Contents

[1. Introduction 1](#_Toc481743524)

[2. Design 1](#_Toc481743525)

[3. Development 2](#_Toc481743526)

[3.1. CitiBanamex Standards and Guidelines 2](#_Toc481743527)

[3.2. One-Time Setup of tools for Development on workstation 2](#_Toc481743528)

[3.2.1. Setup of STS 2](#_Toc481743529)

[3.2.2. Setup of Postman 3](#_Toc481743530)

[3.2.3. Setup of MongoDB (not applicable for all projects) 3](#_Toc481743531)

[3.2.4. GIT Bash or UI client 7](#_Toc481743532)

[3.3. Code Management 8](#_Toc481743533)

[3.3.1. Access to GITHub 8](#_Toc481743534)

[3.3.2. Check-out and Check-in guidelines 9](#_Toc481743535)

[3.4. Project Development 9](#_Toc481743536)

[3.4.1. Download Dependency Projects 9](#_Toc481743537)

[3.4.2. Create Project Source Code Directory 11](#_Toc481743538)

[3.4.3. Spring boot project setup 12](#_Toc481743539)

[3.4.4. Logging 14](#_Toc481743540)

[3.4.5. Swagger Documentation 14](#_Toc481743541)

[3.4.6. Editing project setup and configurations 14](#_Toc481743542)

[3.4.7. TDD development guide 14](#_Toc481743543)

[3.4.8. Local Build and Unit test 16](#_Toc481743544)

[3.5. Useful scripts/snippets 19](#_Toc481743545)

[3.6. Troubleshooting Notes and Experiences 19](#_Toc481743546)

# Introduction

**Mission**

As an engineer working in the API Factory, your mission is to develop solid, reusable, well-designed APIs that

* Follow the CitiBanamex API Guidelines for standardization and security
* Follow REST architectural guidelines for simplicity and consistency
* Are managed and operated as microservices for agility
* Are hosted in Pivotal Cloud Foundry for scalability and availability

To help guide you through the development process this document describes a virtual "day in the life" of an API Factory software developer and provides both principles to follow and steps to take that will achieve your mission.

**Requirements**

The development of APIs will follow an Agile methodology. The fundamentals of Agile are to use short development iterations to generate real customer feedback that guides subsequent development iterations.

In an Agile process, requirements are described at a high level for a subset of the most important features or functions. Requirements are described as outcomes and observed behavior and not as implementation instructions or technical details. Choices about implementation are discussed as part of the development iteration. This discussion takes place between developers and customer representative (the "product owner").

Lightweight tools and processes are to be used and so when clarification of desired behavior occurs, they are not required to be documented in a specification (although lightweight documentation is also acceptable), but rather should be rapidly implemented in software and demonstrated to the product owner for confirmation that it meets the expectations.

**Guidelines**

* Review the high level goals and expected behavior for an Agile iteration – essentially, review the user stories. If the expected behavior (not the internal implementation but the observable behavior) is not clear, then discuss directly with the product owner and the tech lead to clarify. Do not wait for a meeting to discuss, but connect with people directly. If clarity does not occur quickly then the associated requirement or user story should be deprioritized or deferred to a future iteration.
* If the implementation of the desired behavior will result in numerous error conditions or ambiguous situations which are not described in the requirement or user story then discuss with the tech lead and product owner to extend the requirement to take these cases into consideration and provide an expected outcome.

# Design

When developing the APIs based on the requirements described in user stories, the APIs should follow REST design principles and use the 'resource modeling' process.

Guidelines for Resource Modeling

* Simplify the user stories into short, simple phrases with clear nouns and verbs
* Review with tech lead and product owner that these phrases correctly capture the goal of the requirements
* Translate the nouns from these phrases into an 'entity data model', similar to an ERD
* Review the data model to ensure relationships and cardinality accurately represent CitiBanamex business object models
* Translate the ERD into REST Resources and object models (schemas). Schemas should be based on Citi global object schemas, but simplified and standardized
* Defined which HTTP methods will be supported for each resource : GET, PUT, POST, DELETE

# Development

## CitiBanamex Standards and Guidelines

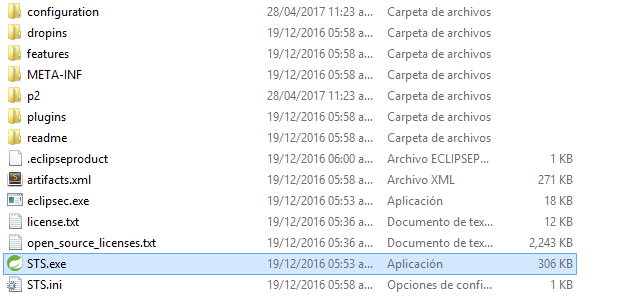


## One-Time Setup of tools for Development on workstation

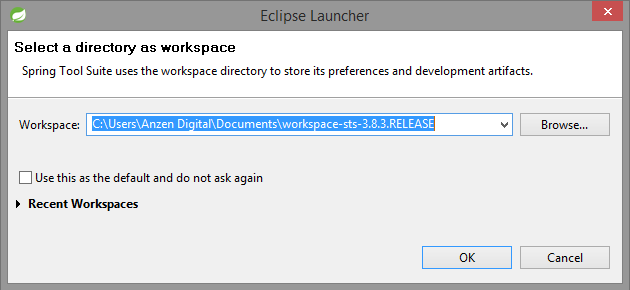
### Setup of STS

The Spring Tool Suite is an Eclipse-based development environment that is customized for developing Spring applications. It provides a ready-to-use environment to implement, debug, run, and deploy your Spring applications, including integrations for Pivotal tc Server, Pivotal Cloud Foundry, Git, Maven, AspectJ, and more.

1. To get the latest version available go to the following link and download the windows version that is suitable for you. <https://spring.io/tools>
2. Once downloaded, unzip the download of STS to the root directory of a drive (this will avoid possible problems with long pathnames).

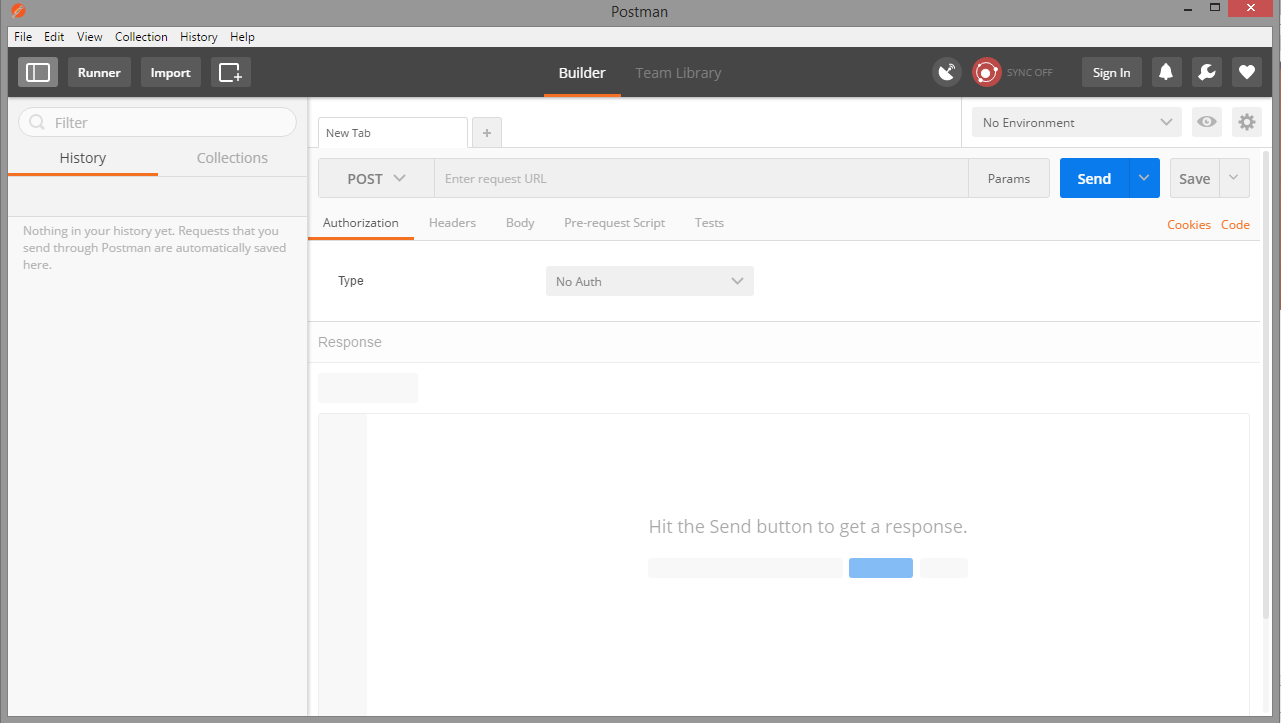


1. To verify the installation, run the STS.exe executable in the unzipped directory and check that STS displays a welcome panel. The first time there may be a short delay due to the initial set-up of indexes.
2. While starting, STS will ask for workspace, like eclipse. STS is built on top of eclipse.



### Setup of Postman

* With Postman, you can construct requests quickly, save them for later use and analyze the responses sent by the API.
* Postman is available as a native app (recommended) for Mac / Windows / Linux, and as a Chrome App. The Postman Chrome app can only run on the Chrome browser. To use the Postman Chrome app, you will first need to install Google Chrome.
* If you already have Chrome installed, head over to Postman’s page on the Chrome Webstore and click ‘Add to Chrome’.
* The download should take a few minutes, depending on your internet connection. Once you’ve downloaded the app, you can launch Postman.
* You can visit below link for more info.
* <https://www.getpostman.com/docs/introduction>



### Setup of MongoDB (not applicable for all projects)

**Install MongoDB Enterprise on Red Hat Enterprise or CentOS**

PLATFORM SUPPORT

This installation guide only supports 64-bit systems. See [Platform Support](https://docs.mongodb.com/manual/release-notes/3.0-compatibility/#compatibility-platform-support)for details.

MongoDB 3.2 deprecates support for Red Hat Enterprise Linux 5.

MongoDB provides officially supported Enterprise packages in their own repository. This repository contains the following packages:

|  |  |
| --- | --- |
| **mongodb-enterprise** | A metapackage that will automatically install the four component packages listed below. |
| **mongodb-enterprise-server** | Contains the [mongod](https://docs.mongodb.com/manual/reference/program/mongod/#bin.mongod) daemon and associated configuration and init scripts. |
| **mongodb-enterprise-mongos** | Contains the [mongos](https://docs.mongodb.com/manual/reference/program/mongos/#bin.mongos) daemon. |
| **mongodb-enterprise-shell** | Contains the [mongo](https://docs.mongodb.com/manual/reference/program/mongo/#bin.mongo) shell. |
| **mongodb-enterprise-tools** | Contains the following MongoDB tools: [mongoimport](https://docs.mongodb.com/manual/reference/program/mongoimport/#bin.mongoimport) [bsondump](https://docs.mongodb.com/manual/reference/program/bsondump/#bin.bsondump), [mongodump](https://docs.mongodb.com/manual/reference/program/mongodump/#bin.mongodump), [mongoexport](https://docs.mongodb.com/manual/reference/program/mongoexport/#bin.mongoexport), [mongofiles](https://docs.mongodb.com/manual/reference/program/mongofiles/#bin.mongofiles), [mongooplog](https://docs.mongodb.com/manual/reference/program/mongooplog/#bin.mongooplog), [mongoperf](https://docs.mongodb.com/manual/reference/program/mongoperf/#bin.mongoperf), [mongorestore](https://docs.mongodb.com/manual/reference/program/mongorestore/#bin.mongorestore), [mongostat](https://docs.mongodb.com/manual/reference/program/mongostat/#bin.mongostat), and [mongotop](https://docs.mongodb.com/manual/reference/program/mongotop/#bin.mongotop). |

The default /etc/mongod.conf configuration file supplied by the packages have bind\_ip set to 127.0.0.1 by default. Modify this setting as needed for your environment before initializing a [replica set](https://docs.mongodb.com/manual/reference/glossary/#term-replica-set).

**Step 1 - Configure repository.**

Create an /etc/yum.repos.d/mongodb-enterprise.repo file so that you can install MongoDB enterprise directly, using yum.

For the latest stable release of MongoDB Enterprise, use the following repository file:

**[mongodb-enterprise]**

name=MongoDB Enterprise Repository

baseurl=https://repo.mongodb.com/yum/redhat/$releasever/mongodb-enterprise/3.4/$basearch/

gpgcheck=1

enabled=1

gpgkey=https://www.mongodb.org/static/pgp/server-3.4.asc

repo files for each release can also be found [in the repository itself](https://repo.mongodb.com/yum/redhat/). Remember that odd-numbered minor release versions (e.g. 2.5) are development versions and are unsuitable for production deployment.

**Step 2 - Install the MongoDB Enterprise packages and associated tools.**

To install the latest stable version of MongoDB Enterprise, issue the following command:

sudo yum install -y mongodb-enterprise

**Step 3 - When the install completes, you can run MongoDB.**

**Install MongoDB Enterprise on Windows**

**Prerequisites**

MongoDB Enterprise Server for Windows requires Windows Server 2008 R2 or later. The .msi installer includes all other software dependencies and will automatically upgrade any older version of MongoDB installed using an .msi file.

**Get MongoDB Enterprise**

**Download MongoDB Enterprise for Windows.**

Download the latest production release of [MongoDB Enterprise](http://www.mongodb.com/products/mongodb-enterprise?jmp=docs).

To find which version of Windows you are running, enter the following commands in the Command Prompt or Powershell:

wmic os get caption

wmic os get osarchitecture

**Install MongoDB Enterprise**

**Interactive Installation of MongoDB Enterprise for Windows**

In Windows Explorer, locate the downloaded MongoDB .msi file, which typically is located in the default Downloads folder. Double-click the .msi file. A set of screens will appear to guide you through the installation process.

You may specify an installation directory if you choose the “Custom” installation option.

**NOTE**

These instructions assume that you have installed MongoDB to C:\ProgramFiles\MongoDB\Server\3.2\.

MongoDB is self-contained and does not have any other system dependencies. You can run MongoDB from any folder you choose. You may install MongoDB in any folder (e.g. D:\test\mongodb).

**Unattended Installation of MongoDB Enterprise for Windows**

You may install MongoDB unattended on Windows from the command line using msiexec.exe.Change to the directory containing the .msi installation binary of your choice and invoke:

msiexec.exe /q /i mongodb-win32-x86\_64-2008plus-ssl-3.4.4-signed.msi ^

INSTALLLOCATION="C:\Program Files\MongoDB\Server\3.4.4\" ^

ADDLOCAL="all"

You can specify the installation location for the executable by modifying the INSTALLLOCATION value.By default, this method installs all MongoDB binaries. To install specific MongoDB component sets, you can specify them in the ADDLOCAL argument using a comma-separated list including one or more of the following component sets:

| Component Set | Binaries |
| --- | --- |
| Server | mongod.exe |
| Router | mongos.exe |
| Client | mongo.exe |
| MonitoringTools | mongostat.exe, mongotop.exe |
| ImportExportTools | mongodump.exe, mongorestore.exe, mongoexport.exe, mongoimport.exe |
| MiscellaneousTools | bsondump.exe, mongofiles.exe, mongooplog.exe, mongoperf.exe |

For instance, to install only the MongoDB utilities, invoke:

msiexec.exe /q /i mongodb-win32-x86\_64-2008plus-ssl-3.4.4-signed.msi ^

INSTALLLOCATION="C:\Program Files\MongoDB\Server\3.4.4\" ^

ADDLOCAL="MonitoringTools,ImportExportTools,MiscellaneousTools"

**Run MongoDB Enterprise**

**Step1 - Set up the MongoDB environment.**

MongoDB requires a [data directory](https://docs.mongodb.com/manual/reference/glossary/#term-dbpath) to store all data. MongoDB’s default data directory path is the absolute path \data\db on the drive from which you start MongoDB. Create this folder by running the following command in a Command Prompt:

md \data\db

You can specify an alternate path for data files using the --dbpath option to [mongod.exe](https://docs.mongodb.com/manual/reference/program/mongod.exe/#bin.mongod.exe), for example:

"C:\Program Files\MongoDB\Server\3.4\bin\mongod.exe" --dbpath d:\test\mongodb\data

If your path includes spaces, enclose the entire path in double quotes, for example:

"C:\Program Files\MongoDB\Server\3.4\bin\mongod.exe" --dbpath "d:\test\mongo db data"

You may also specify the dbpath in a [configuration file](https://docs.mongodb.com/manual/reference/configuration-options/).

**Step2 - Start MongoDB.**

To start MongoDB, run [mongod.exe](https://docs.mongodb.com/manual/reference/program/mongod.exe/#bin.mongod.exe). For example, from the Command Prompt:

"C:\Program Files\MongoDB\Server\3.4\bin\mongod.exe"

This starts the main MongoDB database process. The waiting for connections message in the console output indicates that the [mongod.exe](https://docs.mongodb.com/manual/reference/program/mongod.exe/#bin.mongod.exe) process is running successfully.

Depending on the security level of your system, Windows may pop up a Security Alert dialog box about blocking “some features” of C:\ProgramFiles\MongoDB\Server\3.4\bin\mongod.exe from communicating on networks. All users should select Private Networks, such as my home or work network and click Allowaccess. For additional information on security and MongoDB, please see the [Security Documentation](https://docs.mongodb.com/manual/security/).

**Step 3 - Connect to MongoDB.**

To connect to MongoDB through the [mongo.exe](https://docs.mongodb.com/manual/reference/program/mongo/#bin.mongo) shell, open another Command Prompt.

"C:\Program Files\MongoDB\Server\3.4\bin\mongo.exe

If you want to develop applications using .NET, see the documentation of [C# and MongoDB](https://docs.mongodb.com/ecosystem/drivers/csharp) for more information.

**Step 4 - Begin using MongoDB.**

To help you start using MongoDB, MongoDB provides [Getting Started Guides](https://docs.mongodb.com/manual/#getting-started) in various driver editions. See [Getting Started](https://docs.mongodb.com/manual/#getting-started) for the available editions.

Before deploying MongoDB in a production environment, consider the [Production Notes](https://docs.mongodb.com/manual/administration/production-notes/) document.

Later, to stop MongoDB, press Control+C in the terminal where the [mongod](https://docs.mongodb.com/manual/reference/program/mongod/#bin.mongod) instance is running.

### GIT Bash or UI client

**Bit Bucket Installation**

1. Download Bitbucket Server

Download the installer - www.atlassian.com/software/bitbucket/download.

1. Run the installer

Run the installer. We recommend using a Windows administrator account.

Follow the prompts to install Bitbucket. You'll be asked for the following info:

Type of Bitbucket instance - the type of installation, for these instructions select Standard.

Installation directory - where Bitbucket will be installed.

Home directory - where Bitbucket application data will be stored.

TCP ports - the HTTP connector port and control port Bitbucket will run on.

Once the installer completes launch Bitbucket in a browser.

Set up Bitbucket

The Setup Wizard runs automatically when you visit Bitbucket Server in your browser the first time.

1. Connect to your database

If you've not already done so, it's time to create your database. See the 'Before you begin' section of this page for details.

Select External as your database, then choose a Database Type from the dropdown menu and enter the details of your database.

If you plan to use MySQL, there's an extra step...

1. Add your license key

Follow the prompts to log in to my.atlassian.com to retrieve your license, or enter a license key.

You can also set the base URL at this step, (you can elect to do this later).

1. Create your administrator account

Enter details for the administrator account.

Click either Go to Bitbucket - to go straight to the Bitbucket Server interface, or Integrate with JIRA - to create your connection with an existing JIRA application.

1. Start using Bitbucket Server

That's it! Your Bitbucket Server site is accessible from a URL like this: http://<computer\_name\_or\_IP\_address>:<port>

## Code Management

### Access to GITHub

**Creating a New Repository from scratch**

Open Git Bash.

Change the current working directory to your local project.

Initialize the local directory as a Git repository.

git init

Add the files in your new local repository. This stages them for the first commit.

git add .

# Adds the files in the local repository and stages them for commit. To unstage a file, use 'git reset HEAD YOUR-FILE'.

Commit the files that you've staged in your local repository.

git commit -m "First commit"

# Commits the tracked changes and prepares them to be pushed to a remote repository. To remove this commit and modify the file, use 'git reset --soft HEAD~1' and commit and add the file again.

Copy remote repository URL field at the top of your GitHub repository's Quick Setup page, click to copy the remote repository URL.

In the Command prompt, add the URL for the remote repository where your local repository will be pushed.

git remote add origin remote <repository URL>

# Sets the new remote

git remote -v

# Verifies the new remote URL

Push the changes in your local repository to GitHub.

git push origin master

# Pushes the changes in your local repository up to the remote repository you specified as the origin

**A new repo from an existing project**

* Go into the directory containing the project.
* Type git init.
* Type git add to add all of the relevant files.
* a .gitignore file is created to indicate all of the files you don’t want to track. Use git add .gitignore, too.
* Type git commit.

**Connect local git repos to github**

* Go to [github](http://github.com/).
* Log in to your account.
* Click the [new repository](https://github.com/new) button in the top-right. You’ll have an option there to initialize the repository with a README file, but I don’t.
* Click the “Create repository” button.

Push an existing local repository

$ git remote add origin https://github.com/username/new\_repo

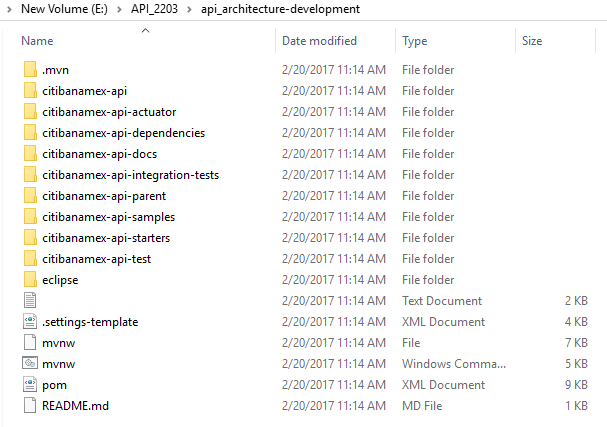
$ git push -u origin master

### Check-out and Check-in guidelines

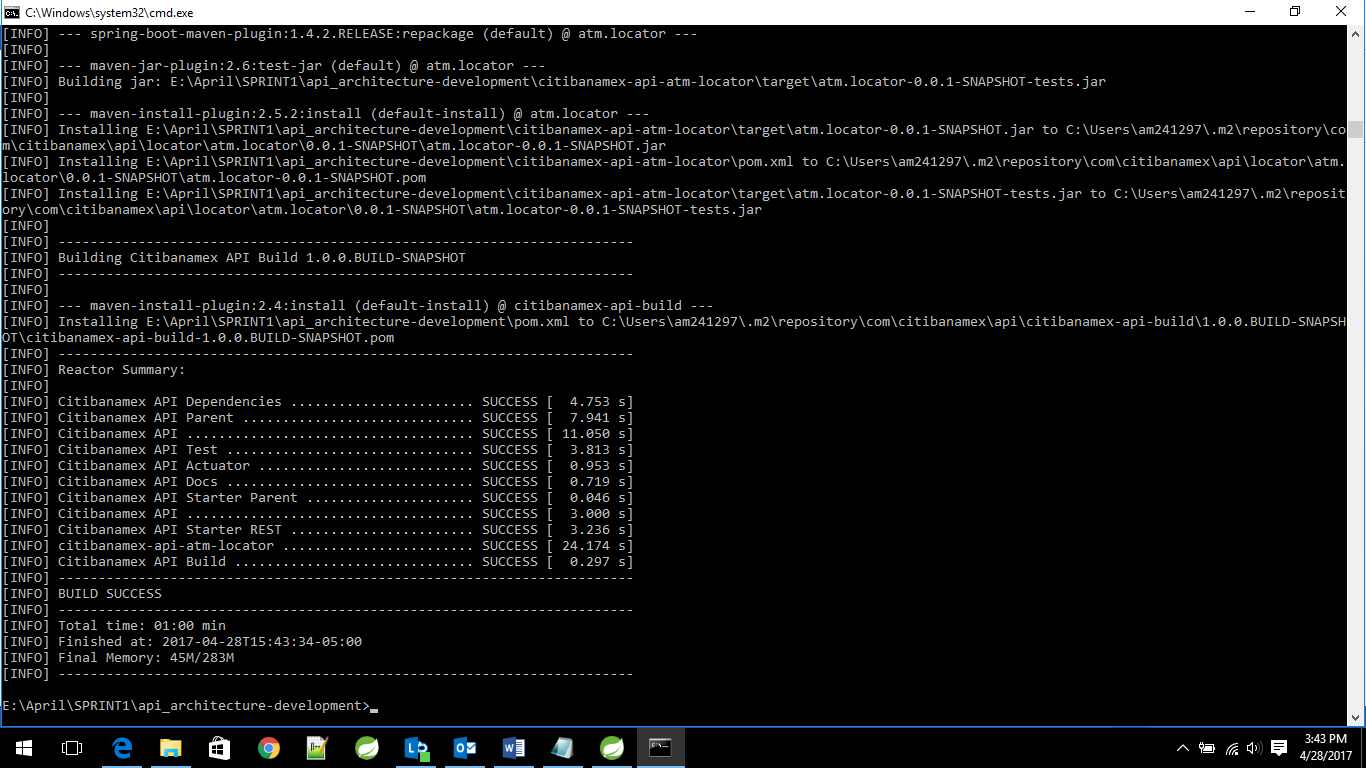
## Project Development

### Download Dependency Projects

* Download “api\_architecture-development” dependency projects from Bit Bucket repository (TBD)
* Your api\_architecture-development folder should contain all the below projects.

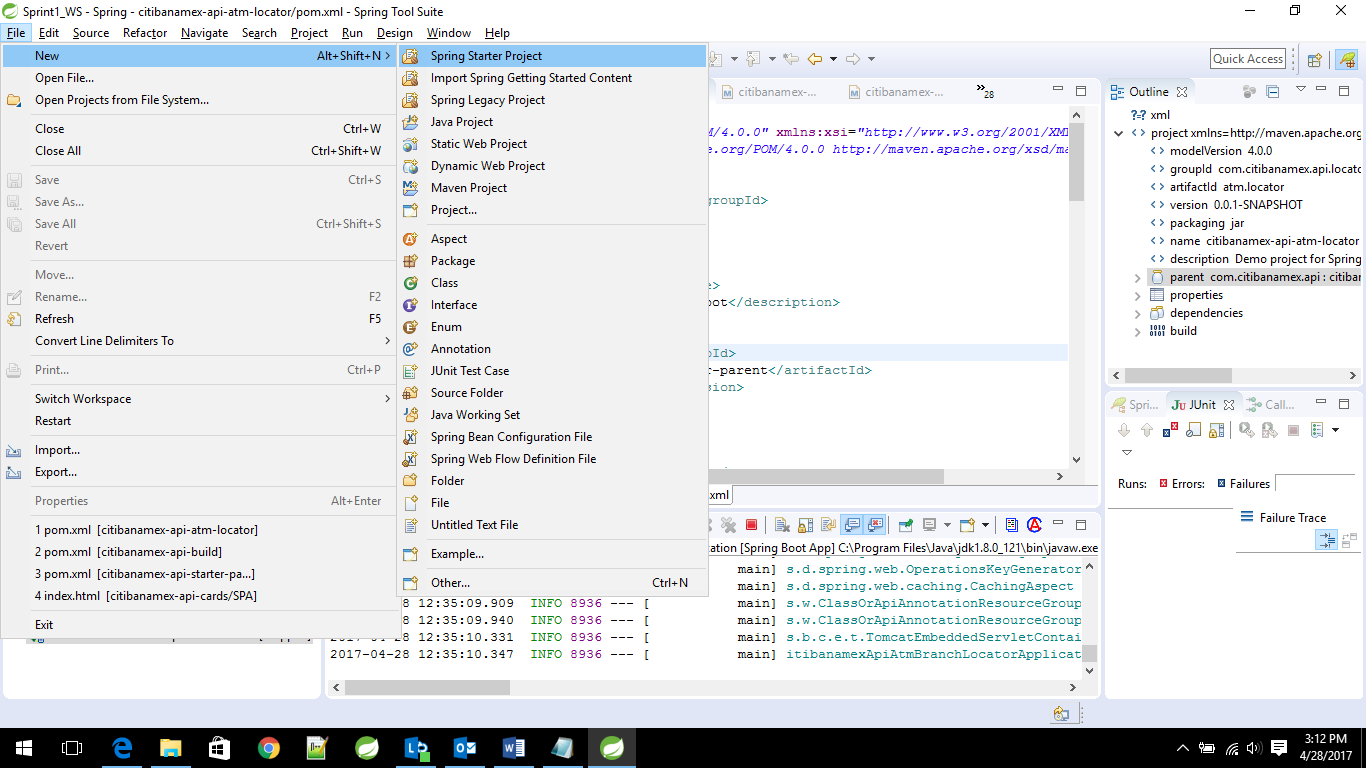


* Open command prompt and run “mvn install” command from root of “api\_architecture-development” folder.
* Run “mvn eclipse:eclipse” to create a .project file.
* Run again “mvn install” and see build successful below.

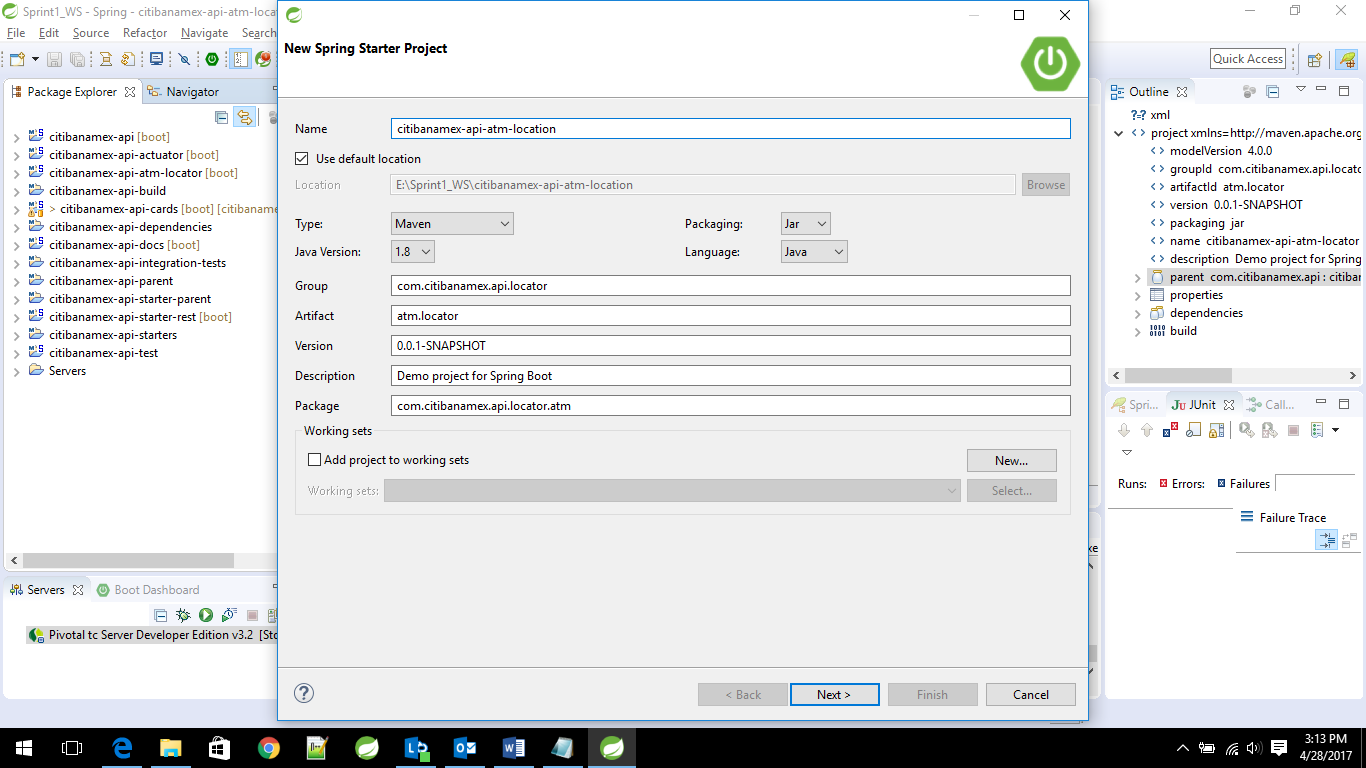


### Create Project Source Code Directory

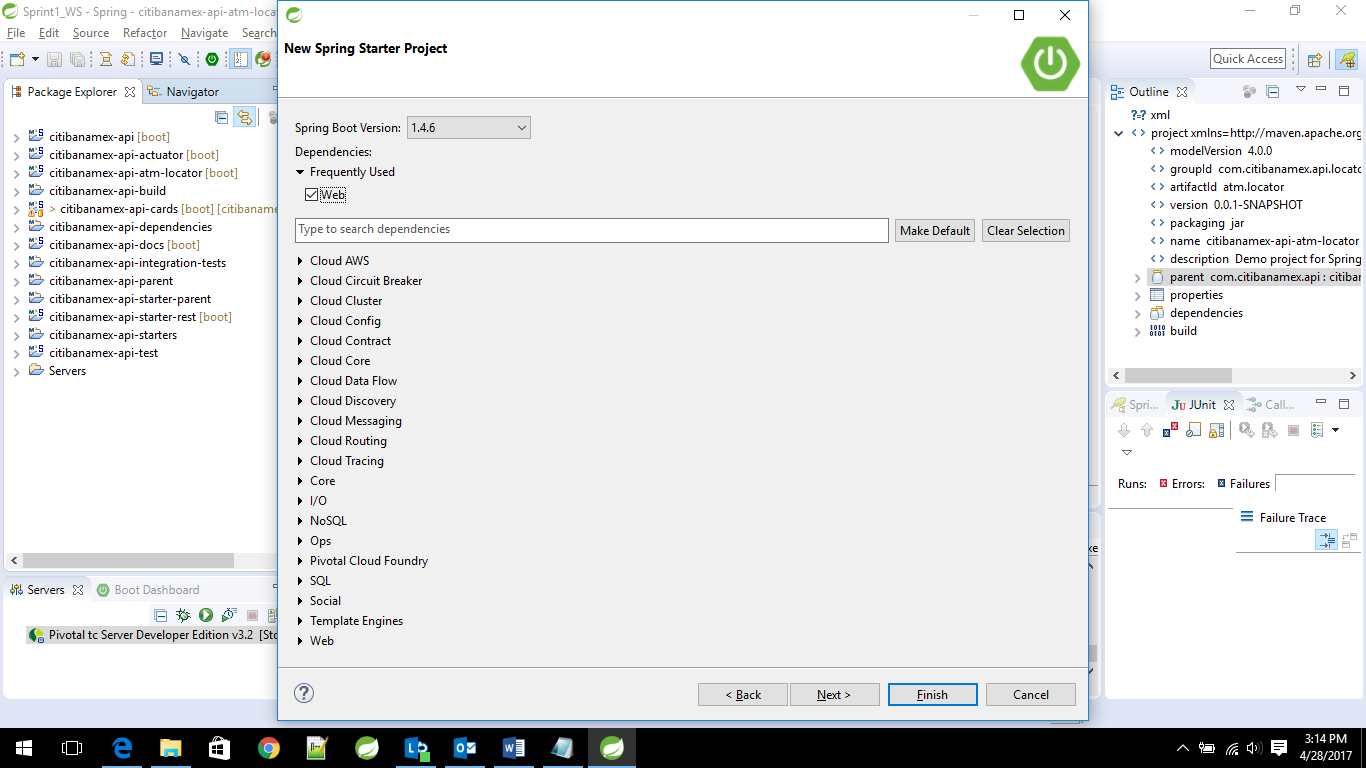
* Now create one Spring Starter project according to your requirement. For example “citibanamex-api-atmlocator” by selecting Maven as a build tool, Java 1.8 as the java runtime environment and 1.4.6 as the Spring boot version.



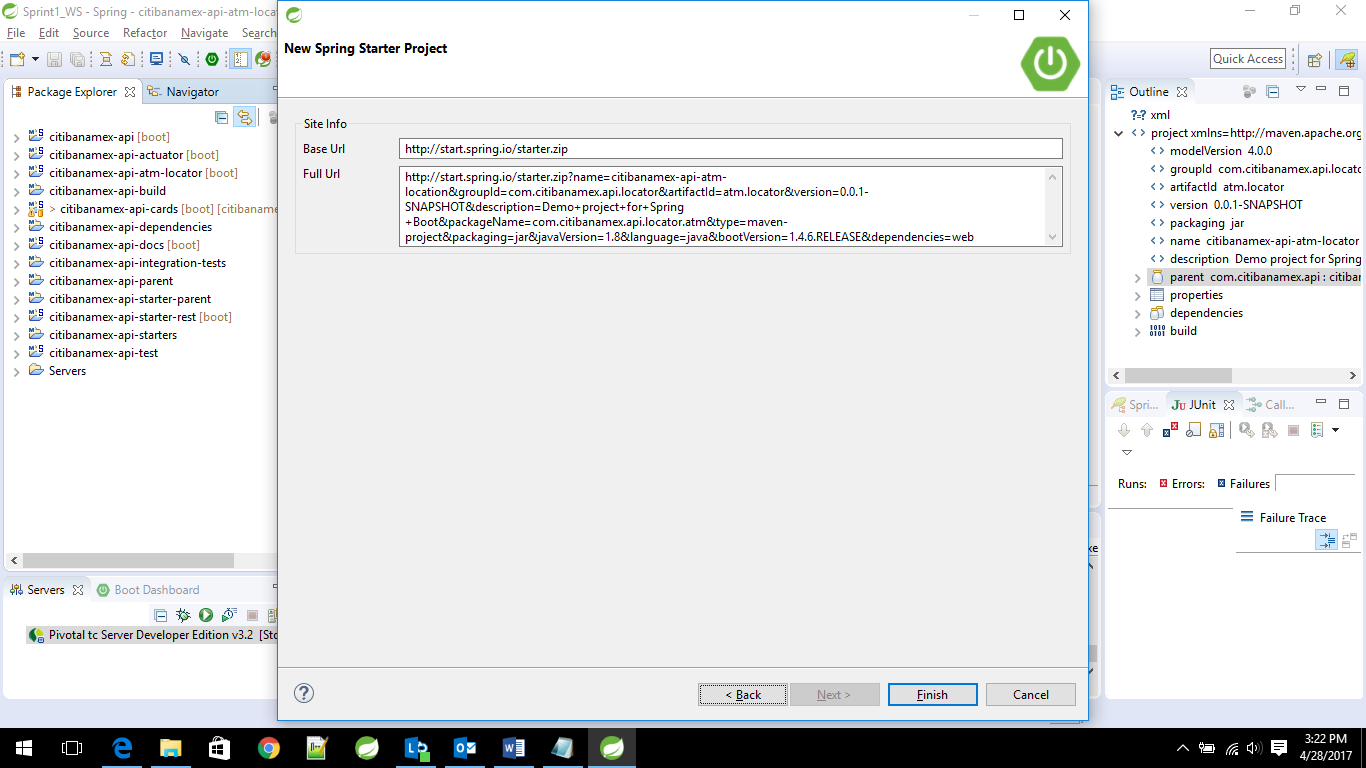
* Update the Group Id, Artifact Id, Description and Package name as per the requirements.



* Select web under dependencies list.



* Click next and then finish.



* It will create your project and download all dependencies.

### Spring boot project setup

Now replace the Spring boot parent tag with “citibanamex-api-starter-parent” as shown below

In your projects pom.xml.

<parent>

<groupId>com.citibanamex.api</groupId>

<artifactId>citibanamex-api-starter-parent</artifactId>

<version>1.0.0.BUILD-SNAPSHOT</version>

<relativePath />

</parent>

Make sure to add the Rest Assured Test dependencies in your project’s pom.xml for writing Test Cases using Rest Assured framework.

<dependency>

<groupId>com.jayway.restassured</groupId>

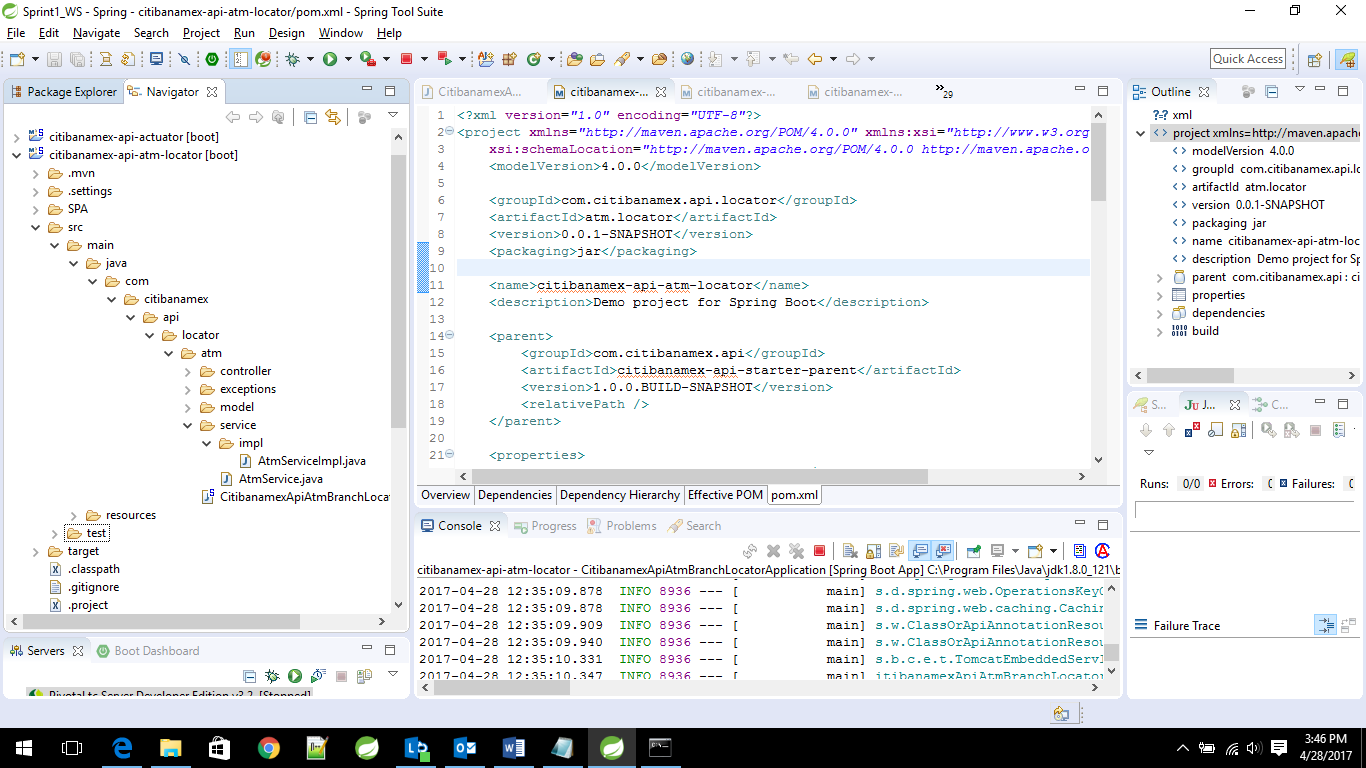
<artifactId>spring-mock-mvc</artifactId>

<version>2.9.0</version>

<scope>test</scope>

</dependency>

* Add your project as a module under citibanamex-api-build project’s pom.xml. For example
* <module>citibanamex-api-atm-locator</module>
* Now run the build for your project using “mvn install” command or go to project clean build your project from STS.
* Once the build is successful, Right click on your project and select run as Spring Boot Application.
* Or Execute the command “mvn spring-boot:run”.
* Your Spring boot application starts successfully. Now create the folder structure like below.



* Create meaningful package structure as per your API requirement and segregate Controller, Model, Service, ServiceImpl and Exception handling folders accordingly.
* Create a RestController and Service Class Skelton for your API.
* Re-launch the application as Spring boot Application.
* Now hit the uri from any browser or from Postman Rest client tool.
* You should be able to see the expected Json response.

### Logging

* **Logging** is the most useful feature for debugging our API Application. We will be using **SL4J** as a logging framework. Get Logger instance using LoggerFactory by passing .class file of your class and then call the appropriate method. For Example as shown below.

private static final Logger log = LoggerFactory.getLogger(AtmController.class);

log.info("\*\*\*\*\*Inside AtmController try block\*\*\*\*\*");

### Swagger Documentation

* **Swagger documentation** is one of the important task to document your APIs. There are multiple ways to document. We can use Online Swagger Editor to write about your API in YAML. Once YAML file is ready we can download Yaml/Json from editor.

<http://editor.swagger.io/#/>

Below attachment shows the demo to document your API details using Swagger api.

### Editing project setup and configurations

### TDD development guide

Test-driven development (TDD), is an evolutionary approach to development which combines test-first development where you write a test before you write just enough production code to fulfill that test and refactoring.

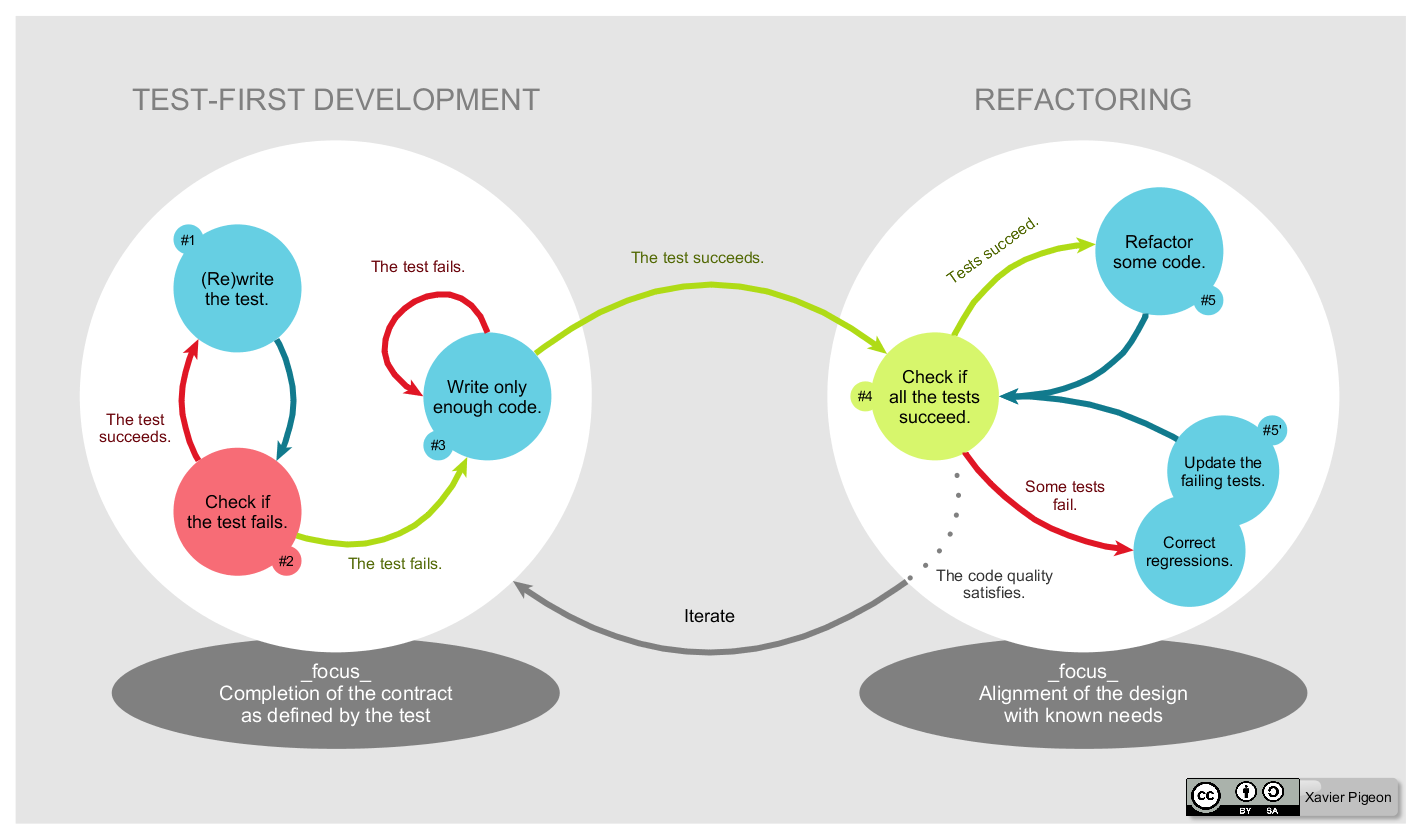
TDD is a programming technique in which one must think through the requirements or design before writing functional code. TDD is both an important agile requirements and agile design technique.

Test-driven development (TDD), also called test-driven design, is a method of [software](http://searchsoa.techtarget.com/definition/software) development in which [unit testing](http://searchsoftwarequality.techtarget.com/definition/unit-testing) is repeatedly done on [source code](http://searchsoa.techtarget.com/definition/source-code). The concept is to "get something working now and perfect it later." After each test, [refactoring](http://searchsoa.techtarget.com/definition/refactoring) is done and then the same or a similar test is performed again. The process is iterated as many times as necessary until each unit is functioning according to the desired specifications. Test-driven development is part of a larger software design paradigm known as [Extreme Programming](http://searchsoftwarequality.techtarget.com/definition/Extreme-Programming) (XP).

Test-driven development can produce [application](http://searchsoftwarequality.techtarget.com/definition/application)s of high quality in less time than is possible with older methods. Proper implementation of TDD requires the developers and testers to accurately anticipate how the application and its features will be used in the real world. Problems are approached in an incremental fashion and tests intended for the same unit of code must often be done many times over. The methodical nature of TDD ensures that all the units in an application have been tested for optimum functionality, both individually and in synergy with one another. Because tests are conducted from the very beginning of the design cycle, time and money spent in [debugging](http://searchsoftwarequality.techtarget.com/definition/debugging) at later stages is minimized.

One of the chief limitations of TDD is the fact that tests can sometimes be incorrectly conceived or applied. This may result in units that do not perform as expected in the real world. Even if all the units work perfectly in isolation and in all anticipated scenarios, end users may encounter situations not imagined by the developers and testers. The final results of TDD are only as good as the tests that have been used, the thoroughness with which they have been done and the extent to which they mimic conditions encountered by users of the final product.

**TDD Cycle**



Creative commons - Lifecycle of the Test-driven development method by Xavier Pigeon

**#1 Write the test**

In test-driven development, each new feature begins with writing a test. Write a test that defines a function or improvements of a function, which should be very succinct. To write a test, the developer must clearly understand the feature's specification and requirements. The developer can accomplish this through use cases and user stories to cover the requirements and exception conditions, and can write the test in whatever testing framework is appropriate to the software environment. It could be a modified version of an existing test. This is a differentiating feature of test-driven development versus writing unit tests after the code is written: it makes the developer focus on the requirements before writing the code, a subtle but important difference.

**#2 Check if the test fails**

This validates that the [test harness](https://en.wikipedia.org/wiki/Test_harness) is working correctly, shows that the new test does not pass without requiring new code because the required behavior already exists, and it rules out the possibility that the new test is flawed and will always pass. The new test should fail for the expected reason. This step increases the developer's confidence in the new test.

**#3 Write only enough code**

The next step is to write some code that causes the test to pass. The new code written at this stage is not perfect and may, for example, pass the test in an inelegant way. That is acceptable because it will be improved and honed in Step 5.

At this point, the only purpose of the written code is to pass the test. The programmer must not write code that is beyond the functionality that the test checks.

**#4 Check if all the tests succeed**

If all test cases now pass, the programmer can be confident that the new code meets the test requirements, and does not break or degrade any existing features. If they do not, the new code must be adjusted until they do.

**#5 Refactor some code**

The growing code base must be cleaned up regularly during test-driven development. New code can be moved from where it was convenient for passing a test to where it more logically belongs. Duplication must be removed. Object, class, module, variable and method names should clearly represent their current purpose and use, as extra functionality is added. As features are added, method bodies can get longer and other objects larger. They benefit from being split and their parts carefully named to improve readability and maintainability, which will be increasingly valuable later in the software lifecycle. Inheritance hierarchies may be rearranged to be more logical and helpful, and perhaps to benefit from recognized design patterns. There are specific and general guidelines for refactoring and for creating clean code. By continually re-running the test cases throughout each refactoring phase, the developer can be confident that process is not altering any existing functionality.

The concept of removing duplication is an important aspect of any software design. In this case, however, it also applies to the removal of any duplication between the test code and the production code—for example magic numbers or strings repeated in both to make the test pass in Step 3.

**#6 Repeat (not in diagram)**

Starting with another new test, the cycle is then repeated to push forward the functionality. The size of the steps should always be small, with as few as 1 to 10 edits between each test run. If new code does not rapidly satisfy a new test, or other tests fail unexpectedly, the programmer should [undo](https://en.wikipedia.org/wiki/Undo) or revert in preference to excessive [debugging](https://en.wikipedia.org/wiki/Debugging). [Continuous integration](https://en.wikipedia.org/wiki/Continuous_integration) helps by providing revertible checkpoints. When using external libraries it is important not to make increments that are so small as to be effectively merely testing the library itself,[[4]](https://en.wikipedia.org/wiki/Test-driven_development#cite_note-Newkirk-4) unless there is some reason to believe that the library is buggy or is not sufficiently feature-complete to serve all the needs of the software under development.

### Local Build and Unit test

**Local Build**

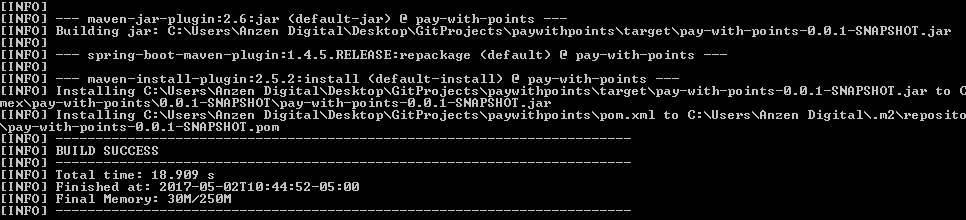
You can build your springboot project either from your IDE or by using command prompt. Here we will show you the second way.

Enter to the path of your project and type the following commands to build the jar file of your application.

mvn clean

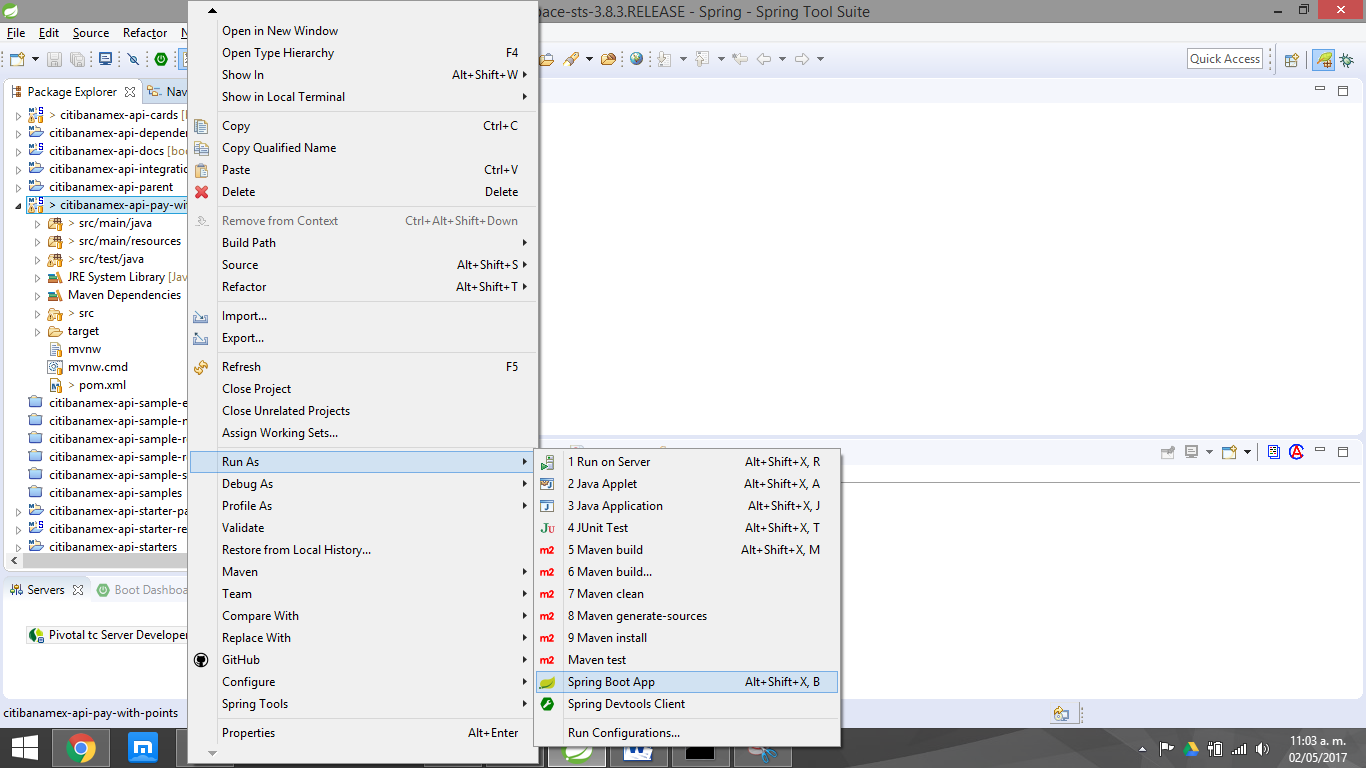
mvn install

If everything is going right with your project, you should see a build success on your cmd.



You can verify the existence of your jar file application by entering to the target folder inside your project folder.

Now that your jar file is built you can launch your application as a spring boot application from your IDE or from command prompt.



To do it from command prompt, get into your folder project and type:

mvn spring-boot: run

**Unit Testing**

To Test the Rest Services we focused on unit and integration testing by using the spring-test module. Spring Boot provides a starter POM named spring-boot-starter-test that automatically adds the spring-test module to a Boot application. Additionally, the starter POM brings in JUnit, Mockito, and Hamcrest libraries.

Unit Testing with Spring MVC Test framework

Central to the test framework is the org.springframework.test.web.servlet.MockMvc class, which can we used to perform HTTP requests. It contains only one method named perform and has the following API signature:

public ResultActions perform(RequestBuilder requestBuilder) throws java.lang.Exception

An example of a unit test is shown below.



To run the test you only have to run it as a JUnit and you will see a http response in the console.

The Spring MVC Test framework provides a lightweight alternative for integration testing MVC applications. In this approach, the entire Spring application context along with the DispatcherServlet and associated MVC infrastructure gets loaded. A mocked MVC container is made available to receive and execute HTTP requests. We interact with real controllers and these controllers work with real collaborators.

The following is an example of an Integration Test class. Note the use of WebApplicationContext dependency, this allows to launch the whole context of your application so your tests will drive through all the layers present in your application when calling an endpoint of your API.



To run the test you only have to run it as a JUnit Test and you will see the http response in the console and green status in JUnit window.

Once your are done with all the unit tests, it is advisable to add the @Ignored annotation to your tests, so they do not get called when you are building the application.

**Code Review and Check-in**

First start with Self Review the code as per the attached Citibanamex API standards document and then send for Peer review.

Implement the Peer review comments and send for ERT.

Implement ERT review comments if any and then push the code to Bit Bucket.

Make sure to maintain all review comments documents with status of comments.

## Useful scripts/snippets

## Troubleshooting Notes and Experiences

Make sure to enable 'access-control-allow-origin' CORS when you are trying to access remote urls. CORS (Cross Origin resource sharing) is a W3C specification supported by most modern browsers to specify when it is safe to allow cross origin requests. In Spring boot, enabling CORS is as easy as adding the @CrossOrigin annotation.

Below is an example to enable CORS for example.com domain and port 80 at controller level

