GOVERNMENT OF KARNATAKA

DEPARTMENT OF COLLEGIATE AND TECHNICAL EDUCATION

GOVERNMENT POLYTECHNIC TUMKUR - 572103

FULL STACK DEVELOPMENT (20CS52I)

A

Lab Manual

Submitted

By

5th Semester

DEPT. OF COMPUTER SCIENCE AND ENGINEERING

UNDER THE GUIDANCE OF

THARA B S,

(COHORT OWNER)

Dept. of Computer Science and Engineering (2022-23)

Signature of course coordinator **signature of program coordinator**

FULL STACK DEVELOPMENT

List of Experiments :-

Sl.no	Experiments
1	Create User Stories ,Product backlog, workflow and start sprint using appropriate tool like Jira
2.	Creation of local Repository on Git and push it into Remote Repository and Working with Git Commands
3.	Setting up React environment and Create and running a React.js app
4.	Setting Up the Environment and Tools for front end development Installing VS Code
5.	Create a form like registration form or feedback form, after submit hide create form and enable the display section using java script.
6.	Create and run simple program in TypeScript
7.	Forms - Use of HTML tags in forms like select, input, file, textarea, etc
8.	Create and run a simple Maven project
9.	Create a Spring Boot Project in Eclipse IDE
10.	Create and Run Servlet class using Tomcat server
.11	Initialize Spring boot Project
12.	Implement navigation using react router
13.	Build single page application(Shopping cart)
14.	Download and Install MongoDB and perform CRUD operation on database
15.	Test Web Application using appropriate automated testing tool like Selenium IDE

Experiment – 1

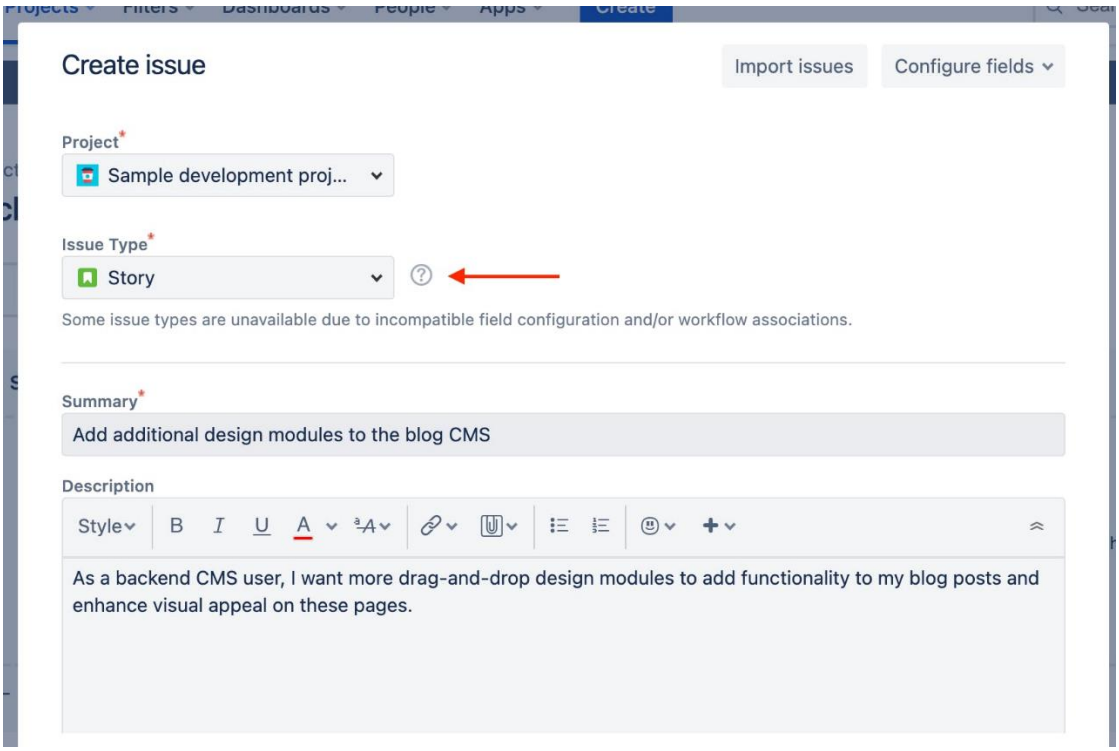
Create User Stories ,Product backlog, workflow and start sprint using appropriate tool like Jira

User Story :-
A user story is defined by Atlassian as being an “informal, general explanation of a software feature written from the perspective of the end user.”

The key here is the term “end user”. A user story should reflect the needs and wants of an individual - who could be a system user, internal team member or customer - and show how the requested functionality will deliver value to this end user.

How to create a user story in Jira
To add a story to your backlog:

- Navigate to the ‘Create’ screen in your Jira Scrum or Kanban project
- Create a new Jira issue and select the appropriate project from the dropdown menu.
- Make sure ‘Story’ is selected as the issue type. This will be the default setting.
- Follow your organisation’s guidelines for formatting your story. Don't worry if you have no guidelines to follow - we give specifics below!.
- Your story will then go into the backlog to be assigned and actioned by the project manager, project leader, product owner or other relevant stakeholders.



A user story will look something like this:

As A [type of user] and I want to do [activity/function] so that I can [achieve specific goal].

It's important to include these details in your story for several reasons:

The user helps the development team understand who they're completing work for and what types of requirements they might have, so they can better tailor work towards this specific user

Here, Some of the Examples of User stories :-

1. As a Sales Professional, I want to generate reports so that I can take a decision on the marketing strategy for the upcoming quarter
2. As a Banking Customer, I want to access net banking, so that I can access my account and make transactions
3. As an Administrator of the software, I want to access master records so that I can make changes to customer data
4. As a Banking Customer, I want to access Savings account so that I can view/make transactions

5. As a Banking Customer, I want to access Credit Card page, so that I can view and make transactions
6. As a Banking Customer, I want to access Loans page so that I can view my loans
7. As a Banking Customer, I want to transfer funds, so that I can move my funds to different accounts within my bank and other banks

Epic in Jira :-

An epic is a large piece of work that can be broken down into several smaller pieces, often called Issues in Jira. Epics can encompass multiple teams working on multiple projects, can be tracked on various boards and are generally employed via a set of sprints.

Epics tend to be focused on agile teams and projects, where efficiency is key. Think of epics as high-level business requirements that are too complicated and large to be delivered over a single sprint. Epics can be a great way to better manage issues for much larger projects, and I'm going to show you how to create one.

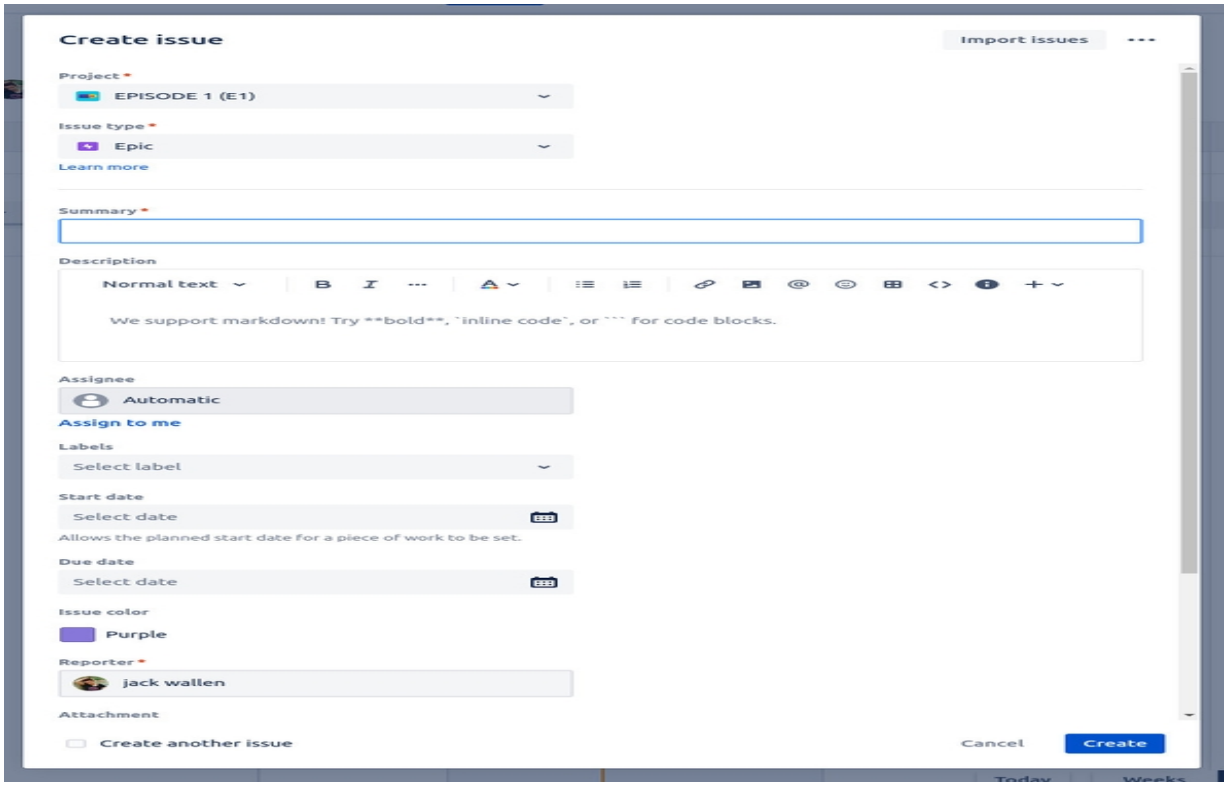
What you'll need to create an epic in Jira

To create your first epic, you'll need a Jira account with the Roadmap enabled. Make sure to log in to your Jira account and either create a new development project for testing or open a previous project that includes a Sprint.

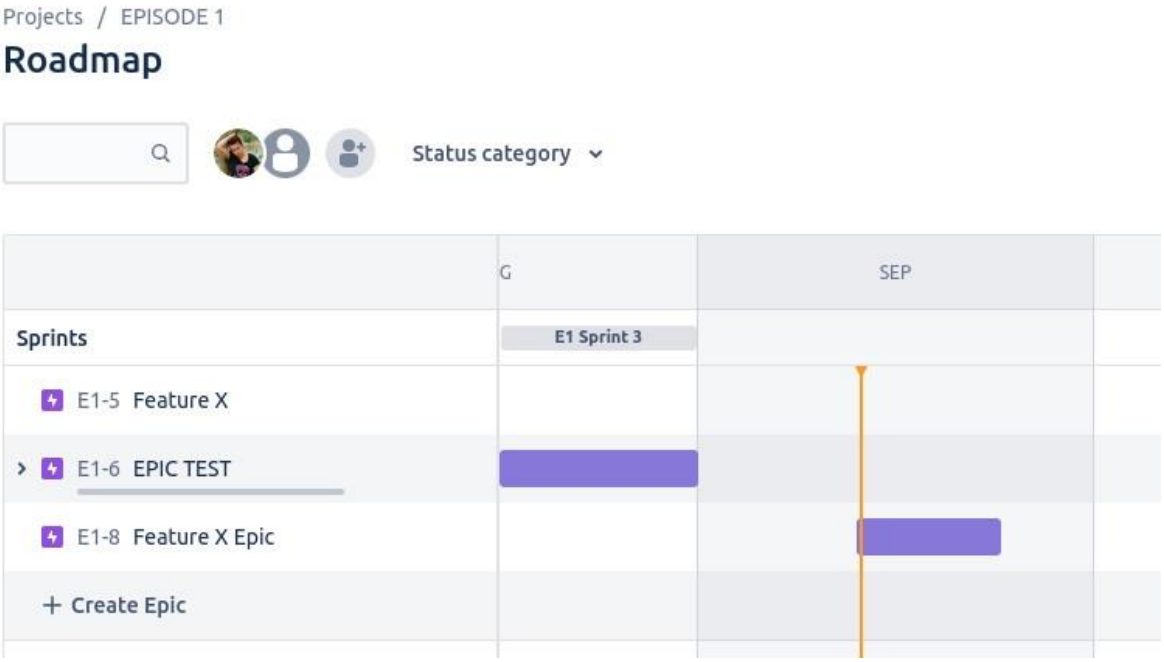
How to create your first epic

Once you've logged in to your Jira account,

- Navigate to the project in question and click onCreate button in the top toolbar. In the resulting window ,make sure to select Epic as the Issue Type and then fill out the rest of the information.



Once you've created the epic, it will appear in your Roadmap

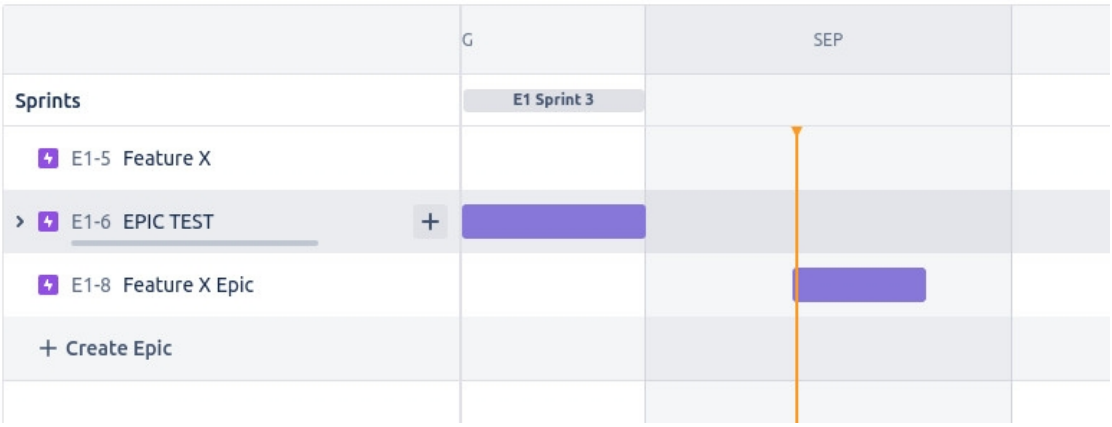


- After you've created the epic, the next step is to add child issues. Here's how:

Stay in the Roadmap view.

- Hover your cursor over the epic and, when the + appears, click it .

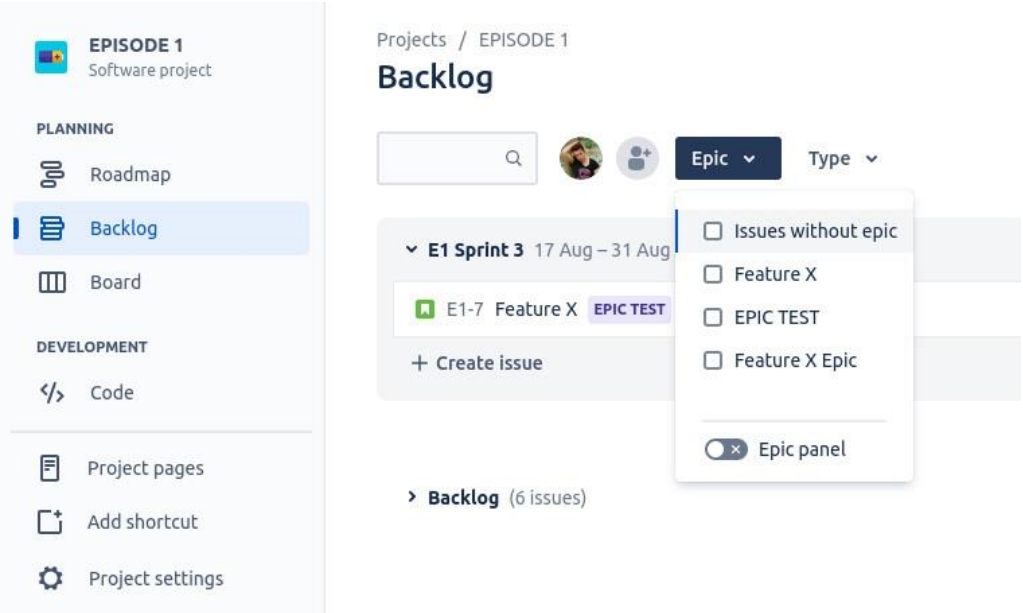
- Type the name of the child issue and hit enter on your keyboard.
- Keep adding child issues until you're satisfied.



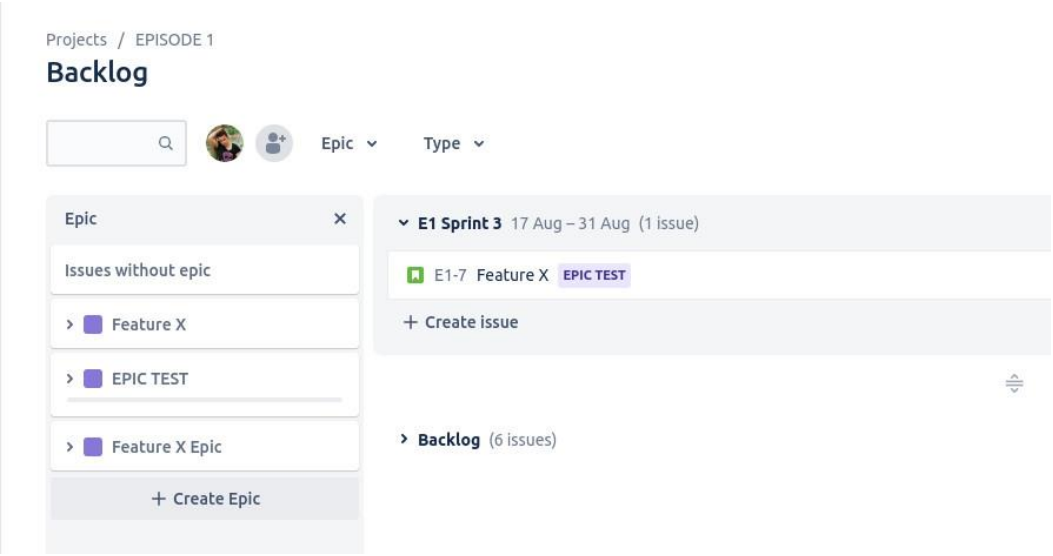
How to enable the Epic panel in Backlogs

Now that you've created your first epic, you might want to enable the Epic panel in the Backlogs view. To do this,

- Click Backlog in the left navigation and then click the Epic drop-down to the right of your profile image.
- From the drop-down, click the Epic Panel On/Off switch until it's in the On position.



With the Epic Panel created ,you can now create epics much more quickly.



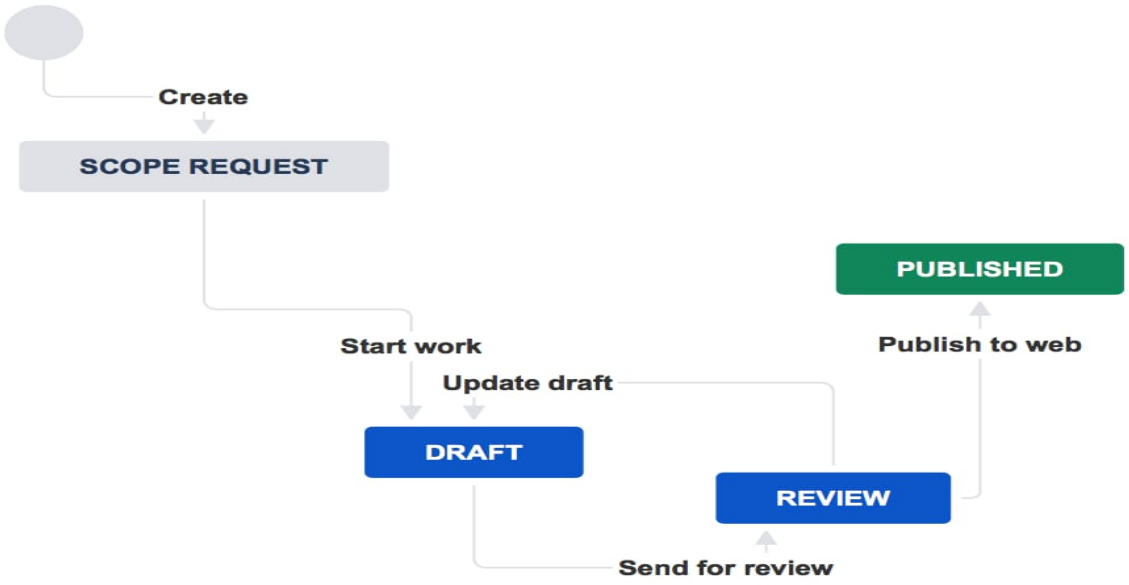
From the Epic Panel, you can click an Epic and then create associated child issues. As you create each child issue, you'll notice that they are automatically tagged with the epic you've selected .

Workflow in Jira :-

A workflow maps out the steps and statuses that a task can go through and defines your process.

If your team does work that's more complicated than "to do, in progress, done", then you'll need to customize the way your typical processes are set up. You can edit the overall workflow used in a project, or modify the way particular task types are handled in the workflow.

Here's an example workflow for making website updates:



Key workflow components

Jira Work Management workflows are made up of several key components:

- Status: current state

- Transitions: change events

Transitions are the actions that move a task between statuses. For example, 'Sent to review'.

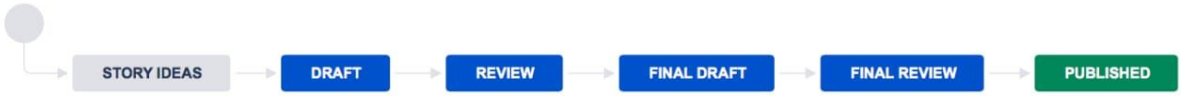
- Resolution: final state

When a task is finalised, and no longer open, it needs a resolution status. The resolution closes the issue and represents the final state of the piece of work. For example, "done", "published", or "rejected".

Keep workflows simple

Jira Work Management workflows are designed to make your work processes easier, so make sure you keep them as simple as you can. Here's an example of a simple editorial process:

Workflow with story ideas, draft, review, final draft, final review, and published statuses.



In the example above, a piece of work can only move forward in the process. However, we can simplify it a lot further by creating a review loop, instead of separate review statuses.

Update a project's workflow

If your project has a workflow that's pretty close to what you need, you can update it by adding statuses and transitions to suit your needs.

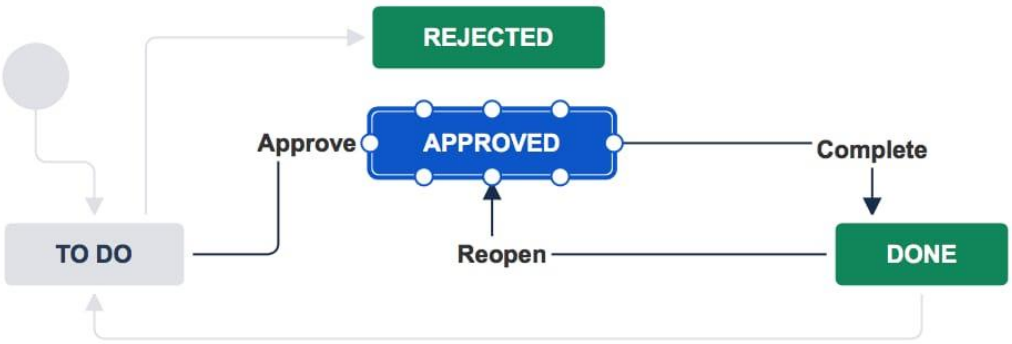
To update the existing project workflow:

- Choose Projects and select a starred or recent project, or choose View all projects and select a project.
- From your project's sidebar, select Project settings > Workflows.
- Click the edit pencil to the right of the workflow.
- Do either or both of the following:

- Click Add status to add a new step to the workflow (you can add existing statuses or create new ones)
- Click Add transition or drag from a node on one status to another to add a transition

➢ Click Publish Draft.
When you publish your new workflow, Jira will ask you if you'd like to save a copy of the original. If you're not sure about your changes, or you just like to be cautious, save a copy of the original workflow so you can re-apply it to your project should you need to.

Here's a shot of the workflow editor in "diagram" mode. It lets you add statuses and draw transitions between them. Here we've added an approval stage and a Rejected resolution for those issues that aren't approved.



In this example, you need to approve an issue (it'll then transition it to the Approved status) before you can complete it (transition it to Done). From Done, if the issue needs more work, you can either reopen it (transition it to Approved again) or send it to be reassessed for approval (transition it to the To Do status).

Create a new workflow

If there isn't an existing workflow to meet your needs, you can create a new one then use it with your project.

1) Go to Settings () > Issues
2) Choose Workflows and click Add workflow.
3) Do both of the following until you've built your workflow:
    - Click Add status to add a new step to the workflow (you can add existing statuses or create new ones)

- Click Add transition or drag from a node on one status to another to add a transition

4) Click Publish Draft.
5) Select the new workflow to use with your project:

Product Backlog :-

The product backlog is an ordered list of requirements that is maintained for a product. It consists of features, bug fixes, non-functional requirements, etc.—whatever needs to be done in order to successfully deliver a viable product. The product backlog items (PBIs) are ordered by the Product Owner based on considerations like risk, business value, dependencies, date needed, etc.

Items added to a backlog are commonly written in story format. The product backlog is what will be delivered, ordered into the sequence in which it should be delivered. It is open and editable by anyone, but the Product Owner is ultimately responsible for ordering the items on the backlog for the Developer to choose.

Example :-

Let us look at a small list of items that could be part of a Product Backlog. Each item in the backlog is essentially a short description of a requirement to meet the need of the customer or add value to customer.

Let us consider a ticket booking system project

Sl.no	Description	Size
1.	Utility to display all Unpaid tickets	15
2.	Utility to display all Cancellations	10
3.	Fix bug in display of Paid Tickets	5
4.	Implement indexing in the database to improve data retrieval time	20
5.	Write User Manual for ticket booking system	10

6.	Update Marketing brochure to display new logo of the booking system	3
7.	Upgrade document management system from Adobe flash 8 to Adobe flash	10

Sprint planning :-

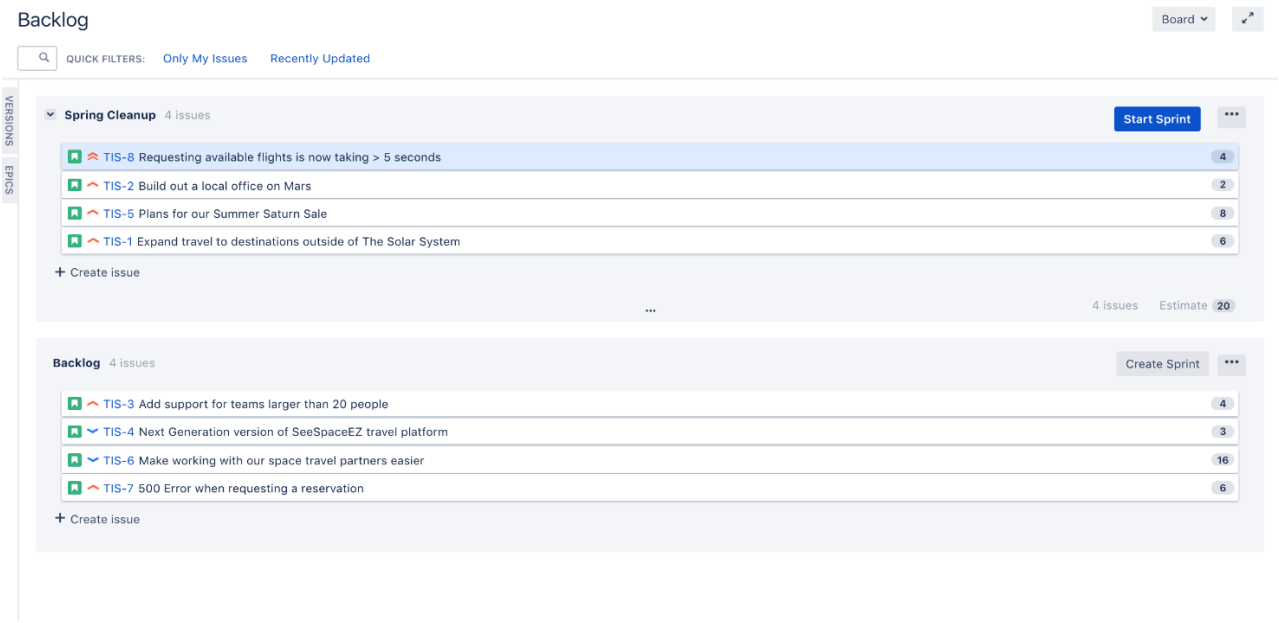
Before creating and starting a sprint, your Scrum team would typically hold a sprint planning meeting. In this meeting, your team should:

- Review the estimates for selected issues
- Break down the selected issues into an initial list of sprint tasks
- Consider upcoming employee time-off, holidays, and other issues that may impact the completion of these sprint tasks
- Gauge the team's capacity team to complete these sprint tasks

By the end of the meeting, your team should be confident enough to commit to completing the work in the sprint.

Create a sprint :-

1. On the Teams in Space board, click **Backlog**.
2. Click **Create Sprint** at the top of the Backlog.
3. Your new upcoming sprint will be added to your board, below any other future sprints. Select the **Sprint 1** text and edit the name of the sprint to 'Spring Cleanup'.
4. The top 4 issues in the Backlog are equal to 20 story points. This is what the team estimated that they could accomplish in the upcoming sprint. Drag and drop the top four issues from the Backlog into your new sprint.



Start your sprint :-

Now that you have created a sprint, you can go ahead and start it.

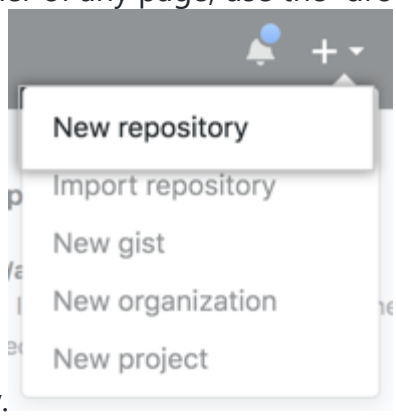
1. Click **Start Sprint**.
2. Today's date and current time become the start date and time for the sprint. For the purpose of this tutorial, enter an end date of 5 minutes from the start date and time.
3. Select **Start** to start the sprint and move the issues into Active sprints.]

Experiment – 2

Creation of local Repository on Git and push it into Remote Repository and Working with Git Commands

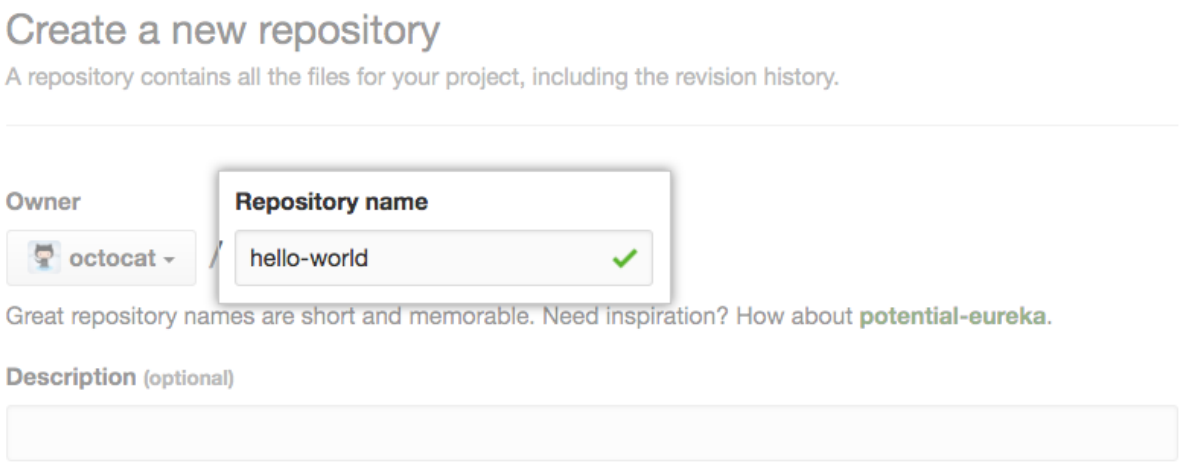
Step 1: Create a remote repo on Github

1. In the upper-right corner of any page, use the drop-down menu, and

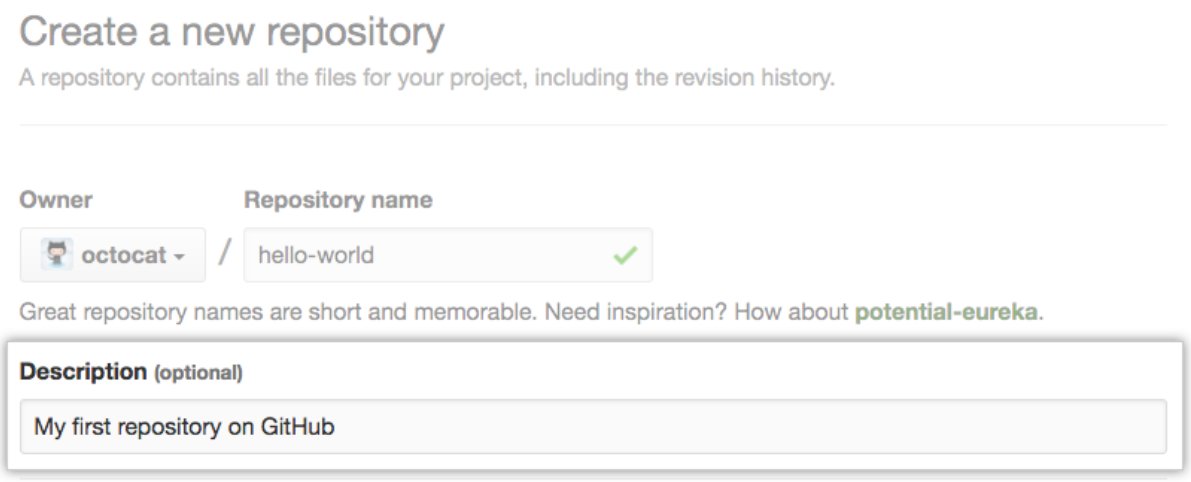


select **New repository**.

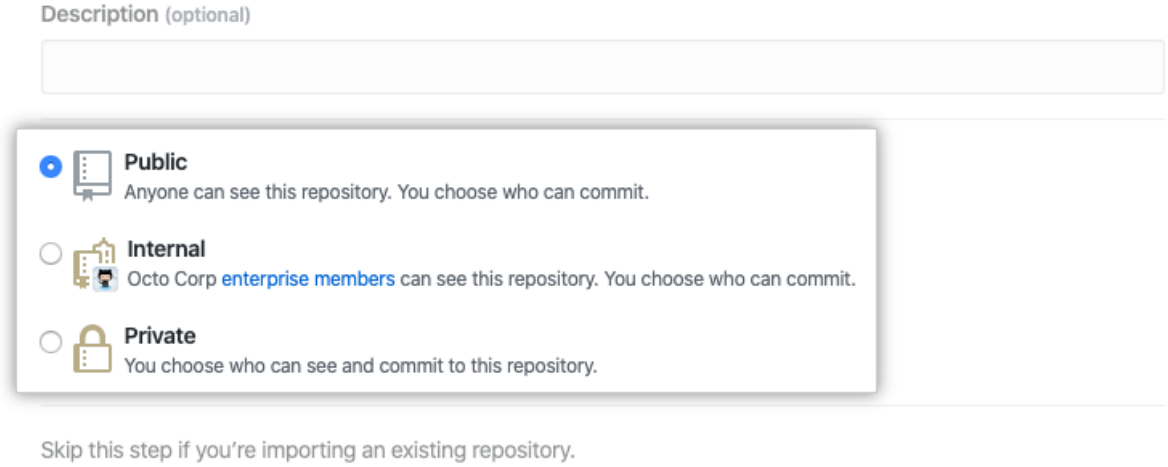
2. Type a short, memorable name for your repository. For example, "hello-world".



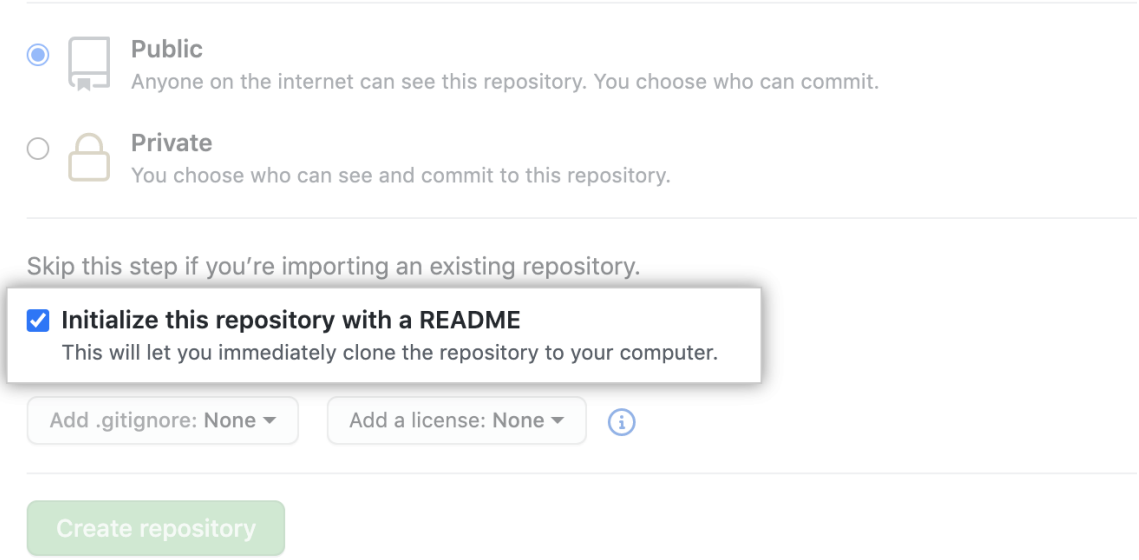
3. Optionally, add a description of your repository. For example, "My first repository on GitHub."



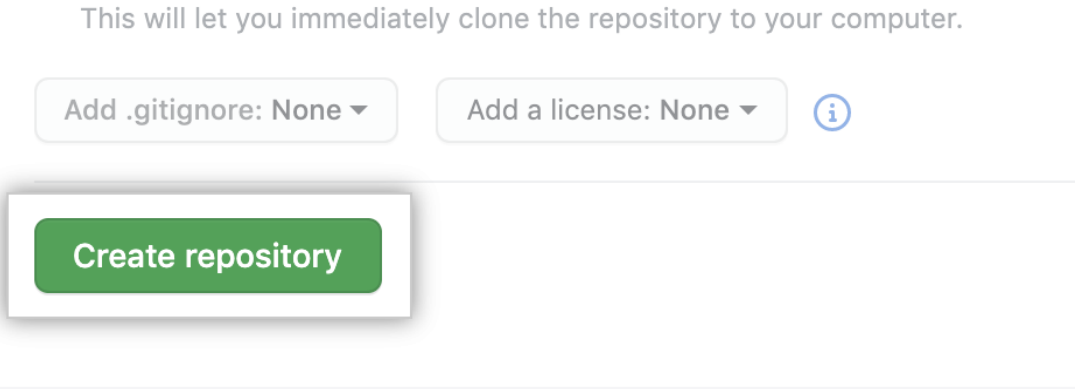
4. Choose a repository visibility. For more information, see "About repositories."



5. Select **Initialize this repository with a README**.



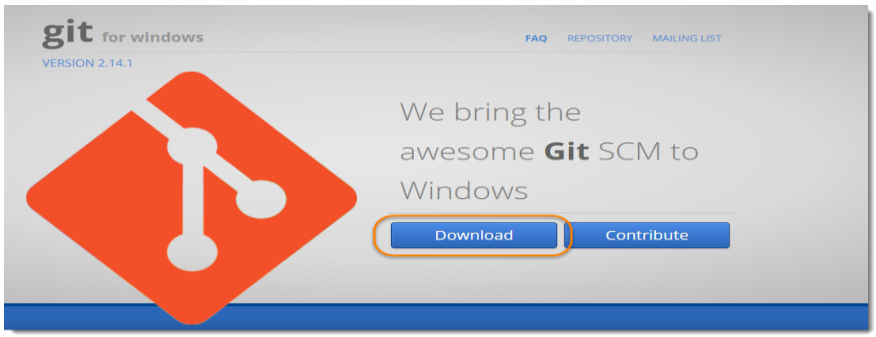
6. Click **Create repository**.



Congratulations! You've successfully created your first repository, and initialized it with a *README* file.

Step 2: Install Git on Windows

1. Download the latest *Git for Windows*.

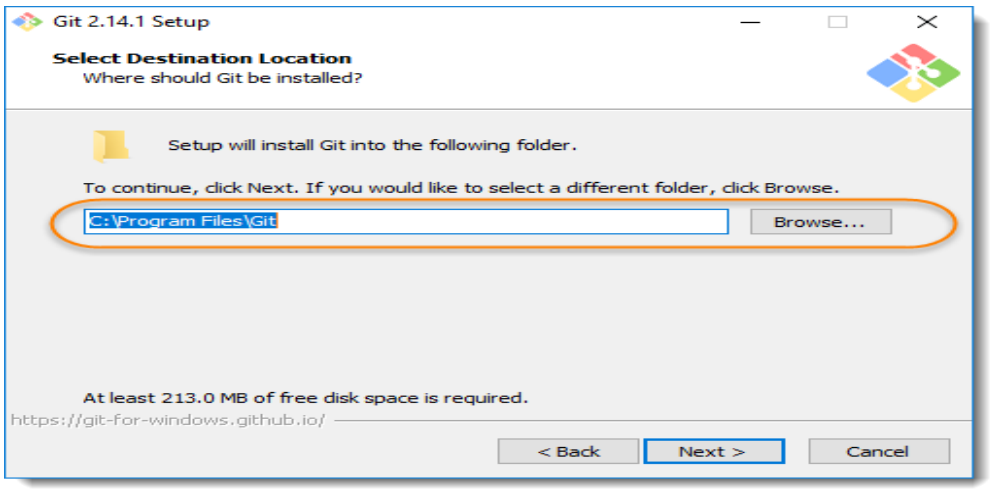


2. Go to the folder where new downloads gets store, at my machine by default folder is *Download* folder. *Double click* on the installer. The installer gets save on the machine as per the Windows OS configuration. My machine is *64 bits*.

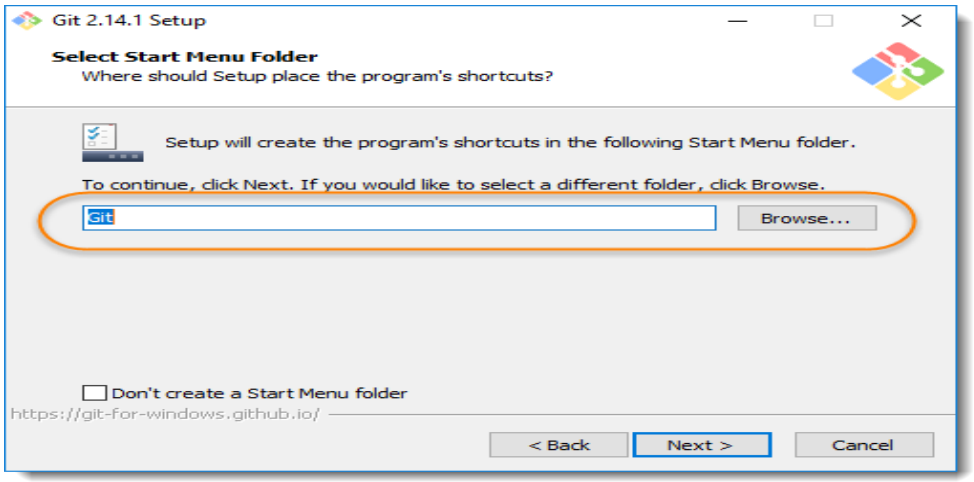
***Note**: When you've successfully started the installer, you should see the Git Setup wizard screen. Follow the Next and Finish prompts to complete the installation. The default options are pretty sensible for most users.*

***Note**: At the time of writing the tutorial on 9th Sep'17, the latest version is **Git-2.14.1**.*

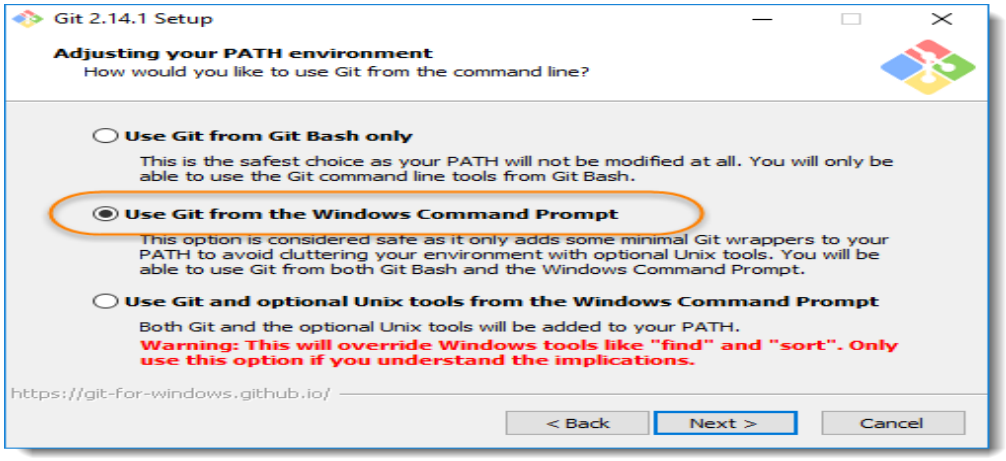
3. You may like to keep the installation to another folder, so here is the chance to do so. I just want to keep it in the suggested default folder in my ***Program Files\Git.***



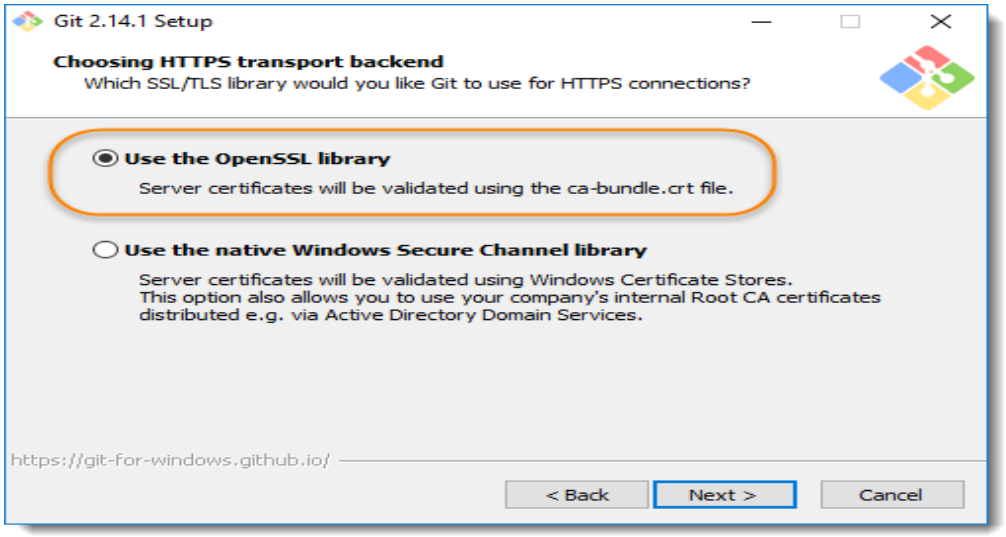
4. This is the option to store the *shortcut of the Git* under the ***Program Menu***.



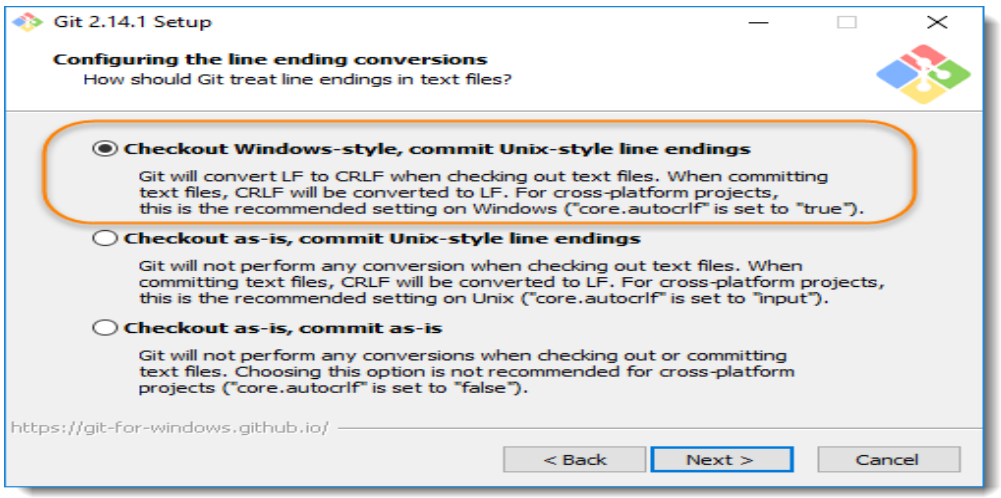
5. This is asking your choice that whether you like to Git from the ***Windows Command Prompt*** or you like to use some other program like ***Git Bash***. As of now just select the *Windows Cmd* for simplicity of the tutorial, later we will cover *Git Bash* and other as well.



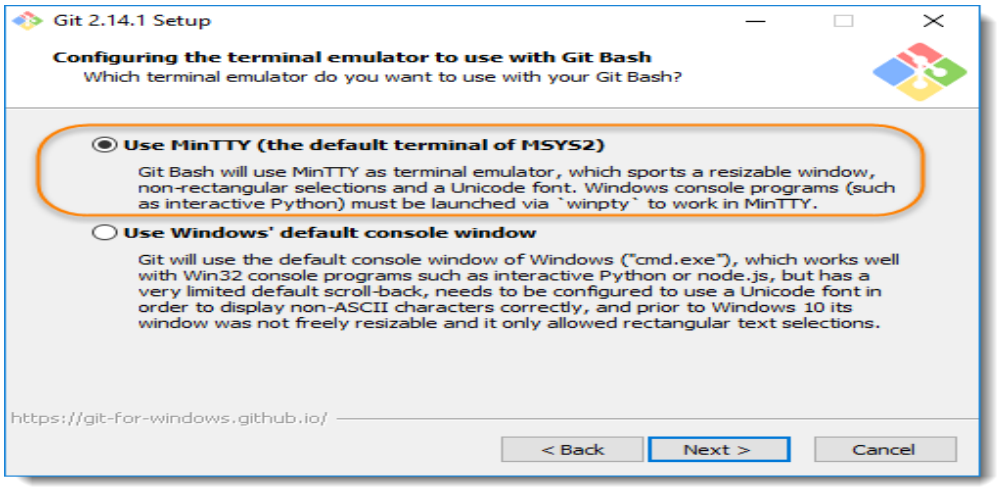
6. If you have *PuTTY/TortoiseSVN* installed, you may see this screen, otherwise just ignore this. Regardless, use ***OpenSSL*** to make things easy.



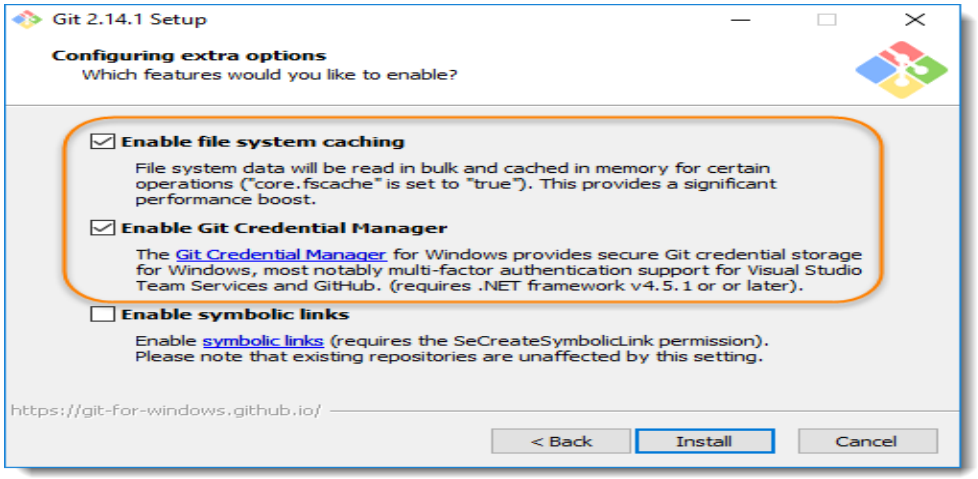
7. Here, we recommend to choose the option of ***Checkout Windows-style, commit Unix-style line endings***. Select next once you have done this.



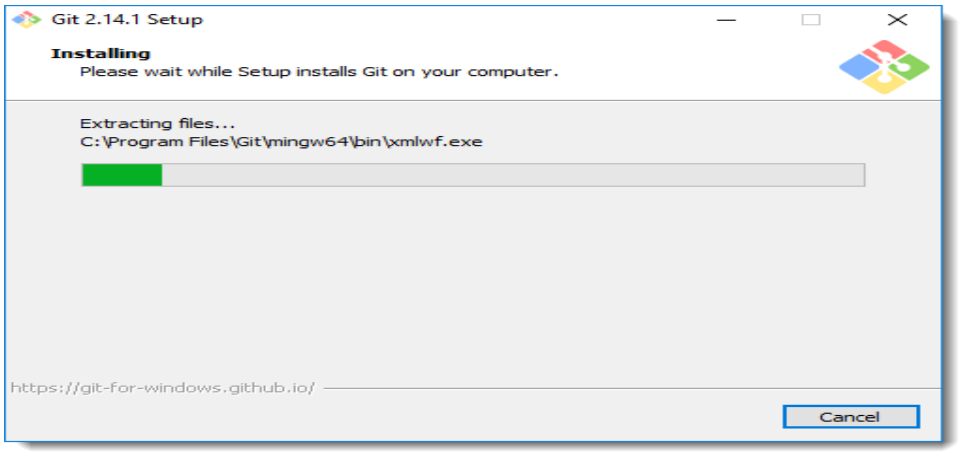
8. Again, just go with default selection and move forward.



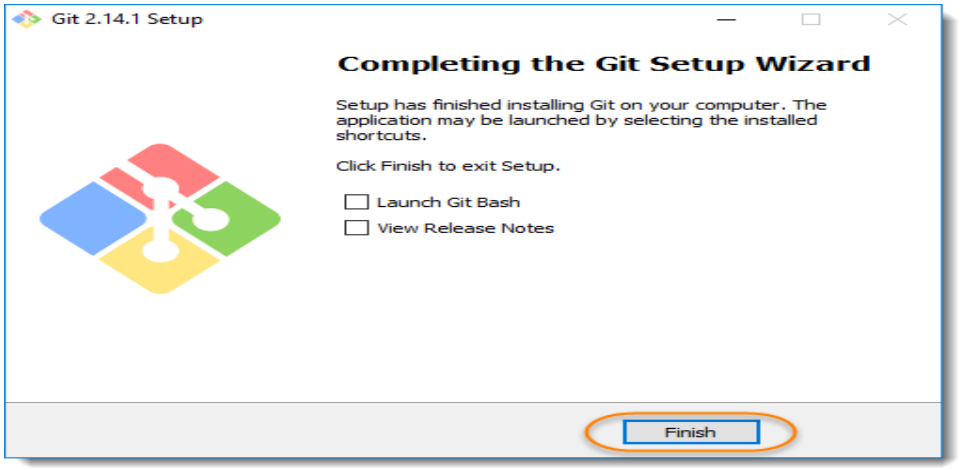
9. Just go with default selections, as we will cover the details in later advance chapter.



10. Now, its all done. This will just take few minutes to complete the installation as per your machine speed.



11. Once done, just click on Finish button.


Git 2.14.1 Setup
Completing the Git Setup Wizard
Setup has finished installing Git on your computer. The application may be launched by selecting the installed shortcuts.
Click Finish to exit Setup.
Launch Git Bash
View Release Notes
Finish

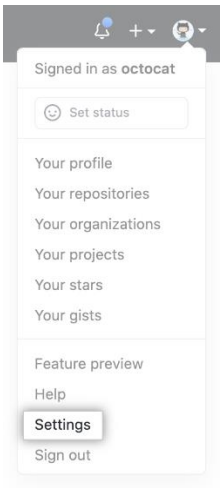
Step 3: Create of local Repository on Git and push it into Remote Repository

```
Mithun Aradya@DESKTOP-VO1RN53 MINGW64 ~
$ mkdir gitcmds

Mithun Aradya@DESKTOP-VO1RN53 MINGW64 ~
$ cd gitcmds

Mithun Aradya@DESKTOP-VO1RN53 MINGW64 ~/gitcmds
$ git init
Initialized empty Git repository in C:/Users/Mithun Aradya/gitcmds/.git/

Mithun Aradya@DESKTOP-VO1RN53 MINGW64 ~/gitcmds (master)
$ git config --global user.name "MITHUN"

Mithun Aradya@DESKTOP-VO1RN53 MINGW64 ~/gitcmds (master)
$ git config --global user.email "mithunaradya@gmail.com"

Mithun Aradya@DESKTOP-VO1RN53 MINGW64 ~/gitcmds (master)
$ git remote add mithun "https://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@github.com/MIthun-aradya/reglogjs.git"

Mithun Aradya@DESKTOP-VO1RN53 MINGW64 ~/gitcmds (master)
$ nano simple.txt

Mithun Aradya@DESKTOP-VO1RN53 MINGW64 ~/gitcmds (master)
$ echo simple.txt >> README.md

Mithun Aradya@DESKTOP-VO1RN53 MINGW64 ~/gitcmds (master)
$ git add simple.txt README.md
warning: in the working copy of 'README.md', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'simple.txt', LF will be replaced by CRLF the next time Git touches it

Mithun Aradya@DESKTOP-VO1RN53 MINGW64 ~/gitcmds (master)
$ git commit -m "this is my first commit"
[master (root-commit) a8b7155] this is my first commit
 2 files changed, 4 insertions(+)
 create mode 100644 README.md
 create mode 100644 simple.txt
```

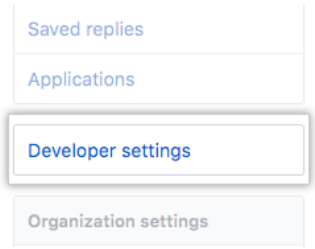
```
Mithun Aradya@DESKTOP-VO1RN53 MINGW64 ~/gitcmds (master)
$ git push -u mithun master
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 4 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (4/4), 345 bytes | 345.00 KiB/s, done.
Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'master' on GitHub by visiting:
remote:      https://github.com/MIthun-aradya/reglogjs/pull/new/master
remote:
To https://github.com/MIthun-aradya/reglogjs.git
 * [new branch]      master -> master
branch 'master' set up to track 'mithun/master'.

Mithun Aradya@DESKTOP-VO1RN53 MINGW64 ~/gitcmd (master)
$ git clone "https://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@github.com/MIthun-aradya/reglogjs.git"
Cloning into 'reglogjs'...
remote: Enumerating objects: 8, done.
remote: Counting objects: 100% (8/8), done.
remote: Compressing objects: 100% (6/6), done.
remote: Total 8 (delta 0), reused 8 (delta 0), pack-reused 0
Receiving objects: 100% (8/8), done.
```

Step 4: Creating a personal access token

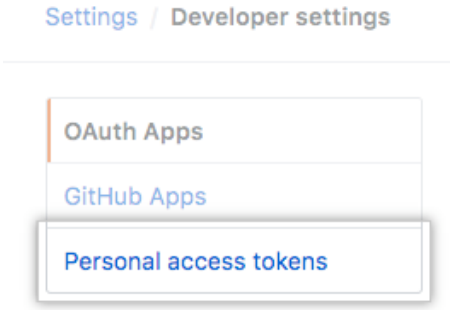
1. In the upper-right corner of any page, click your profile photo, then click **Settings**.



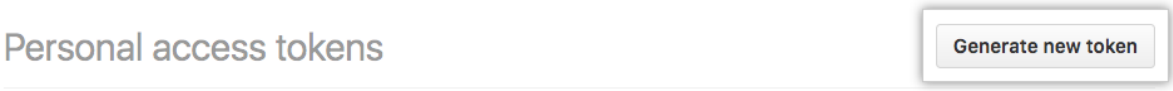
2. In the left sidebar, click **Developer settings**



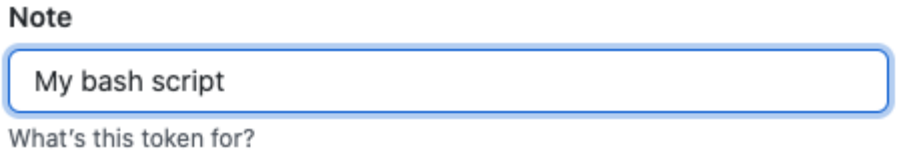
3. In the left sidebar, click **Personal access tokens**.



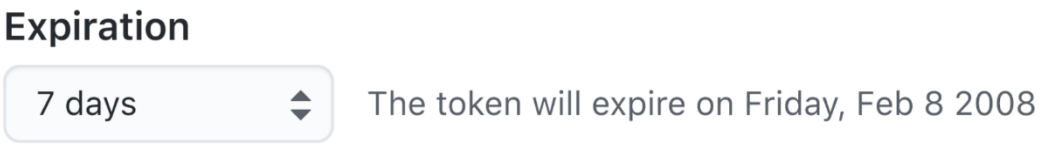
4. Click **Generate new token**.



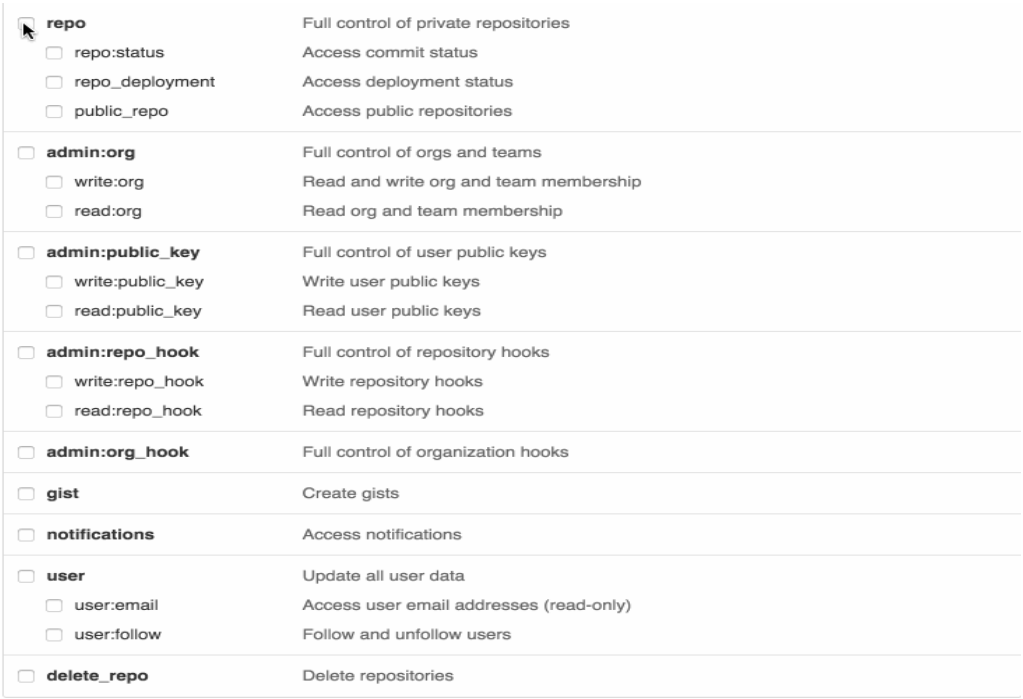
5. Give your token a descriptive name.



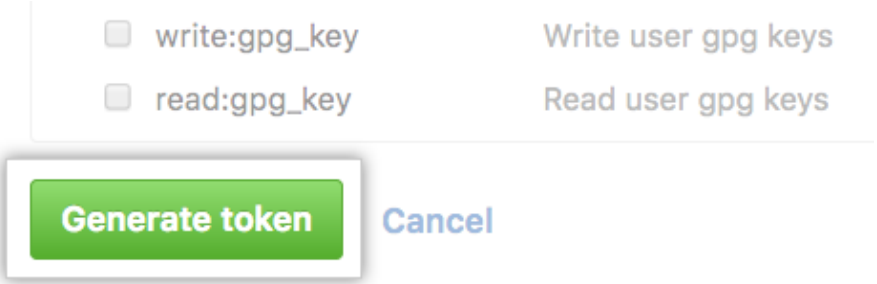
6. To give your token an expiration, select the **Expiration** drop-down menu, then click a default or use the calendar picker.

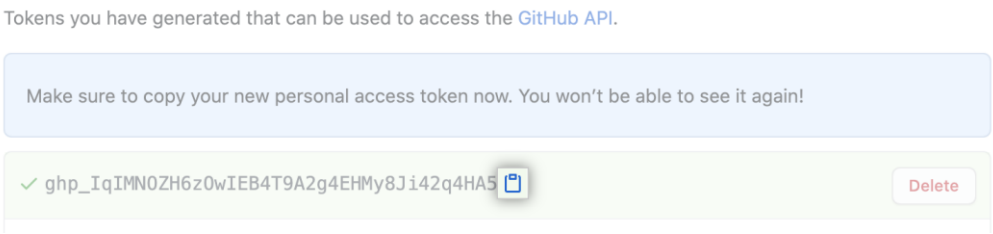


7. Select the scopes you'd like to grant this token. To use your token to access repositories from the command line, select **repo**. A token with no assigned scopes can only access public information. For more information, see "Available scopes".



8. Click **Generate token**.





Working with Git Commands

1. Git clone

Git clone is a command for downloading existing source code from a remote repository (like Github, for example). In other words, Git clone basically makes an identical copy of the latest version of a project in a repository and saves it to your computer.

Syntax : git clone https://name-of-the-repository-link

2. Git branch

Branches are highly important in the git world. By using branches, several developers are able to work in parallel on the same project simultaneously. We can use the git branch command for creating, listing and deleting branches.

Creating a new branch:

Syntax : git branch <branch-name>

3. Git checkout

This is also one of the most used Git commands. To work in a branch, first you need to switch to it. We use **git checkout** mostly for switching from one branch to another. We can also use it for checking out files and commits.

Syntax:

git checkout <name-of-your-branch>

4. Git status

The Git status command gives us all the necessary information about the current branch.

Syntax: git status

5. Git add

When we create, modify or delete a file, these changes will happen in our local and won't be included in the next commit (unless we change the configurations).

To add a single file:

git add <file>

To add everything at once:

git add -A

6. Git commit

This is maybe the most-used command of Git. Once we reach a certain point in development, we want to save our changes (maybe after a specific task or issue).

Syntax :

git commit -m "commit message"

7. Git push

After committing your changes, the next thing you want to do is send your changes to the remote server. Git push uploads your commits to the remote repository.

Syntax

git push <remote> <branch-name>

8. Git pull

The **git pull** command is used to get updates from the remote repo. This command is a combination of **git fetch** and **git merge** which means that, when we use git pull, it gets the updates from remote repository (git fetch) and immediately applies the latest changes in your local (git merge).

Syntax

git pull <remote>

9. Git revert

Sometimes we need to undo the changes that we've made. There are various ways to undo our changes locally or remotely (depends on what we need), but we must carefully use these commands to avoid unwanted deletions.

Syntax

git revert 3321844

10. Git merge

When you've completed development in your branch and everything works fine, the final step is merging the branch with the parent branch (dev or master). This is done with the git merge command.

Git merge basically integrates your feature branch with all of its commits back to the dev (or master) branch. It's important to remember that you first need to be on the specific branch that you want to merge with your feature branch.

For example, when you want to merge your feature branch into the dev branch:

First you should switch to the dev branch:

git checkout dev**Before**

merging, you should update your local dev branch:

git fetch**Finally**

you can merge your feature branch into dev:

git merge <branch-name>

Experiment – 3

Setting up React environment and Create and running a React.js app

React Introduction

ReactJS is a declarative, efficient, and flexible JavaScript library for building reusable UI components.

It is an open-source, component-based front end library responsible only for the view layer of the application.

It was created by Jordan Walke. It was initially developed and maintained by Facebook and was later used in its products like WhatsApp & Instagram.

REACT FEATURES:

Currently, ReactJS gaining quick popularity as the best JavaScript framework among web developers. It is playing an essential role in the front-end ecosystem. The important features of ReactJS are as following.

- JSX: JSX stands for JavaScript XML. It is a JavaScript syntax extension. Its an XML or HTML like syntax used by ReactJS. This syntax is processed into JavaScript calls of React Framework.
- Components:ReactJS is all about components. ReactJS application is made up of multiple components, and each component has its own logic and controls. These components can be reusable which help you to maintain the code when working on larger scale projects.
- One-way Data Binding:ReactJS is designed in such a manner that follows unidirectional data flow or one-way data binding. The benefits of one-way data binding give you better control throughout the application
- Virtual DOM:A virtual DOM object is a representation of the original DOM object. It works like a one-way data binding
- Simplicity:ReactJS uses JSX file which makes the application simple and to code as well as understand.

Performance: when we create a component, we did not write directly to the DOM. Instead, we are writing virtual components that will turn into the DOM leading to smoother and faster performance.

Setting up React environment

Requirements

To create a React Project using create-react-app, we need to have installed the following things in your system.

1. Node version >= 8.10
2. NPM version >= 5.6
3. Visual-studio/ATOM/sublime or any IDE

Run the following command to check the Node version in the command prompt

$ node -v

$ npm -v

Install React

We can install React using npm package manager by using the following command. There is no need to worry about the complexity of React installation. The create-react-app npm package manager will manage everything, which needed for React project.

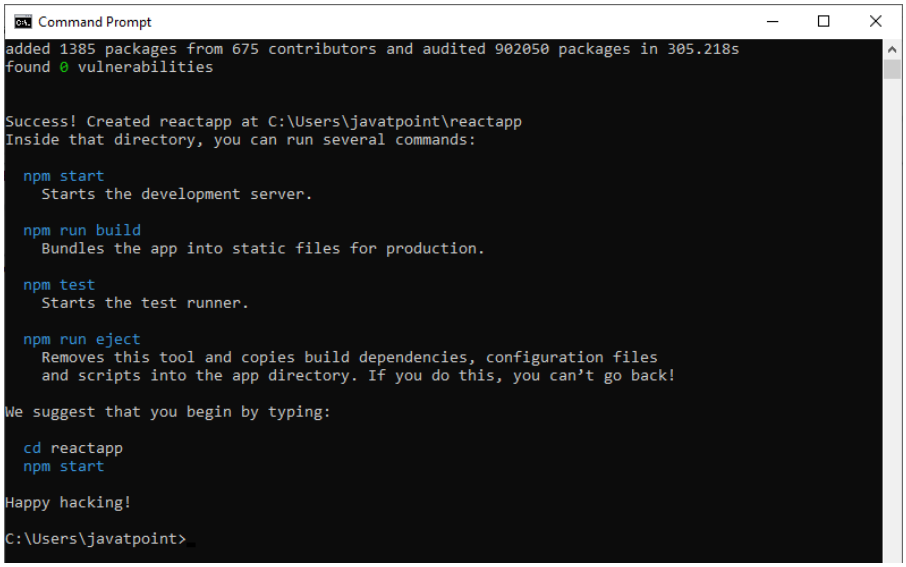
C:\Users\javatpoint> npm install -g create-react-app

Create a new React project

Once the React installation is successful, we can create a new React project using create-react-app command.

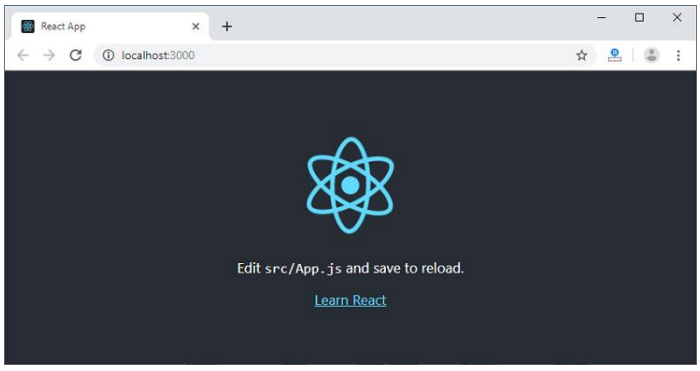
C:\Users\javatpoint> create-react-app reactapp

The above command will take some time to install the React and create a new project with the name "reactapp." Now, we can see the terminal as like below.



The above screen tells that the React project is created successfully on our system. Now, we need to start the server so that we can access the application on the browser. Type the following command in the terminal window.

1. $ cd reactapp
2. $ npm start
   NPM is a package manager which starts the server and access the application at default server http://localhost:3000. Now, we will get the following screen.



Create a login page(using react)

Index.html

```
<html>
<head>
<title>LOGIN PAGE</title>
</head>
<body>

<div class="form-box">
<h2>Login Here</h2>
  <form class="myform" method="post" name="form" onsubmit="return
validated()" action=message.html>
  <div class="input-group">
  <label>User name:</label>
  <input autocomplete="off" type="text" name="username" value="">

  </div>
  <div class="input-group">
  <label>password</label>
  <input autocomplete="off" type="text" name="password" id="pass">

  </div>
  <p><a href="#">Forget password?</a></p>
  <button type="submit">Login &#10132;</button>
  </form>
 <h1> Not registerd? <a
href="file:///C:/Users/Admin/OneDrive/Pictures/OneDrive/Desktop/html/form.h
tml">
 Register</h1>
 </a></div></body></html>
```

Index.css

```
*{
  margin: 0;
  padding: 0;
 }
body{
    background:  green;
```

```
    background-position: center;
    background-size: cover;
}
.form-box{
   width: 350px;
   box-shadow: 0 0 10px 0 #000;
   margin: 100px auto;
   background: #fff;
   padding: 50px 25px;
}
input{
  width: 100%;
  height: 40px;
  border-radius: 30px;
  outline: none;
  border: 1px solid #999;
  padding: 20px;


}
.input-group{
  margin: 20px auto;
  position: relative;
}
label{
  position: absolute;
  top: -12px;
  left: 20px;
  background: #fff;
  padding: 0 5px;

}
button{
  text-align: center;
  padding: 10px 20px;
  border: none;
  background: #ff1100;
  font-size: 14px;
  color: #fff;
  cursor: pointer;
  border-radius: 30px;
}
button:hover{
```

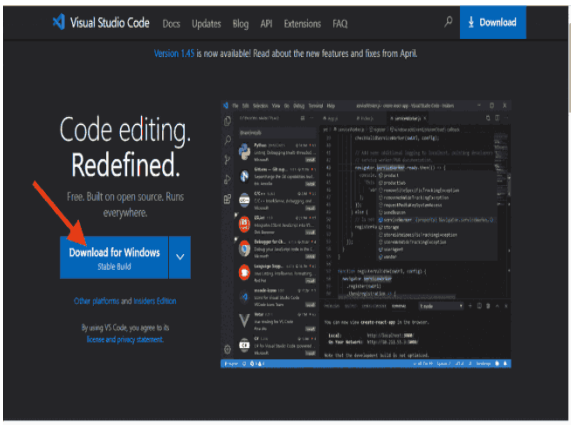
```
   background: #fff;
   transition: 0.5;
}
h2{
 text-align: center;
 width: 200px;
 box-shadow: 0 0 10px 0 #000;
 margin: -80 auto 30px;
 height: 40px;
 padding-top: 16px;
 border-radius: 30px;
 background: #fff;
}
```

Experiment – 4

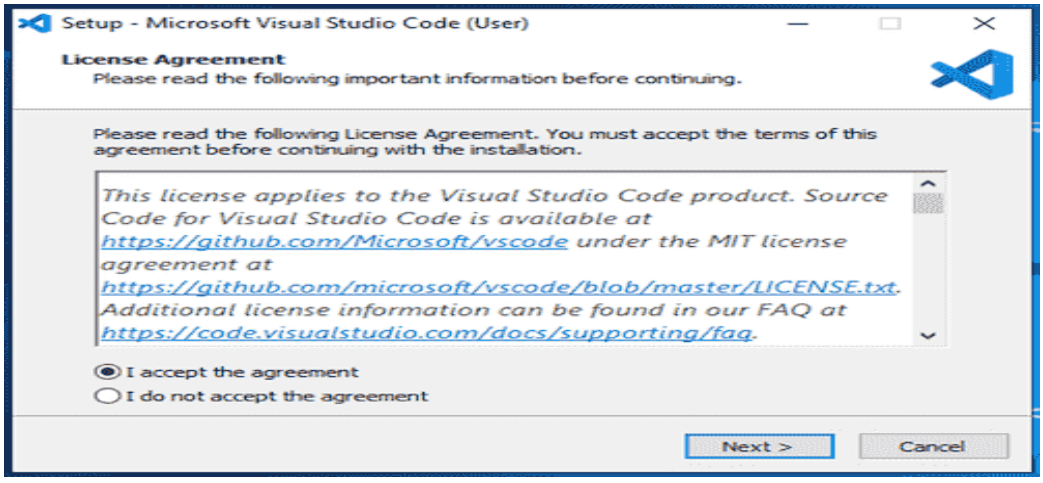
Setting Up the Environment and Tools for front end development Installing VS Code

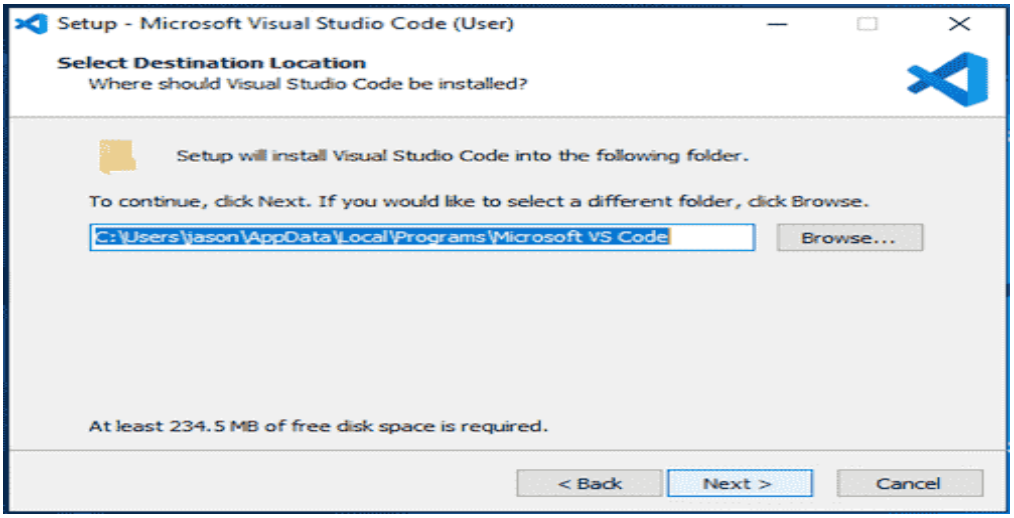
VS Code is a free code editor that runs on Windows, Mac and Linux.

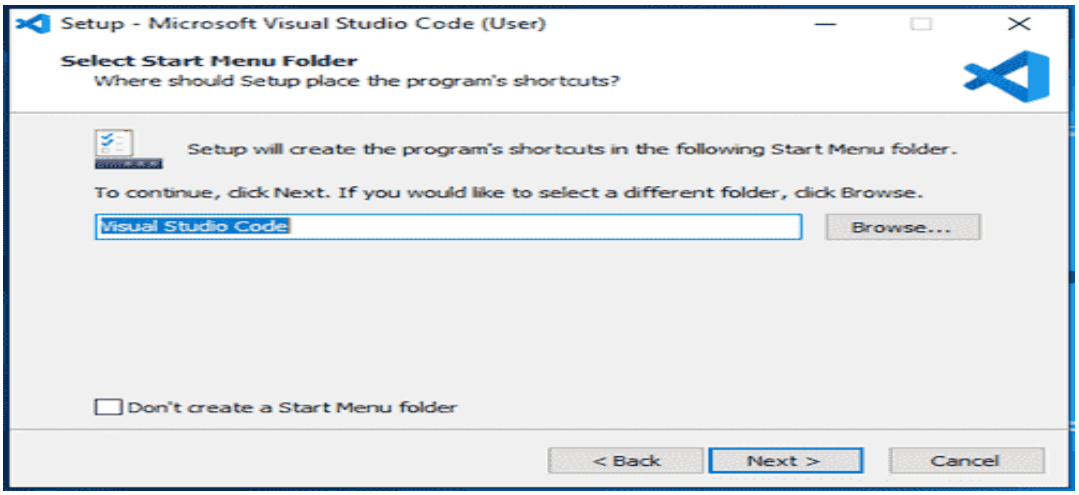
Download VS Code from https://code.visualstudio.com/.

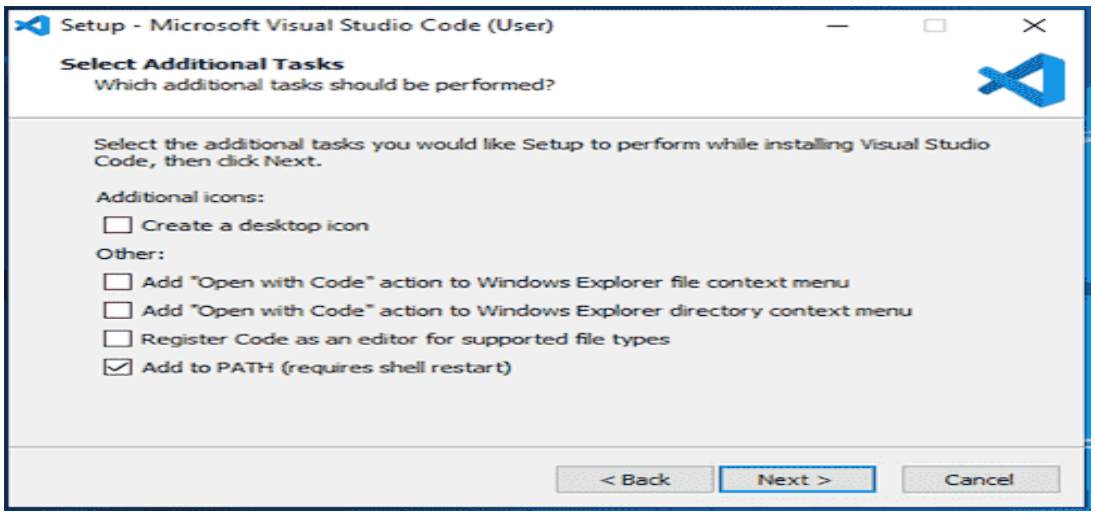


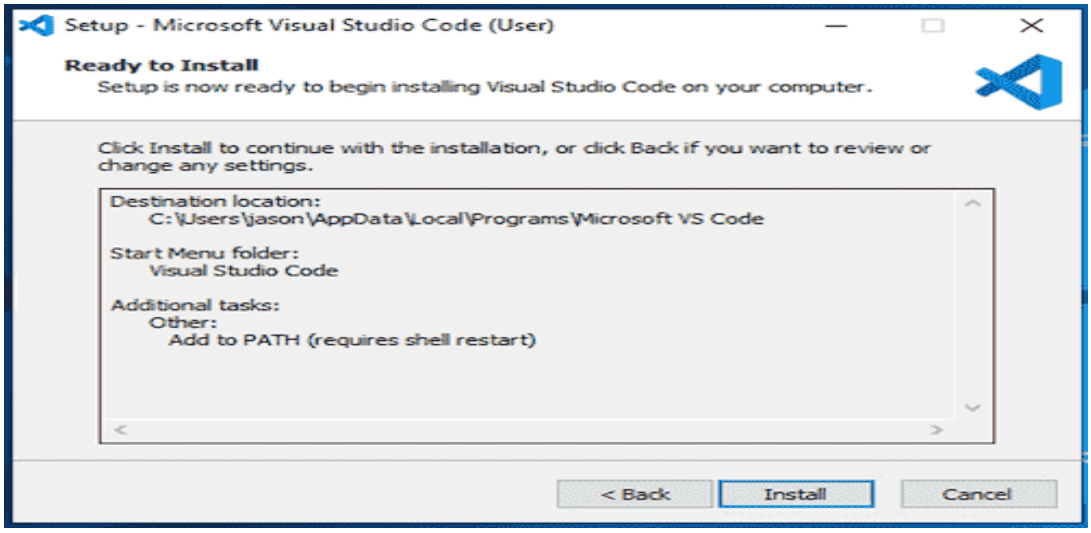
Install Visual Studio Code by opening the downloaded setup file and following the prompts.

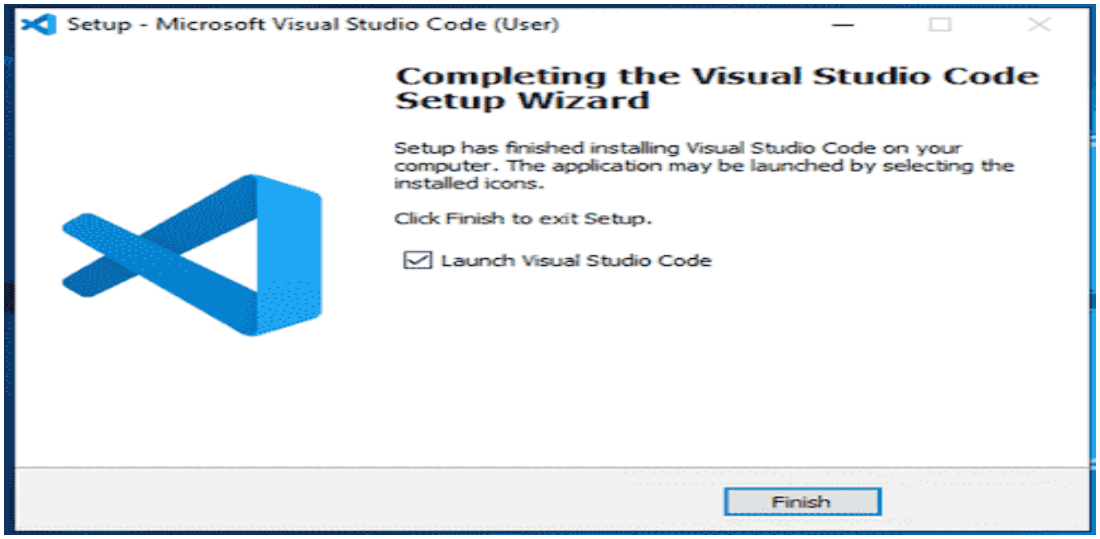



Setup - Microsoft Visual Studio Code (User)
Select Destination Location
Where should Visual Studio Code be installed?
Setup will install Visual Studio Code into the following folder.
To continue, click Next. If you would like to select a different folder, click Browse.
C:\Users\jason\AppData\Local\Programs\Microsoft VS Code
Browse...
At least 234.5 MB of free disk space is required.
< Back
Next >
Cancel


Setup - Microsoft Visual Studio Code (User)
Select Start Menu Folder
Where should Setup place the program's shortcuts?
Setup will create the program's shortcuts in the following Start Menu folder.
To continue, click Next. If you would like to select a different folder, click Browse.
Visual Studio Code
Browse...
Don't create a Start Menu folder
< Back
Next >
Cancel


Setup - Microsoft Visual Studio Code (User)
Select Additional Tasks
Which additional tasks should be performed?
Select the additional tasks you would like Setup to perform while installing Visual Studio Code, then click Next.
Additional icons:
Create a desktop icon
Other:
Add "Open with Code" action to Windows Explorer file context menu
Add "Open with Code" action to Windows Explorer directory context menu
Register Code as an editor for supported file types
Add to PATH (requires shell restart)
< Back
Next >
Cancel


Setup - Microsoft Visual Studio Code (User)
Ready to Install
Setup is now ready to begin installing Visual Studio Code on your computer.
Click Install to continue with the installation, or click Back if you want to review or change any settings.
Destination location:
C:\Users\jason\AppData\Local\Programs\Microsoft VS Code
Start Menu folder:
Visual Studio Code
Additional tasks:
Other:
Add to PATH (requires shell restart)
< Back
Install
Cancel


Setup - Microsoft Visual Studio Code (User)
Completing the Visual Studio Code Setup Wizard
Setup has finished installing Visual Studio Code on your computer. The application may be launched by selecting the installed icons.
Click Finish to exit Setup.
Launch Visual Studio Code
Finish

Experiment – 5

Create a form like registration form or feedback form, after submit hide create form and enable the display section using java script.

Registration.html

```
<html> <head>
      <title> Registration Form</title>
      <script>
         function passvalues()
         {
                  var name = document.getElementById("name").value;
                    var email = document.getElementById("email").value;
                    var address = document.getElementById("address").value;
                    localStorage.setItem("name",name);
                    localStorage.setItem("email",email);
                    localStorage.setItem("address",address);
                    return;
         }
      </script>
   </head>
   <body>
<h1>Registrtion Form</h1>
      <form action="Details.html">
<fieldset>
   <legend>Registration</legend>
 <label> Name </label>
      <input type="text" id="name"/><br><br>
 <label> Email ID </label>
      <input type="email" id="email"/><br><br>

 <label> Address </label>
      <input type="address" id="address"/><br><br>
      <input type="submit" value="submit" onclick="passvalues()"/>
</fieldset>
      </form>
   </body>
```

```
</html>
```

Details.html

```
<html>
   <head>
      <title> Details</title>
   </head>
   <body>
<form>
   Your Name is:<p id="name"></p><br>
      Your email is:<p id="email"></p><br>
      Your address is:<p id="address"></p>
<script>
         document.getElementById("name").innerHTML = localStorage.getItem("name");
         document.getElementById("email").innerHTML = localStorage.getItem("email");
         document.getElementById("address").innerHTML = localStorage.getItem("address");

      </script>
</form>

   </body>
</html>
```

Experiment – 6

Create and run simple program in TypeScript

Install TypeScript using Node.js Package Manager (npm)

Step-1 Install Node.js. It is used to setup TypeScript on our local computer.

To install Node.js on Windows, go to the following link: **https://www.javatpoint.com/install-nodejs**

Step-2 Install TypeScript. To install TypeScript, enter the following command in the Terminal Window.

- npm install typescript --save-dev //As dev dependency
- npm install typescript -g //Install as a global module

  or
- npm install -g typescript
- npm install typescript@latest -g //Install latest if you have an older version

Step-3 To verify the installation was successful, enter the command **$ tsc -v** in the Terminal Window.

Install Live server

```
npm install -g live-server
```

Create and run first program in TypeScript

- open command prompt
- go to d: drive(any drive)
- d:\>mkdir typescript
- d:\>cd typescript
- d:\typescript> npm install typescript --save-dev
- open visual studio code
- file-open folder-choose typescript folder from d:
- create new file- save it as types.ts(any name.ts)

- Write the below code and save it
- console.log("Hello World");
- go to command prompt and compile the program
- tsc types.ts
- run the program
- node types.js
- Observe the output

Experiment – 7

Forms - Use of HTML tags in forms like select, input, file, textarea, etc.

```
<html>
<head>
<title>Form Elements</title>
</head>
<body>
<form>
<lable>Text Box</lable>
<input type="text" id="t1" name="name" value=""/><br><br>

Radio Button: <br>
<input type="radio" id="r1" name="" value=""/>Male<br> <br>
<input type="radio" id="r1" name="" value=""/>FeMale<br><br>

Check Box:<input type="checkbox" id="c1" name="" value=""/><br><br>

File:<input type="file" id="e1" name="file" value=""/><br><br>

Select:<br>
<label>Sem</label>
<select name="sem" id="sem">
  <option value="1">1 Sem</option>
  <option value="2">2 Sem</option>
</select><br><br>

Text Area:<br>
<textarea id="ta1" name="textarea" rows="4" cols="50">
At w3schools.com you will learn how to make a website.
</textarea><br><br>


<fieldset>
   <legend>Personal Details:</legend>
```

```
    <label>First name:</label>
    <input type="text" id="fname" name="fname"><br><br>
    <label>Last name:</label>
    <input type="text" id="lname" name="lname"><br><br>
  </fieldset><br><br>
Button:<input type="button" id="t1" name="" value="Submit"/><br>
</form>
</body>
</html>
```

Experiment -8

Create and run a simple Maven project

Maven Installation Steps

1. JDK and JAVA_HOME
Make sure JDK is installed, and the JAVA_HOME environment variable is configured.

2. Download Apache Maven
Visit Maven official website, download the Maven zip file, for example, **apache-maven-3.6.3-bin.zip**.

3. Add MAVEN_HOME system variable
Go to the Environment variables -> System variables section in your windows 10 machine.
Add a MAVEN_HOME system variable, and point it to the Maven folder.

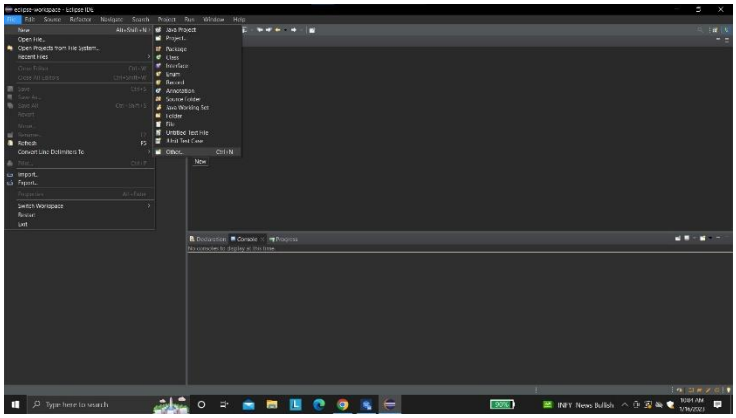
MAVEN STEPS TO RUN IN ECLIPSE

Import a maven project in Eclipse

- Open Eclipse.

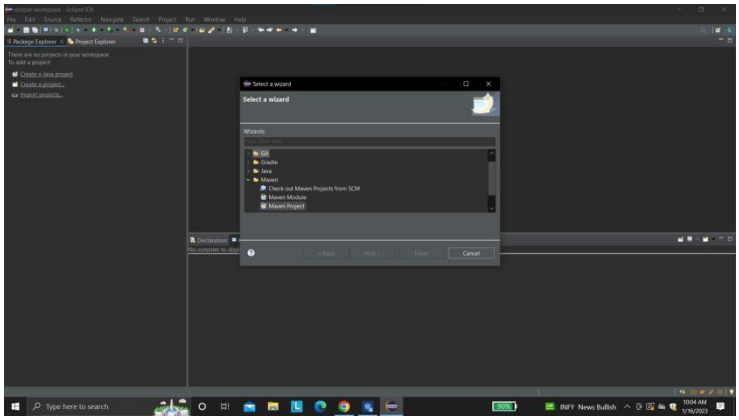
- Select **File > Import >** option.

- Select Maven Projects Option. Click on Next Button.

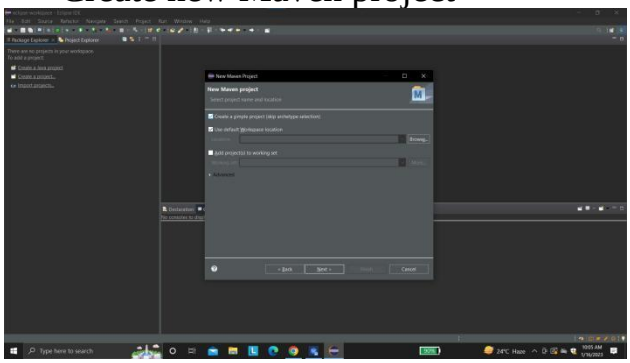


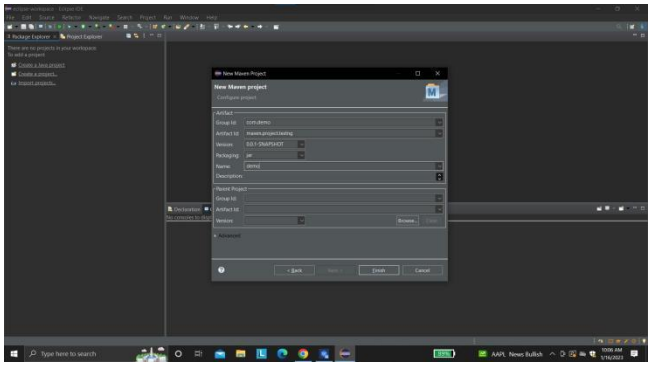
- Select Project location, where a project was created using Maven.

- Click Finish Button.

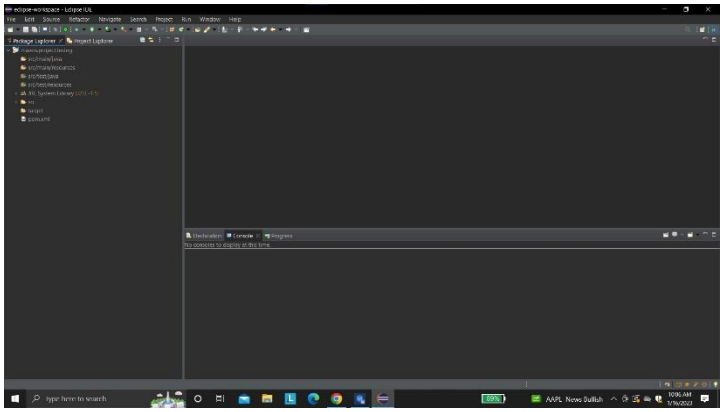


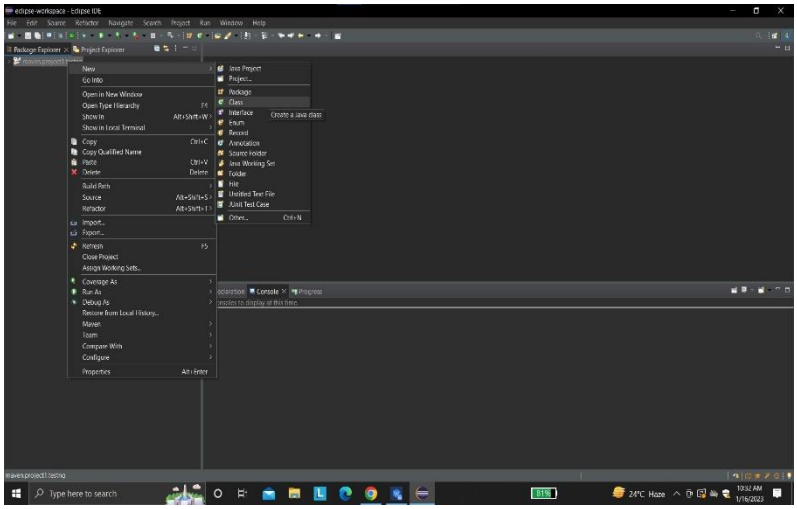
- Create new Maven project



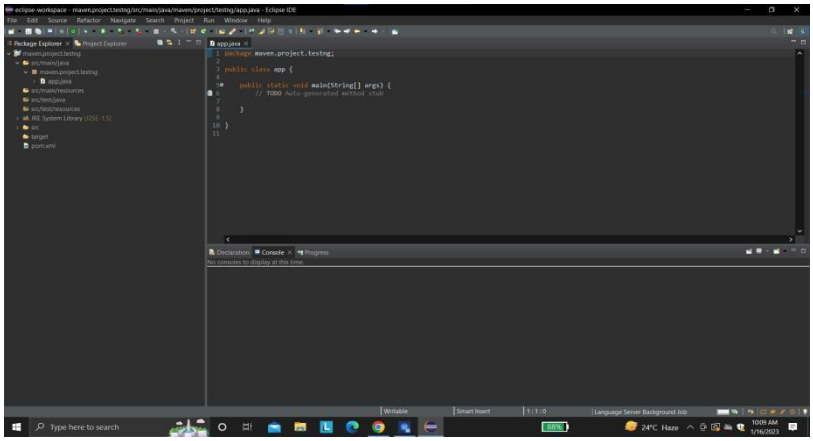


- Click Finish Button.

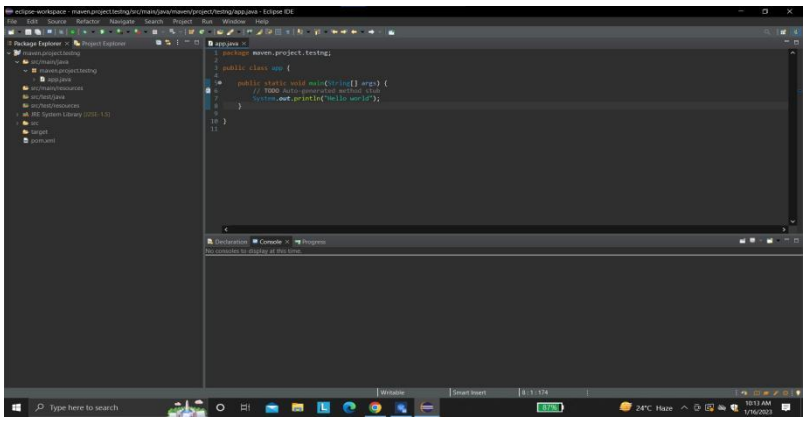




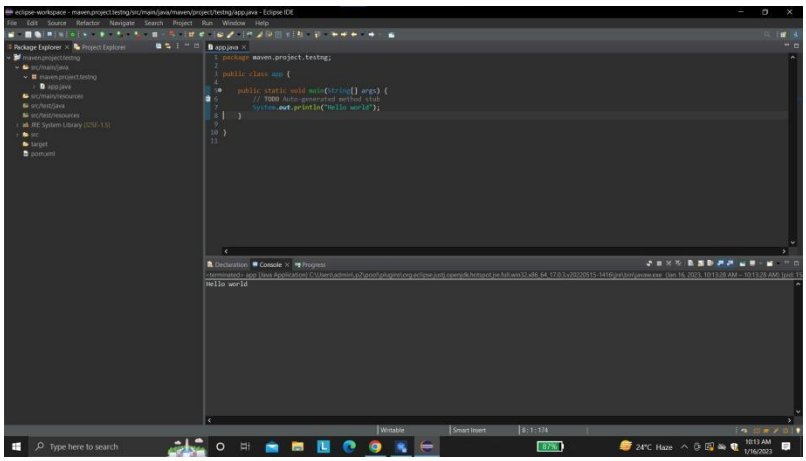
- Run program using class



- Insert the print statement



- Run the Project



Experiment – 9

Create a Spring Boot Project in Eclipse IDE

Introduction for springboot:
Spring Boot is a microservice-based framework and making a production-ready application in it takes very little time.

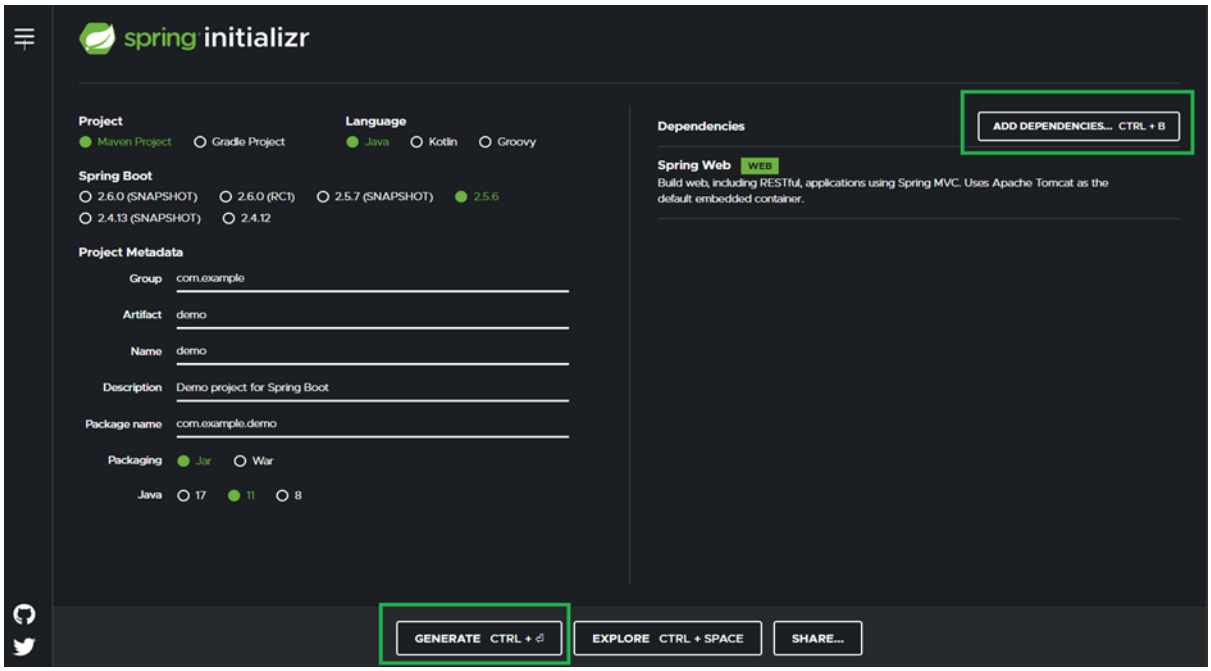
Following are some of the features of Spring Boot:
1.It allows avoiding heavy configuration of XML which is present in spring
2.It provides easy maintenance and creation of REST endpoints
3.It includes embedded Tomcat-server
Deployment is very easy, war and jar files can be easily deployed in the tomcat server.

The Eclipse IDE is famous for the Java Integrated Development Environment (IDE), but it has a number of pretty cool IDEs, including the C/C++ IDE, JavaScript/TypeScript IDE, PHP IDE, and more.
Procedure:
1. Install Eclipse IDE for Enterprise Java and Web Developer
2. Create a Spring Boot Project in Spring Initializr
3. Import Spring Boot Project in Eclipse IDE
4. Search “maven” and choose Existing Maven Project
5. Choose Next
6. Click on the Browse button and select the extracted zip
7. Click on the Finish button and we are done creating the Spring Boot project

Let us discuss these steps in detail alongside visual aids

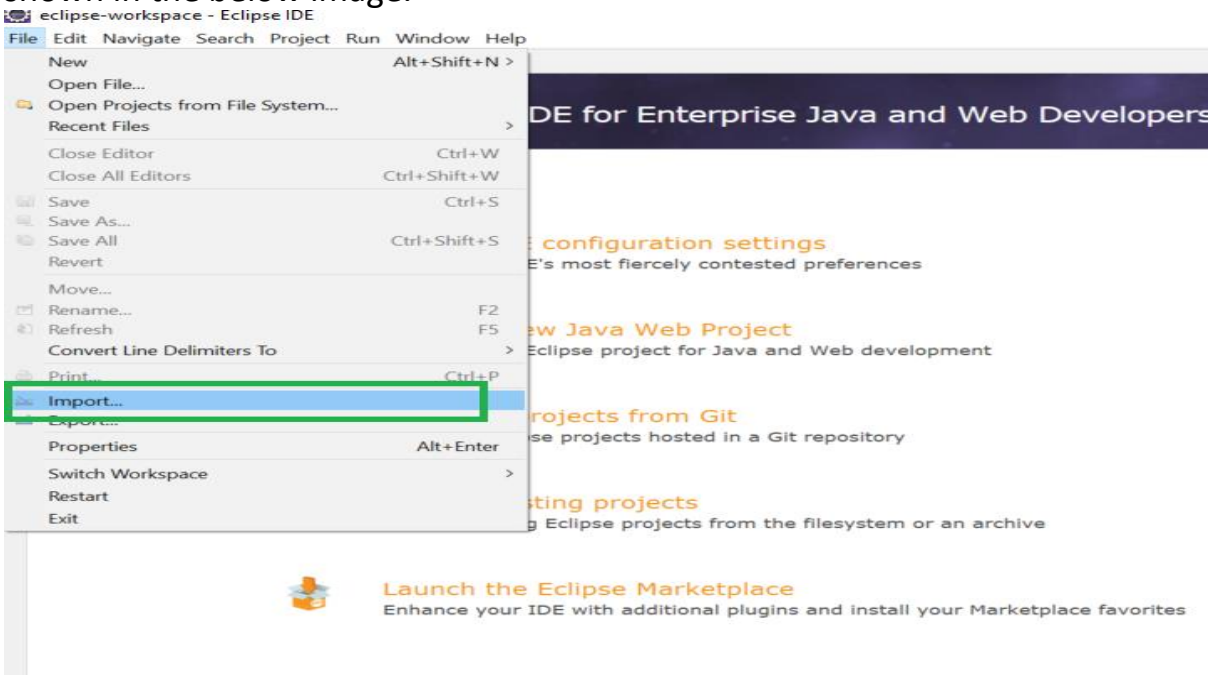
Step 1: Install Eclipse IDE for Enterprise Java and Web Developer
Please refer to this article How to Install Eclipse IDE for Enterprise Java and Web Development and install the Eclipse IDE.
Step 2: Create a Spring Boot Project in Spring Initializr
Go to this link and create a Spring Boot project. Please fill in all the details accordingly and at last click on the GENERATE button below. This will download your Spring Boot project in zip format. Now extract the folder into your local machine. For more details in Spring Initializr refer to this article:

Spring Initializr

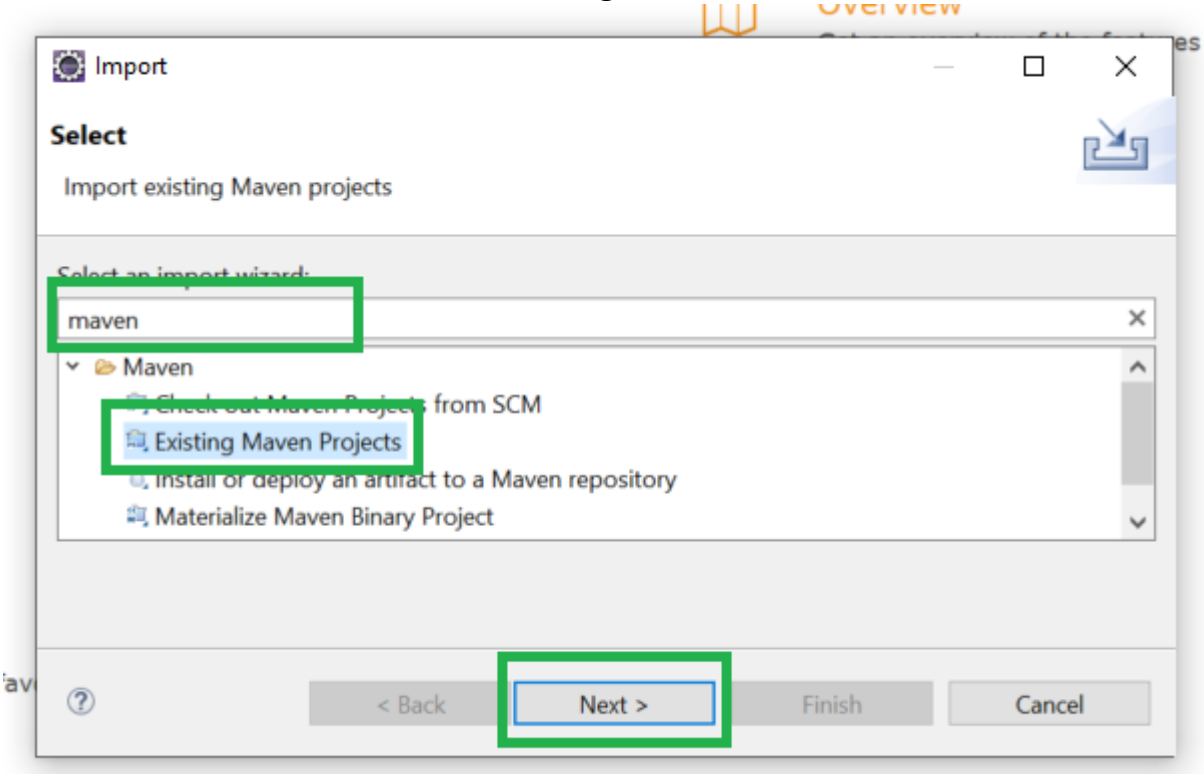


Step 3: Import Spring Boot Project in Eclipse IDE

Go to the Eclipse IDE for Enterprise Java and Web Developer > File > Import as shown in the below image.

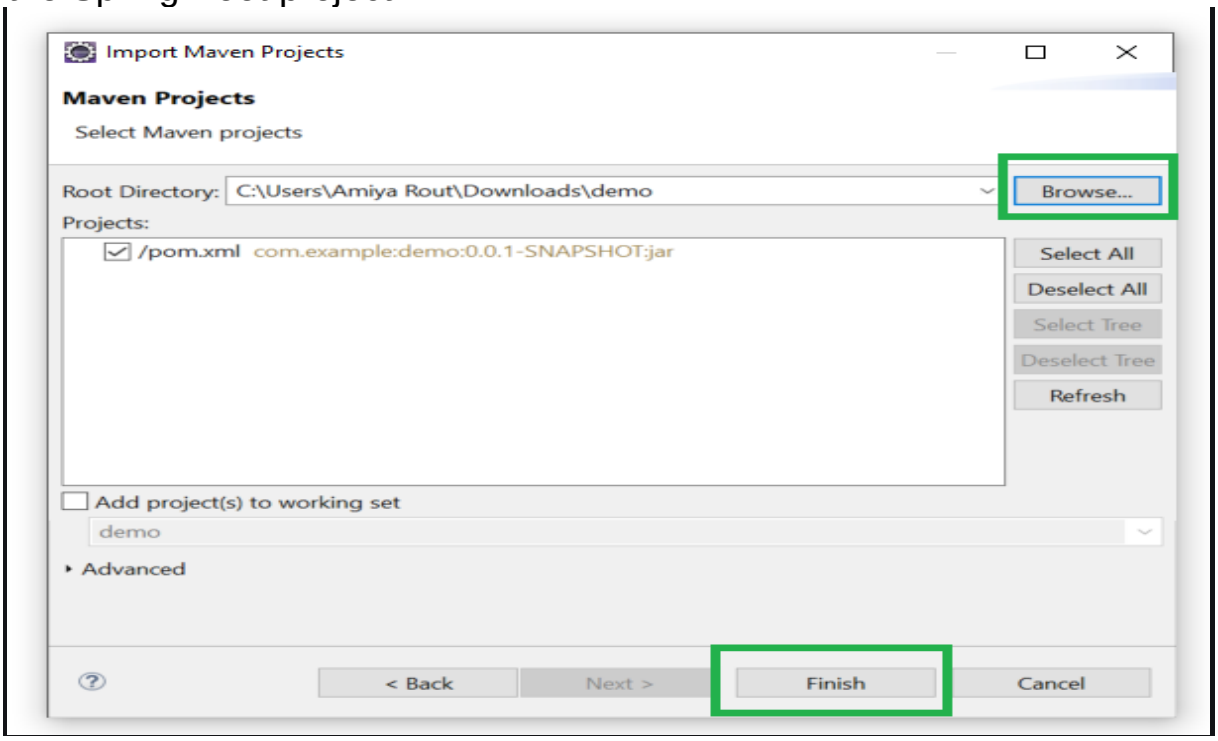


Step 4: Search “maven” and choose Existing Maven Project and click on the Next button as shown in the below image.

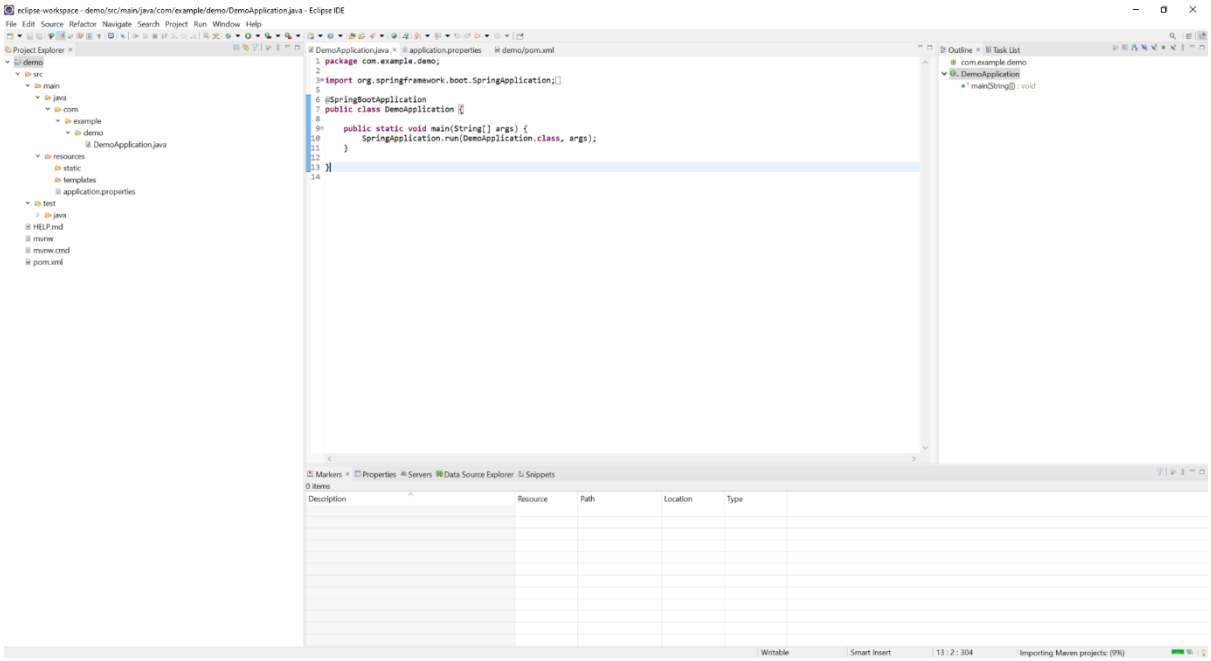


Step 5: Now click on the Browse button and select the extracted zip file that has been generated.

Step 6. And at last click on the Finish button and we are done creating the Spring Boot project.



By now, Spring Boot project has been created as depicted in the below media



Create a Spring Boot Project in STS IDE

Spring Tool Suite (STS) is a java IDE tailored for developing Spring-based enterprise applications. It is easier, faster, and more convenient. And most importantly it is based on Eclipse IDE. STS is free, open-source, and powered by VMware. Spring Tools 4 is the next generation of Spring tooling for the favorite coding environment. Largely rebuilt from scratch, it provides world-class support for developing Spring-based enterprise applications, whether you prefer Eclipse, Visual Studio Code, or Theia IDE.

Procedure:

1. Install Spring Tool Suite IDE

2. Create a new Spring project

3. Fill details in the pop-up window and press Next.

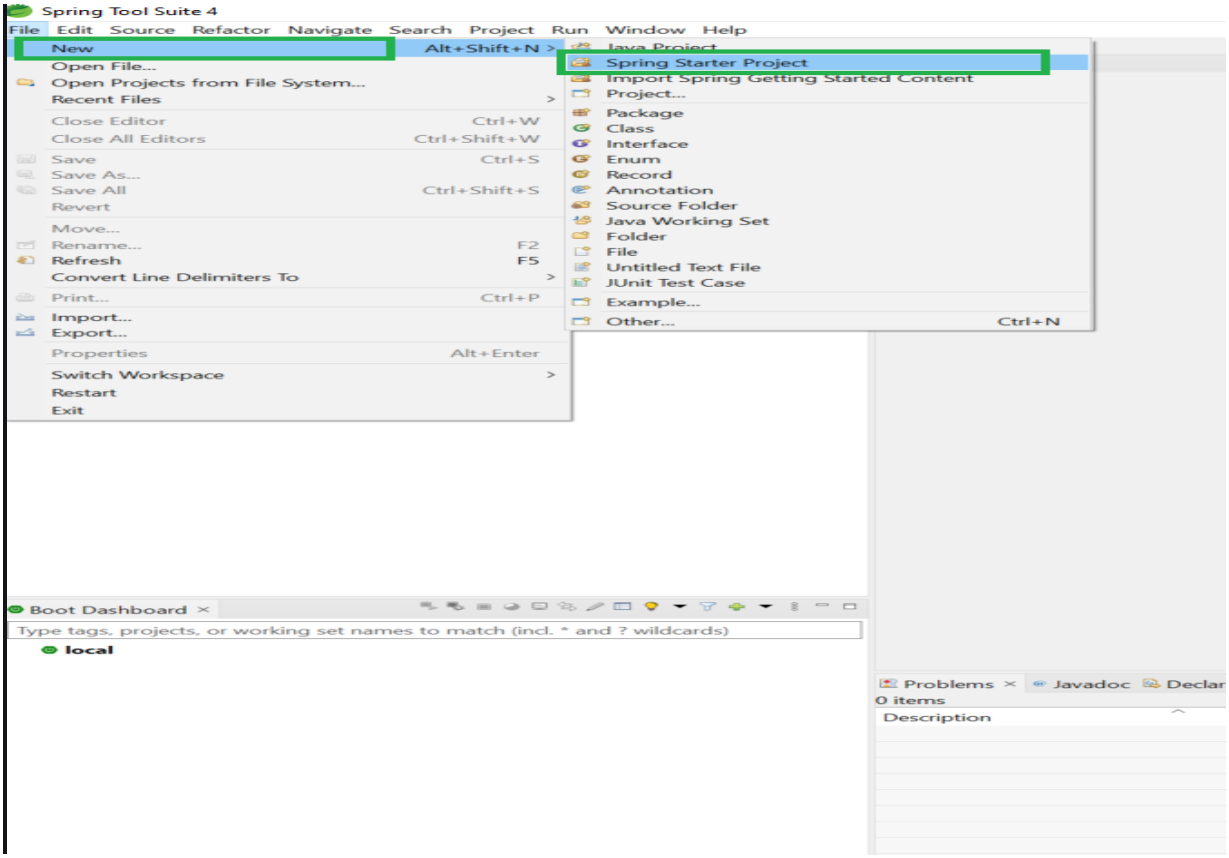
4. Choose Spring Boot version and select dependencies and press Next.

5. Click on the ‘Finish’ button.

Step 1: Install Spring Tool Suite (Spring Tools 4 for Eclipse) IDE

For this user must have pre-requisite knowledge of downloading and installing Spring Tool Suite IDE

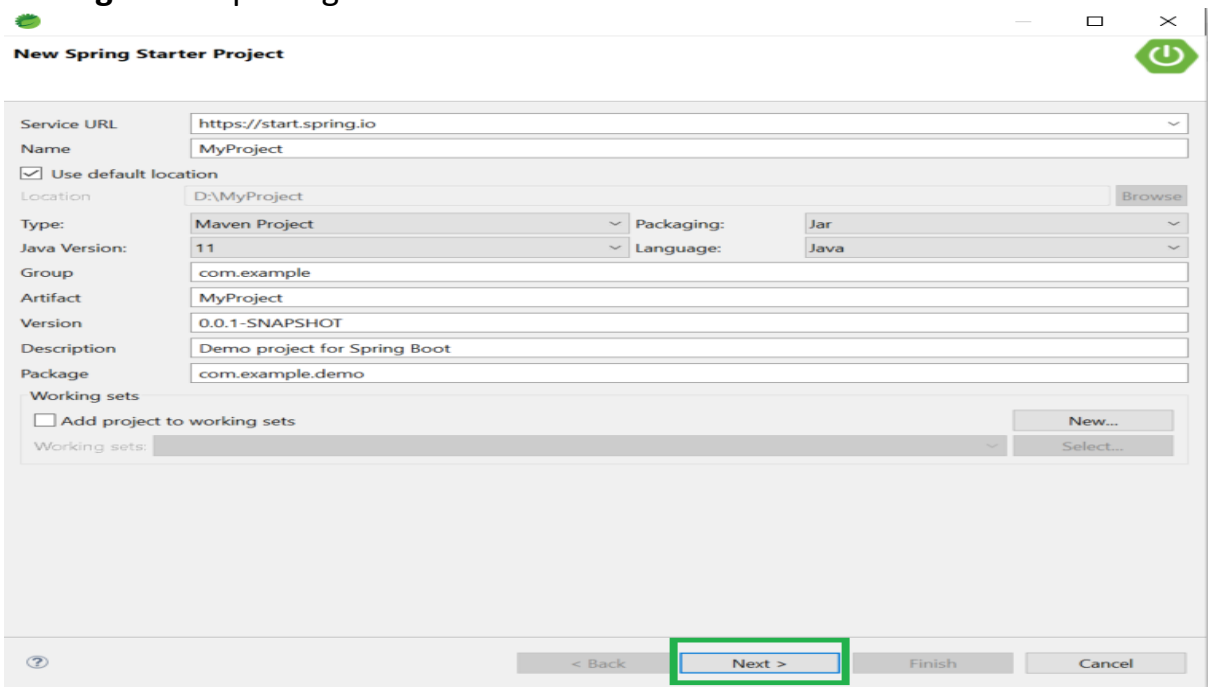
Step 2: Go to the File > New > Spring Starter Project as shown in the below image as follows:



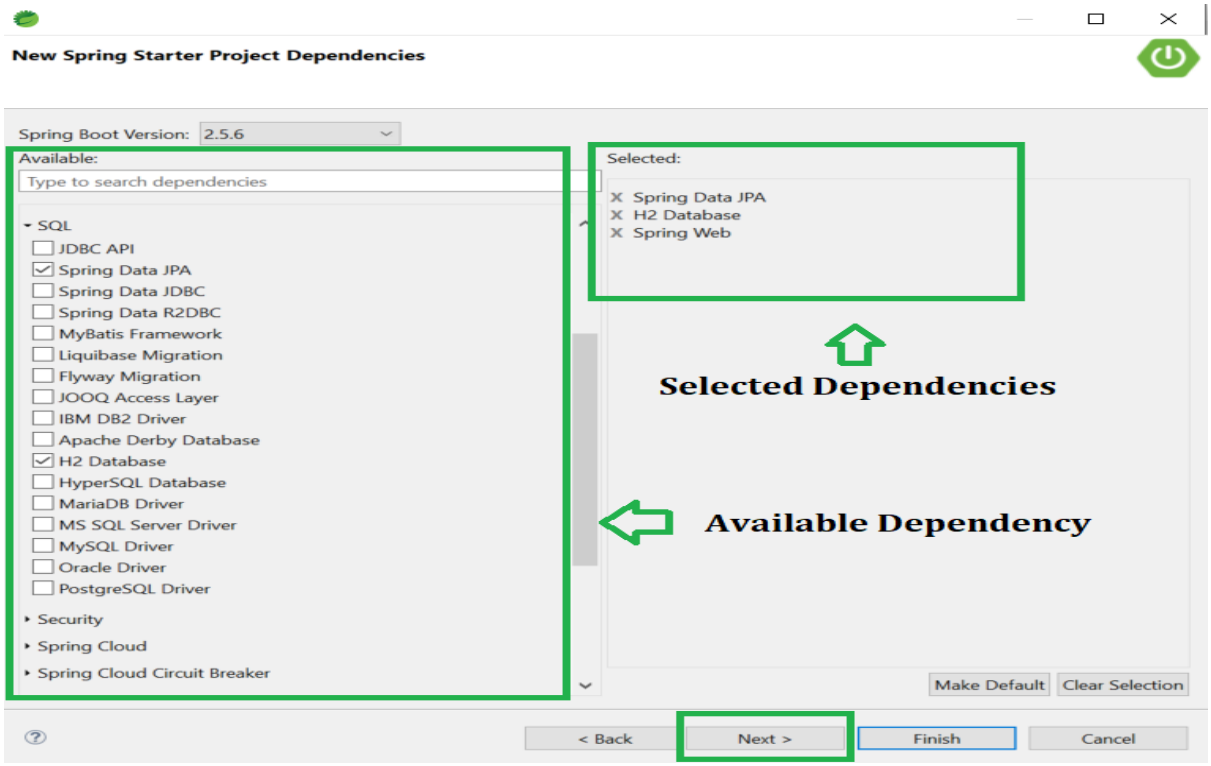
Step 3: In this pop-up window fill the detail below as follows and further click on the Next button as shown in the below image.

Service URL: Default
Name: Your Project Name
Type: Maven Project
Java Version: 11 or greater than 11
Packaging: As your need
Language: As your need
Group: A unique base name of the company or group that created the project
Artifact: A unique name of the project
Version: Default
Description: As your need

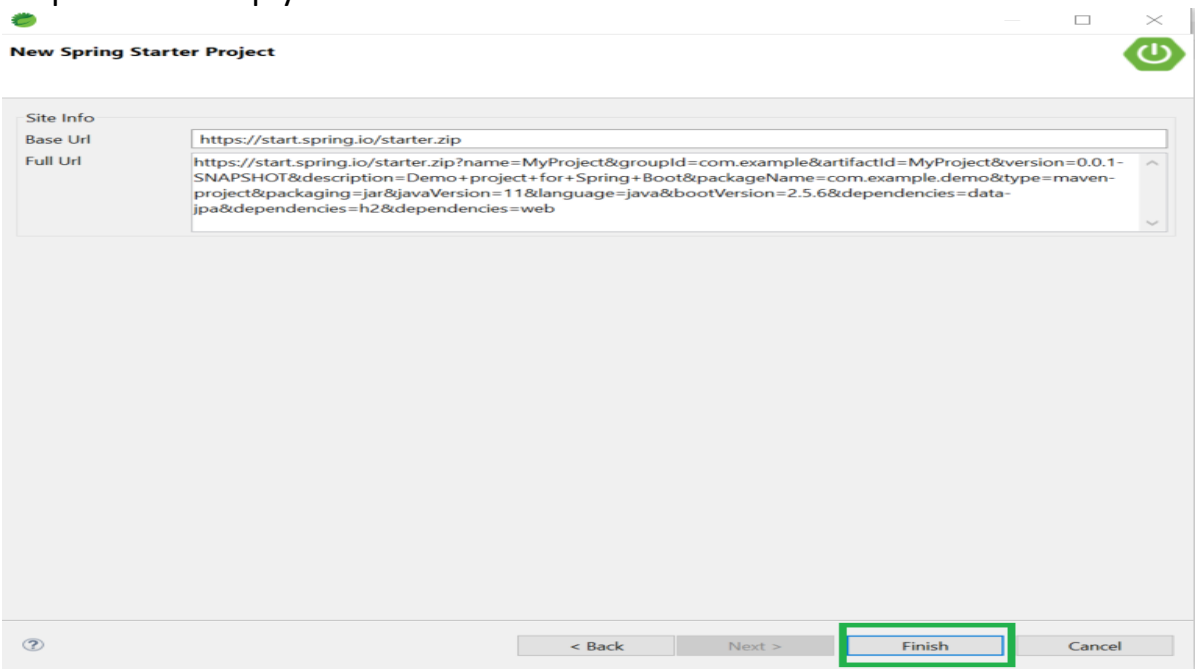
Package: Your package name



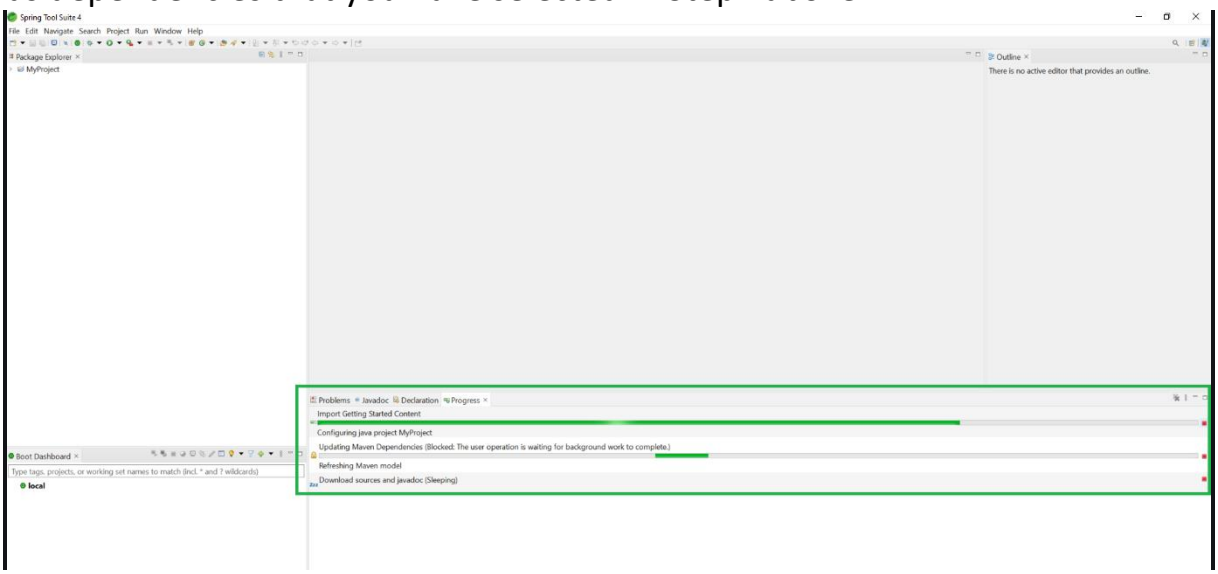
Step 4: Choose your required Spring Boot Version and select your dependencies as per your project requirement. And at last click on the Next button.



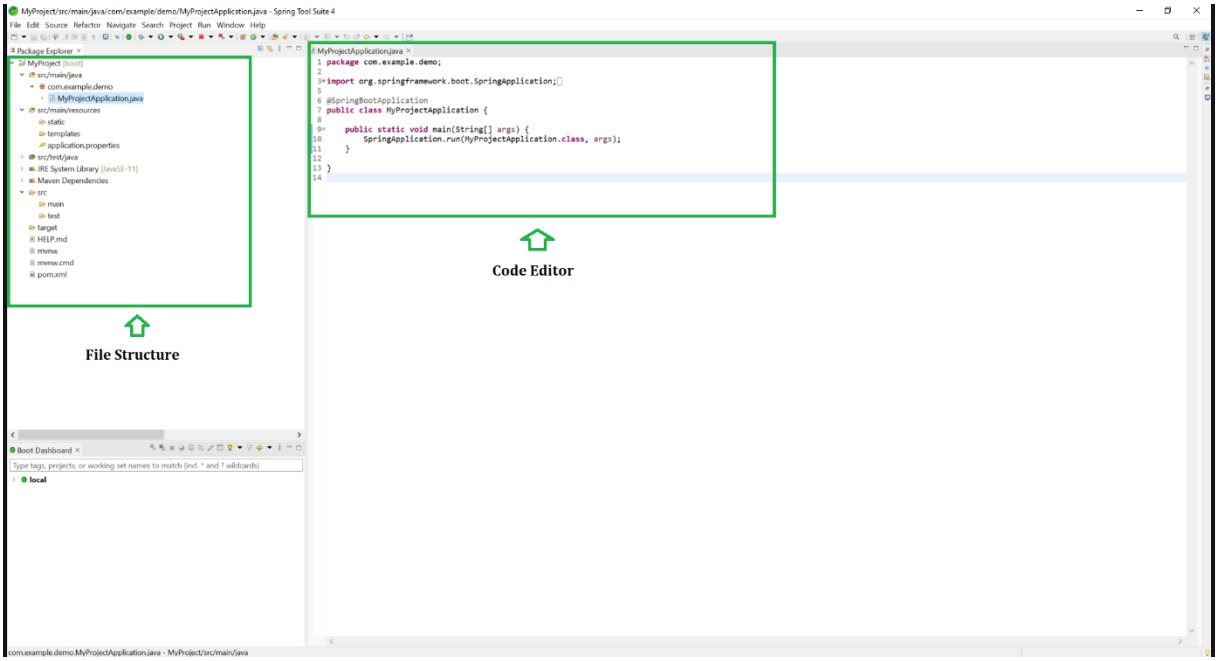
Step 5: Now simply click on the Finish button.



Here now, please wait for some time to download all the required files such as dependencies that you have selected in Step4 above.



Below is the Welcome screen after you have successfully Created and Setup Spring Boot Project in Spring Tool Suite


File Structure
Code Editor

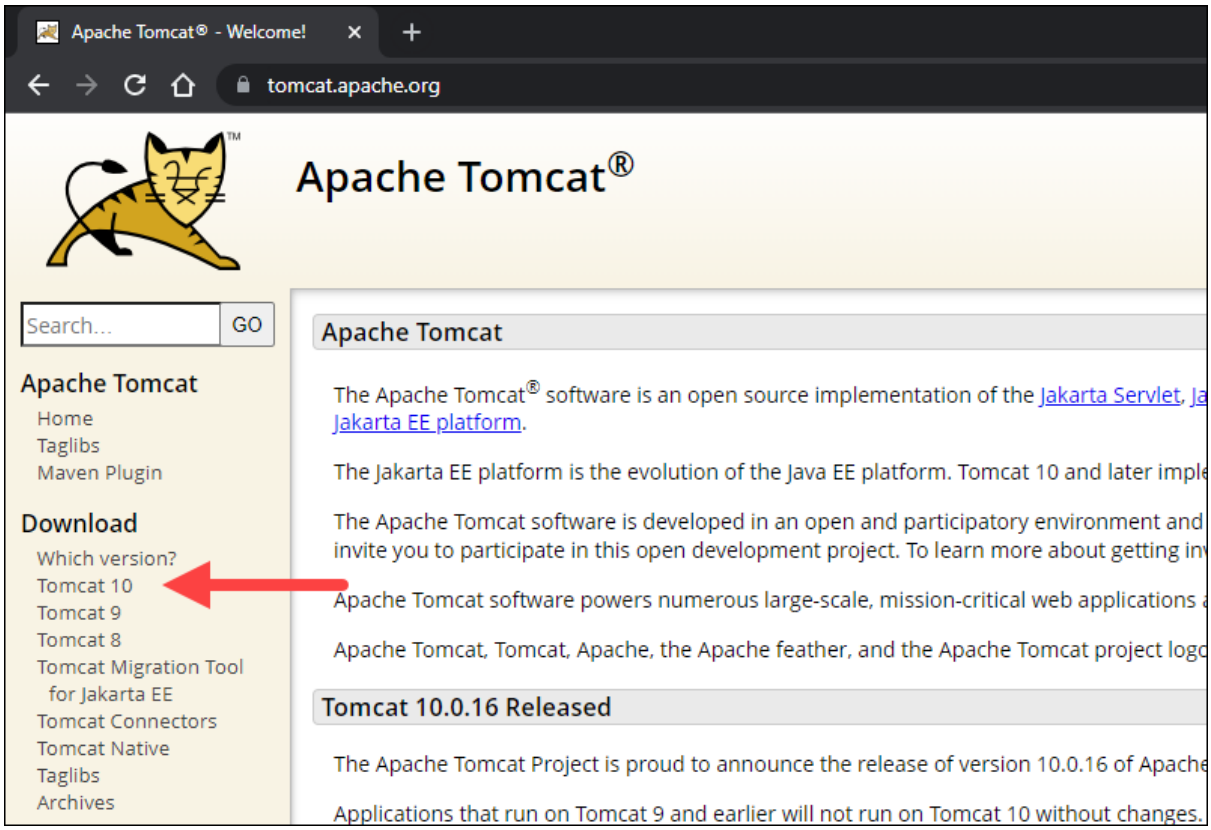
Experiment – 10

Create and Run Servlet class using Tomcat server

Step 1: Download Tomcat for Windows

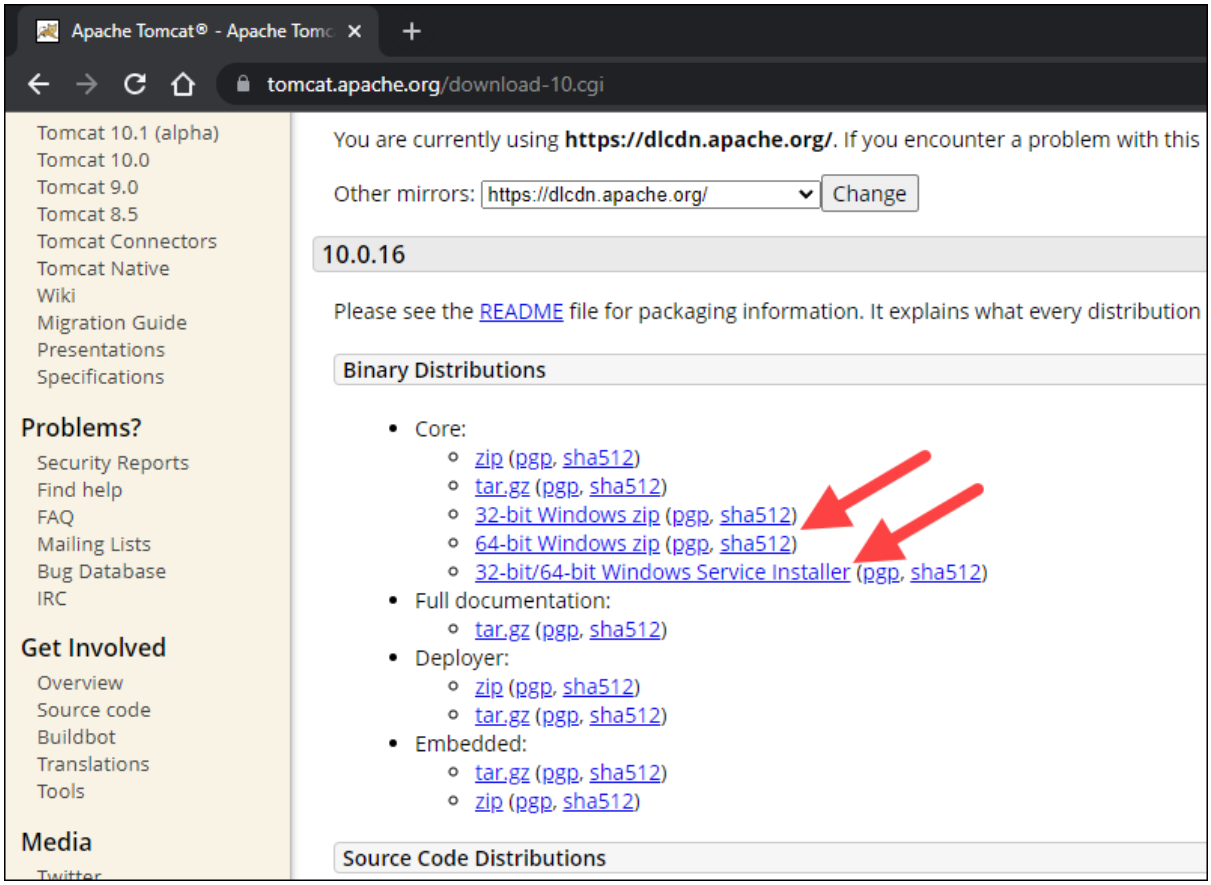
To download the Tomcat installation file, follow the steps below:

1. Browse to the official Apache Tomcat website. Locate the *Download* section and click the **latest Tomcat version** available. At the time of writing this article, the latest Tomcat version was version 10.



2. On the *Download* page, scroll down and locate the *Binary Distributions* area.

In the *Core* list, depending on the installation type you prefer, click the download link for the **Windows Service Installer** or the **32bit/64bit Windows zip file**.



Step 2: Install Tomcat

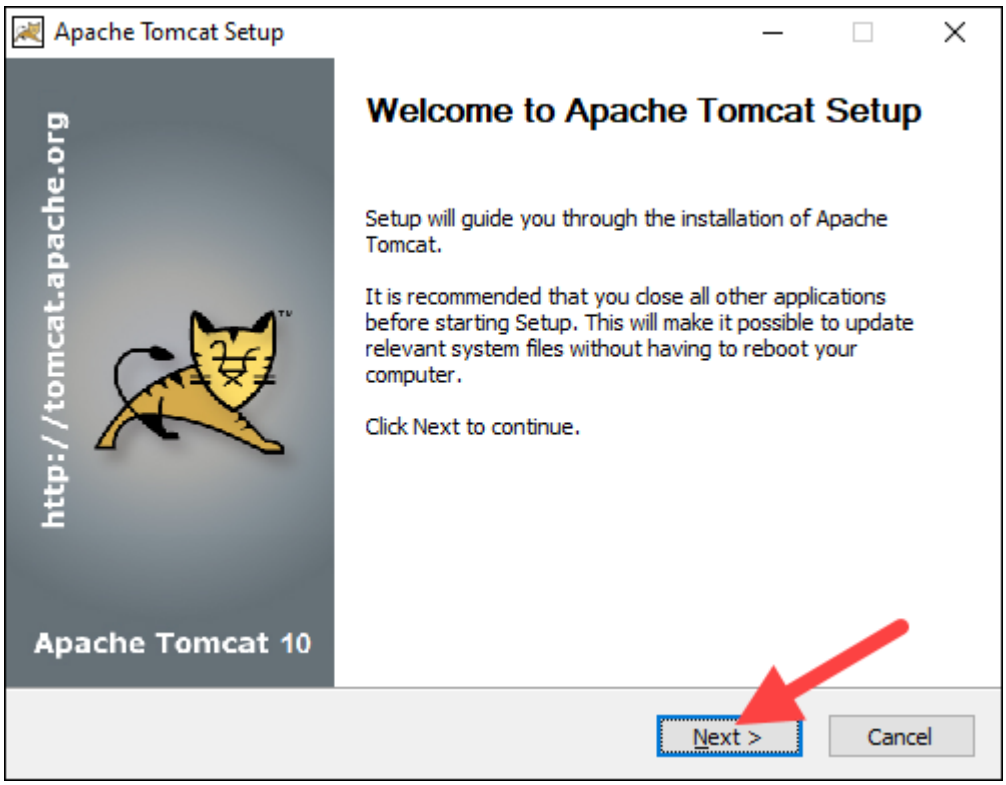
Install Tomcat via the **Windows Service Installer** for an automated and wizard-guided experience. The service installer installs the Tomcat service and runs it automatically when the system boots.

Method 1: Install Tomcat Using the Windows Service Installer

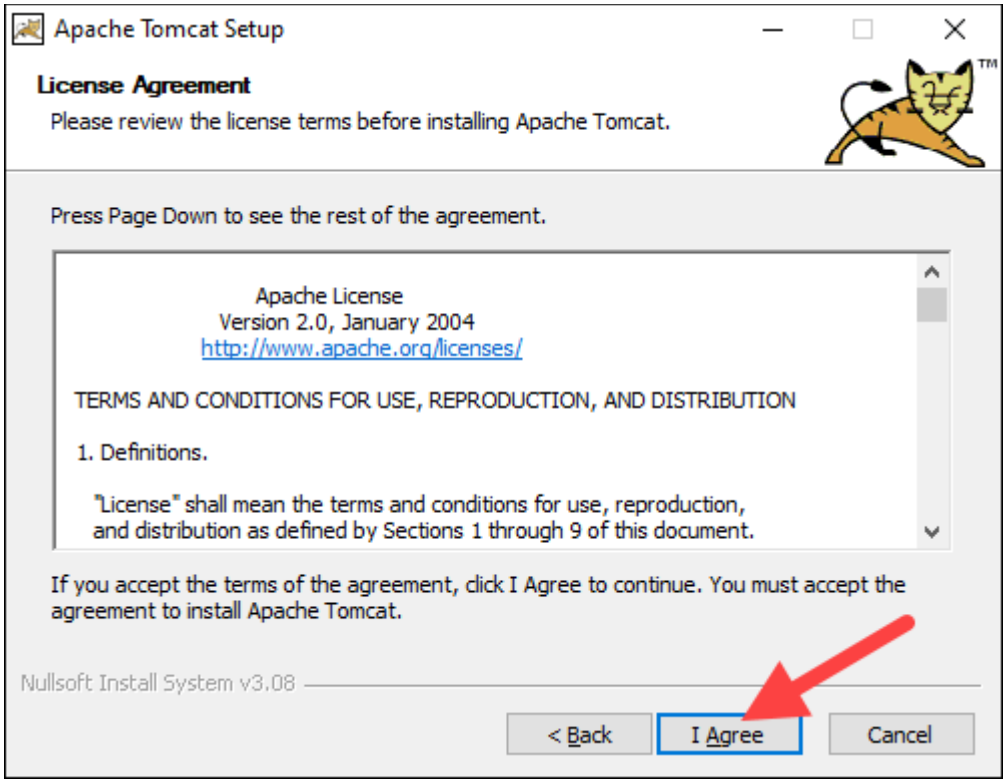
Follow the steps below to install Tomcat using the Windows Service Installer.

1. Open the downloaded **Windows Service Installer** file to start the installation process.

2. In the Tomcat Setup welcome screen, click **Next** to proceed.

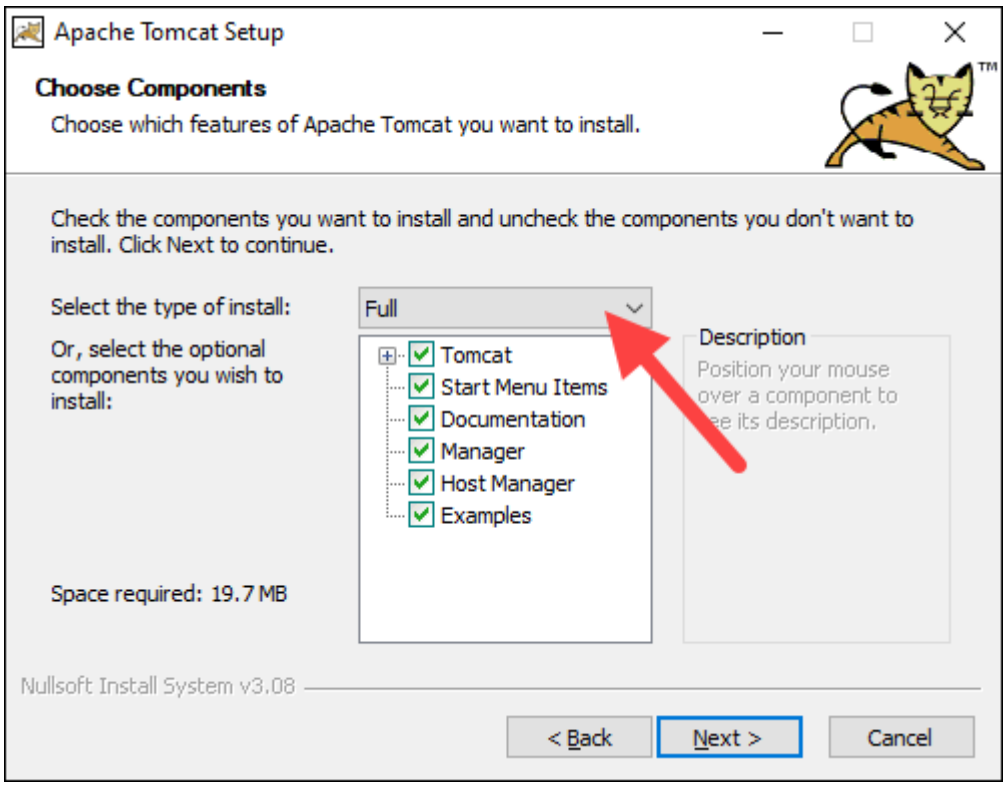


3. Read the License Agreement and if you agree to the terms, click **I Agree** to proceed to the next step.

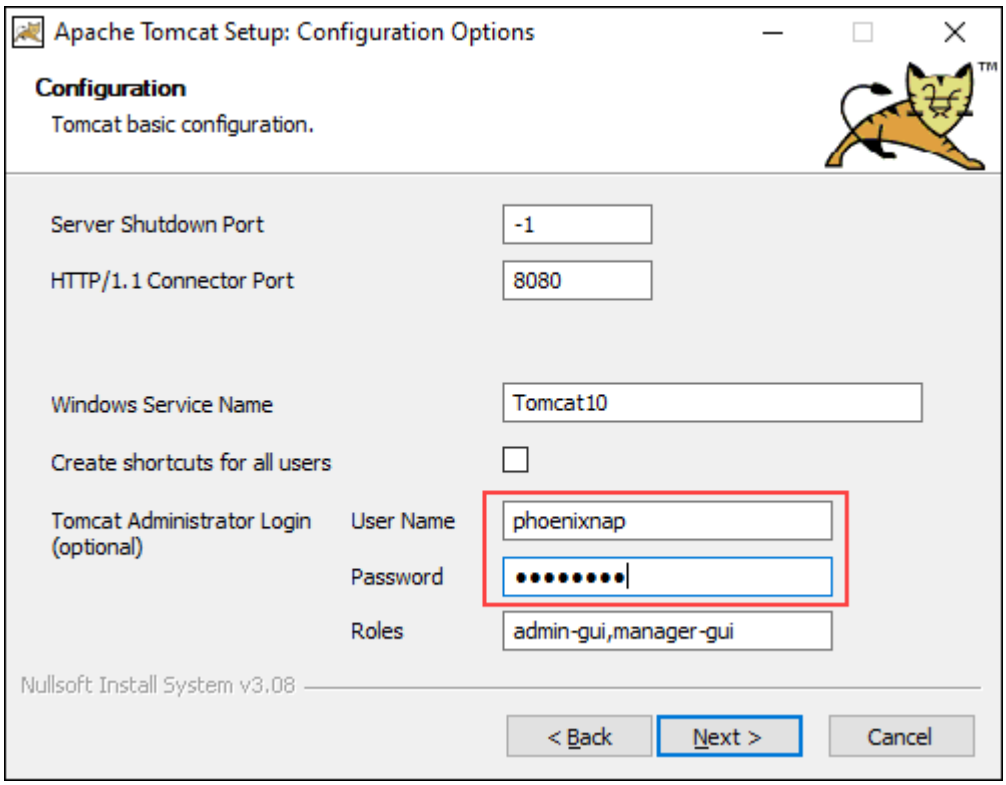


4. In the Tomcat component selection screen, choose **Full** in the dropdown menu to ensure the wizard installs the Tomcat Host Manager and Servlet and

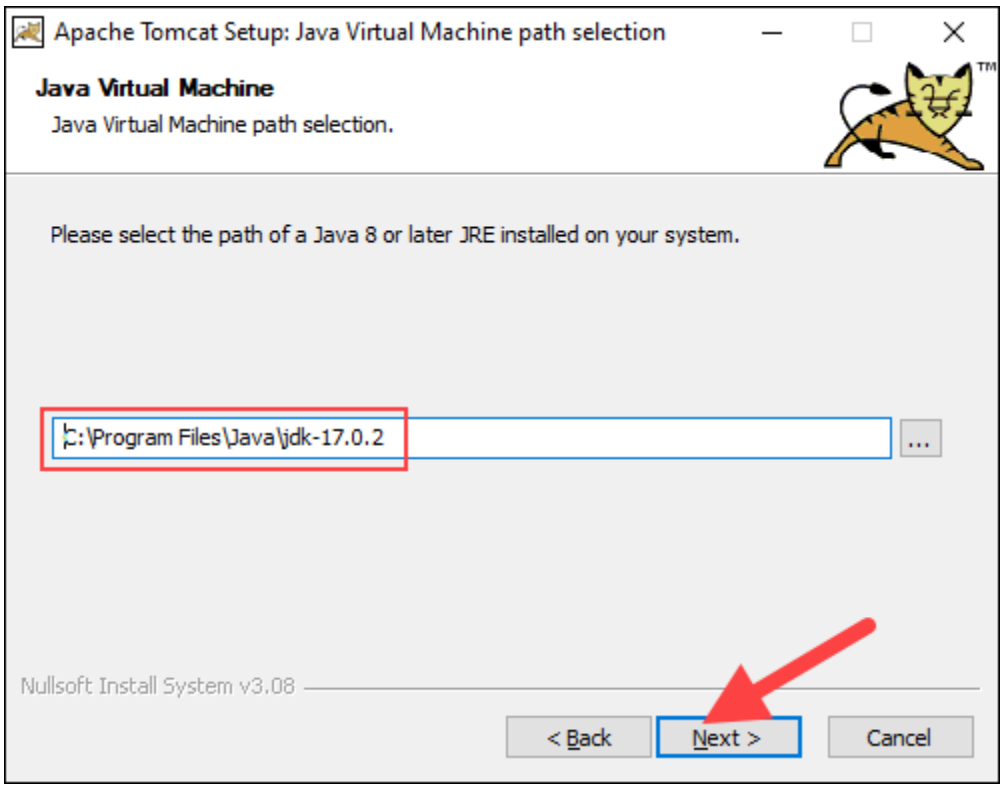
JSP examples web applications. Alternatively, keep the default **Normal** installation type and click **Next**.



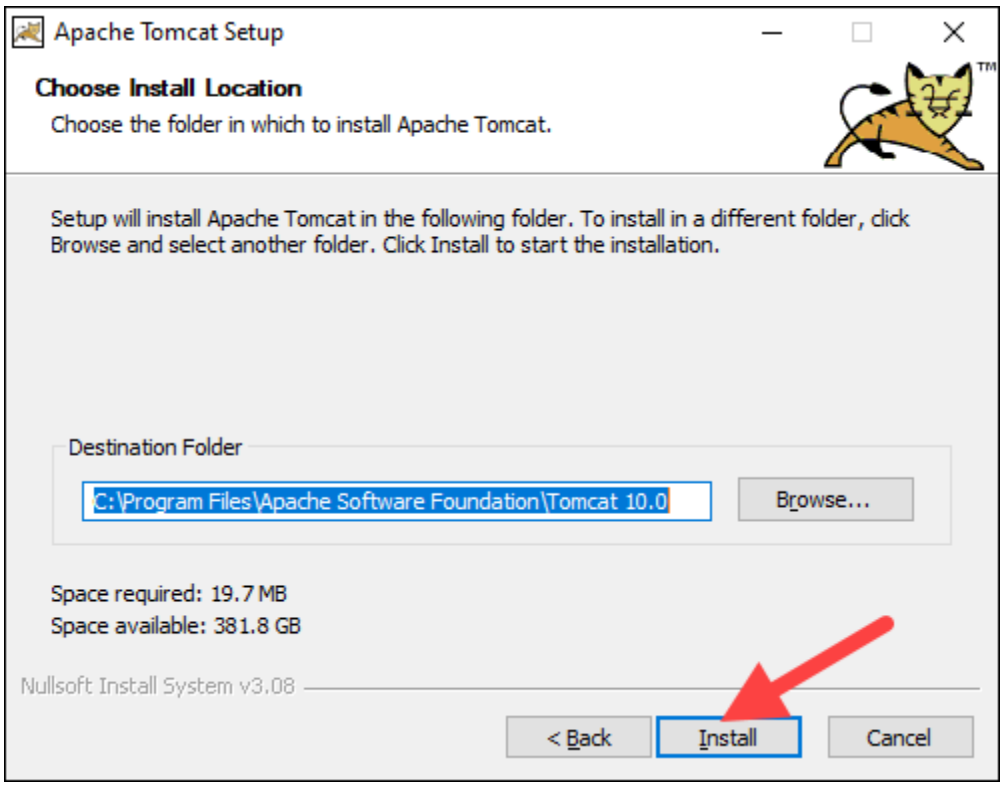
5. The next step configures the Tomcat server. For instance, enter the **Administrator login credentials** or choose a different **connection port**. When finished, click **Next** to proceed to the next step.



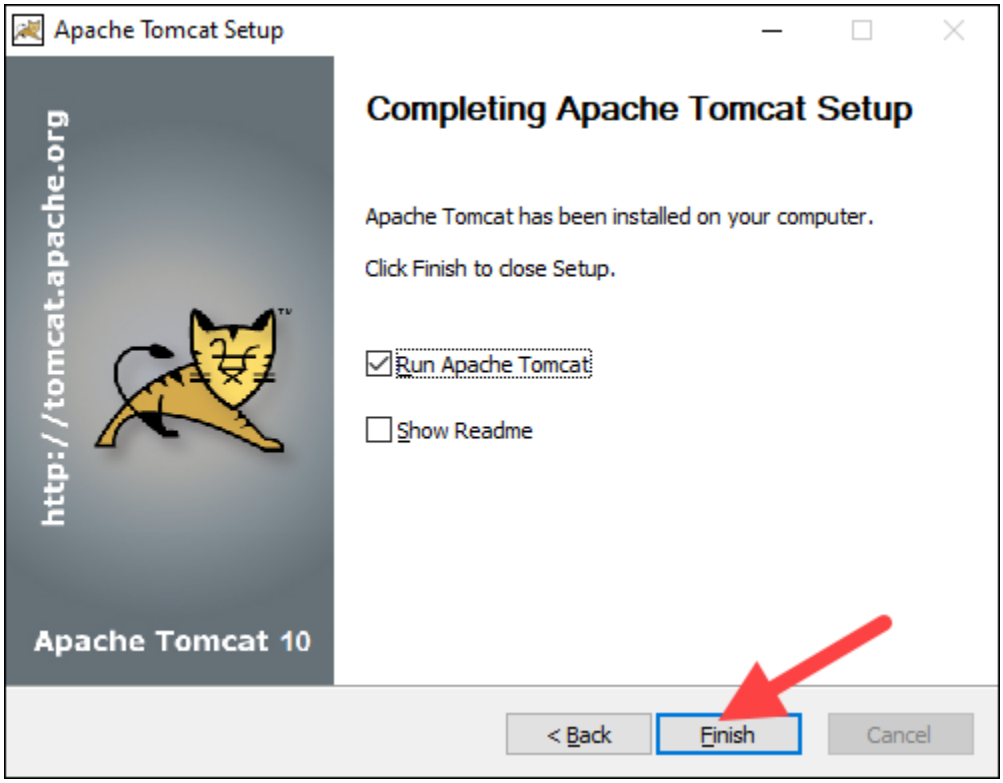
6. The next step requires you to enter the full path to the JRE directory on your system. The wizard auto-completes this if you have previously set up the Java environment variables. Click **Next** to proceed to the next step.



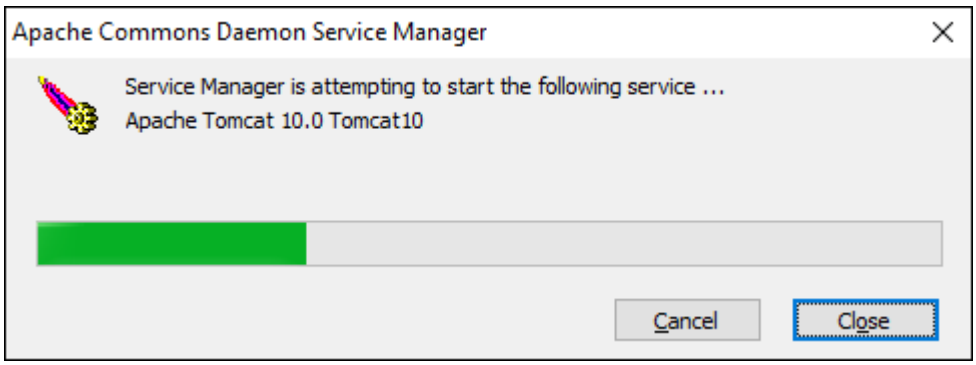
7. Choose the Tomcat server install location or keep the default one and click **Install**.



8. Check the **Run Apache Tomcat** box to start the service after the installation finishes. Optionally, check the **Show Readme** box to see the Readme file. To complete the installation, click **Finish**.



9. A popup window appears that starts the Tomcat service. After the process completes, the window closes automatically. The Apache Tomcat web server is now successfully installed .



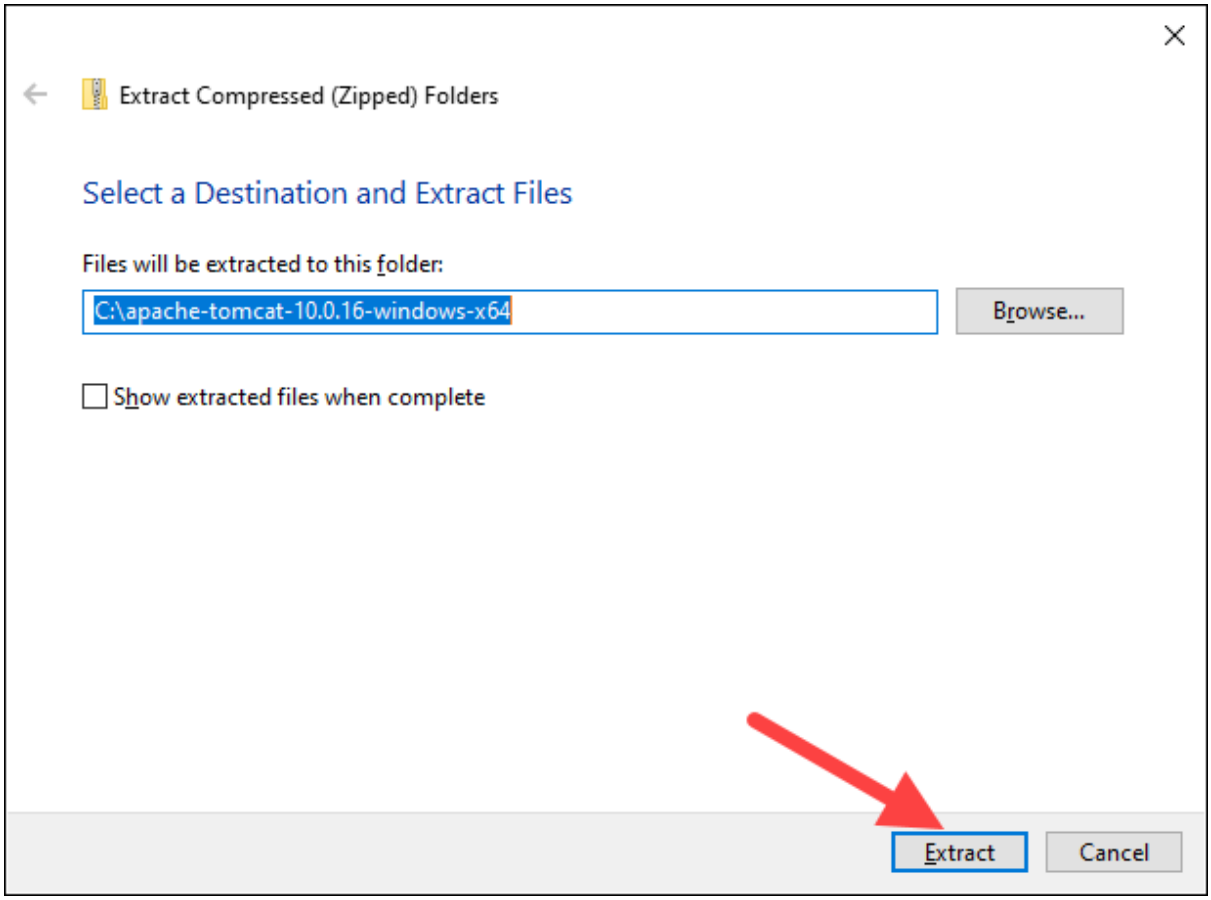
Method 2: Install Tomcat Using the zip Archive

Follow the steps below to set up the Tomcat server using the **zip archive**.

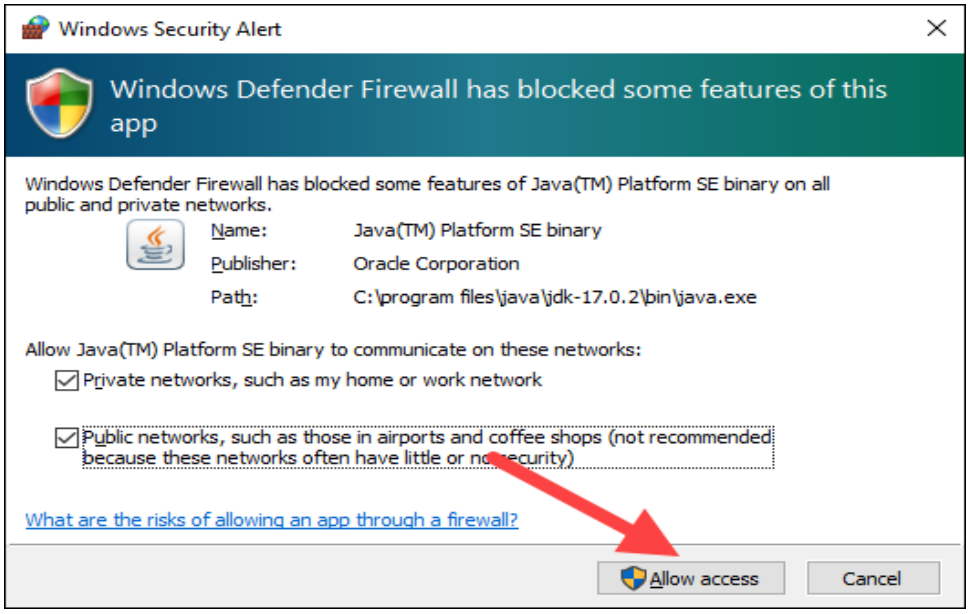
Note: Apache is part of the popular LAMP stack.

1. After downloading the **32bit/64bit Windows zip file,** depending on your Windows version, **unzip** the downloaded file. Right-click the file and select **Extract all...**

2. Choose where to extract the archive contents. For easier navigation, we recommend extracting it to the hard drive's root. Optionally, give the directory a shorter name to facilitate server configuration later. Click **Extract** to start the process.

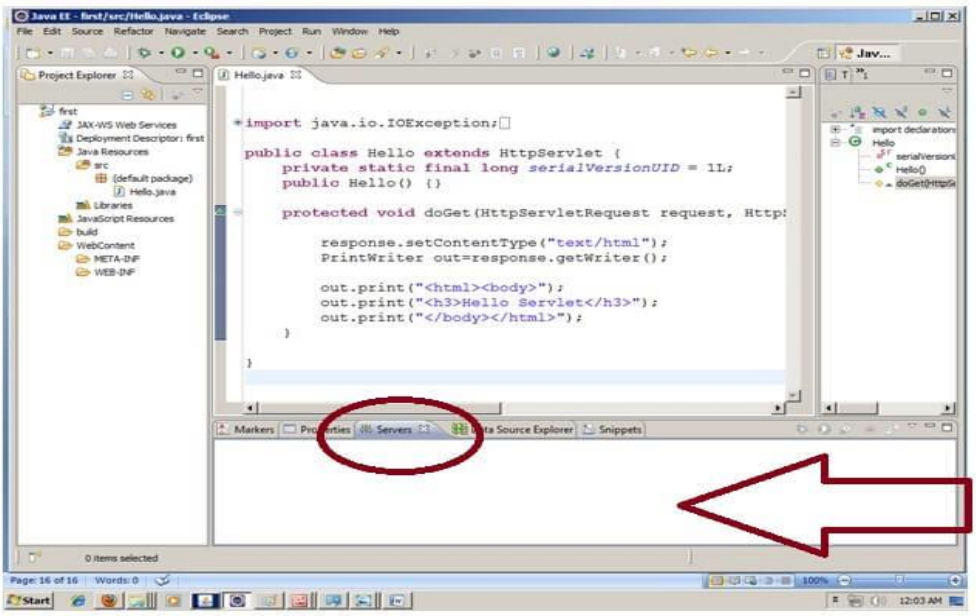


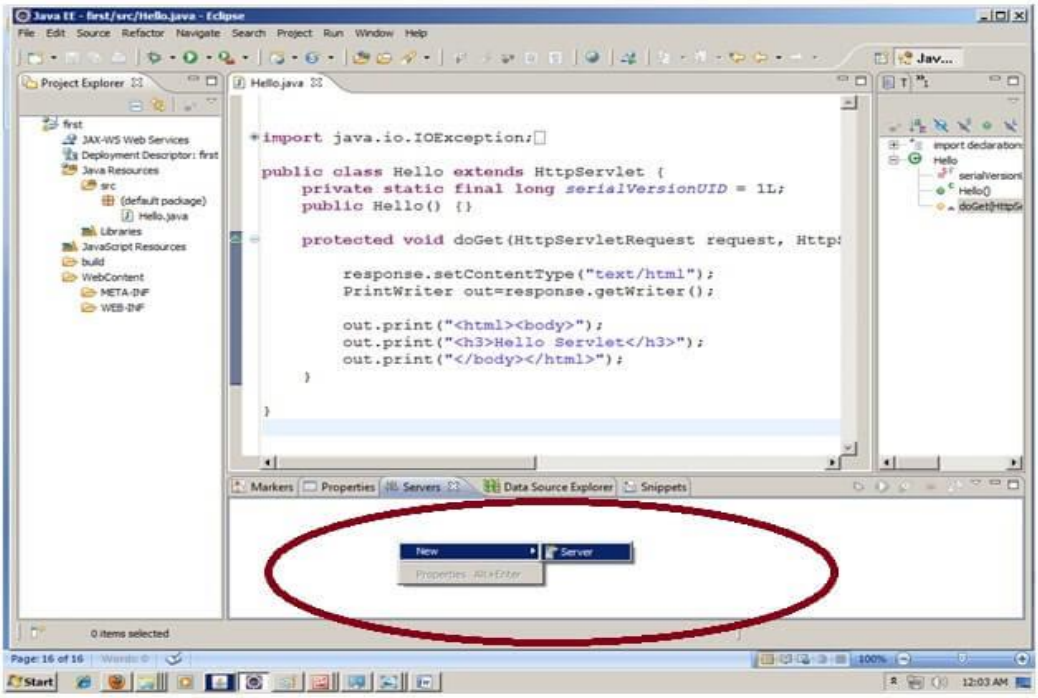
3. Add an exception for Tomcat in the firewall:

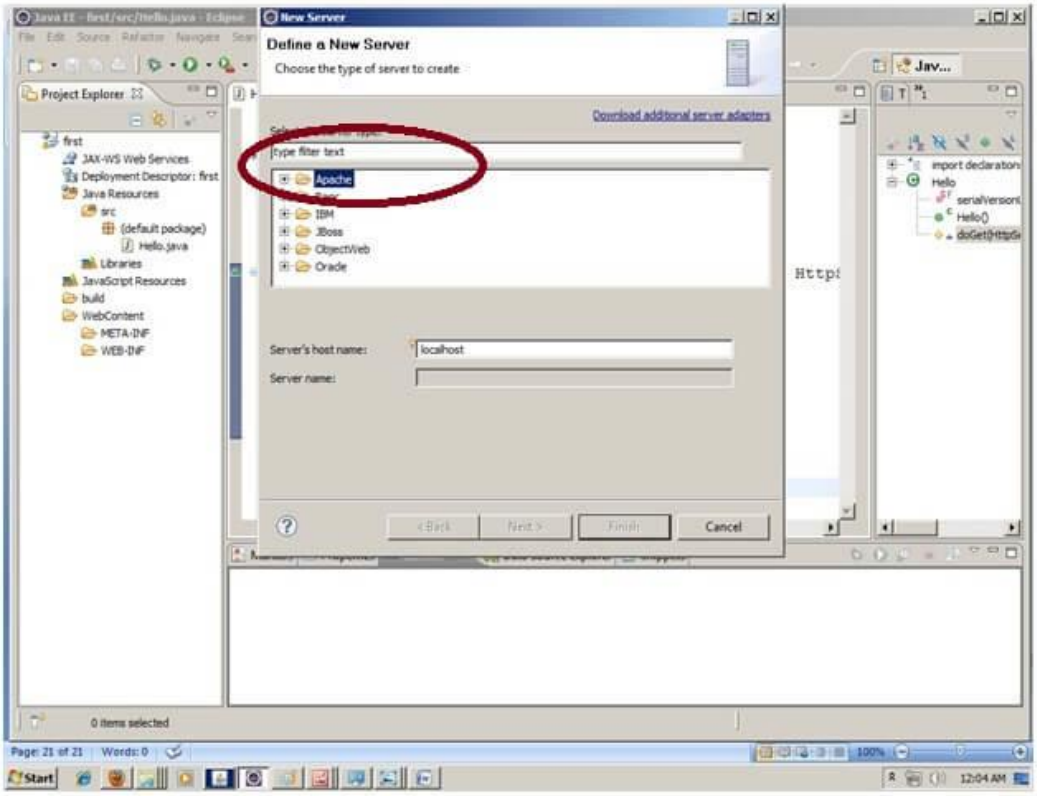


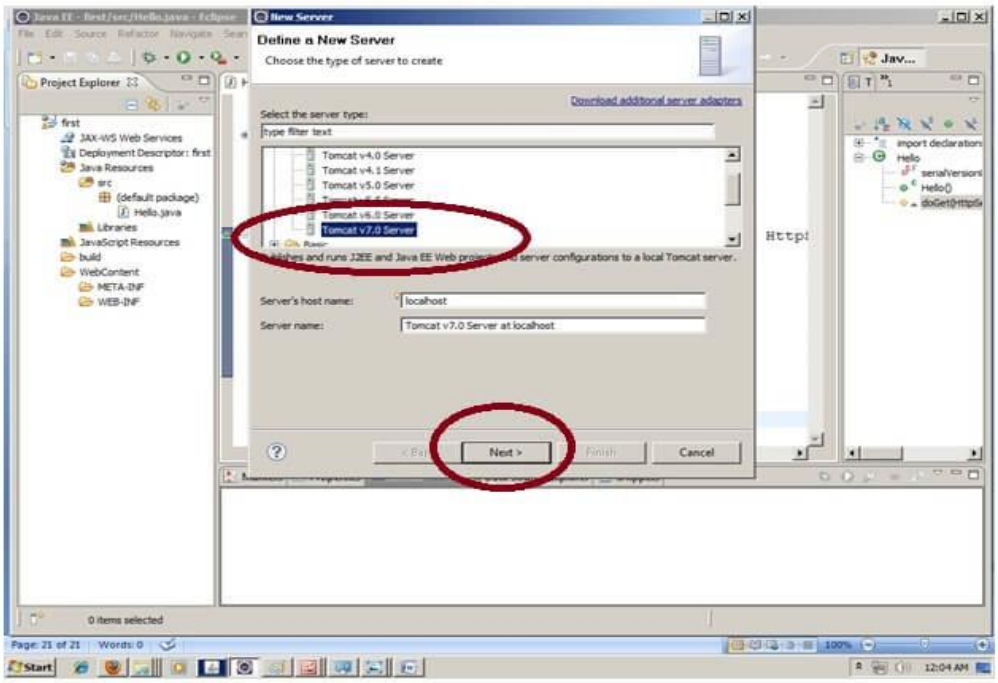
How to configure tomcat server in Eclipse IDE

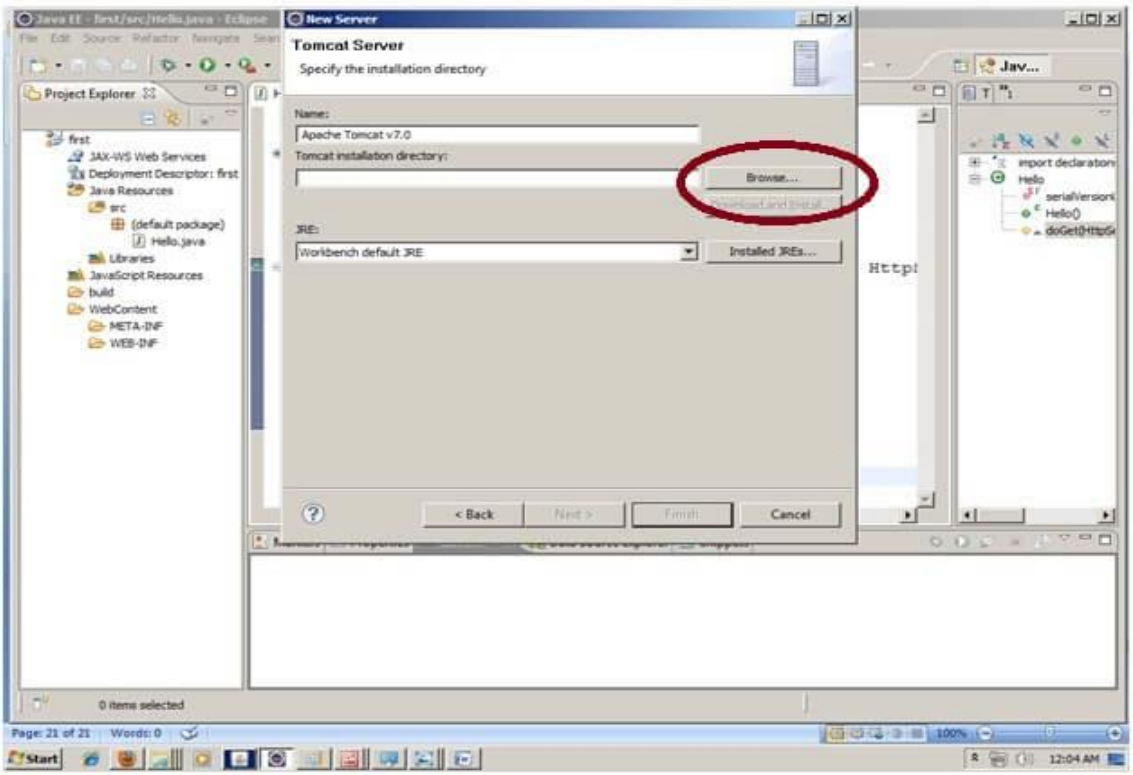
If you are using Eclipse IDE first time, you need to configure the tomcat server First. For co server in eclipse IDE, click on servers tab at the bottom side of the IDE -> right click on bl Servers -> choose tomcat then its version -> next -> click on Browse button -> select the apacl previous to bin -> next -> addAll -> Finish.

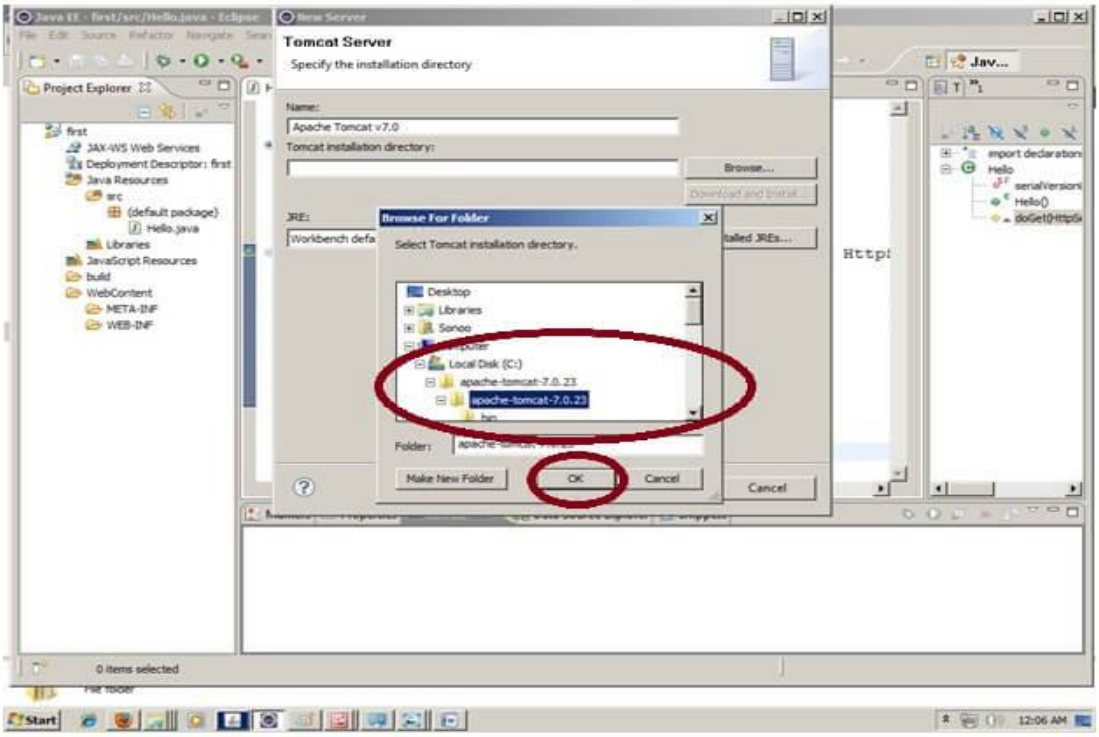


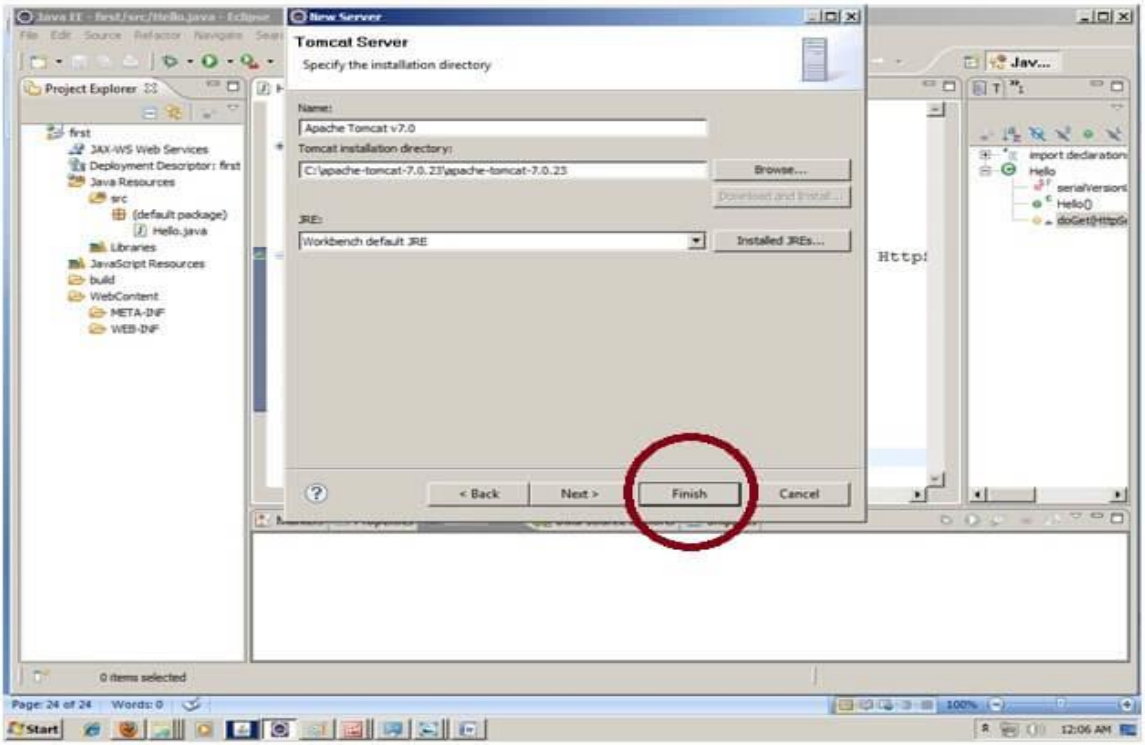

Java EE - first/src/Hello.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Project Explorer
Hello.java
import java.io.IOException;
public class Hello extends HttpServlet {
private static final long serialVersionUID = 1L;
public Hello() {}
protected void doGet(HttpServletRequest request, Http
response.setContentType("text/html");
PrintWriter out=response.getWriter();
out.print("<html><body>");
out.print("<h3>Hello Servlet</h3>");
out.print("</body></html>");
}
}
Markers Properties Servers Data Source Explorer Snippets
New
Server
0 items selected
Start
12:03 AM

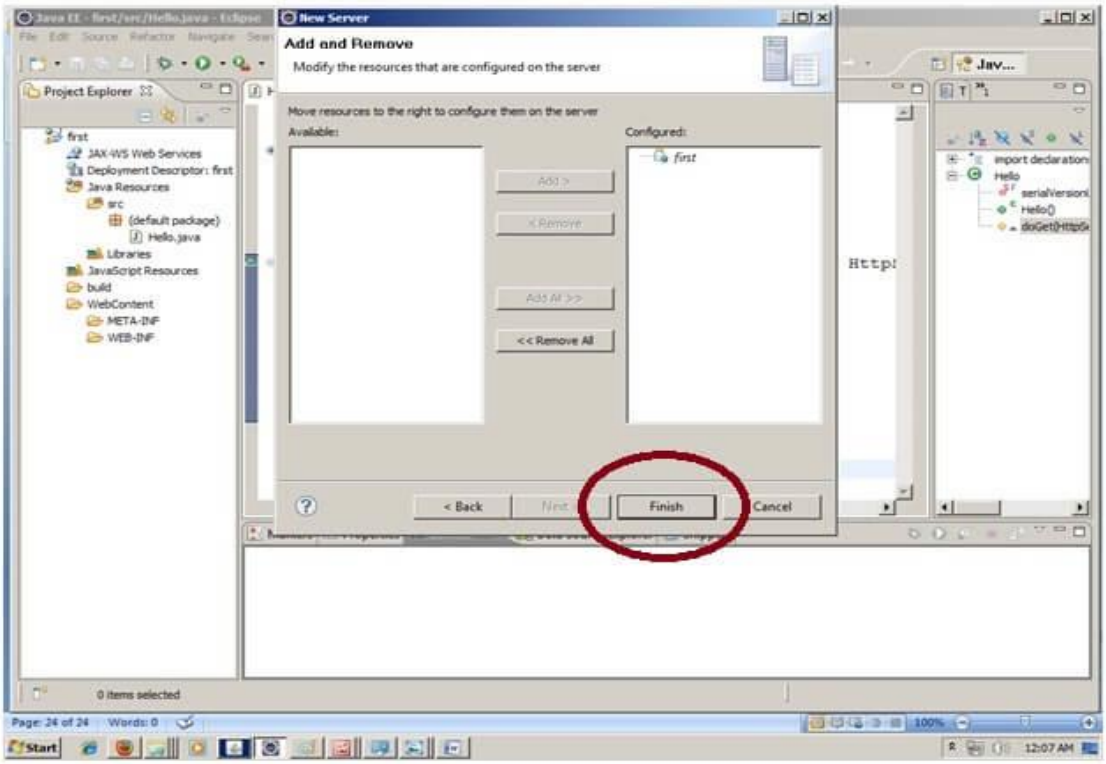

New Server
Define a New Server
Choose the type of server to create
Download additional server adapters
type filter text
Apache
IBM
JBoss
ObjectWeb
Oracle
Server's host name:
localhost
Server name:
< Back
Next >
Finish
Cancel
0 items selected
Start
12:04 AM

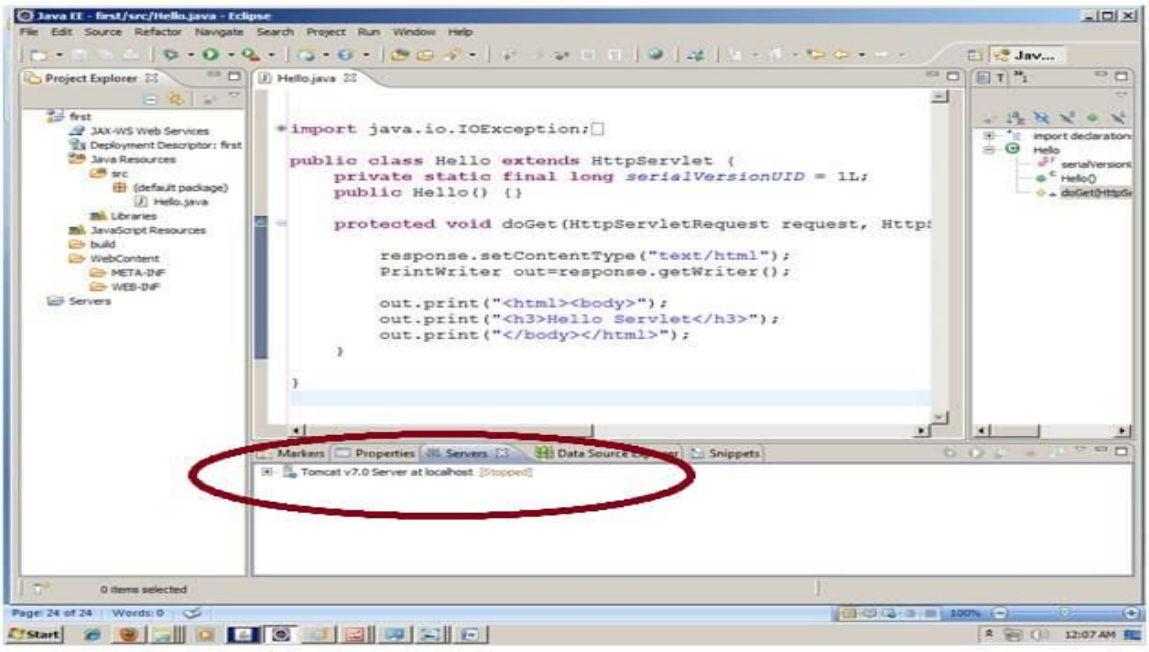

New Server
Define a New Server
Choose the type of server to create
Select the server type:
Tomcat v4.0 Server
Tomcat v4.1 Server
Tomcat v5.0 Server
Tomcat v6.0 Server
Tomcat v7.0 Server
Server's host name:
localhost
Server name:
Tomcat v7.0 Server at localhost
Next >
Cancel


New Server
Tomcat Server
Specify the installation directory
Name:
Apache Tomcat v7.0
Tomcat installation directory:
Browse...
JRE:
Workbench default JRE
Installed JREs...
< Back
Next >
Finish
Cancel


Tomcat Server
Specify the installation directory
Name:
Apache Tomcat v7.0
Tomcat installation directory:
Browse...
JRE:
Browse For Folder
Select Tomcat installation directory.
Desktop
Libraries
Local Disk (C:)
apache-tomcat-7.0.23
bin
Folder:
Make New Folder
OK
Cancel


Tomcat Server
Specify the installation directory
Name:
Apache Tomcat v7.0
Tomcat installation directory:
C:\apache-tomcat-7.0.23\apache-tomcat-7.0.23
Browse...
JRE:
Workbench default JRE
Installed JREs...
< Back
Next >
Finish
Cancel

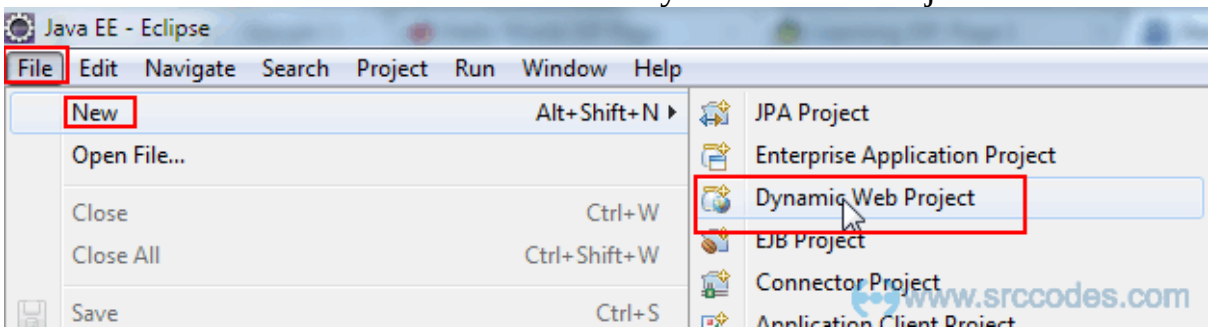

New Server
Add and Remove
Modify the resources that are configured on the server
Move resources to the right to configure them on the server
Available:
Configured:
first
Add >
< Remove
Add All >>
<< Remove All
< Back
Next >
Finish
Cancel
Project Explorer
first
JAX-WS Web Services
Deployment Descriptor: first
Java Resources
src
(default package)
Hello.java
Libraries
JavaScript Resources
build
WebContent
META-INF
WEB-INF
0 items selected


Java EE - first/src/Hello.java - Eclipse
Project Explorer
Hello.java
import java.io.IOException;
public class Hello extends HttpServlet {
private static final long serialVersionUID = 1L;
public Hello() {}
protected void doGet(HttpServletRequest request, Https
response.setContentType("text/html");
PrintWriter out=response.getWriter();
out.print("<html><body>");
out.print("<h3>Hello Servlet</h3>");
out.print("</body></html>");
}
}
Markers
Properties
Servers
Snippets
Tomcat v7.0 Server at localhost [Stopped]
0 items selected

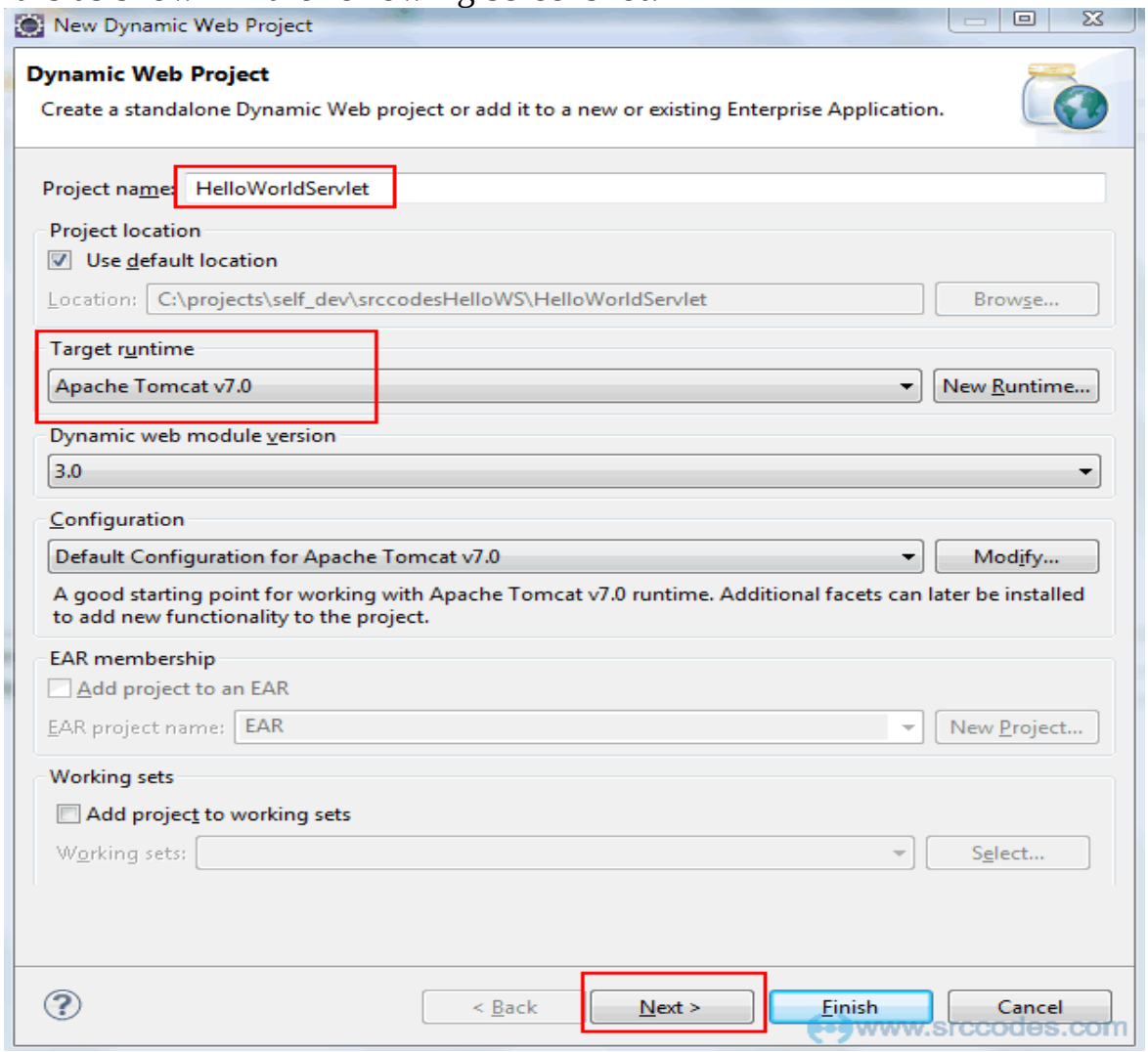
•

Create Dynamic Web Project

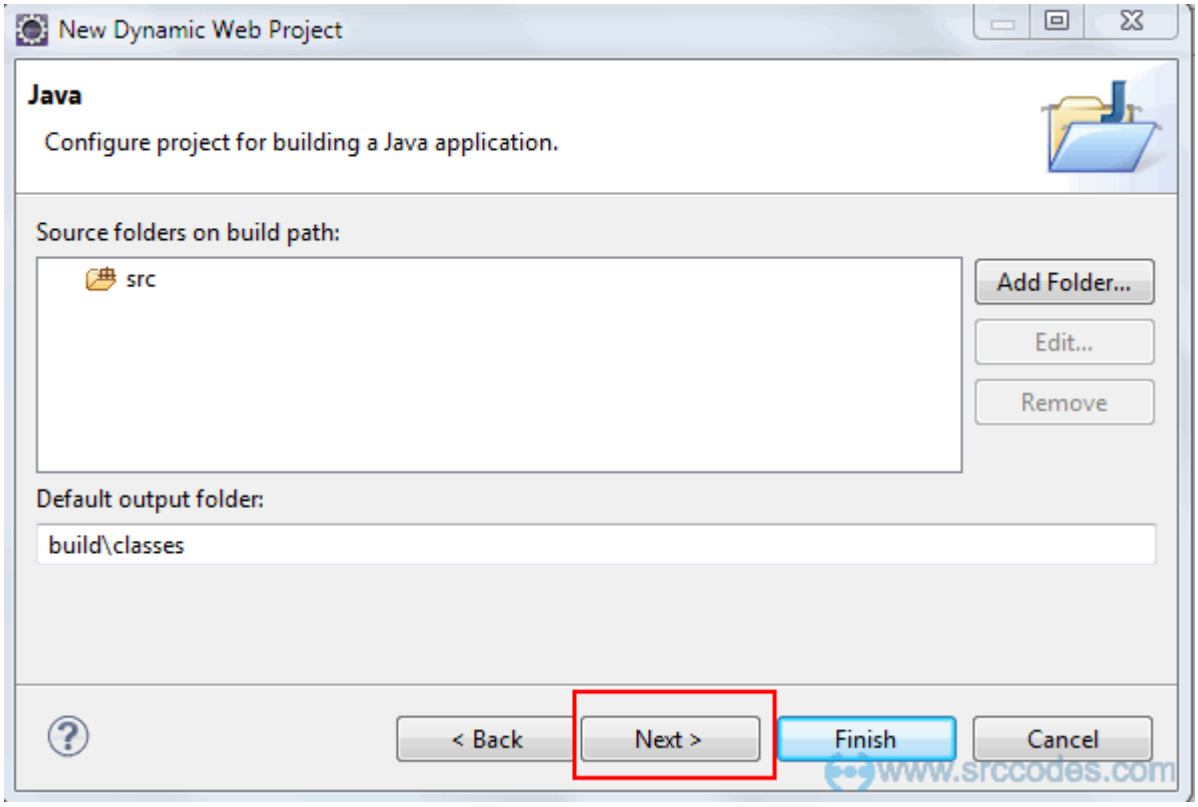
Select from the menu File --> New --> Dynamic Web Project.



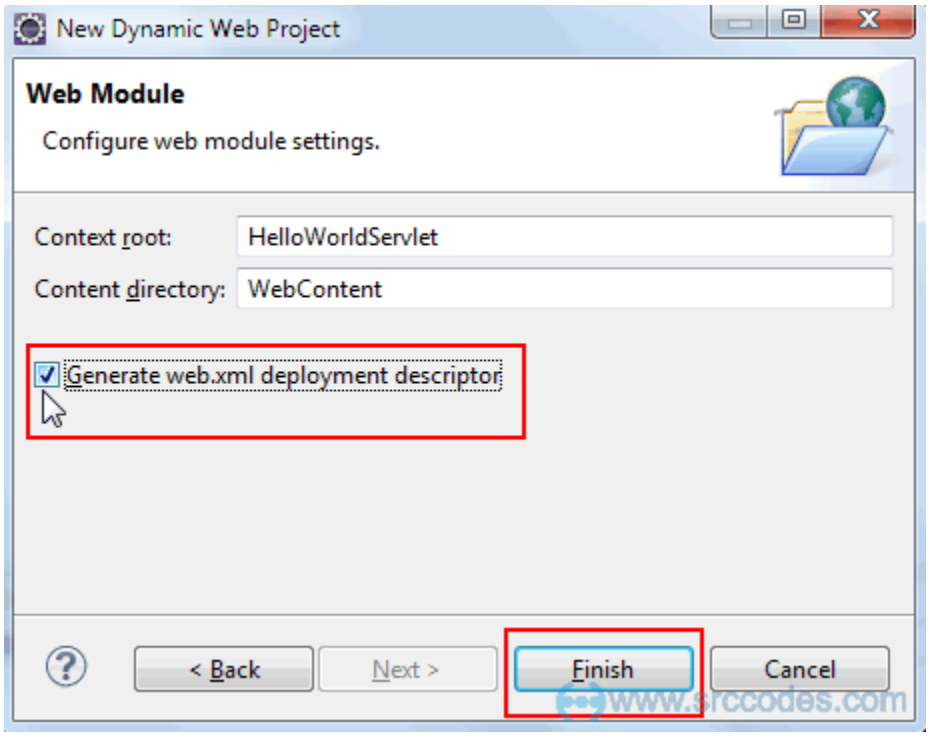
Enter "HelloWorldServlet" as the project name. Keep rest of the settings as it is as shown in the following screenshot.



Click "Next" button.

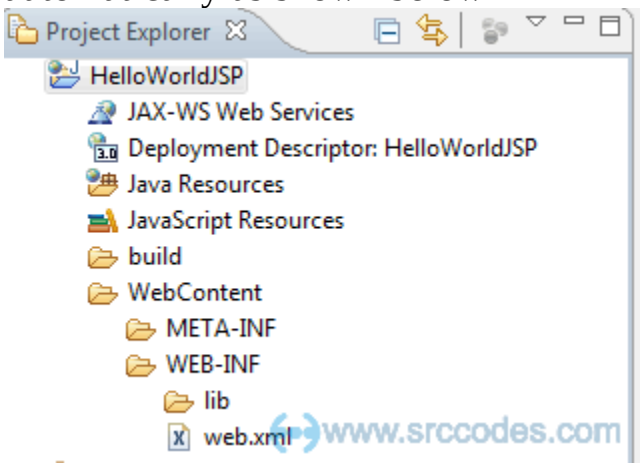


Click "Next" button.



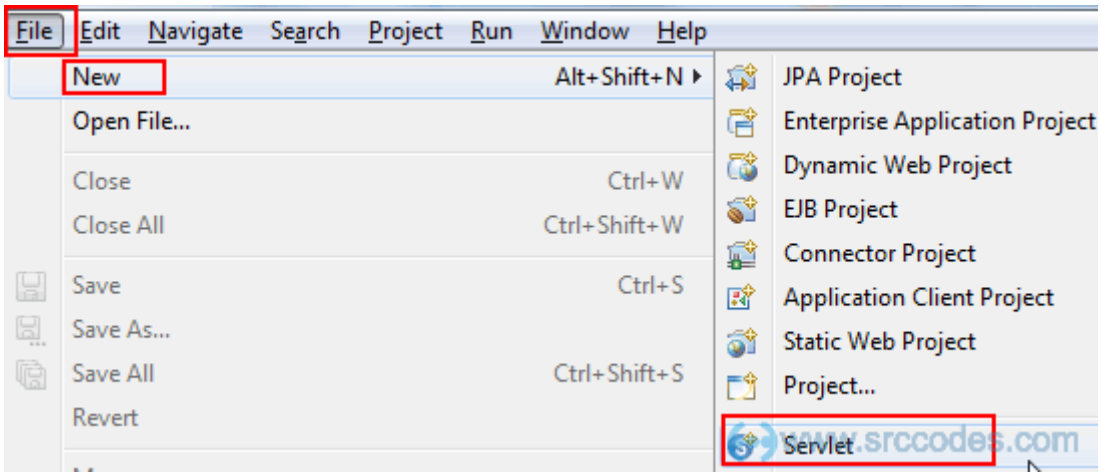
Check 'Generate web.xml deployment descriptor' checkbox and click "Finish" button and Eclipse IDE will generate the web project

automatically as shown below

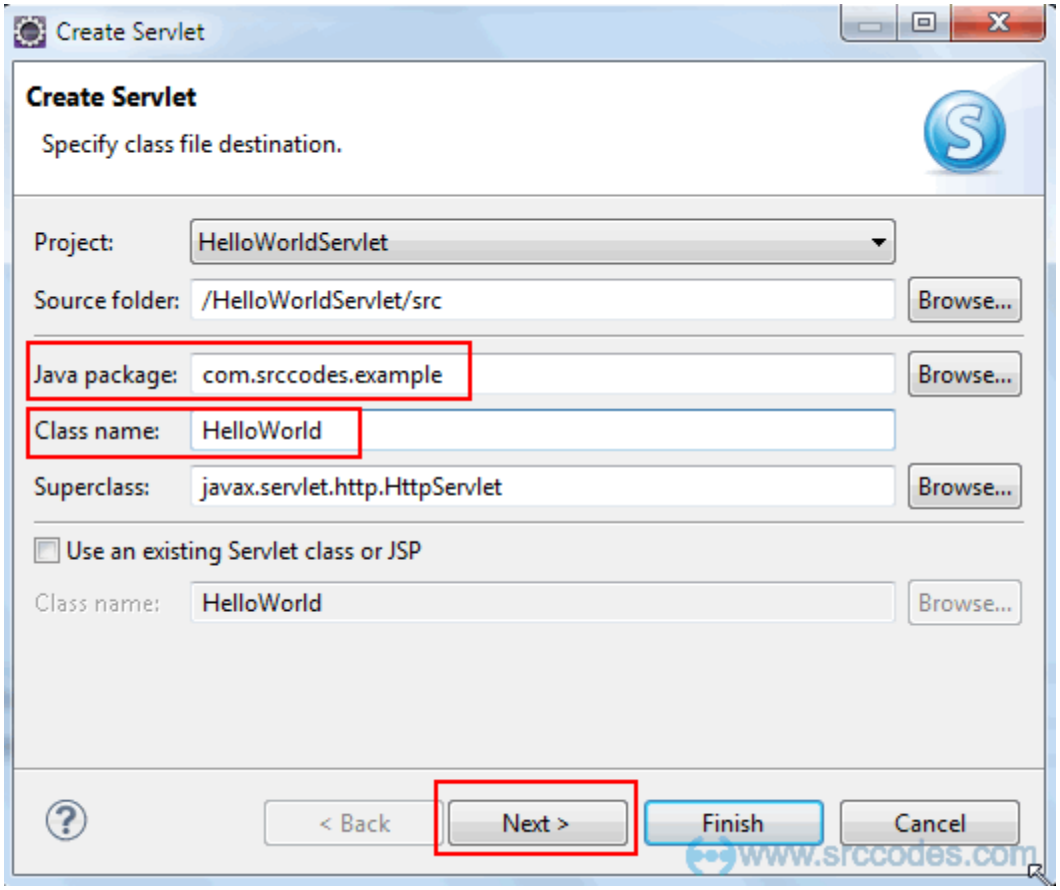


. Create Servlet Class

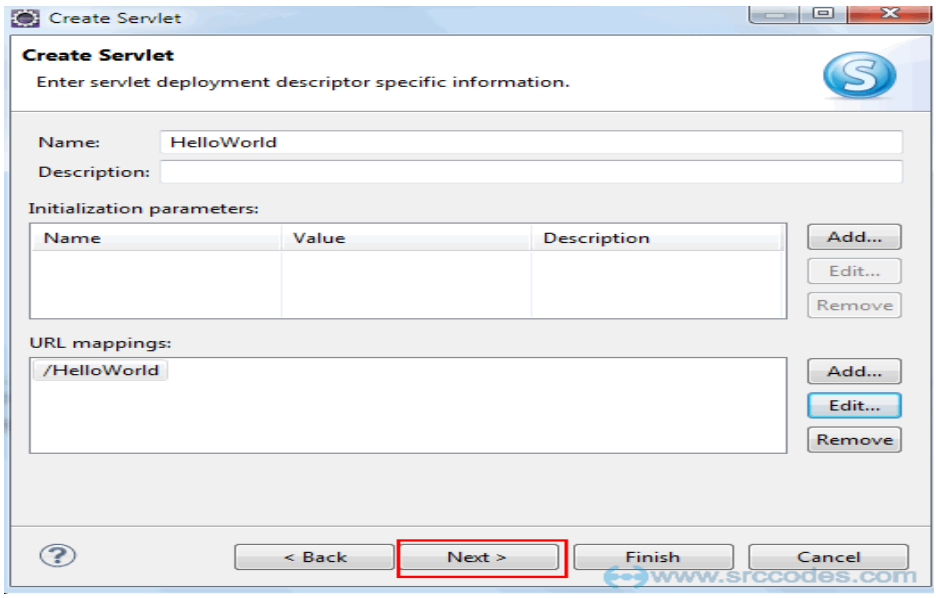
Select from the menu File --> New --> Servlet.



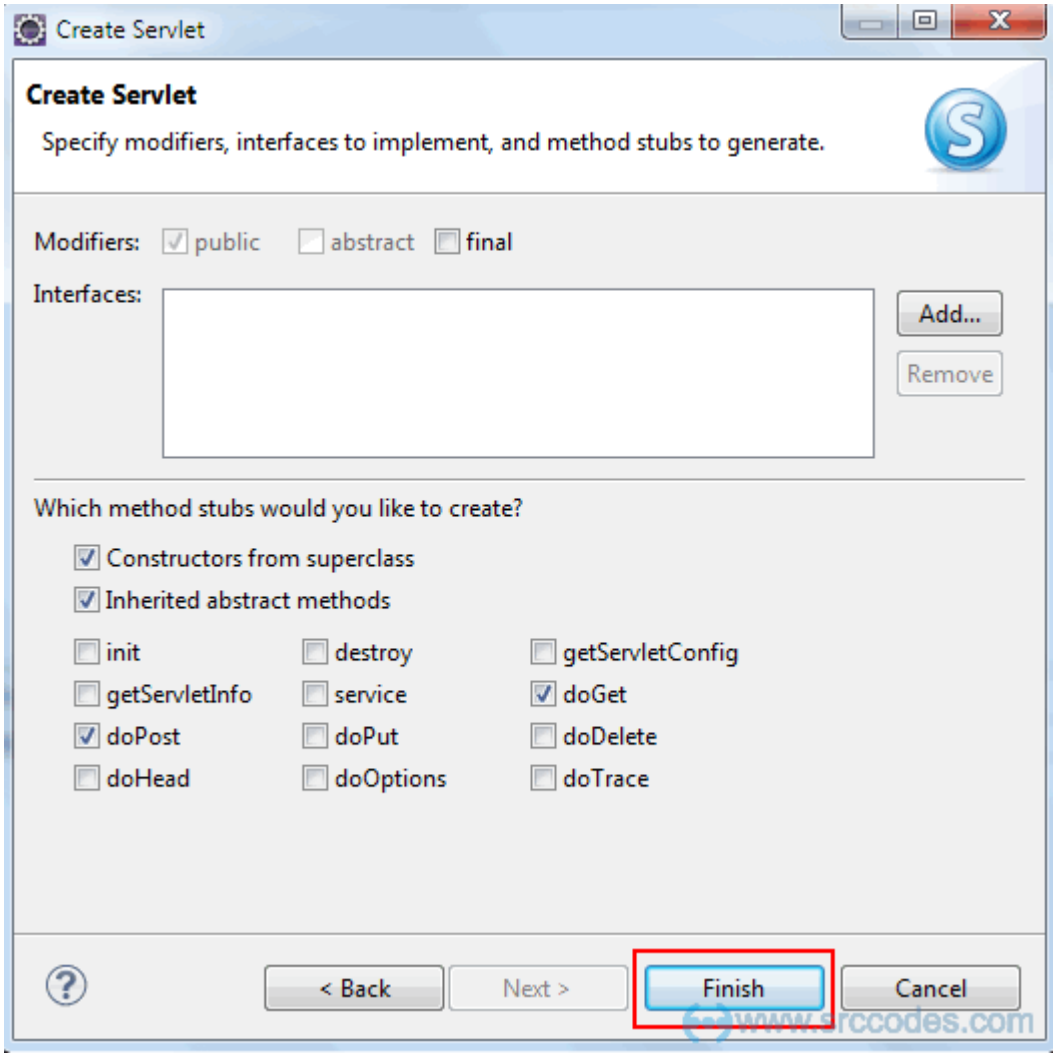
Write "com.srccodes.example" in the 'Java Package' field and "HelloWorld" in the 'Class Name' field. Click 'Next' button.



We can specify deployment descriptor (web.xml) specific information in the following screen. Just keep every thing as it is for the time being. Click "Next" button.



Click 'Next' button.



Eclipse will generate a Servlet class based on the configuration / input we provided in the previous steps and open the same in the Java Editor as shown below

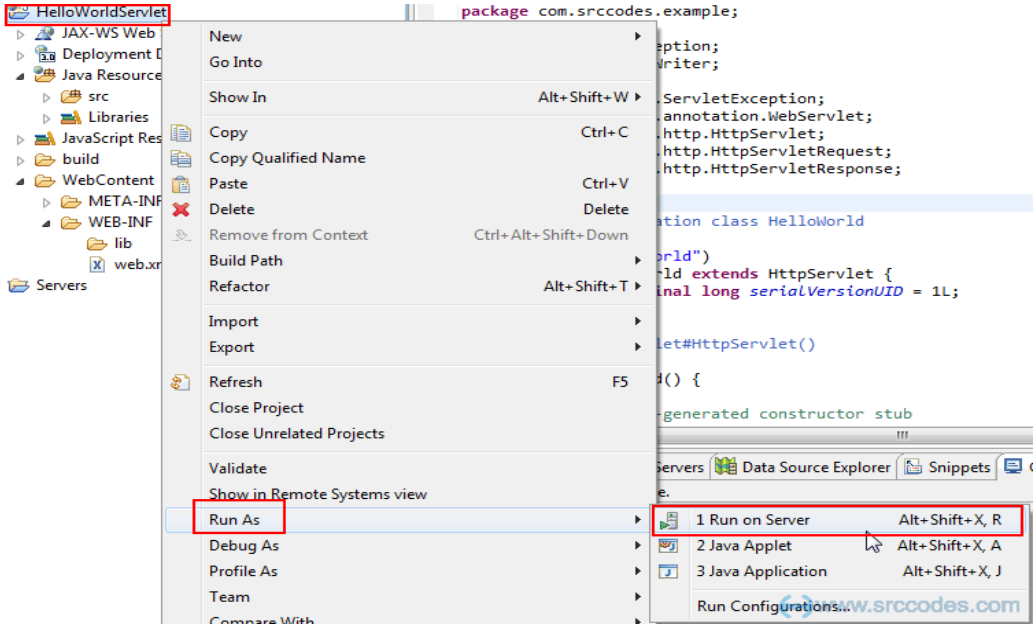
```
package com.srccodes.example;


import java.io.IOException;

import javax.servlet.ServletException;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;
```

```
import javax.servlet.http.HttpServletResponse;


/**
 * Servlet implementation class HelloWorld
 */
@WebServlet("/HelloWorld")
public class HelloWorld extends HttpServlet {
    private static final long serialVersionUID = 1L;


    /**
     * @see HttpServlet#HttpServlet()
     */
    public HelloWorld() {
        super();
        // TODO Auto-generated constructor stub
    }


    /**
     * @see HttpServlet#doGet(HttpServletRequest request,
HttpServletResponse response)
     */
    protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        // TODO Auto-generated method stub
```

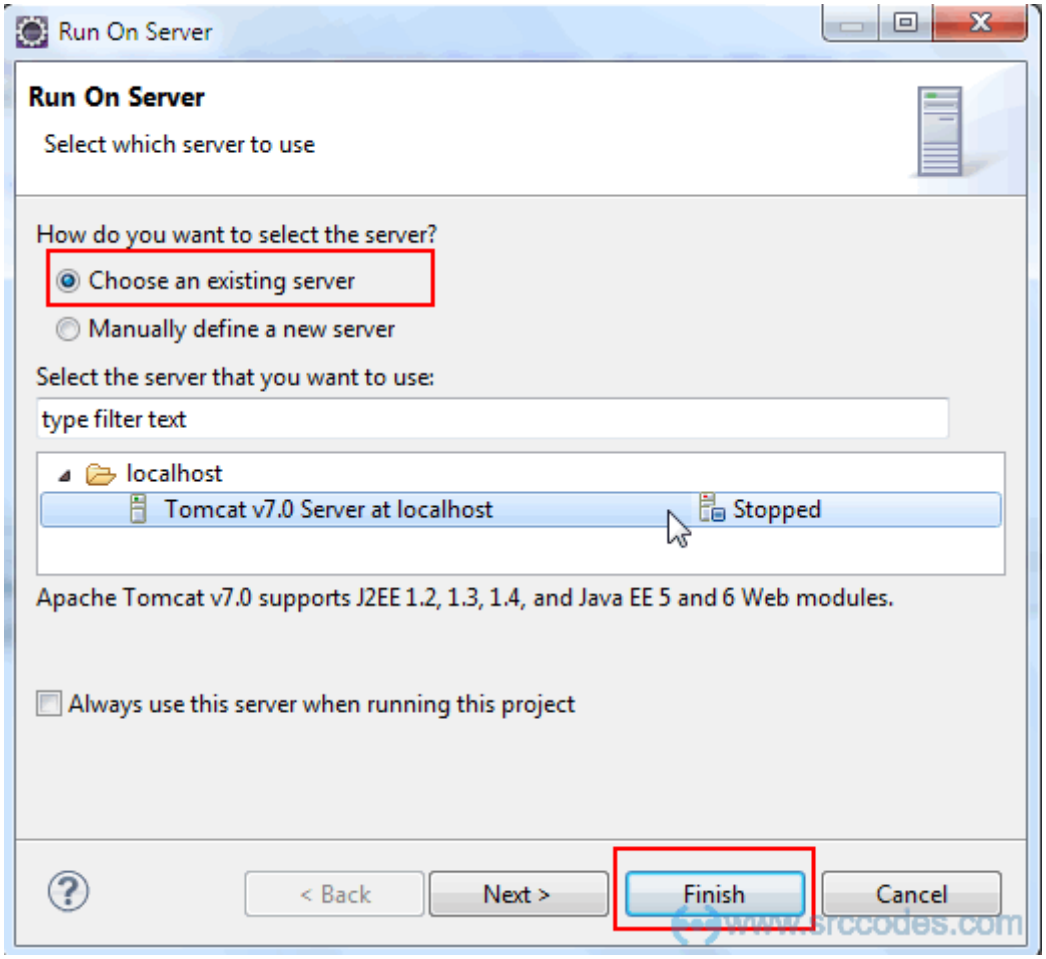
```
    }


    /**
     * @see HttpServlet#doPost(HttpServletRequest request,
HttpServletResponse response)
     */
    protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        // TODO Auto-generated method stub
    }


}
```

Run Your Servlet Code

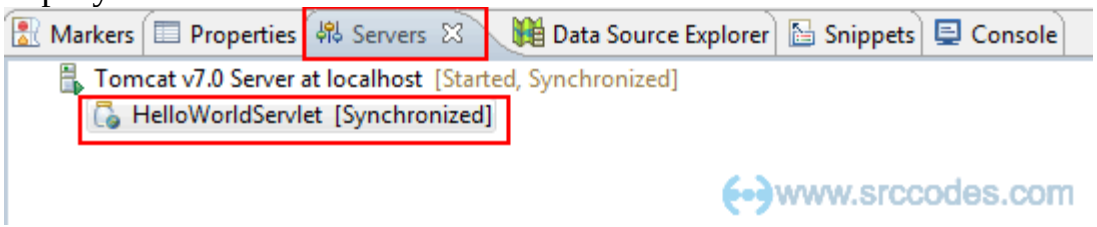
Right click on the project 'HelloWorldServlet' and select from context menu 'Run As' --> 'Run on Server'.



Select the existing tomcat server. If not available then manually define a new web server.



Click "Finish" button. HelloWorldServlet web application will be deployed in the tomcat web server.



.

Browser Output

Eclipse will open a browser and your server side code will print 'Hello World!' in the browser.

Experiment – 11

Initialize Spring boot Project

Creating a Spring Boot Application

Step 1: Open the Spring Initializr https://start.spring.io/.

Step 2: Select the Spring Boot version **2.2.2.BUILD-SNAPSHOT.**

Step 3: Provide the **Group** name. We have provided the Group name **com.javatpoint.**

Step 4: Provide the **Artifact**. We have provided the Artifact **spring-boot-application-run.**

Step 5: Add the **Spring Web** dependency.

Step 6: Click on the **Generate** button. When we click on the Generate button, it wraps all the specifications related to application into a **Jar** file and downloads it to the local system.

Step 7: Extract the jar file.

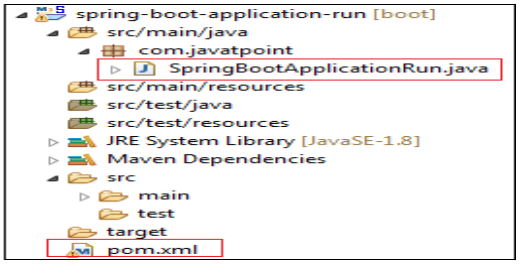
Step 8: Copy the folder and paste it in the STS workspace.

Step 9: Import the project.

File -> Import -> Existing Maven Projects -> Next -> Browse -> Select the folder spring- spring-boot-application-run -> Select Folder -> Finish

It takes time to import the project. When the project imports successfully, we can see it in the **Package Explorer** section of the IDE**.**

We see that two files are automatically created, one is **pom.xml,** and the other is **Application.java** file.



The **pom.xml** file contains the all the **dependencies**, **application name, Spring Boot version, group name, artifact,** and other **plugins.**

The **main** class is a class that contains the main() method. It starts the Spring ApplicationContext. It is the class that we run for the execution of the application.

Experiment - 12

Implement navigation using react router

Add React Router

- To add React Router in your application, run this in the terminal from the root directory of the application:

```
npm i -D react-router-dom
```

Index.js

```
import ReactDOM from "react-dom/client";
import { BrowserRouter, Routes, Route } from "react-router-dom";
import Layout from "./pages/Layout";
import Home from "./pages/Home";
import Blogs from "./pages/Blogs";
import Contact from "./pages/Contact";
import NoPage from "./pages/NoPage";
export default function App() {
 return (
  <BrowserRouter>
   <Routes>
    <Route path="/" element={<Layout />}>
     <Route index element={<Home />} />
     <Route path="blogs" element={<Blogs />} />
     <Route path="contact" element={<Contact />} />
     <Route path="*" element={<NoPage />} />
    </Route>
   </Routes>
  </BrowserRouter>
```

```
    );
  }
  const root = ReactDOM.createRoot(document.getElementById('root'));
  root.render(<App />);
```

Create a folder name called pages. Within a pages create following files.

Blogs.js

```
const Blogs = () => {
   return <h1>Blog Articles</h1>;
 };
   export default Blogs;
```

Contact.js

```
const Contact = () => {
   return <h1>Contact Me</h1>;
 };
 export default Contact;
```

Home.js

```
const Home = () => {
   return <h1>Home</h1>;
 };
 export default Home;
```

Layout.js

```
import { Outlet, Link } from "react-router-dom";
const Layout = () => {
 return (
   <>
     <nav>
       <ul>
         <li>
           <Link to="/">Home</Link>
         </li>
         <li>
           <Link to="/blogs">Blogs</Link>
         </li>
         <li>
           <Link to="/contact">Contact</Link>
         </li>
       </ul>
     </nav>
```

```
      <Outlet />
    </>
  )
};export default Layout;
```

NoPage.js

```
const NoPage = () => {
   return <h1>404</h1>;
  };
   export default NoPage;
```

App.css

```
ul {
  list-style-type: none;
  margin: 0;
  padding: 0;
  overflow: hidden;
  background-color: #04AA6D;
}

li {
  float: left;
  border-right:1px solid #bbb;
}

li a {
  display: block;
  color: white;
  text-align: center;
  padding: 14px 16px;
  text-decoration: none;
```

```
}

li a:hover:not(.active) {
  background-color: #111;
}
```

Experiment - 13

Build single page application(Shopping cart)

App.js File

```
import { useState } from "react";
import "./App.css";
function App() {
  const [list, setList] = useState([]);
  const [value, setValue] = useState("");
  const addToList = () => {
    let tempArr = list;
    tempArr.push(value);
    setList(tempArr);
    setValue("");
  };
 const deleteItem = (index) => {
 let temp = list.filter((item, i) => i !== index);
    setList(temp);
  };
  return (
    <div className="App">
        <fieldset>
        <h>Add Product to List</h><br></br>
    <input type="text" value={value} onChange={(e) =>
setValue(e.target.value)}/>
    <button onClick={addToList}> Click to Add
</button><br></br><br></br>
    <h>Product Catalog</h><br></br>
```

```
    <ol>
        {list.map((item, i) => <li onClick={() => deleteItem(i)}>{item}
</li>)}
    </ol>
    <h>Click on Product to Delete</h><br></br>
    </fieldset></div>
  );
}export default App;
```

index.js file

```
 import React from 'react';
import ReactDOM from 'react-dom/client';
import './index.css';
import App from './App';
import reportWebVitals from './reportWebVitals';
const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
  <React.StrictMode>
    <App/>
  </React.StrictMode>
);
```

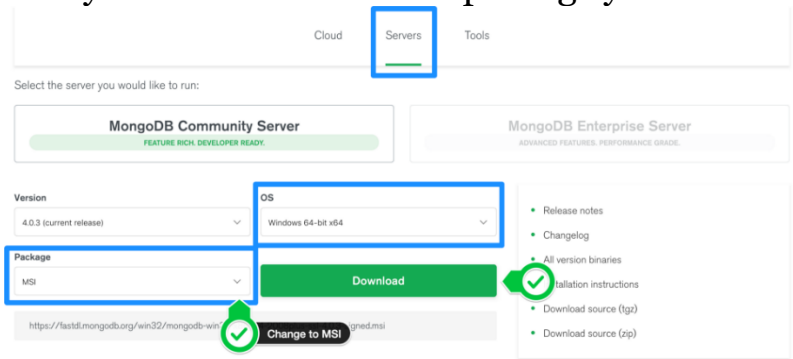
Experiment – 14

Download and Install MongoDB and perform CRUD operation on database

Step 1: Download & Install MongoDB on Windows

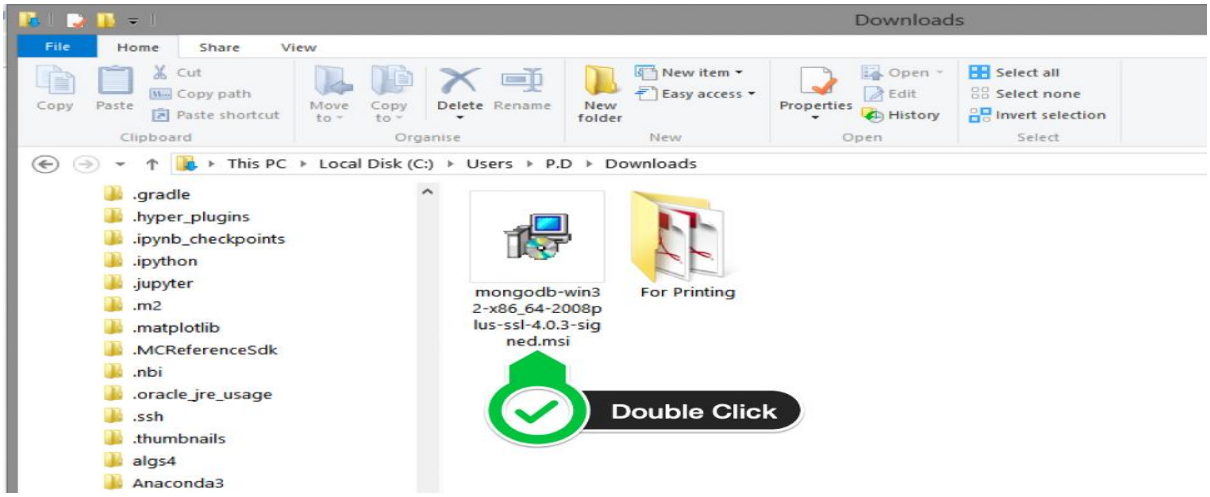
Step 1 — Download the MongoDB MSI Installer Package

Head over here and download the current version of MongoDB. Make sure you **select MSI** as the package you want to download.

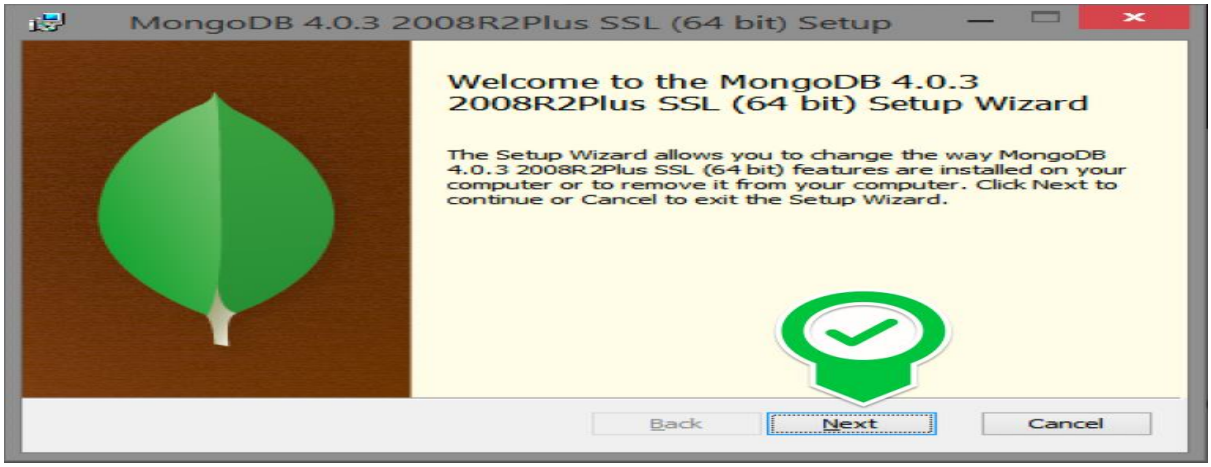


Step 2 — Install MongoDB with the Installation Wizard

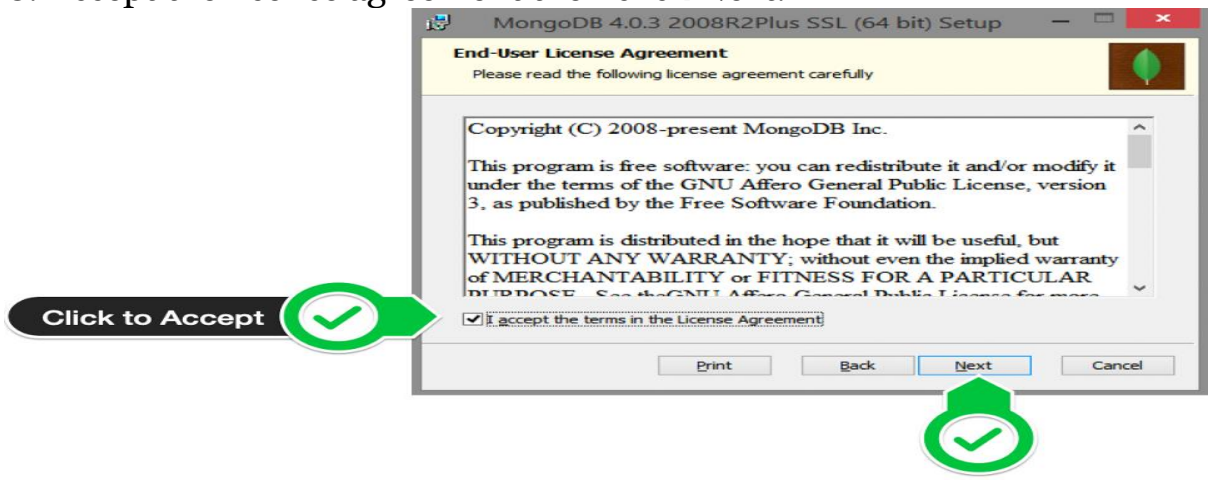
A. Make sure you are logged in as a user with Admin privileges. Then navigate to your downloads folder and double click on the .msi package you just downloaded. This will launch the installation wizard.



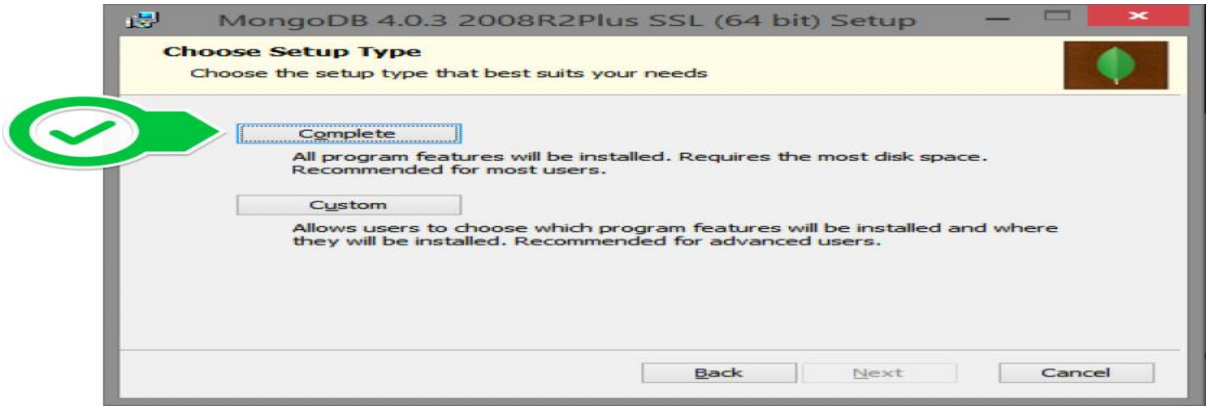
B. Click Next to start installation.



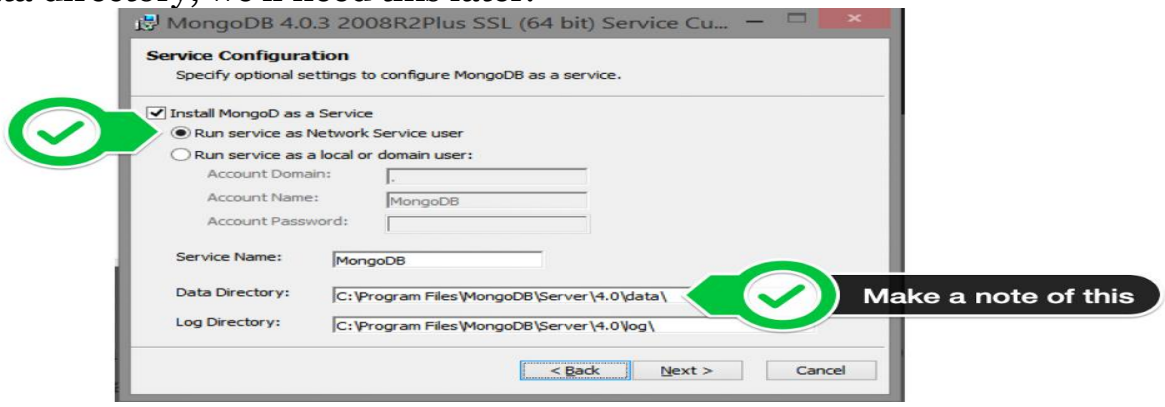
C. Accept the licence agreement then click Next.



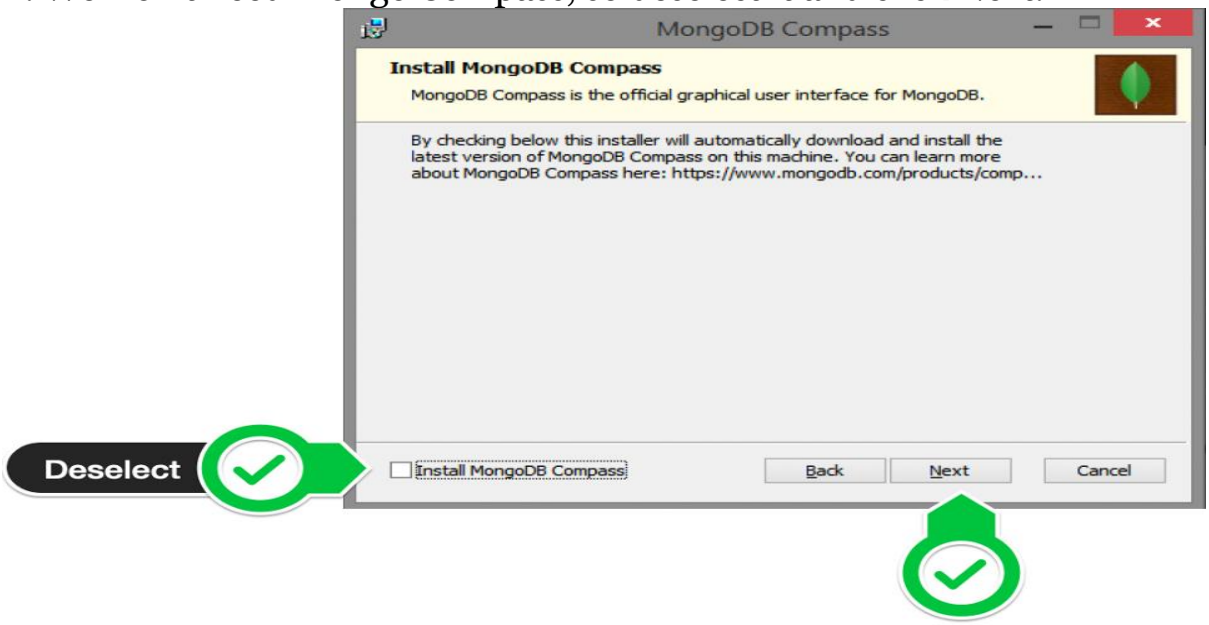
D. Select the Complete setup.



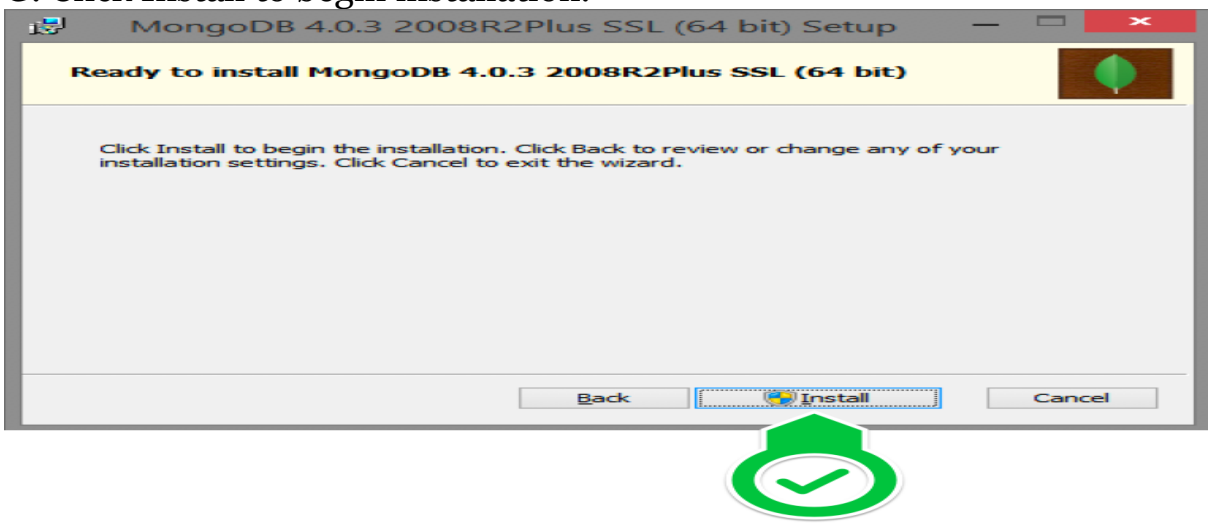
E. Select "Run service as Network Service user" and make a note of the data directory, we'll need this later.



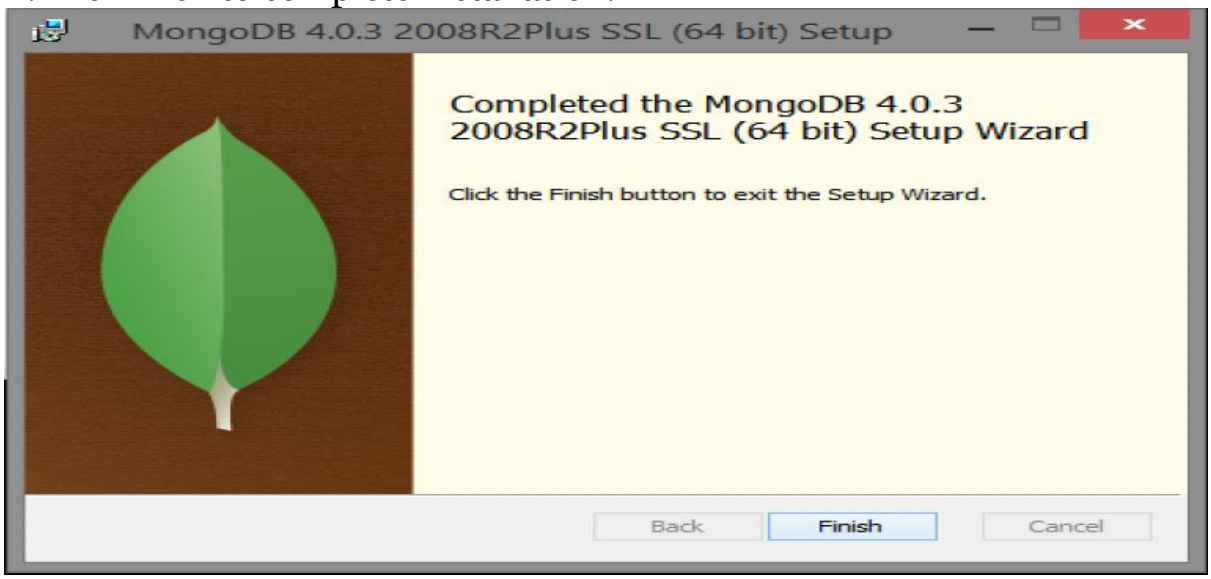
F. We won't need Mongo Compass, so deselect it and click Next.



G. Click Install to begin installation.



F. Hit Finish to complete installation.



Step 2: Open MongoDB Shell

Create operation on document

In Mongodb to create /insert operation ,add new documents to the collection

>use sampleDB

Switched to db sampled

>db.createCollection(“mycollection1”)

{“ok”:1}

We use following methods :

1. insertOne():Single document insertion

```
>db.mycollection1.insertOne(
{ name:”Arnav”,
age:19,
sem:5
}
)
```

2.insertMany(): many document insertion into collection

```
> db.mycollection1.insertMany([
{ name:”TEJRAJ”,
age:18,
sem:5
},
{ name:”Yuvraj”,
age:20,
sem:5
}
])
3.insert({ name:”Sushurut”,
age:18,
sem:5
})
```

In the above cases we have created database sampledDB, collection by name mycollection1,followed by inserting 4 documents using insertOne,insertMany and
insert(used for single or multiple documents).

READ OPERATION 0N DOCUMENT

Following method retrieves all the documents in the collection –

```
>db.mycollection1.find()
```

:it fetches 4 documents from the mycollection1 and displays continuously.

The pretty() Method

To display the results in a formatted way, you can use pretty() method.

Syntax

```
>db.COLLECTION_NAME.find().pretty()
```

Example

Following example retrieves all the documents from the collection named mycol and
arranges them in an easy-to-read format.

```
> db.mycollection1.find().pretty()
```

{4 documents….
}
>db.mycollection1.find(→collection
{age:{$gt :19}}, -→query
{name :1,sem:1} →projection
).limit(1) →cursor modifier
The above command displays documents with age greater than 19 ie
{ name:”Yuvraj”,
age:20,
sem:5
}
The findOne() method
Apart from the find() method, there is findOne() method, that returns only one document.
Syntax
>db.COLLECTIONNAME.findOne()
Example
Following example retrieves the document with title MongoDB Overview.
> db.mycol.findOne({title: "MongoDB Overview"})
{ "_id" : ObjectId("5dd6542170fb13eec3963bf0"),
"title" : "MongoDB Overview",
"description" : "MongoDB is no SQL database",
"by" : "tutorials point",
"url" : "http://www.tutorialspoint.com",
"tags" : ["mongodb", "database", "NoSQl”
},
“likes” :100
}
Operation Syntax Example RDBMS Equivalent

- Equality →{<key>:{$eg:<value>}} db.mycol.find({"by":"tutorials point"}).pretty()

where by = 'tutorials point' Less Than→{<key>:{$lt:<value>}
db.mycol.find({"likes":{$lt:50}}).pretty() where likes < 50

- Less Than Equals→ {<key>:{$lte:<value>}} db.mycol.find({"likes":{$lte:50}}).pretty()

where likes <= 50

- Greater Than → {<key>:{$gt:<value>}} db.mycol.find({"likes":{$gt:50}}).pretty()

where likes > 50

- Greater Than Equals→ {<key>:{$gte:<value>}}
  db.mycol.find({"likes":{$gte:50}}).pretty()

where likes >=50

- Not Equals→ {<key>:{$ne:<value>}}
  db.mycol.find({"likes":{$ne:50}}).pretty()

where likes != 50

- Values in an array→ <key>{:{$in:<value1>[,<value2> ,……]}}

db.mycol.find({"name":{$in:["Raj", "Ram", "Raghu"]}}).pretty() Where name matches any of
the value in :["Raj",”ram”,”Arnav”]

UPDATE OPERATION ON DOCUMENT

In mongodb to modify existing documents in th collection we use following methods
1. updateOne() : First single document to update
>db.students.insertMany(
[
{”_id” :1,”grades” :[95,92,90]},
{“_id” :2,”grades” :[98,100,102]},
{“_id” :3,”grades” :[95,110,100]}
]}
>db.students.updateOne(
{grades : {$gte : 100}}, →FILTER PART
{$set : {“grades.$[element]” :100}}, →UPDATE PART
{arrayFilters:[{“element” :{$gte : 100} →OPTION PART
}]}
)
o/p:
{”_id” :1,”grades” :[95,92,90]},
{“_id” :2,”grades” :[98,100,100},
{“_id” :3,”grades” :[95,110,100]}
2. UPDATEMANY() METHOD:
>db.students.updateMany()
{grades : {$gte : 100}},
{$set : {“grades.$[element]” :100}},
{arrayFilters:[{“element” :{$gte : 100}
}]})
o/p:
{”_id” :1,”grades” :[95,92,90]},

{“_id” :2,”grades” :[98,100,100},
{“_id” :3,”grades” :[95,100,100]}

DELETE ON DOCUMENTS

In mongodb to delete documents we use following methods :
1.deleteOne() :first document with the match will be deleted.
>db.students.deleteOne(
{grades :{$gte :100}})
o/p:
{”_id” :1,”grades” :[95,92,90]},
{“_id” :3,”grades” :[95,100,100]}
> db.students.deleteMany({}) :deletes all the documents.
>db.students.drop() :It is used when it capped collection

Experiment – 15

Test Web Application using appropriate automated testing tool like Selenium IDE

Selenium IDE is an open-source, automated testing tool used to test web applications across various browsers. It's primarily built in Java and supports several browsers and programming languages.

Key features of Selenium IDE:

- Support for both Chrome and Firefox
- Improved locator functionality
- Parallel execution of tests using Selenium command line runner
- Provision for control flow statements
- Automatically waits for the page to load
- Supports embedded JavaScript code-runs

Installation of Selenium IDE

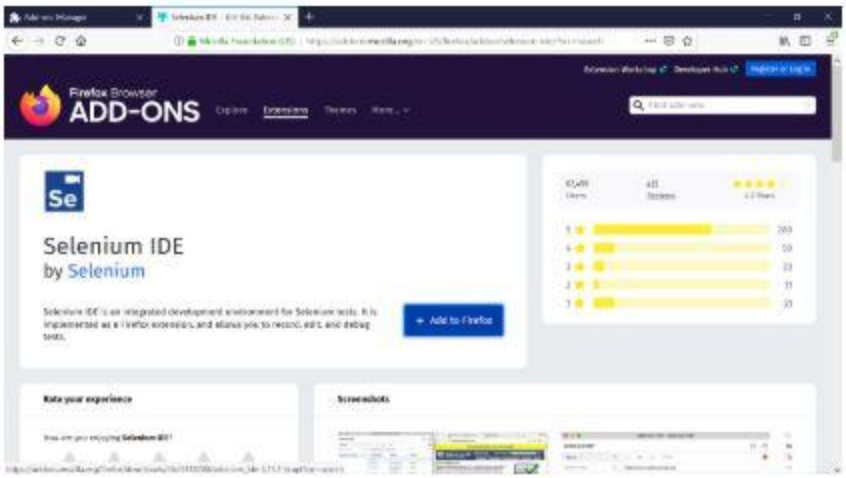
Step 1- Open the Firefox browser.

Step 2- Click on the menu in the top right corner.

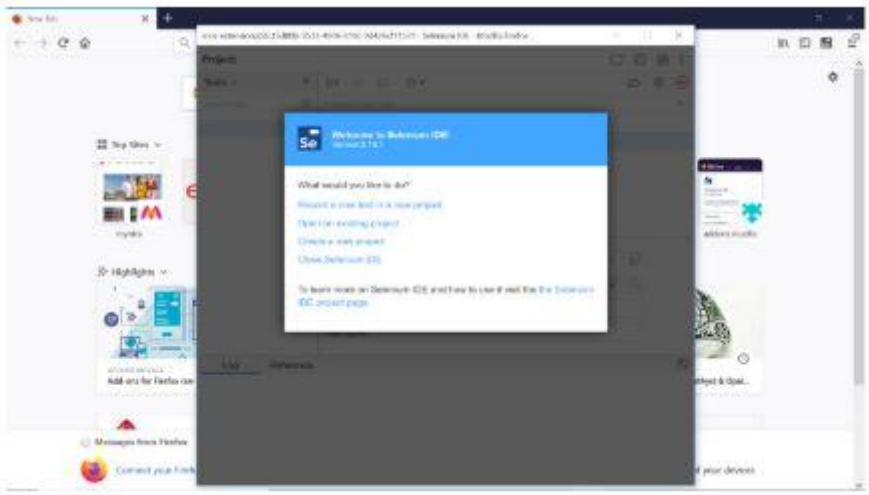
Step 3- Click on Add-ons in the drop-down box.

Step 4- Click on find more add-ons and type “Selenium IDE.”

Step 5- Click on Add to Firefox



Once installed, the Selenium IDE icon appears on the top right corner of the browser. When clicked, a Welcome message appears.



Now that we went through the installation, let's create our first test. Consider the following use case for the tutorial:

- Navigate to https://www.facebook.com
- Provide a dummy userID and password
- Log in with these credentials

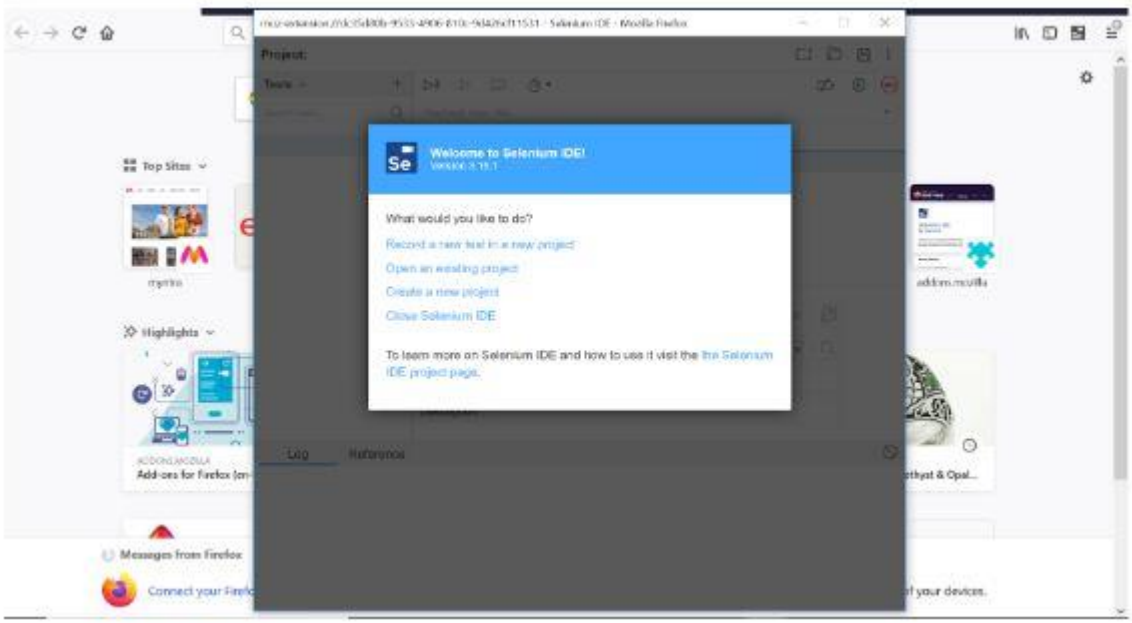
- Assert title of the application

A Simple Demo

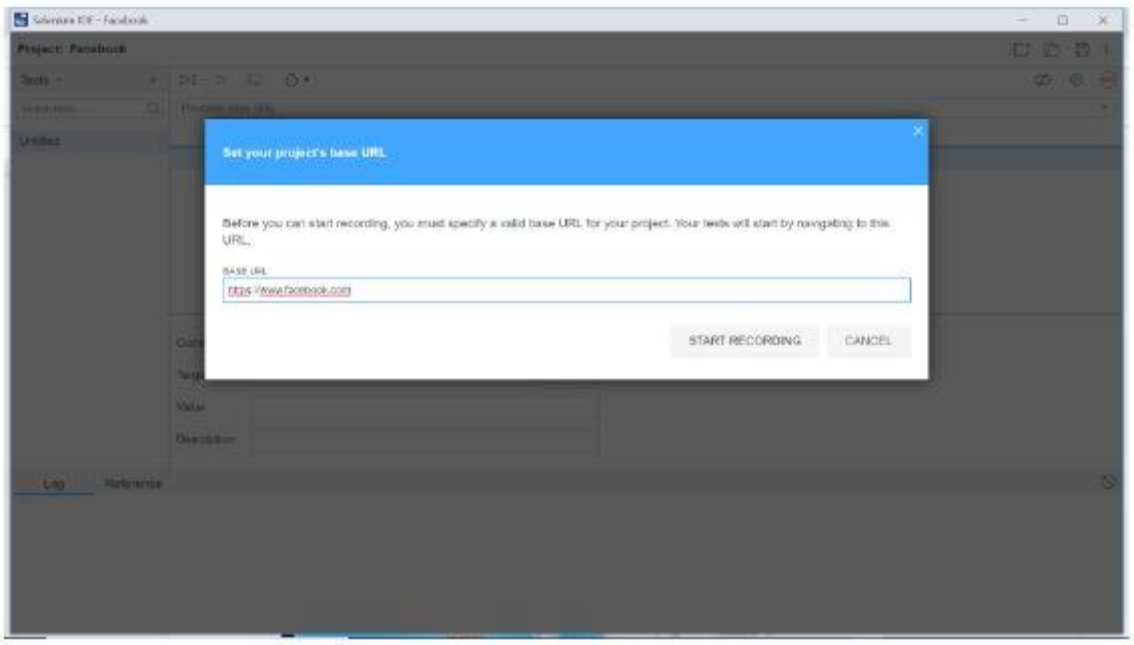
Recording

Step 1 - Launch your Firefox menu and open the Selenium IDE plugin.

Step 2 - Select “Record a test in a new project.” Provide the name for your test.



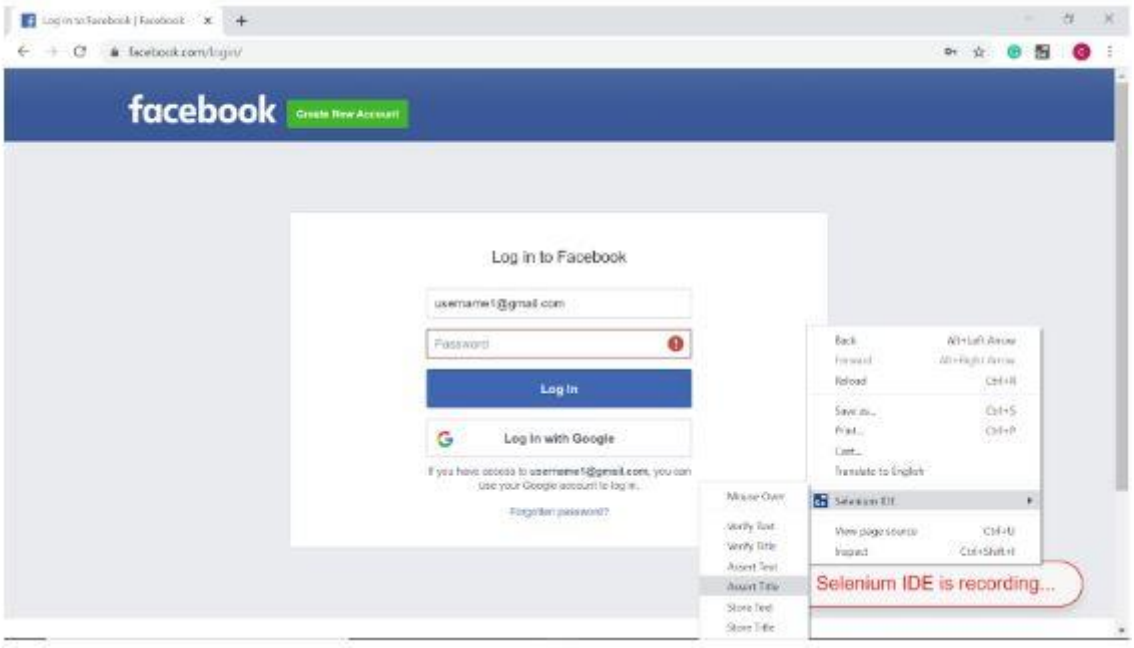
Step 3 - Provide a link to the Facebook webpage. The IDE starts recording by navigating to this web page. To continue, click on, OK.



Step 4 - Once the website opens, type “username1” in the username field and a dummy password for the password field.

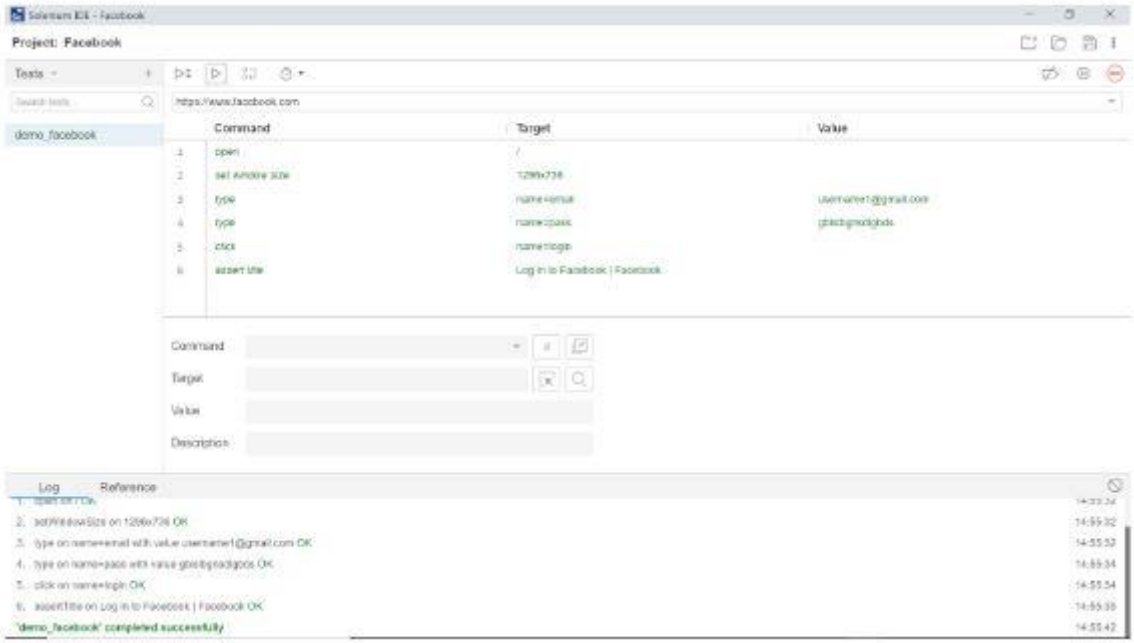
Step 5 - Click on “Log In.”

Step 6 - Now, we verify the title of our application. To do that, Right click>>Selenium IDE>>Assert title. As soon as this is done, a test step would be appended in the IDE editor.



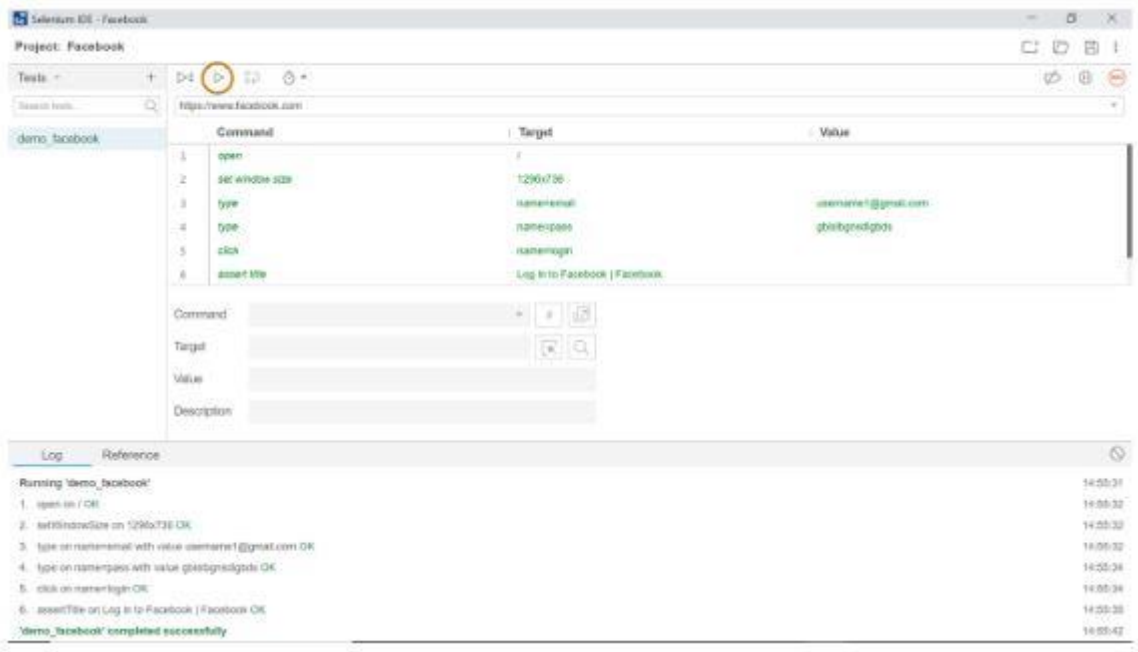
Now you can go back to the IDE editor and click on the Stop icon on the top right corner. With this, we've successfully recorded our test case.

Once the recording is stopped, the editor will look something like this:



Playing Back

Now that the recording is done, we can play it back to verify if the script executes correctly and the browser behaves accordingly. To do this, we can click on the play icon on the menu bar.



The commands successfully executed are color-coded in green, and the log at the bottom indicates any errors that occurred during the execution. If the script runs successfully, it indicates this by displaying a message.

Saving

Once the test script is successfully executed, you can then save it.

Once clicked, you can browse the location and save it in an appropriate folder. The project is stored with a “.side” extension — this can be used for future runs and regressions.