**Installation guideline for Booktera**

# System requirements

**.NET based components**

* Application Core, ASP.NET MVC web server
  + Windows 8.1
  + Visual Studio 2015
  + MS SQL Server 2012
  + .NET Framework 4.5
* Windows Phone client
  + Windows Phone SDK 8.0
* Windows 8.1 client
  + *(nothing extra)*

**Java based components**

* + Java SE Development Kit 1.8
* Play web server
  + Play Framework 2.3.0 (= activator 1.2.2)
  + Apache Maven 3.2.3
* Android client
  + IntelliJ IDEA 13[[1]](#footnote-1)
  + Android SDK Tools r24.1.2

# Setup application’s core

There are 2 important things before start:

* Always run Visual Studio with administrator rights to avoid misfunctionality caused by access restrictions (the correct functioning is not guaranteed without this; also WcfHost cannot be hosted without admin rights)
* There have to be a Default Database Instance with "." alias, using Windows authentication, and Booktera will need privileges to access it

Through the following steps you will install the core application. This means you will have the full backend (db with data, wcf layer ready to host [see in the next chapter]), and you can run the ASP.NET MVC 4 based web application, and the Windows 8.1 client.

1. **Copy** to your disk the BookTera folder
   1. Do not alter the folder structure under it!
   2. The path to it must not contain special characters or white spaces
2. **Start** Visual Studio with administrator privileges, and open the project. The project file’s location: \BookTera\Solution\theSolution\BookTera.sln
   1. Register for Windows Phone Developer license (it’s free) and whatever VS ask you to do for the first time
   2. If appears, ignore the warning that the WEB project needs SQL Server Express.
3. **Build** the solution (Ctrl+Shift+B)
   1. At the first build, a NuGet Restore process will be executed. This way NuGet downloads the referenced packages, and copies the dll-s of those to a local directory (\Solution\theSolution\packages) , so the projects can reach them.
4. Browse to project **Tools/InitSolution**, then right click on it 🡪 Set as startup project; then press Ctrl+F5 to run the project without debugging
   1. If you choose to start any project with debugging, first check this: General troubleshooting / [1]
   2. Note: We can follow in the Console window the flow of the process. The most time-consuming task is here to download a ~5MB archive file. This will take time depends upon your internet bandwidth.
5. Browse to project **DataBase/DB**, then publish it with ebookDB.publish.xml (double click/right click, publish; then click the Publish button)
   1. This will create our (empty) database, with name “BookTera”
6. Browse to project **Tools/TestData**, then right click on it 🡪 Set as startup project; then press Ctrl+F5 to run the project without debugging
   1. If you choose to start any project with debugging, first check this: General troubleshooting / [1]
   2. By running the TestData project, we fill the empty database with data. Most of it is mock data for books and transactions, but there is valid, important data as well, like categories, type enumerations, etc.
   3. Note: We can follow in the Console window the flow of the process. It will take about 0.5 – 2 minutes. There will be exception messages listed with green and red during generation mock transaction data, it’s expected, don’t worry about it.

## Troubleshooting

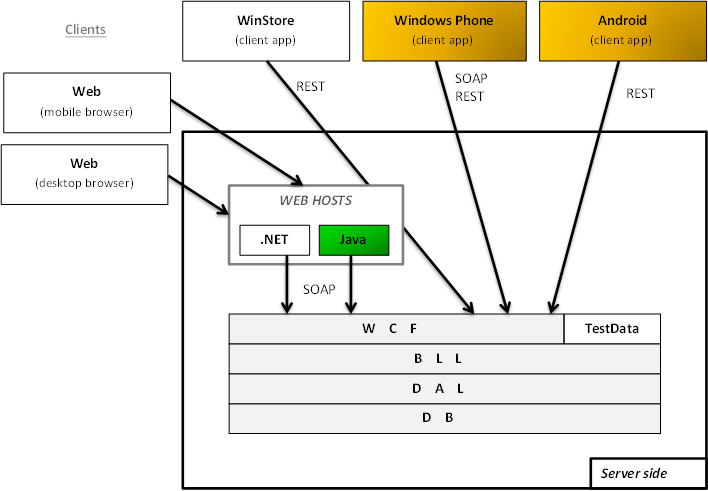
* If something went wrong, you can purge the db via step 4, then refill it via step 5.
* Troubleshooting for step InitSolution.
  + It (re)creates junctions pointing to content directories. If any of these would fail, first check you have started VS with admin privileges. If so, you can do this step manually.  
    You can create the junctions yourself using the “mklink /J” command prompt command. The junctions to create:  
    BookTera\Java\WebPlay\public\images < > BookTera\Solution\WEB\Content\Images  
    BookTera\Java\WebPlay\public\javascripts < > BookTera\Solution\WEB\Scripts\   
    BookTera\Java\WebPlay\public\stylesheets\shared< > BookTera\Solution\WEB\Content\themes
  + It also downloads resources for the TestData project. If it fails, you can manually download them from here:  
    <https://drive.google.com/uc?export=download&id=0B2aHj_zBJI1KRGFQbnhOdDJ0WW8>  
    This is an archive file. Decompress it’s content directly to here:  
    \Solution\Tools\TestData\Resources\

# Initialize the ground for the applications

The application BookTera, as mentioned before, consists of 5 runnable submodules/runnables:

* ASP.NET MVC 4 based web application
* Windows 8.1 client
* Play Framework 2.3.0 based web application
* Windows Phone 8 client
* Android 4.4 client

As you can see below on the project’s architecture image, these modules all depend on the WCF module (layer).



Therefore, **whatever runnable you want to use, first you have to start the WCF module**, which will start an ASP.NET MVC 4 based web host for the wcf services (via IIS Express). If you try to start any of the runnables before wcf process is hosted, you will get some socket layer exceptions, and the applications can’t run at all.

This (starting the WcfHost) you can achieve by browsing to project WCF/WcfHost, then right click 🡪 Set as StartUp project; then press Ctrl+F5 to run without debugging. (You may run it in debug mode, but it’s not necessary). This will also open a new tab in your default browser and browse to the root of the wcf project; and you also can see in your system tray that IIS Express started as well.

To start the ASP.NET MVC 4 web application, simply run the WEB project (right click, set as startup project, ctrl+F5)

## Troubleshooting

There can be some (magic) errors by starting the WcfHost project.

* Sometimes a service can’t be activated (they are activated the first time requesting them). Don’t be afraid about this, the project is hosted via IIS Express… All you have to do is rebuild (and with it republish) the WcfHost project
* At a time, there were instability problems with Newtonsoft.Json package. It has sometimes caused some exceptions with this message:  
  A(z) „Newtonsoft.Json” fájl vagy szerelvény, illetve annak egyik függősége nem tölthető be. A megtalált szerelvény jegyzékdefiníciója nem egyezik a szerelvény hivatkozásával. (A kivétel HRESULT-értéke: 0x80131040)  
  Probably cause of the error was that there were different versions installed to different projects. Luckily, after unifying the versions, this problem did not come up again. But in case the source of the problem was still something other, if you see the above error message, try the followings:

1. Reload the requested page
2. Rebuild the project owns the error
3. Reinstall the NuGet package.   
   In Visual Studio: Tools/Nuget Package Manager/ Package Manager Console  
   Update-Package Newtonsoft.Json -ProjectName {...} -Reinstall

# Mock users and passwords

The application has many mock users, you can sing in with any of them.  
All of them have the password: "**asdqwe123**".  
An example username is: "**ZomiDudu**"

# Setup and run subapplications

## Play

This section covers the necessary steps to be able to try out the WebPlay project. If you also want to use an IDE for it, you can find an internal guideline next to this document: "*Play/How to import WebPlay project into IntelliJ.docx*". This was made for own use, so it’s not too verbose. After all, you do not need any IDE to try out this project.

### Dependencies

First of all, you will need to download Play Framework 2.3.0 **(= Typesafe Activator 1.2.2)** and **Apache Maven 3.2.3**. Below are the links for these. **Very important note:** do not use any other version of the Play Framework! The work of the app is not guaranteed at all if you use a different version, even if it differs only in minor version numbers!

<http://downloads.typesafe.com/typesafe-activator/1.2.2/typesafe-activator-1.2.2.zip>

<http://archive.apache.org/dist/maven/maven-3/3.2.3/binaries/apache-maven-3.2.3-bin.zip>

After downloading and decompressing them, you will have to setup Environment Variables.

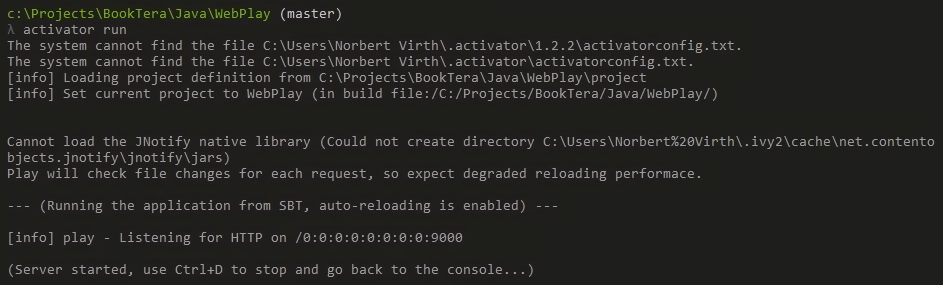
* **PATH**: Add the Play Framework’s root directory to the PATH Environment Variable. If you forget to do so, you won’t be able to use the framework, compile the project; and the error message won’t say you’ve forgotten this step!
* **JAVA\_HOME**: Create a new Environment Variable named “JAVA\_HOME”, and add the JDK 8 root directory to it as value. Both Maven and Play[[2]](#footnote-2) need this.
* (Optional) **PATH**: Add the Maven’s bin directory’s path to the PATH Environment Variable. This way, you can reach the mvn.bat anywhere simply with “*mvn”*, so you don’t have to use its full path. This guideline assumes you accomplished this step.

### Prepare and run the project

Before launching the WebPlay application, you have to install the ServiceClientProxy module to your local maven repository. The WebPlay project will seek for it there. To achieve this, follow these steps:

1. Browse to the directory: BookTera\Java\ServiceClientProxy\
2. Start a command prompt (cmd)
3. Type in: **mvn clean install**

To compile and start the WebPlay project, follow these steps:

1. Browse to the directory: BookTera\Java\WebPlay\
2. Start a command prompt (cmd)
3. Type in: **activator compile**
   1. Don’t worry if you get warnings like “The system cannot find the file …\activatorConfig.txt”. These appeared by me, but caused no problem.
   2. The first run will be very slow. Very, very slow. Despite you've downloaded already 400MB with the activator binaries, it will download a lot of things yet, maybe the whole framework again, I don’t know. Sometimes, by me, some Connection timeout errors also appear in the console; but the process will continue. You have to be **very patient** with it. Even if you monitor your network traffic, you won't see every time downloading - sometimes it just stands with 0KB up- or download rate, the console also stops. In this case, just wait. The whole process can take **10-40 minutes**!
4. Type in: **activator run**Wait a bit, and you will see soon something like this:  
   
5. You can now browse to the web application at <http://localhost:9000>

### Troubleshooting

* If “maven clean install” wouldn’t finish successfully the first time, run it again. I once got a “connection refused” error by some dependencies; but the second time, it worked
* If you have problems with the character encoding, set the JVM’s default encoding to UTF8 via this: <http://stackoverflow.com/questions/361975/setting-the-default-java-character-encoding/623036#623036>. It should not be happen, the Play project’s default char encoding is set to UTF8
* Don’t do a clean-and-then-make (or recompile) via any IDE. Play hacks the bytecode in the background. If you clean the project, first you will only be able to recompile it via Play.
* Don’t worry, if the IDE marks for you some errors, but the project compiles and runs. The framework’s IDE support is not perfect yet.

## Setup for mobile clients

### Setup Firewall and applicationhost.config

You have to complete – with some help below – the  **steps 1-4 (only 1-4!)** here at the „*Quick solution with IIS Express*” section here:  
<http://msdn.microsoft.com/en-us/library/windowsphone/develop/jj684580%28v=vs.105%29.aspx>  
(I suggest rather the saved version: Mobile clients/**How to connect to a local web service from WP8.mht**). Despite of this source’s name, it’s also a necessary setup for the Android client.

Help for the setup:

* **Step 1**: Create a firewall exception to allow HTTP requests through the firewall on the port that IIS Express is using. These ports are **50135** (*WcfHost*) and **50308** (*WEB*). Note: by me, also an OS restart was necessary after the creation of the firewall exception. Anyway, if you encounter some errors, probably due to firewall activity, you can temporarily disable the firewall while testing the clients.
* **Step 4**
  + Before starting this step, be sure you have at least once deployed (started) the *WcfHost* and the *WEB* projects. Without this, the *applicationhost.config* file will not contain the necessery sections to modify
  + You have to see something like this (\* the underlined rows are to be inserted by you):

<sites>

<site name=**"WEB"** id=**"2"**>

<application path=**"/"** applicationPool=**"Clr4IntegratedAppPool"**>

<virtualDirectory path=**"/"** physicalPath=**"C:\Norbi\Development\BookTera\Solution\WEB"** />

</application>

<bindings>

<binding protocol=**"http"** bindingInformation=**"\*:50308:localhost"** />

<binding protocol=**"http"** bindingInformation=**"\*:50308:192.168.1.102"** />

</bindings>

</site>

<site name=**"WcfHost"** id=**"3"**>

<application path=**"/"** applicationPool=**"Clr4IntegratedAppPool"**>

<virtualDirectory path=**"/"** physicalPath=**"C:\Norbi\Development\BookTera\Solution\WCF\WcfHost"** />

</application>

<bindings>

<binding protocol=**"http"** bindingInformation=**"\*:50135:localhost"** />

<binding protocol=**"http"** bindingInformation=**"\*:50135:192.168.1.102"** />

</bindings>

</site>

</sites>

* + Of course, you have to write your own local IP address into the *applicationhost.config* file, the above is just my example to make things easier.

### Setup your local IP address

After you are done with these first steps, you have to run the **SetupLocalIp**program, which you can find here: Solution/Tools/SetupLocalIp. This app will guide you and tell everything it has to/has done. Once you have fully built the Solution, you can access this tool’s executable file under SetupLocalIp/bin/Debug, so you can run it also manually, outside of the IDE. Once you’ve run it, you have to restart Visual Studio, to be able to bind the sites to the new local IP addresses.[[3]](#footnote-3)

Note: Later, **every time your local IP address changes, you have to re-run this** *SetupLocalIp* tool; and then restart the Visual Studio, redeploy Windows Phone and Android clients.

### General notes

* To use the application, the devices/emulators have to have a common LAN access with the computer hosting the WcfHost and the Web projects.

## Windows Phone

You have to download and install the **Windows Phone SDK 8.0[[4]](#footnote-4)**, and complete the ”*Setup for mobile clients”* section to be able to run the Windows Phone application.

### Running on device

The Windows Phone 8 Emulator requires Windows 8 Pro edition or greater, and a processor that supports HyperV and SLAT (Second Level Address Translation). If you don’t have the proper equipment for these, you can run the app on a physical device.

First, you have to **developer unlock** your device. Follow this guidelines to achieve it:  
<http://msdn.microsoft.com/library/windows/apps/ff769508(v=vs.105).aspx>  
(I suggest rather the saved version: Mobile clients/How to register your phone for development for Windows Phone 8.mht)

If you get here, you are able now to deploy (run) the application to the device. Congratulations ☺

Note, that if you also want to use the application, the phone’s wifi have to be on; and the phone have to have a common LAN access with the computer.

### Troubleshooting

* Hint: I experienced a strange debug error once. It was on a Nokia Lumia 520. This phone can get crazy sometimes. So, the debugging have not worked for me: it started successfully, then after ~10-15s, it exited without prompts or logs. But if I started the app without debugging, nothing wrong happened. The solution was just to restart the phone…

## Android

You have to download and install the **Android SDK Tools r24.1.2**, and complete the ”*Setup for mobile clients”* section to be able to run the Windows Phone application.

### Android SDK Tool dependencies

Once you have downloaded the Android SDK Tools, a tool named ”*Android SDK Manager*” will be available for you. With this, download and install the followings:

* Tools
  + Android SDK Plaform-tools, version 22
  + Android SDK Build-tools, version 22 and 21.1.2
* Android 5.0.1 (API 21)
* Extras
  + Android Support Repository, version 11
  + Android Support Library, version 22
  + Google USB Driver, version 11[[5]](#footnote-5)

### Setup for the project (with IntelliJ IDEA)

#### Provide SDKs for the project

You have to provide the followings for the project:

* Java SDK, at least version 1.7[[6]](#footnote-6)
* Android SDK, at least at API level 21

You can setup these in IntelliJ here: File/Project Structure/SDKs; click the green + icon, and add an Android SDK, and a JDK

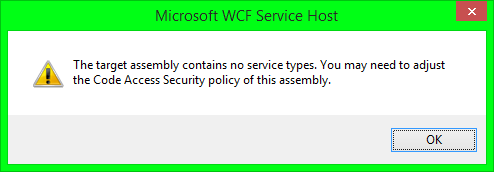
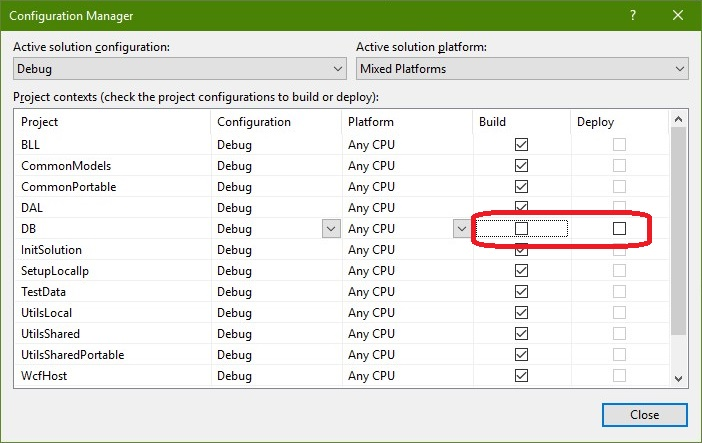
#### Setup Gradle

* First, you have to provide the Android SDK’s home path for gradle:
  + You can setup an Environment Variable named ANDROID\_HOME, pointing to the Android SDK’s home. After this, of course, you have to restart your OS.
  + If you don’t want a restart, you can provide this information locally for the Gradle. Create this file:   
     *BookTera\Java\Android\local.properties*with this content:  
     sdk.dir=<Android SDK’s home path>  
    For example:  
     *sdk.dir=C\:/Program Files (x86)/Android/android-sdk*
* After this, go to *BookTera\Java\Android\build.gradle*. This is the Android project’s main *build.gradle* file. If you open this in IntelliJ the first time, you will see an error message that you have to sync Gradle. Click "sync now" to download dependencies and build the project (first).
  + If there is any error, Gradle will tell you how to fix it. E.g. if you haven't installed Build-tools 21.1.2, only 22, it will tell you do so; in fact, it will even provide you a "click here to solve the problem" solution as well.

Note: Sometimes, after a build, a "Language level changes..." alert appears, and the prompt request you to reload the project. Click No here, it’s not necessary to reload the project. In fact, it’s rather recommended to restore the language level to Java 8; so the IDE won’t complain about lambda expressions. It is because the project uses RetroLambda to be able to use lambda expressions from Java 8; but Android is not compatible yet with that language level, only with Java 7.

You can restore the language level in IntelliJ here: *File/Project Structure/Project/Project language level*

# General troubleshooting

1. If you debug any project in this solution, you might see this alert window:  
     
   Do not take much attention to this, it’s a framework bug; it will cause no problem for this application. Except one thing: if you switch to another application after starting the Debug process, then this alert popups in the background (you will only hear a windows sound); you won’t be able to seek this windows by switching back to VS. First you will have to go to the desktop (win + D, or click in the bottom-right corner), and only then to VS; other way this windows won’t appear; and until you don’t click OK for it, the whole process will pause.
2. The project originally was made in Visual Studio 2013. When migrated it into VS 2015, some solution settings had to be reset.  
   The project DB must not be built and deployed automatically:  
   (you can reach this via *Build/Configuration Manager*)  
     
   These have been set in VS 2013, but had to be reset in VS 2015.

1. Of course you can build this project with Eclipse/Android Studio as well. This guide only contains information about how to install the project with IntelliJ [↑](#footnote-ref-1)
2. For Play – in theory – it would be enuogh to put the JDK’s root into PATH, but you will need JAVA\_HOME for Maven anyway. [↑](#footnote-ref-2)
3. This is only necessary when the *applicationhost.config* file has been modified during the process [↑](#footnote-ref-3)
4. It may have been installed with Visual Studio though [↑](#footnote-ref-4)
5. I suppose this last is only necessary when you will use physical devices to deploy the app. [↑](#footnote-ref-5)
6. The recommended version is 1.8. [↑](#footnote-ref-6)