This behaviour is termed application scope.

If the `ProfileDataKey` is not specified the control will remember it’s previous state only for the page it appears on. This behaviour is termed page scope.