

Readme

Windows Azure Toolkit for Social Games

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | Version: | 1.0.0 | | Last updated: | 10/21/2011 | |  |

## Contents

[Overview 2](#_Toc305167120)

[Get Started 2](#_Toc305167121)

[The Solution 15](#_Toc305167122)

Overview

Building a social game is a tough challenge. From the first iteration developers must plan for and deal with issues like high concurrency, real time interaction, and rapid growth. The Windows Azure Toolkit for Social Games provides you with the tools you need to ramp up your game development quickly on an architecture that will help you face future challenges. The toolkit leverages the power of Windows Azure and game development best practices to handle even the most demanding social games.

This version of the toolkit consists of a generic game play service API, and two sample games that consumes the operations exposed by the game service: “Tic-Tac-Toe” a “Four in a Row”. The game play service is responsible for handling the generic game operations like user authentication, joining multiple players to a game, and persisting the game live state.

This guide will walk you through the steps for running the sample game locally using the Windows Azure emulator.

## Prerequisites

* 1. The following software is required to run this toolkit:
  + [Microsoft Visual Web Developer 2010 Express or Microsoft Visual Studio 2010](http://www.microsoft.com/express/Web/)
  + [Microsoft .NET Framework 4.0](http://www.microsoft.com/downloads/details.aspx?FamilyID=ab99342f-5d1a-413d-8319-81da479ab0d7)
  + Internet Information Services 7 , with ASP.NET Feature enabled
  + [Windows Azure SDK and Tools for Visual Studio (September 2011) version 1.5](http://go.microsoft.com/fwlink/?LinkID=128752)
  + [Microsoft SQL Server 2008 (Express edition or greater)](http://www.microsoft.com/express/Database/InstallOptions.aspx)
  + [Windows Identity Foundation Runtime](http://support.microsoft.com/kb/974405)
  + An HTML 5 capable browser – for example Internet Explorer 9

## Learning more about the Windows Azure Platform

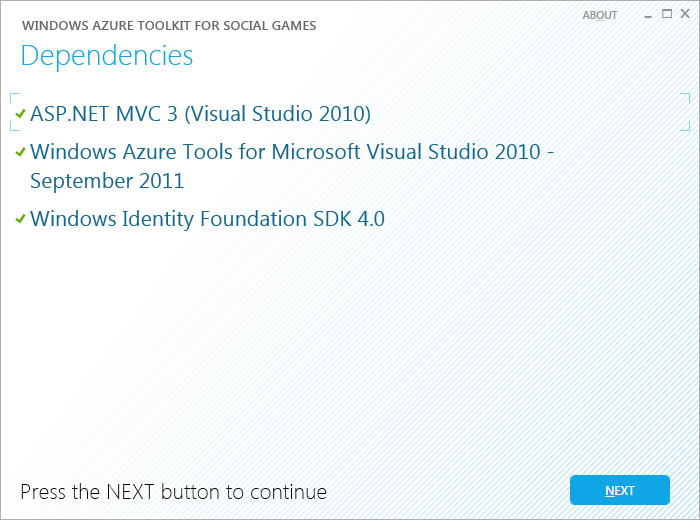
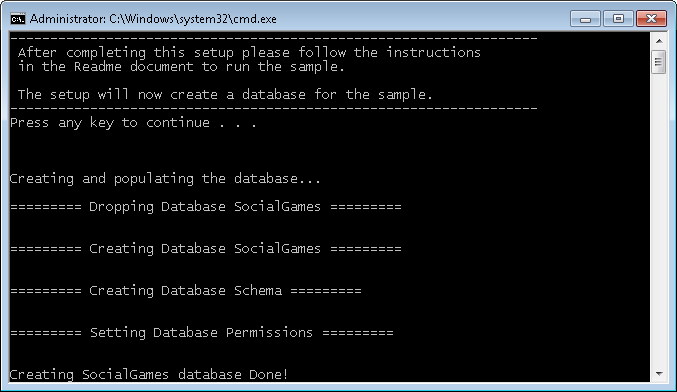
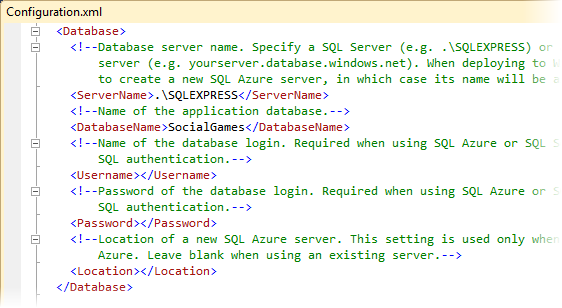
To learn more about the Windows Azure Platform and AppFabric, check these resources:

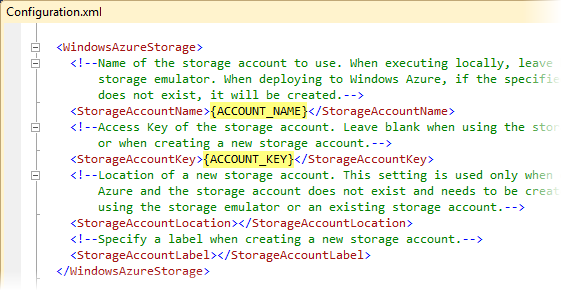
* + Complete the hands-on Labs in the Windows Azure Platform Training Course online on [MSDN](http://go.microsoft.com/fwlink/?LinkID=207018).
  + Learn how to build applications with the Windows Azure Platform Training Kit you can [Download Here](http://go.microsoft.com/fwlink/?LinkID=130354).

Get Started

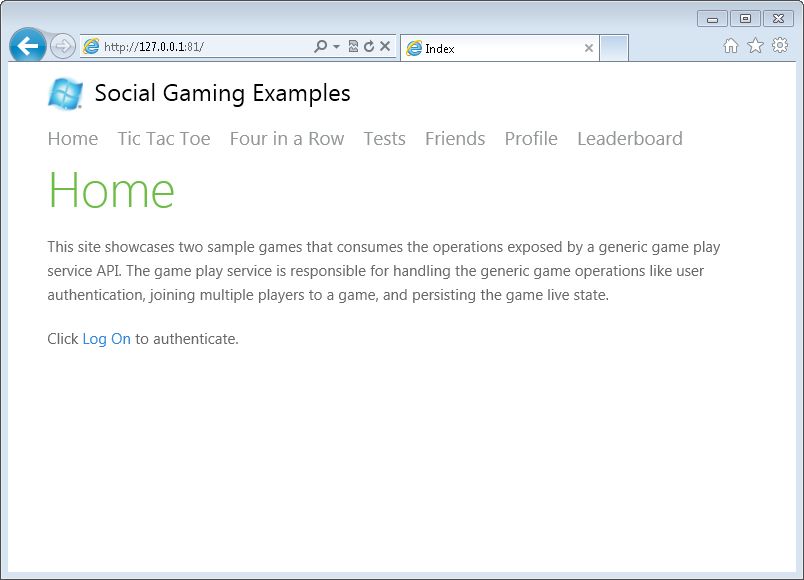
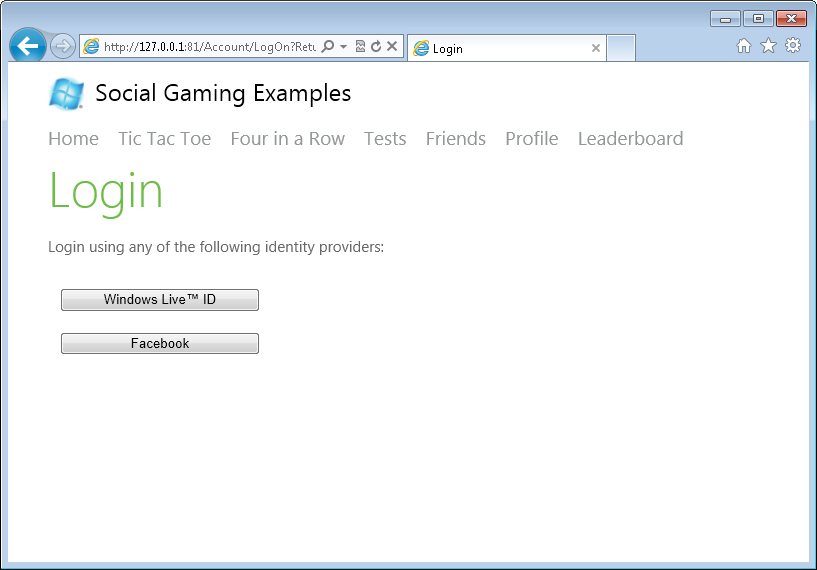
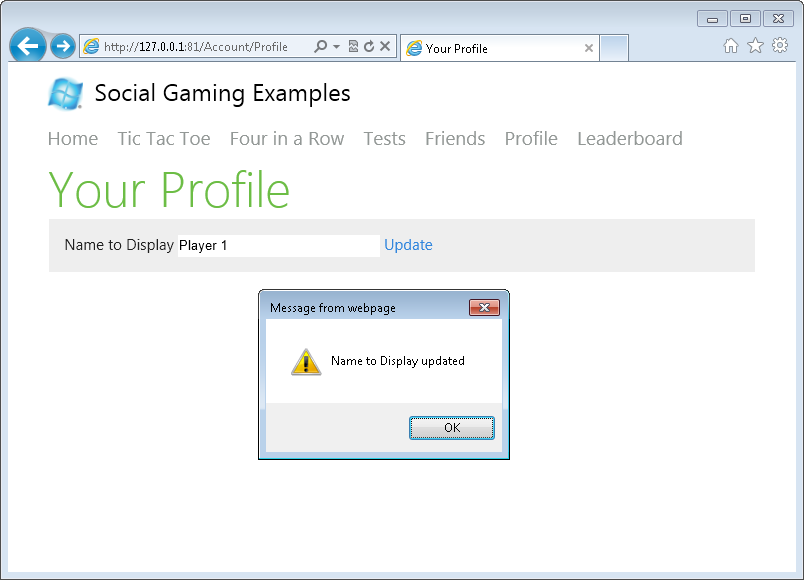
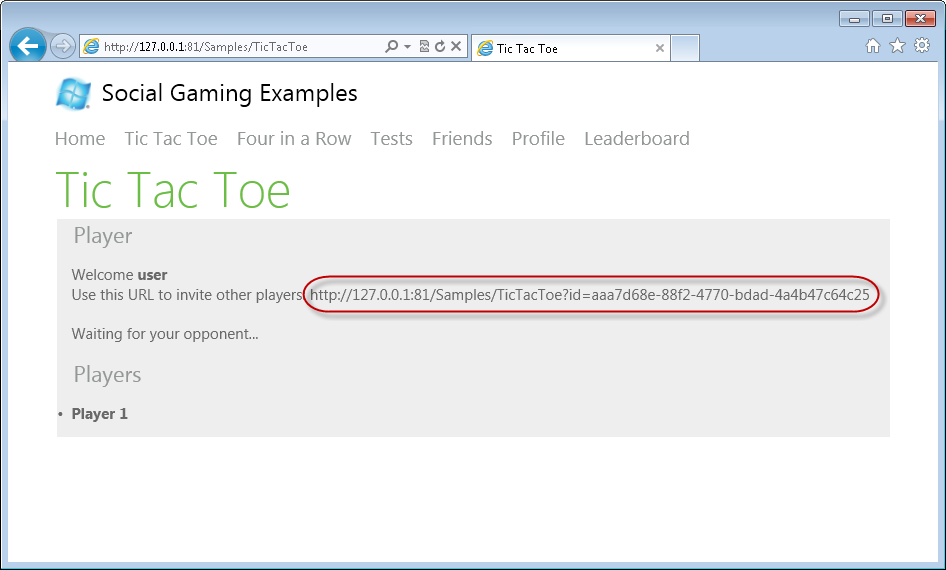
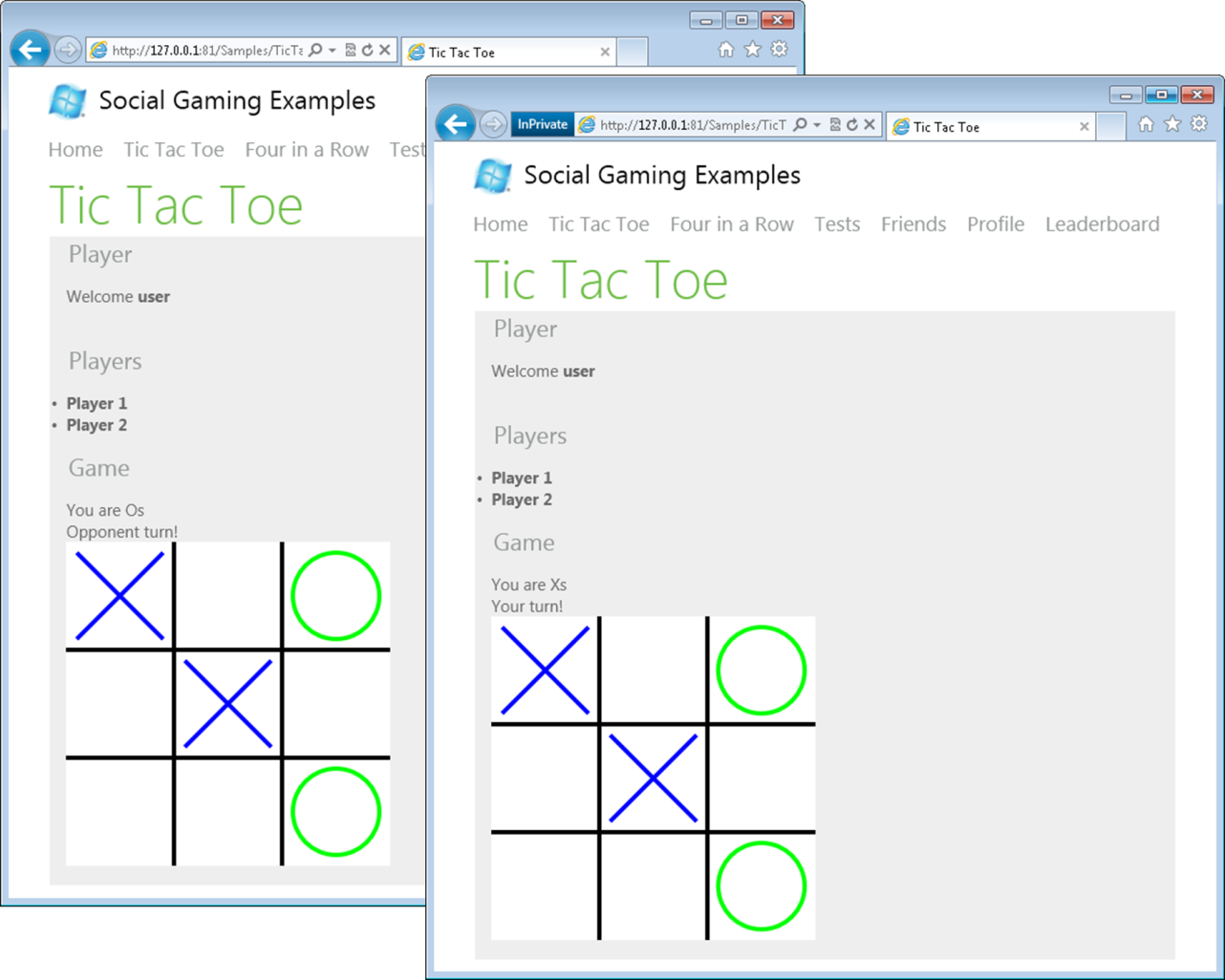
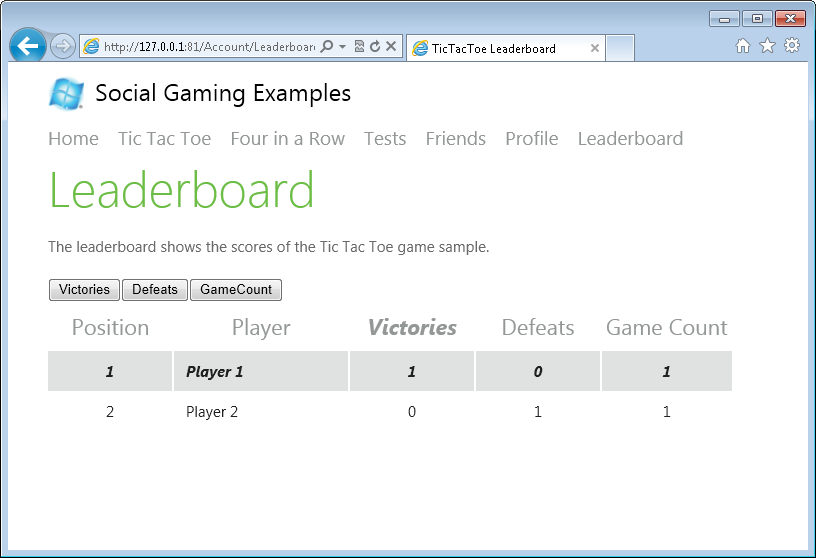
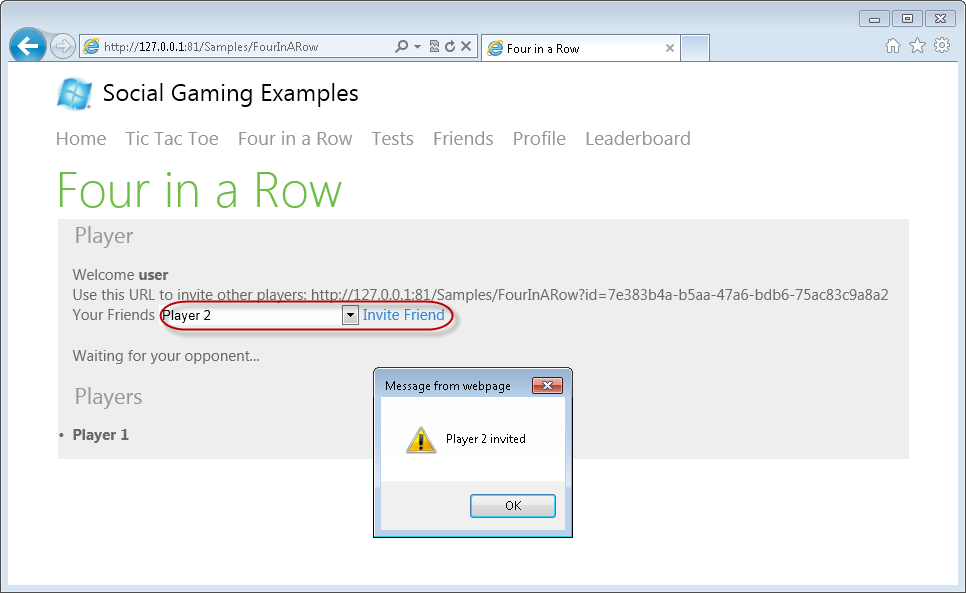
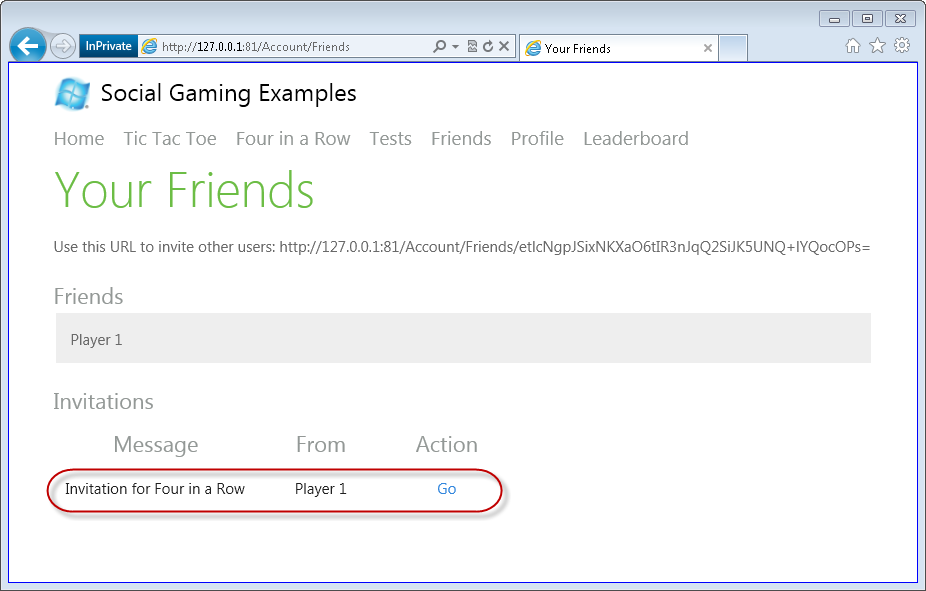
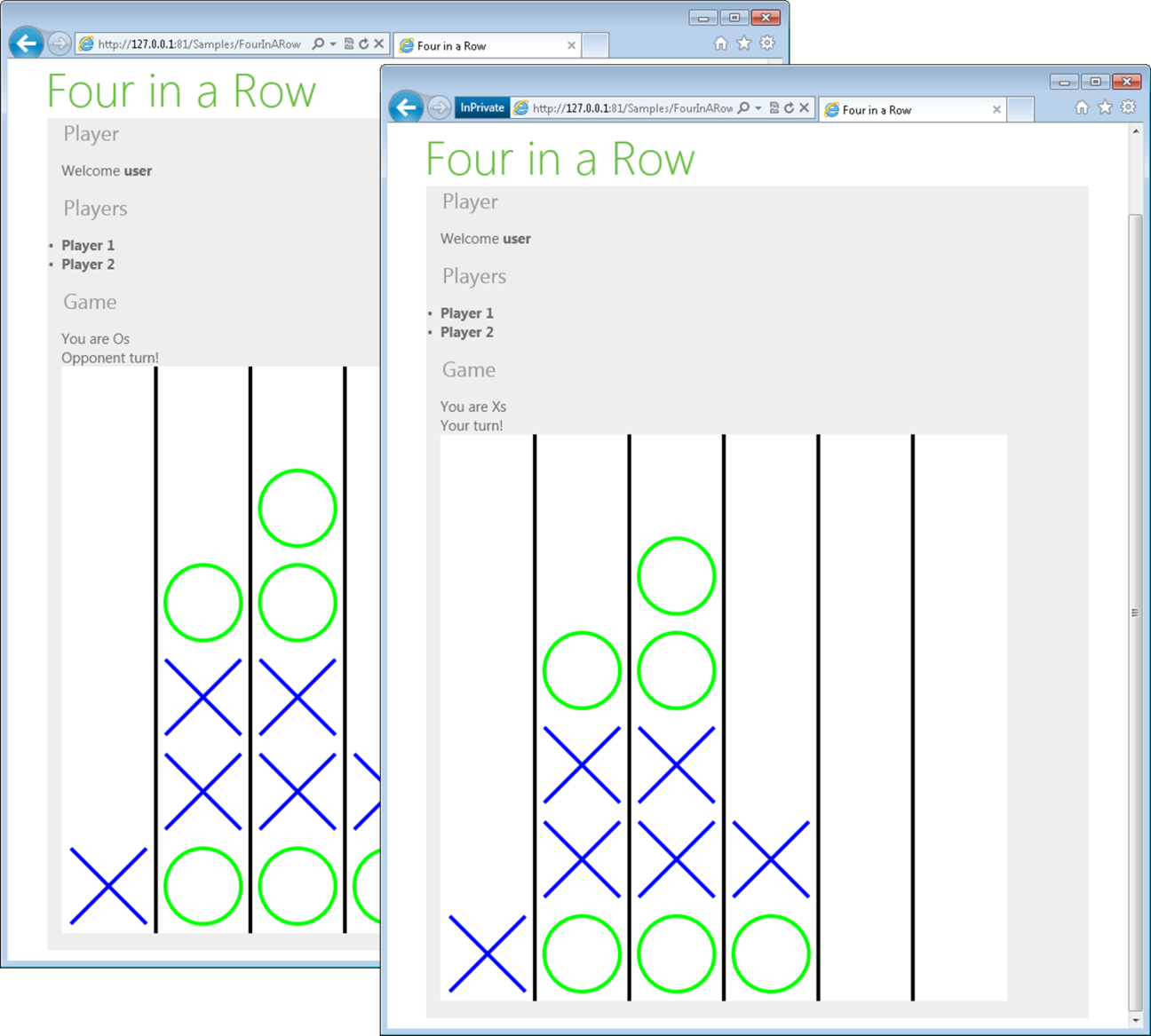
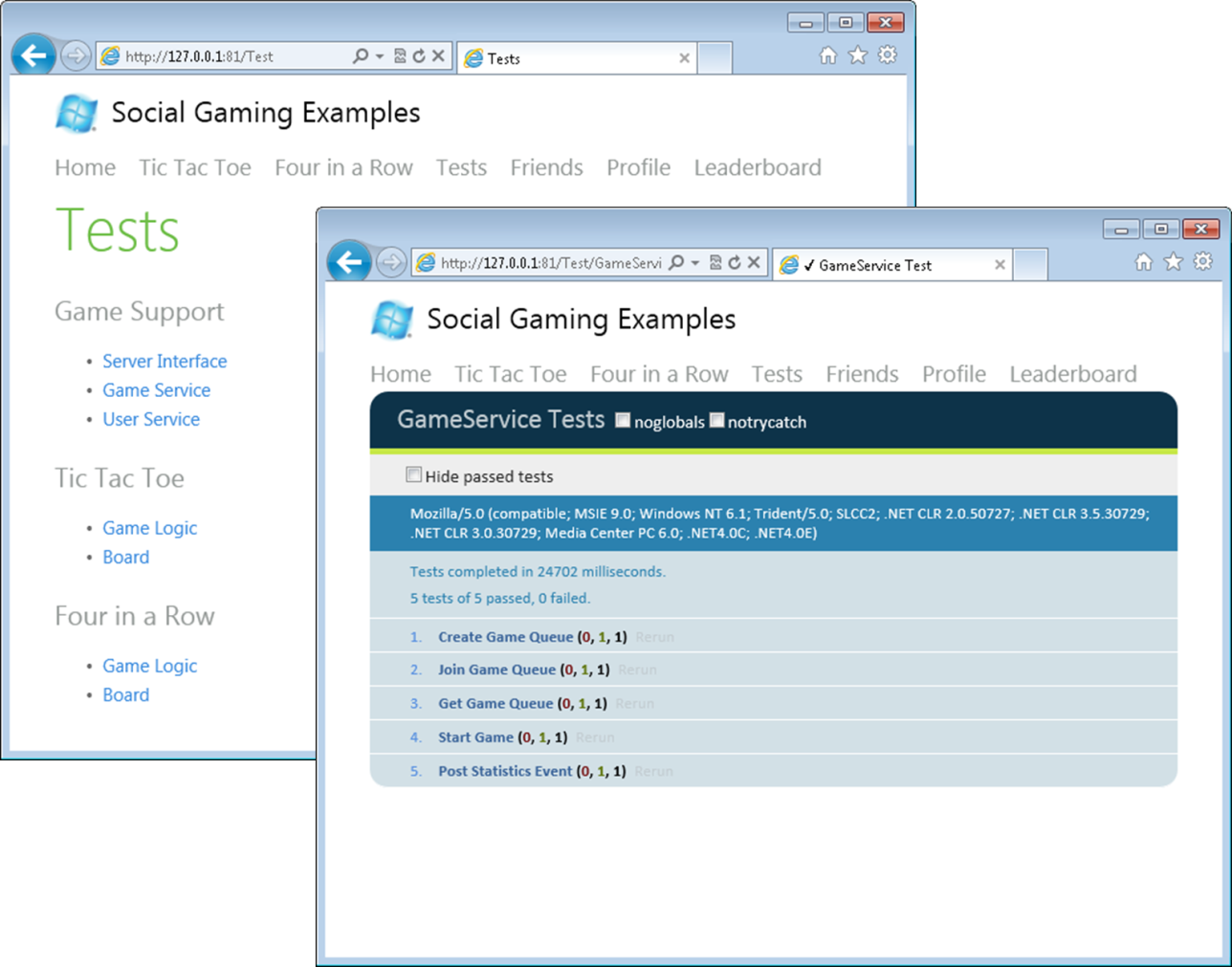
To get started with the Windows Azure Toolkit for Social Games you can run it locally using the Windows Azure compute emulator and your local SQL Server.

**Running the Setup**

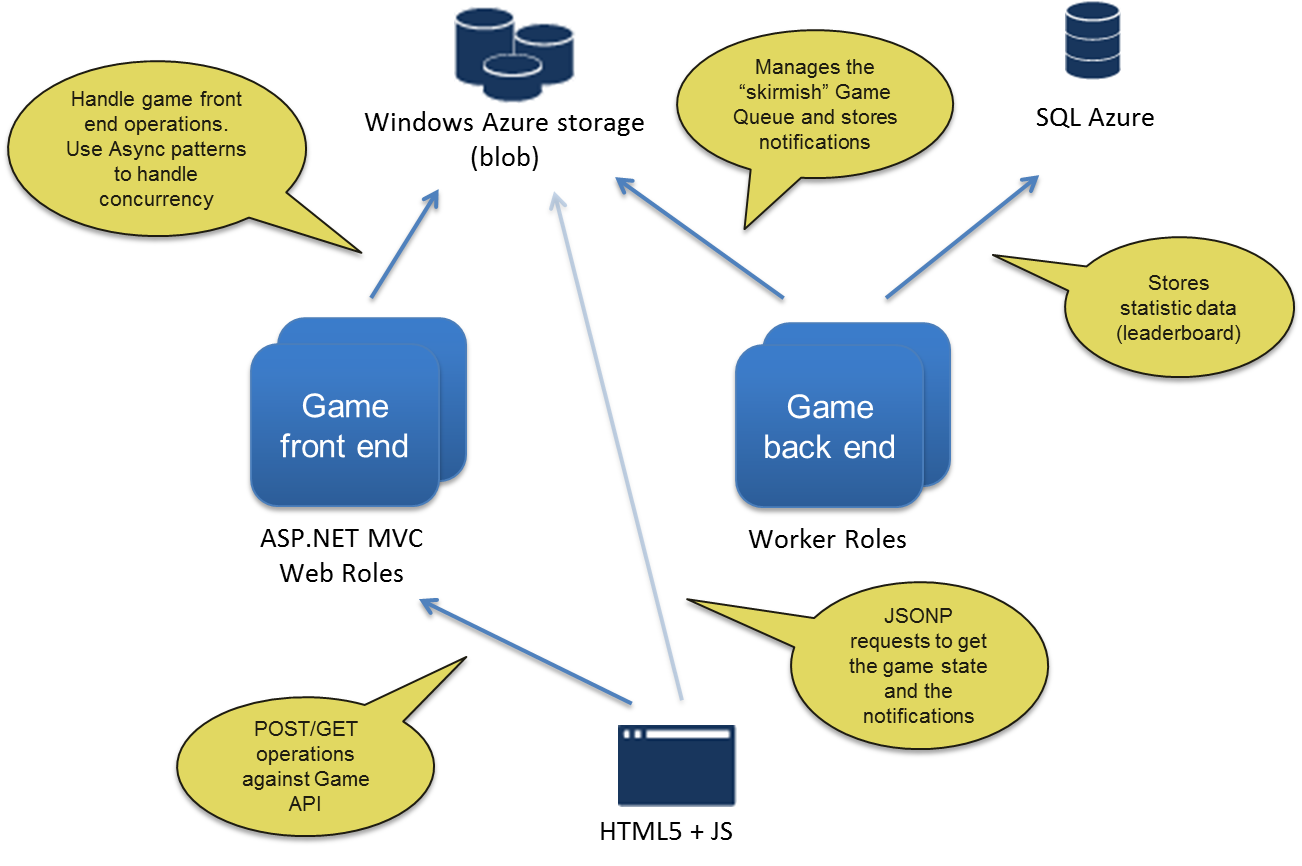
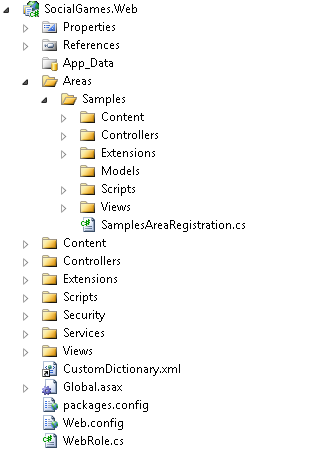
* 1. Before launching the application, you first need to verify that your computer is properly configured and has all the necessary software to build and use the toolkit.
  2. Launch **Setup.exe** located in the root folder where you extracted the sample package. Notice that it requires administrator privileges.
  3. When prompted, click **ACCEPT** if you agree to the terms of the license; otherwise, you will be unable to proceed with the installation and use the toolkit.
  4. The next step involves checking your machine for the prerequisites. If you do not have the necessary configuration or dependencies, the dependency checker will either provide a link to download any required software or launch the Web Platform Installer to install the prerequisite for you. After installing a missing dependency, click **Refresh** to initiate the detection process again.
     1. 
     2. Figure 1
     3. Checking your system for prerequisites
     4. **Note:** By default, you will not be able to proceed with the setup procedure unless the dependency checker has successfully verified every dependency. You can, however, instruct it to ignore a missing dependency by choosing the **SKIP** or the **SKIP ALL** option, but be aware that the sample might not run correctly if you do this. This option is mainly used in cases where a dependency has been superseded by a more recent software package not contemplated in the original dependency scripts.
  5. Once the detection process is complete and you have verified every prerequisite, click **Next** to launch the setup script.
     1. 
     2. Figure 2
     3. Setup scripts
  6. (Optional) The setup script configures the toolkit to run locally using the compute emulator. It creates a *SocialGames* database using your local SQL Express server instance (.\SQLExpress), and Windows Authentication to connect. To use a different database server instance for the toolkit, open the **Configuration.xml** file, which is located inside the root folder of this package, find the **Database** section, and update the **ServerName** setting to point to the correct SQL Server or SQL Azure instance. Similarly, configure the **DatabaseName** setting to use a different database.
     1. 
     2. Figure 3
     3. Configuring database settings
     4. **Note:** The **UserName** and **Password** settings are not required when using integrated authentication. The **Location** setting is only used when creating a new SQL Azure database server when deploying to Windows Azure.
     5. Additionally, the setup scripts configure the toolkit to use the local Windows Azure Storage emulator. If you want to use a Windows Azure Storage account instead, locate the **WindowsAzureStorage** section of the **Configuration.xml** file and set the **StorageAccountName** and **StorageAccountKey** settings to the name and access key of the storage account.

1. 
   * 1. Figure 4
     2. Configuring storage account settings
2. After changing the settings, please run the setup again.

**Running the Samples**

* 1. Open Visual Studio as administrator from **Start** | **All Programs** | **Microsoft Visual Studio 2010** by right clicking the Microsoft Visual Studio 2010 shortcut and choosing **Run as administrator**.
  2. Using Visual Studio, open the **SocialGames.sln** solution located under the **code** folder of the Toolkit package.
  3. Make sure that the **SocialGames.Cloud** project is selected as the start-up project (shown in **bold**).
     1. 
     2. Figure 5
     3. Solution Explorer showing the SocialGames solution
  4. Press **CTRL+F5** to build and deploy the application to the compute emulator. Your default Web browser should open pointing to <http://127.0.0.1:81/> and showing the Social Gaming Examples home page.
     1. **Note:** By default, the application is configured to use port 81, so you should make sure this port is free before running the application.
     2. 
     3. Figure 6
     4. Social Gaming Examples home page
  5. Click **Log On** to authenticate. You will be redirected to the login page.
  6. Log in with your Windows Live ID or Facebook:
     1. 
     2. Figure 7
     3. Login page
     4. **Note:** This sample uses a pre-configured AppFabric Access Control Service for managing user identity and access control. If you have a Windows Azure account and you want to use your own service you can do so, by updating the settings in the **Web.config** file of the **SocialGames.Web** project and the **ServiceConfiguration.cscfg** file in **SocialGames.Cloud**.
  7. Click **Profile** and change your display name to something else.
     1. 
     2. Figure 8
     3. Profile page
     4. **Note:** For Live ID users, the display name shown by default is empty the first time they authenticate. This is because there is no claim associated with the Live ID user name in the ACS namespace configuration. You should update the display name manually in the Profile page.
  8. The browser is redirected back to the Tic Tac Toe invite page. Copy the invite URL provided in the screen.
     1. 
     2. Figure 9
     3. Invite page
  9. Open another browser window using InPrivate mode (**Ctrl+Shift+P** in Internet Explorer) and browse to the invite URL. Log in using a different account. Wait until the board is displayed and start playing in turns with the two different windows.
     1. 
     2. Figure 10
     3. Online multiplayer Tic Tac Toe game
     4. **Note:** For Live ID users, the display name shown by default is empty the first time they authenticate. This is because there is no claim associated to the Live ID user name in the ACS namespace configuration. You should update the display name manually in the Profile page.
  10. Once the game finishes, click on the **Leaderboard** menu option. A board is shown, listing the players and their current scores.
      1. 
      2. Figure 11
      3. Leaderboard page
  11. Now, click on the **Four in a Row** menu option to play a different game. Make sure to select the player to invite from the combo box, and click on **Invite Friend**.
      1. 
      2. Figure 12
      3. Inviting a friend
      4. **Note:** After accepting an invitation, the game sets a friend relation between the players of an invited game.
  12. Switch to the browser of the invited player, and click on the **Friends** menu option. The invitation to the game should be displayed in the Invitations section. Click on **Go** to start the game.
      1. 
      2. Figure 13
      3. Invite notifications
  13. Wait until the board is displayed and start playing in turns with the two different windows.
      1. 
      2. Figure 14
      3. Online multiplayer Four in a Row game
  14. The sample web site also provides a set of tests for the different client components of the game. To execute the tests, click the **Tests** menu option and select the test fixture to run.
      1. 
      2. Figure 15
      3. Running the tests

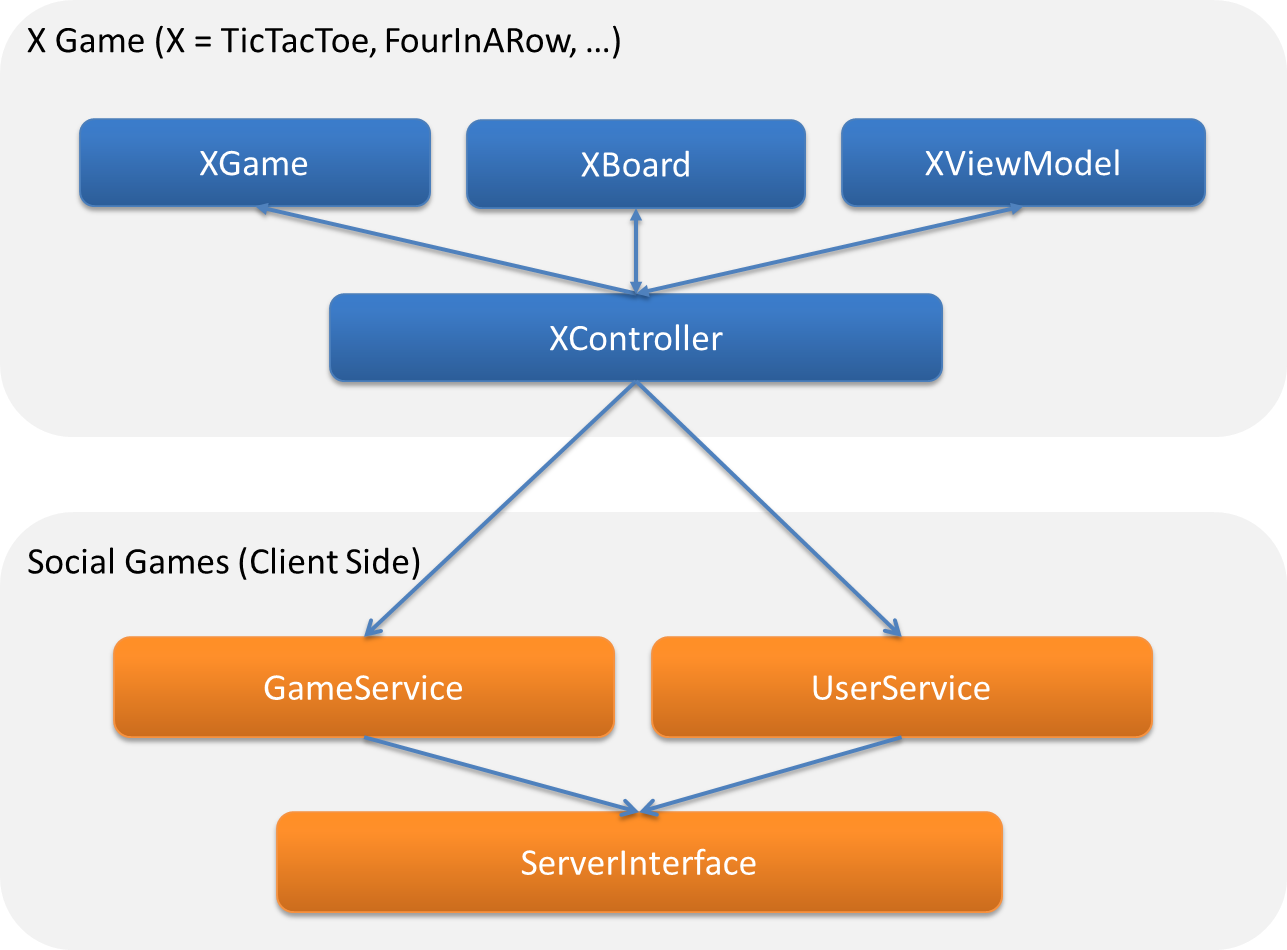
The Solution

* 1. The design of the Toolkit is organized in a way to facilitate other developers identify the reusable components to build their own social games.
  2. **High Level Architecture**
  3. 
  4. Figure 16
  5. High level architecture
  6. **Solution Structure**
  7. Open Visual Studio from **Start** | **All Programs** | **Microsoft Visual Studio 2010**.
  8. Open the **SocialGames.sln** solution located under the **code** folder of the Toolkit package. The solution includes the following projects:
     + **SocialGames.Core**: This project contains the model of the Social Game API. Here is where the all the game entities and the repositories are located.
     + **SocialGames.Worker:** This project contains all the processes that perform background tasks, like storing invitations, collecting game statistics, etc.
     + **SocialGames.Web:** This is an ASP.NET MVC project that hosts the Game Service API endpoints, as well as the “Tic Tac Toe” and “Four in a Row” games samples shipped with the Toolkit.
     + **SocialGames.Cloud:** This is the Windows Azure project associated to the role projects in the solution (**SocialGames.Web** and **SocialGames.Worker** projects). The local configuration is already in place to run the services on the Windows Azure Emulator.
     1. 
     2. Figure 17
     3. SocialGames solution
  9. Let’s take a look in more detail to the **SocialGames.Web** project. In Solution Explorer, expand the **SocialGames.Web** project node.
     1. This project contains, at the root level, all the common and reusable components to build social games, and uses an MVC Area for all the code related to a specific game implementation. In this case, the code specific to the “Tic Tac Toe” and the “Four in a Row” games are placed in that MVC Area.
     2. 
     3. Figure 18
     4. SocialGames.Web project
  10. Expand the **Views** folder at the root level of the web project. Do the same for the **Views** folder located inside the **Samples** MVC area.
      + The **common views** are for the features of the Social Game API that are common to all the games, like authentication, support for managing friends and the user profile, and a simple leaderboard.
      + The **game specific views** render the game play flow of different type of games.

|  |  |
| --- | --- |
| Common views | Game specific views |
|  |  |

* 1. Expand the **game** subfolder inside the **Script** folder at the root level. Do the same for the **game** folder inside the **Script** folder at the **Samples** MVC area.
     + The **common JavaScript game files** interact with the service operations provided by Game Service API. They take care of handling the communication through HTTP requests and send the appropriate response to the game specific components.
     + The **game specific JavaScript files** manage the behavior of a particular game. For example, they know how to render the game board, and execute the game rules, etc.

|  |  |
| --- | --- |
| Common JavaScript game files | Game specific JavaScript files |
|  |  |

* + 1. Below a diagram illustrating the interaction between the common and game specific JavaScript components:
    2. 
    3. Figure 19
    4. JavaScript components diagram