

Project 1: Setup AWS, MySQL, JDBC, Tomcat, Start Fabflix

Due Apr 16 by 11:45pm **Points** 100

▼ Task 1: GitHub Repository Creation

This course uses GitHub for version control. The students are expected to use GitHub, as explained below.

- Create an account on GitHub if you don't have one.
- Accept the course project through [GitHub Classroom \(https://classroom.github.com/a/MgEsXTpv\)](https://classroom.github.com/a/MgEsXTpv).

UCI-Chenli-teaching-CS122B

Accept the group assignment —
CS122B

Before you can accept this assignment, you must create or join a team. Be sure to select the correct team as you won't be able to change this later.

Create a new team

team-999

+ Create team

- GitHub Classroom will create an empty repo with the name `cs122b-#`
- For the second team member (if it exists), please select the correct team to join when you accept the assignment. You cannot change the team after the selection, so please double-check before proceeding.


Join an existing team

team-999 0 students

Join

▼ Contribute with Git in Team

Suppose the CS122B team 999 contains 2 members (Bob and Alice). We will see how they use Git to collaborate in the CS122B class.

1. Bob installs a git client on his local machine by following the instructions [here](https://git-scm.com/book/en/v2/Getting-Started-Installing-Git)  (<https://git-scm.com/book/en/v2/Getting-Started-Installing-Git>).
2. He then does the following:

```
shell> mkdir mycs122b-projects
shell> cd mycs122b-projects
shell> git init
shell> echo "Cool project" > README.md
shell> git status
shell> git add README.md
shell> git status
shell> git commit -m "First Commit"
shell> git status
```

- creates a README file
- check the status of the repository
- The file is initially untracked by Git. 'git add' moves it to staged.
- check the status of the repository after staging the file
- The staged files are committed locally.
- check the status of the repository after committing the file

3. The local repository now has to be linked to the remote repository. For that Bob does the following:

```
shell> git remote add origin https://github.com/UCI-Chenli-teaching/cs122b-fall21-team-999.git
shell> git push -u origin main
```

4. Bob now wants to start on project 1. He creates a new branch from the main branch for this task.

```
shell> git branch
shell> git checkout -b bob-feature1
shell> mkdir project1
shell> cd project1
shell> echo "SELECT * FROM stars" > mysql-script.sql
shell> git add mysql-script.sql
shell> git commit -m "added mysql command"
shell> git push --set-upstream origin bob-feature1
shell>
```

- This command is used to check which branch you are on and what branches are there in your repository. main should be highlighted as you are on main branch.
- This command creates a new branch and copies all the code from the previous (i.e. 'main' in our case) branch into the new branch.
- commits changes locally to bob-feature1
- creates a remote tracking branch

5. Alice wants to contribute too. First Bob needs to invite Alice as a contributor to this repository on the Github website. Then she can see the repository. She does the following:

```
shell> mkdir gitclones
shell> cd gitclones
shell> git clone https://<Alice's username>@github.com/UCI-Chenli-teaching/cs122b-fall21-team-999.git
```

```
shell> cd cs122b-fall21-team-999
shell> git checkout bob-feature1
shell> cd project1
MODIFY THE FILE add mysql-script.sql
shell> git add mysql-script.sql
shell> git status
shell> git config user.email "alince@alice.com". - Email used for GitHub.com
shell> git config user.name "Alice Smith" - GitHub account name
shell> git commit -m "minor changes"
shell> git push - pushes the commit to bob-feature1 remote branch
shell>
```

6. Bob wants to continue coding. Before modifying files, he needs to do 'git pull' so that the local branch pulls the latest code from the remote branch. In particular, Bob does:

```
shell> git branch - to see which branch he is on. He sees he is on bob-feature1 branch.
shell> git pull - pulls the latest code. Bob now sees the changes that Alice pushed.
```

7. Bob and Alice can also use GitHub to create a pull request from the bob-feature1 branch to the main branch to do code reviews. Check this

[video](https://www.youtube.com/watch?v=oFYyTZwMyAg)  [_ \(https://www.youtube.com/watch?v=oFYyTZwMyAg\)](https://www.youtube.com/watch?v=oFYyTZwMyAg)



[_ \(https://www.youtube.com/watch?v=oFYyTZwMyAg\)](https://www.youtube.com/watch?v=oFYyTZwMyAg)

to learn this process.

▼ Task 2: Setup an AWS Instance

You need to launch an Amazon [AWS EC2 instance \(https://aws.amazon.com/console/\)](https://aws.amazon.com/console/). Make sure to use the **free-tier 64-bit Ubuntu instances** so that we can make our future instructions consistent. You are welcome to participate in the [AWS Educate \(https://aws.amazon.com/education/awseducate/\)](https://aws.amazon.com/education/awseducate/) program, which can provide \$100 AWS credits per student. (UCI is a member institution.)


Generally, similar to many other tasks this quarter, we expect you to figure out how to do many tasks by reading online materials.

(<https://aws.amazon.com/free/>) This [website](#) (<https://aws.amazon.com/free/>) is an excellent place to start from.

Launch a free AWS EC2 instance

1. Go to [AWS Console \(https://aws.amazon.com/console/\)](https://aws.amazon.com/console/) to sign up. You will need to enter a valid credit card. Don't worry; if you choose a free-tier instance and remove it after the end of the quarter, you will not be charged.
2. Log in to the AWS console when you are done.
3. Launch a new **Ubuntu** free-tier **t2.micro** EC2 instance. Notice that you need to generate and download a key to ssh to the machine, and it may take a few minutes to initialize the instance.
4. After the instance runs, you will see a public IP address assigned to it. Keep this IP: you are required to give us this IP to the demo project
5. When viewing the list of instances, click the "connect" button on the top to get instructions on using SSH to connect to the instance. By default, only SSH port 22 is open. You must open the corresponding ports to get other services (e.g., HTTP, HTTPS, and Tomcat) to be available to other machines. **When the instance is checked, select the security group, go to the "inbound" tab, and add more rules.**

The name of the instance you will be launching is: **Ubuntu Server LTS (HVM), SSD Volume Type**

 **Ubuntu Server 20.04 LTS (HVM), SSD Volume Type** - ami-031b673f443c2172c (64-bit x86) / ami-07711e34185f62b48 (64-bit Arm)

Free tier eligible

Ubuntu Server 20.04 LTS (HVM), EBS General Purpose (SSD) Volume Type. Support available from Canonical (<http://www.ubuntu.com/cloud/services>).

Root device type: ebs Virtualization type: hvm ENA Enabled: Yes

Select

☒ 64-bit (x86)
☐ 64-bit (Arm)

▼ Task 3: Install MySQL and Apache Tomcat

▼ Development Machine

1. Install [Java@11 \(https://www.oracle.com/java/technologies/javase/jdk11-archive-downloads.html\)](https://www.oracle.com/java/technologies/javase/jdk11-archive-downloads.html)
2. (<https://www.oracle.com/java/technologies/javase/jdk11-archive-downloads.html>) Setup MySQL:
 - A. Install [MySQL@8.0 \(https://dev.mysql.com/doc/refman/8.0/en/installing.html\)](https://dev.mysql.com/doc/refman/8.0/en/installing.html)
 - B. (<https://dev.mysql.com/doc/refman/8.0/en/installing.html>) Enable logging on MySQL:

- Only For Windows: Install [Cygwin \(https://cygwin.com/install.html\)](https://cygwin.com/install.html) and use the following commands in the Cygwin terminal as administrator (you do not need to use sudo with any command)
- By default, MySQL doesn't enable logging. To enable it, run the following:

```
shell> mysql -u root -p
mysql> SET GLOBAL general_log = 'ON';
mysql> exit
```

- Run the following command to find the MySQL file log file and its path:

```
shell> mysql -u root -p -se "SHOW VARIABLES" | grep -e log_error -e general_log -e slow_query_log -e datadir
```

Pay attention to the path in the **general_log_file** line or **datadir** line (in case of Windows). For example (Linux/MacOS):

```
general_log_file      /usr/local/mysql/data/dhcp-v044-069.log
```

Example (Windows):

```
datadir C:\\ProgramData\\MySQL\\MySQL Server 8.0\\Data\\
```

- Monitor the ".log" files under this folder. In Linux and Mac, the following is an example command using "tail":

```
shell> sudo tail -f /usr/local/mysql/data/dhcp-v044-069.log
```

Example for Windows (no need to use sudo):

```
tail -f "C:\\ProgramData\\MySQL\\MySQL Server 8.0\\Data\\dhcp-v044-069.log"
```

3. Setup Tomcat:

I. Install [Tomcat@10 \(https://tomcat.apache.org/download-10.cgi\)](https://tomcat.apache.org/download-10.cgi)

II. Enable Tomcat Logging:

By default, Tomcat logging is enabled.

Log files are located in the logs folder in your Tomcat installation directory. Monitor log files by running the command (replace the path with your installation directory):

A. For macOS

```
sudo tail -f HOME_OF_TOMCAT/logs/*
```

B. For Linux:

```
sudo tail -f /var/lib/tomcat/logs/*
```

C. For Windows:

```
tail -f "C:\\Program Files\\Apache Software Foundation\\Tomcat\\logs\\*"
```

▼ Production (AWS) Machine

This part assumes you have SSH access to the Ubuntu AWS instance.

1. Java 11

I. Update system repositories:

```
sudo apt update
```

II. Install Java 11:

```
sudo apt install default-jdk
```

2. MySQL 8

I. Update system repositories:

```
sudo apt update
```

II. Install MySQL 8:

```
sudo apt install mysql-server
```

III. Log in to the MySQL server as the root user:

```
sudo mysql
```

IV. Create a test user:

```
CREATE USER 'mytestuser'@'localhost' IDENTIFIED BY 'My6$Password';
```

V. Grant privileges for the test user:

```
GRANT ALL PRIVILEGES ON * . * TO 'mytestuser'@'localhost';
```

VI. Enable MySQL Logging (same as Development Environment)

3. Tomcat 10

I. Update system repositories:

```
sudo apt update
```

II. Add the repository with the tomcat 10 package:

```
sudo add-apt-repository -S deb http://cz.archive.ubuntu.com/ubuntu lunar main universe
```

III. Install Tomcat 10:

```
sudo apt install tomcat10 tomcat10-admin
```

IV. Open ports for Apache Tomcat Server:

```
sudo ufw allow from any to any port 8080 proto tcp
```

V. Open tomcat user config file:

```
sudo nano /etc/tomcat10/tomcat-users.xml
```

VI. Add the following lines before the tag </tomcat-users> (before the last line):

```
<role rolename="manager-gui"/>  
<user username="admin" password="mypassword" roles="manager-gui"/>
```

VII. Restart the Apache Tomcat Server:

```
sudo systemctl restart tomcat10
```

VIII. Test application app manager using the URL

Open port 8080 on AWS security group.

Open http://YOUR_PUBLIC_IP:8080/manager/html  [\(http://YOUR_PUBLIC_IP:8080/manager/html\)](http://YOUR_PUBLIC_IP:8080/manager/html) on your local machine browser.

IX. Enable Tomcat Logging (same as Development Environment)


Here are two cheat sheets of Linux/Ubuntu commands:

- http://cli.learncodethehardway.org/bash_cheat_sheet.pdf ([\(http://cli.learncodethehardway.org/bash_cheat_sheet.pdf\)](http://cli.learncodethehardway.org/bash_cheat_sheet.pdf))
 - <http://www.cheat-sheets.org/saved-copy/ubuntu-ref.pdf> ([\(http://www.cheat-sheets.org/saved-copy/ubuntu-ref.pdf\)](http://www.cheat-sheets.org/saved-copy/ubuntu-ref.pdf))
-

▼ Task 4: Setup IDE on the development machine

▼ Install IntelliJ IDEA Ultimate

Apache Tomcat makes hosting your applications easy. IntelliJ IDEA makes development easy. We recommend you use the latest version of IntelliJ IDEA Ultimate (UCI provides the license for students so that you can use the paid version for free)

1. Obtain a [Free Educational Licenses](https://www.jetbrains.com/community/education/#students)  [\(https://www.jetbrains.com/community/education/#students\)](https://www.jetbrains.com/community/education/#students)
2. Download [IntelliJ IDEA \(choose Ultimate\)](https://www.jetbrains.com/idea/download)  [\(https://www.jetbrains.com/idea/download\)](https://www.jetbrains.com/idea/download)
3. Install IntelliJ IDEA Ultimate
4. Register your license to IntelliJ IDEA Ultimate (log in with your license)

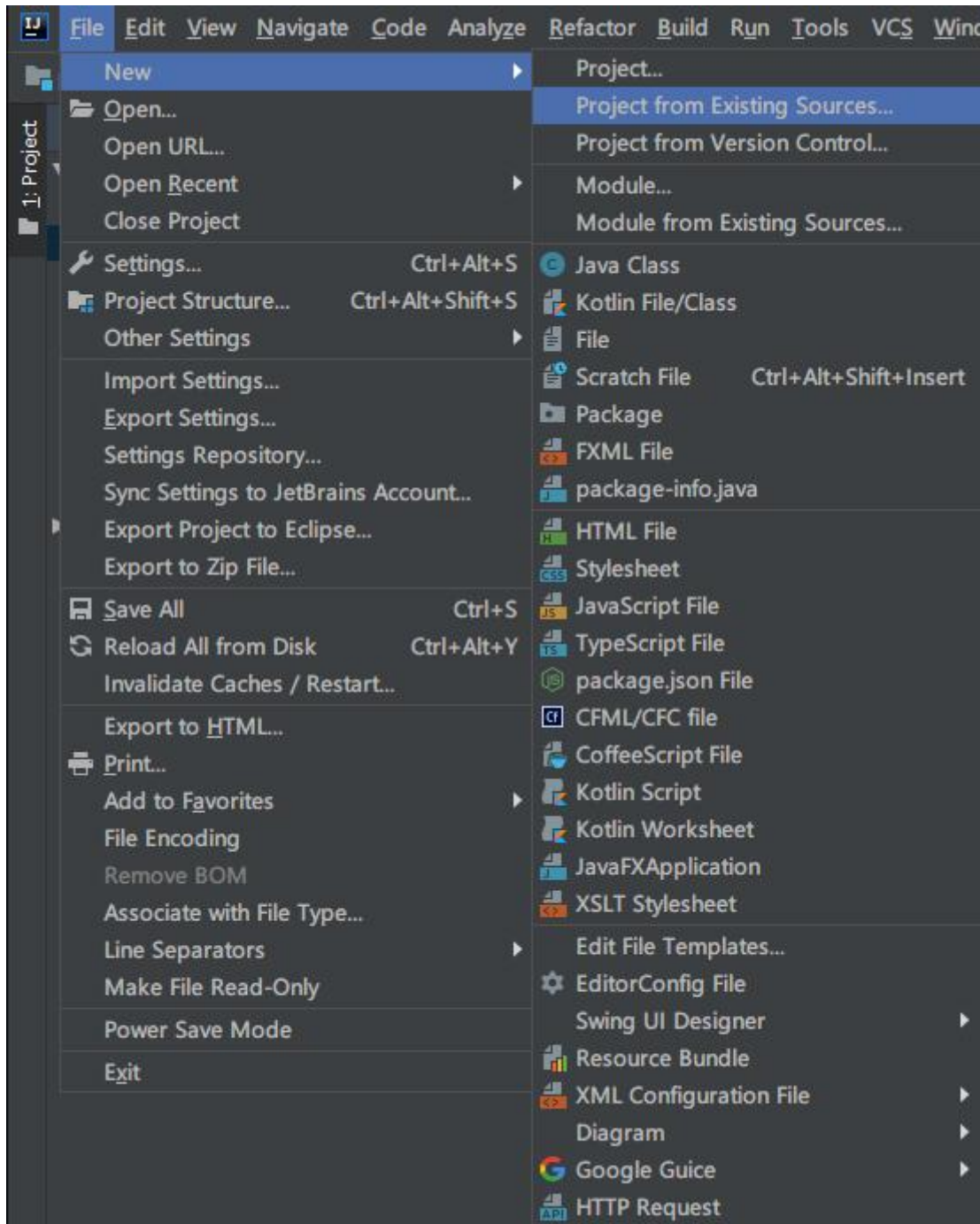
▼ Import Project to IntelliJ

If you want to import a project to IntelliJ instead of working from scratch, refer to the following tutorial.

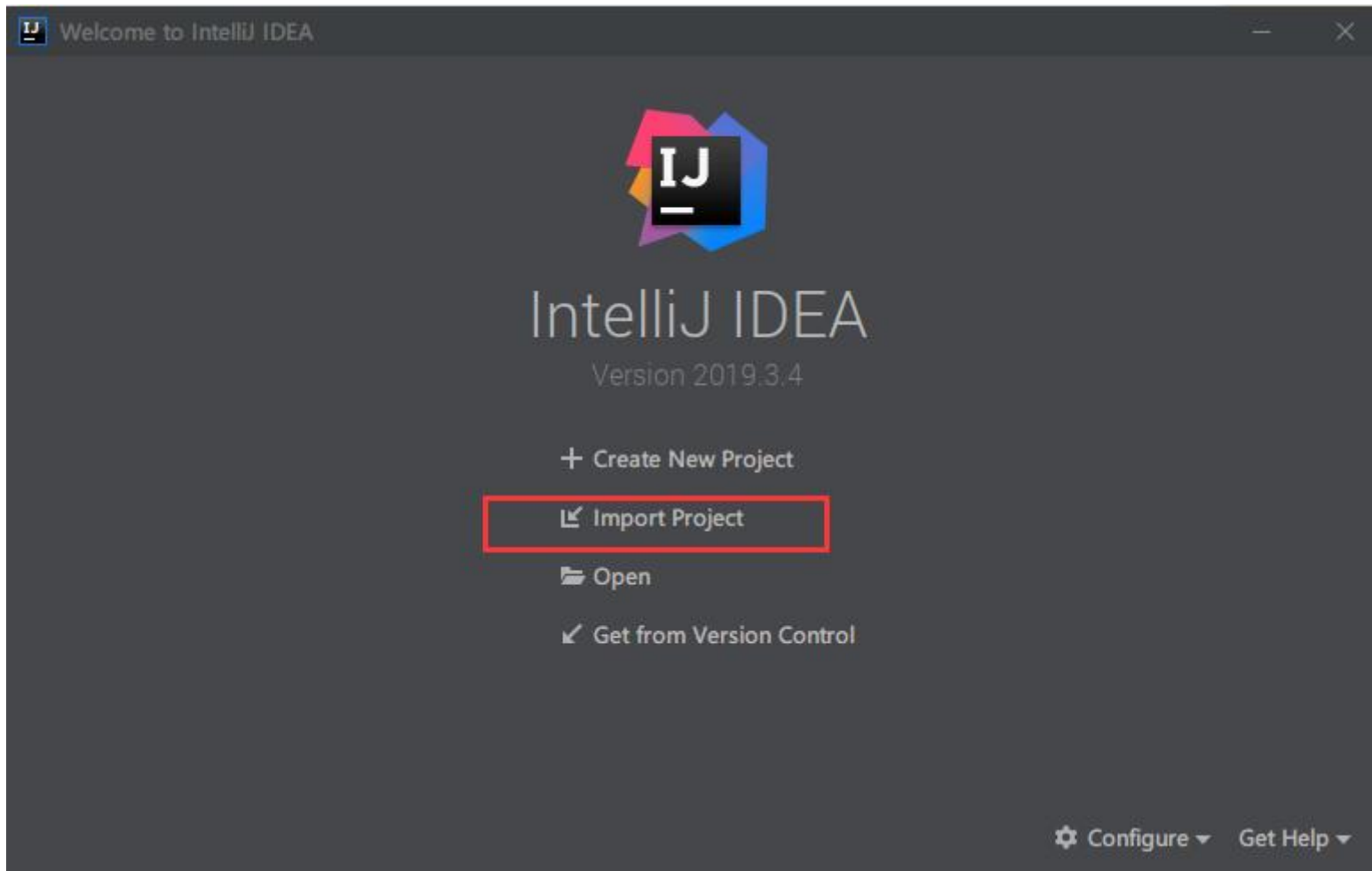
We will walk through the steps to import the project1-star-example web Project to IntelliJ.

Most of the web app examples throughout the quarter will be hosted on Github and will be using maven.

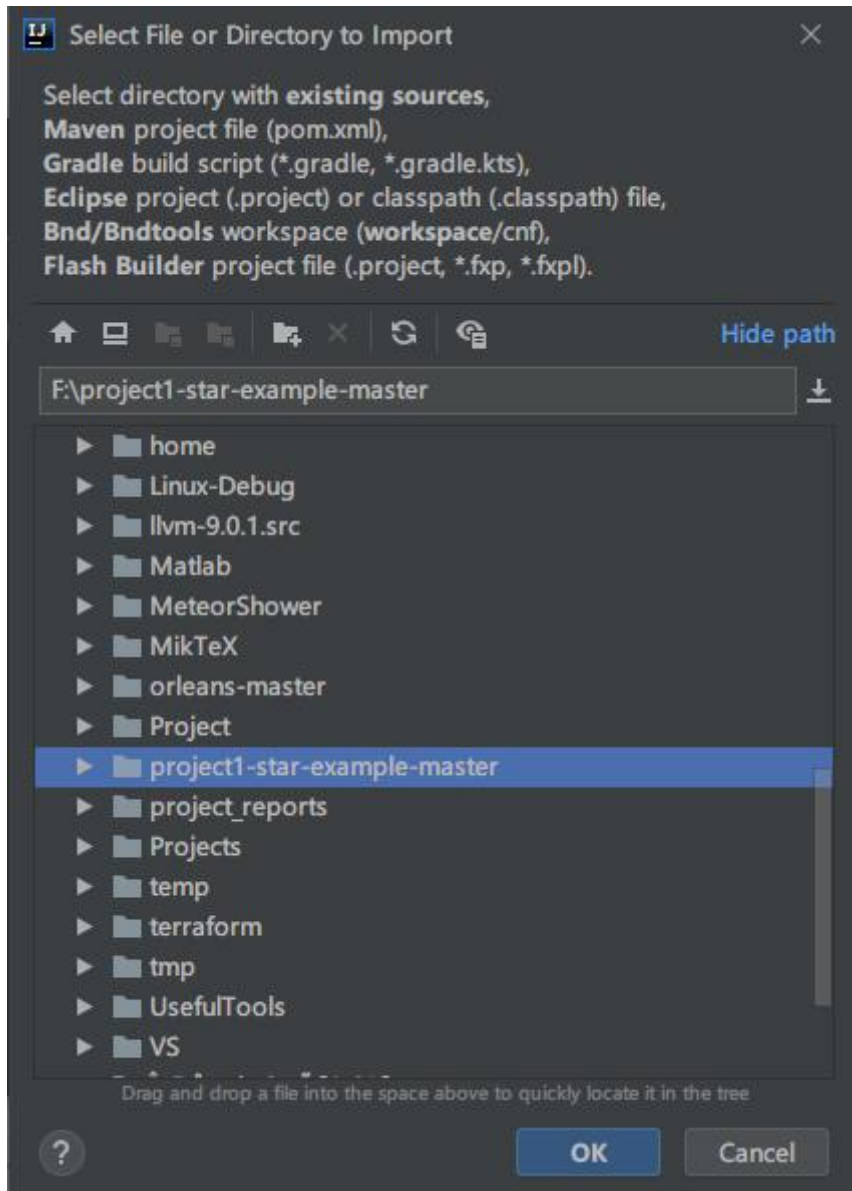
1. Clone project1-star-example repo, or any other example repo we provided. You can also download the codebase as a zip file and unzip it.
2. Open IntelliJ -> **File -> New -> Project from Existing Sources...**



if this is the first time you open IntelliJ, you can just click "**Import Project**"



3. Unzip the zip file and select the folder in your file system.



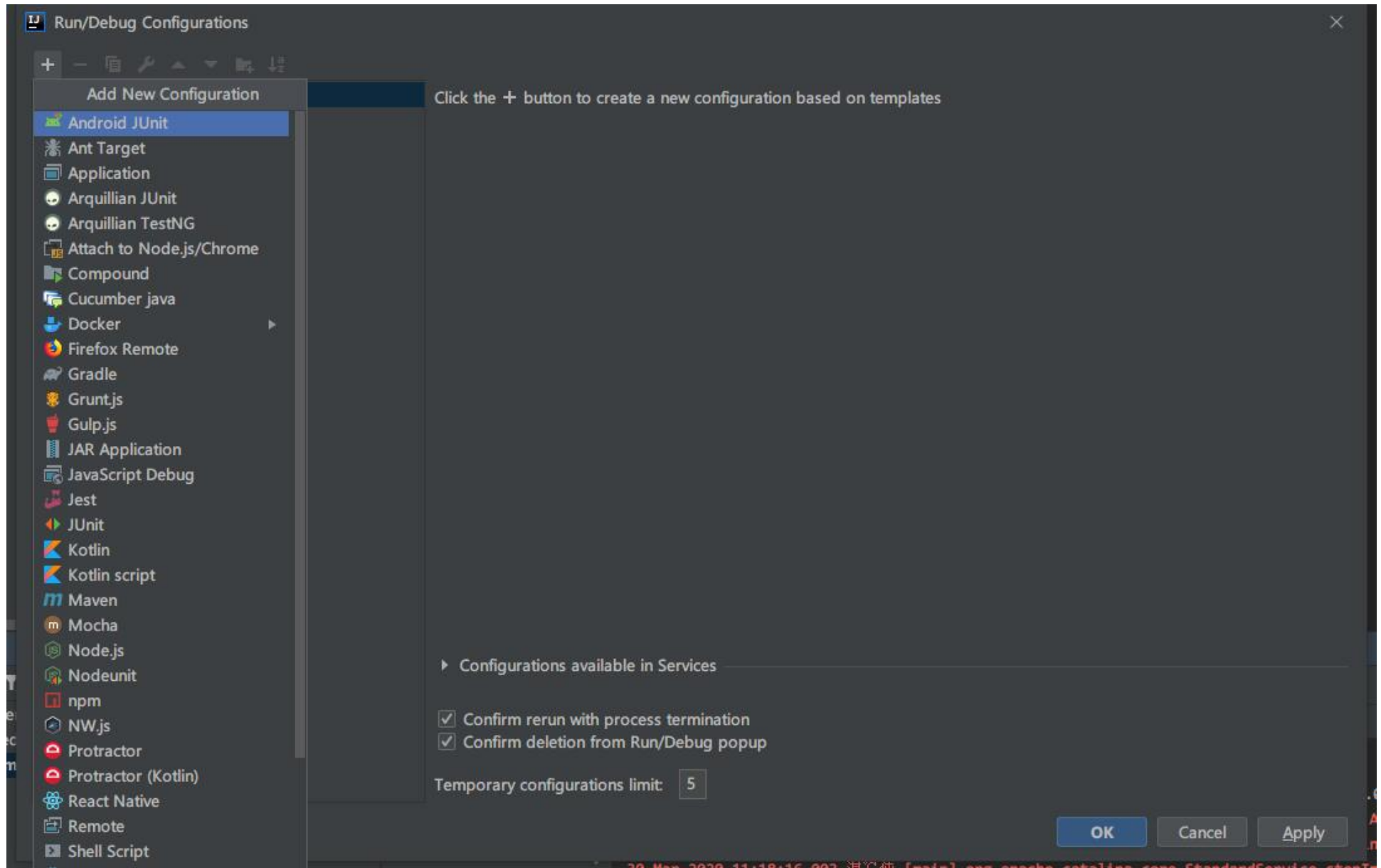
4. In "src" folder, open src->(default package)->StarServlet.java. Change the mysql username and password and make sure you have the moviedb database.

5. In order to run the application, you still need to configure Tomcat with IntelliJ.

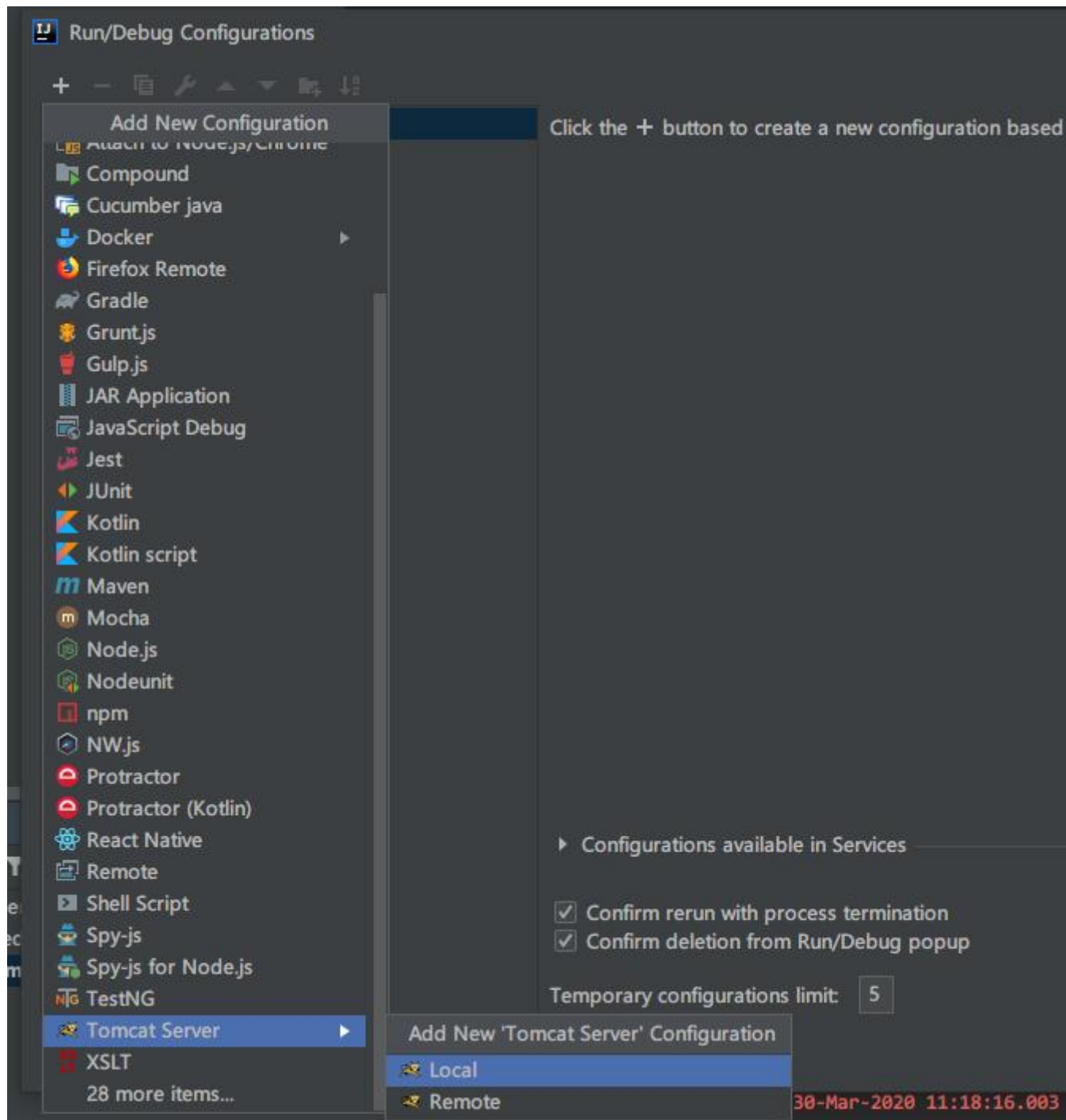
▼ Config Tomcat in IntelliJ

The following tutorial is a quick start guide and will help you set up your development environment in IntelliJ IDEA. **Use Tomcat version 10 instead of the shown version in the tutorial.**

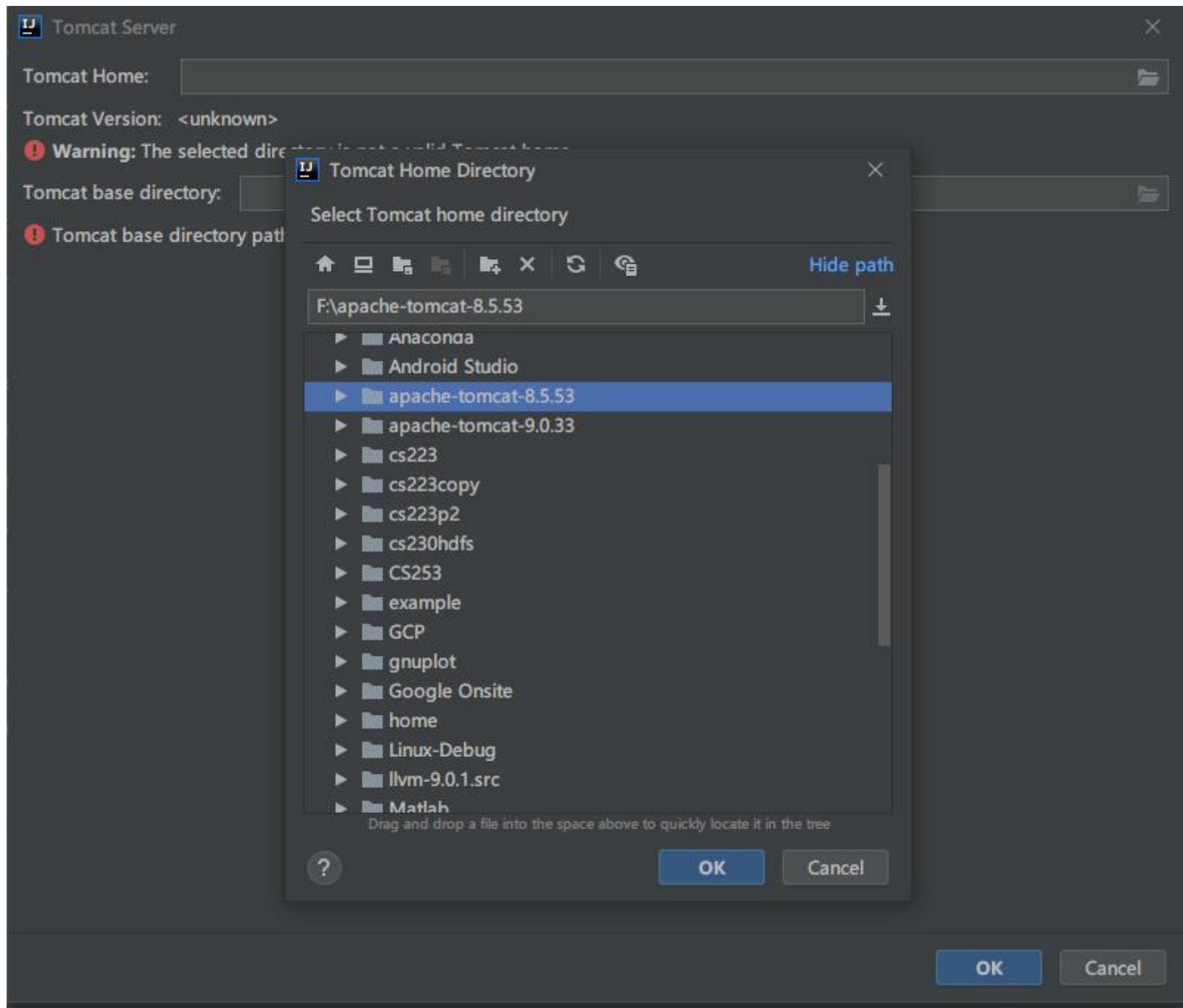
1. Click Run → Edit Configurations, you will see a popup window as following.



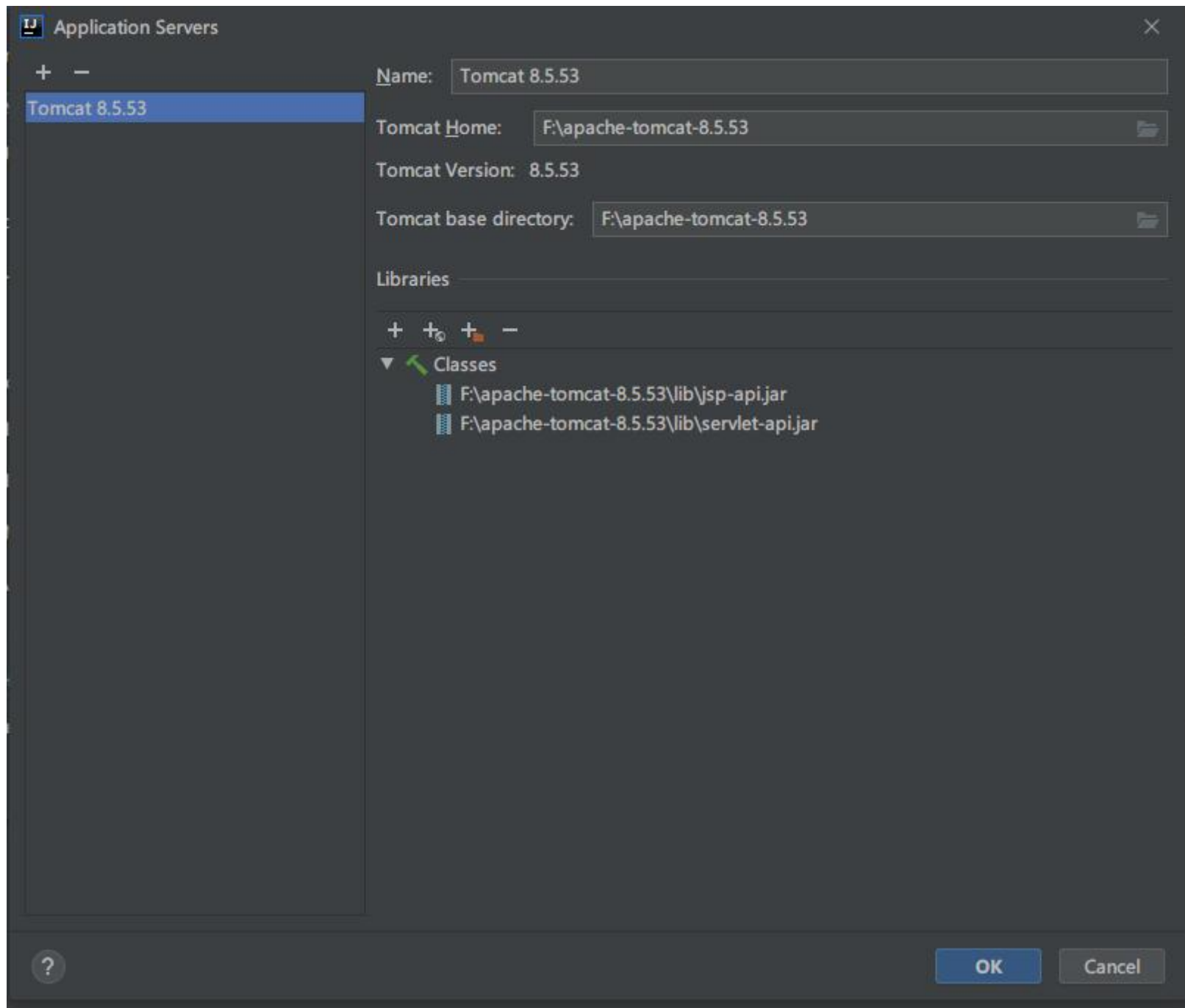
2. choose Tomcat Server -> Local (If you cannot find Tomcat Server, click **# more items..**)



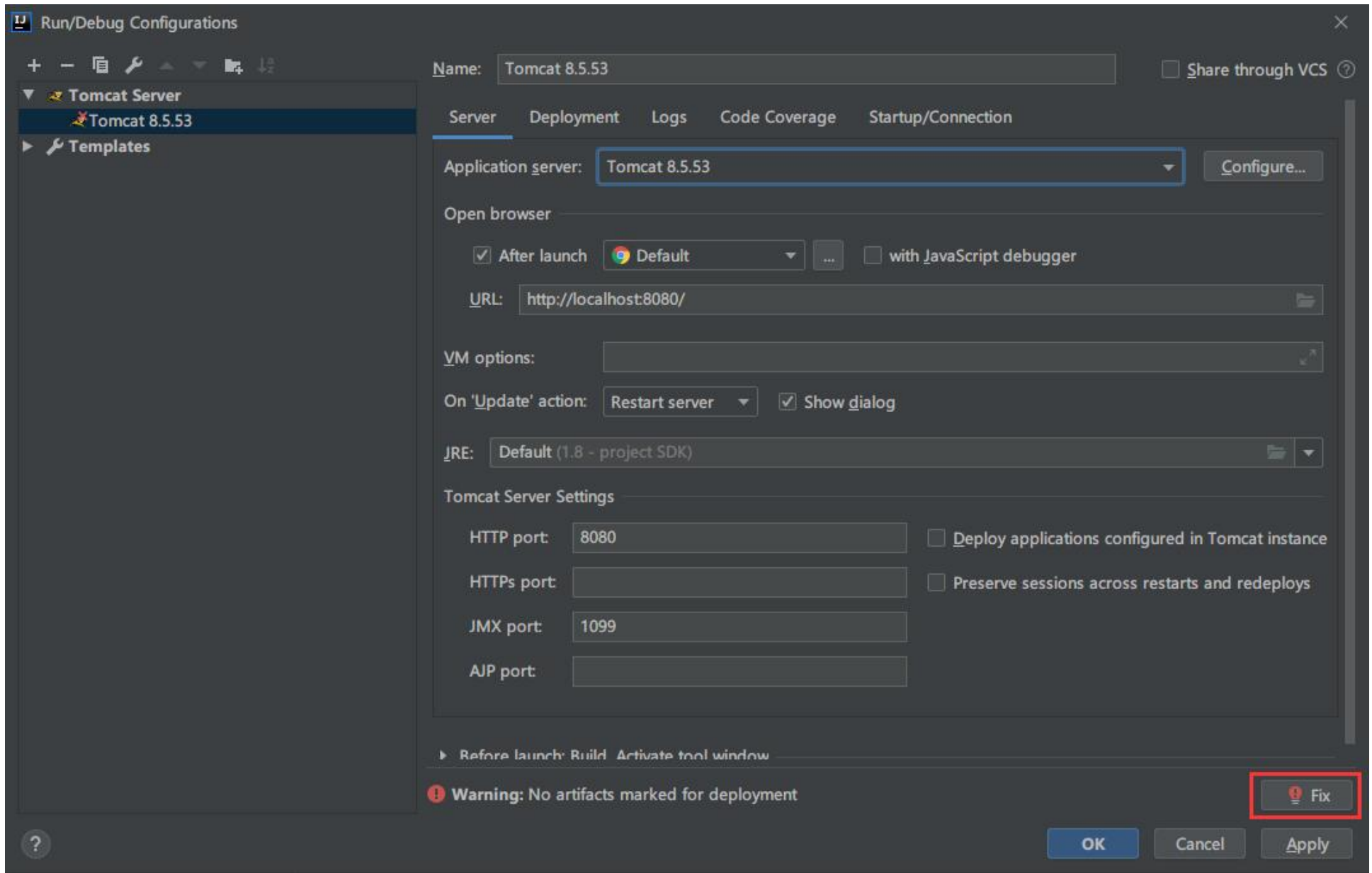
3. Click **Configure** on the right side of the **application server** and you will see a popup window. Click the **folder icon** on the right side of **Tomcat Home**. Then choose the path of your Apache Tomcat folder.



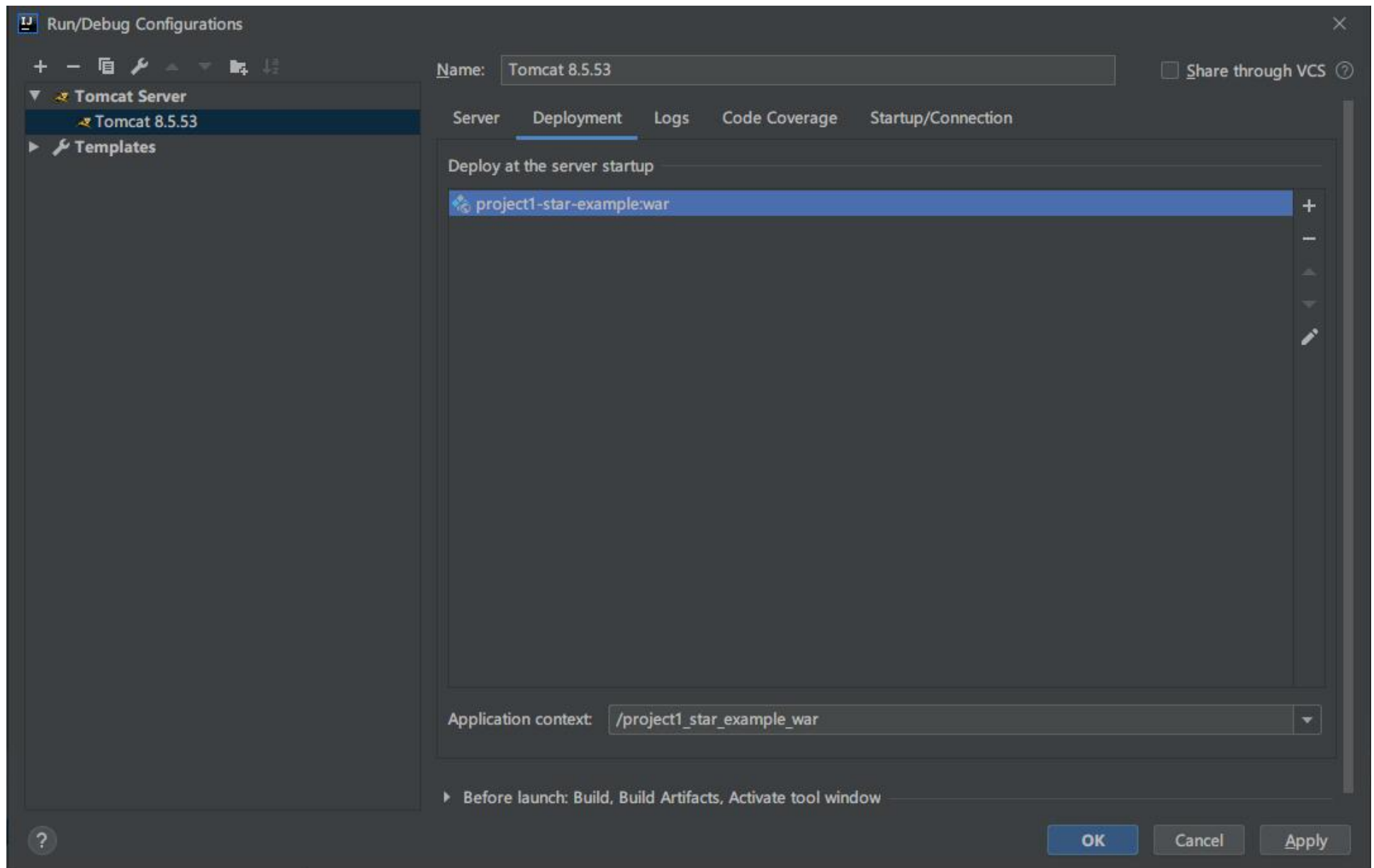
4. After clicking OK, you will see the following. (version should be 9.0.x)



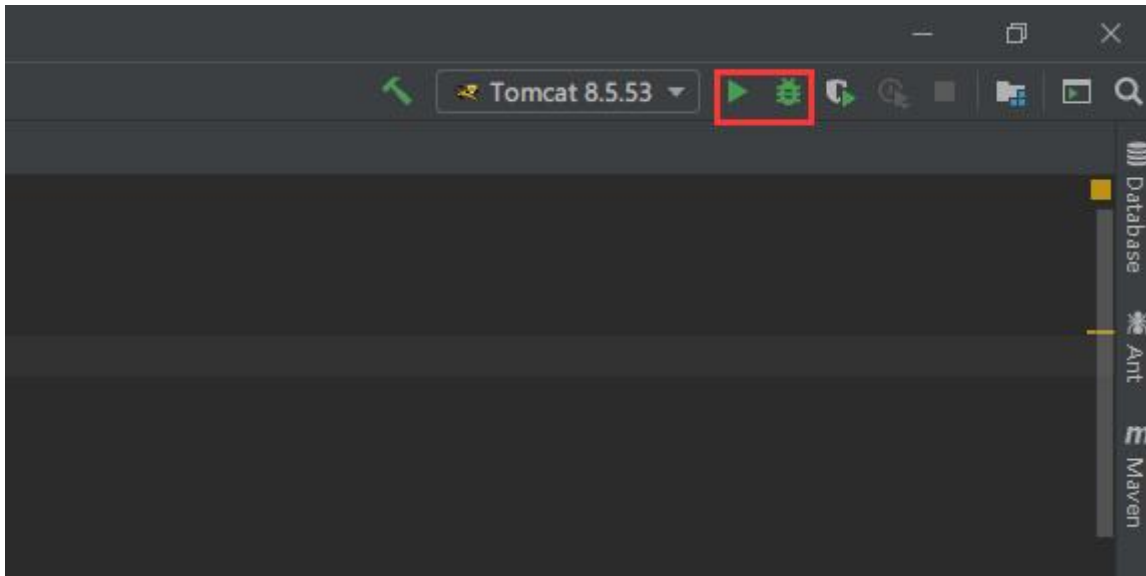
5. Click OK again, you will see a Warning on the bottom of the window. Click **Fix**.



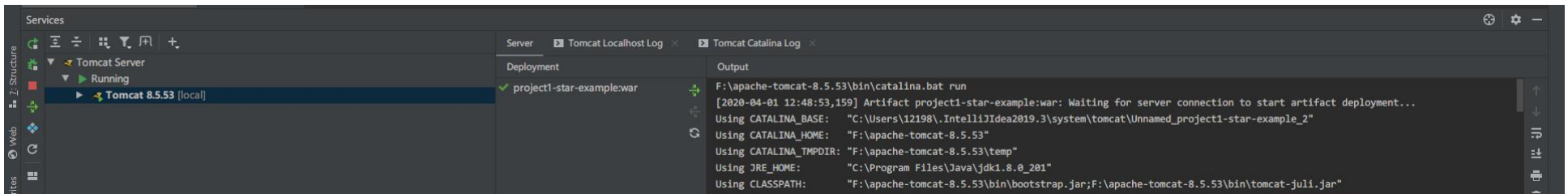
6. Click Apply then OK. Now you've finished the configuration. (You can choose either **war** or **war exploded** or both)



7. You can use the Run/Debug button in the right upper corner to Run/Debug your program.



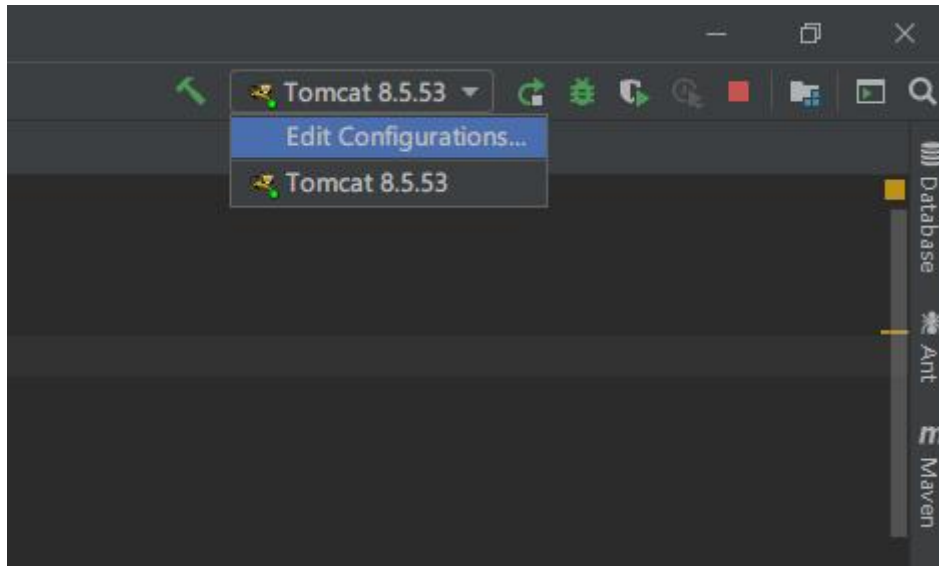
8. This is a screenshot of running the program



Your WAR file should be in the `target/project1-star-example.war`

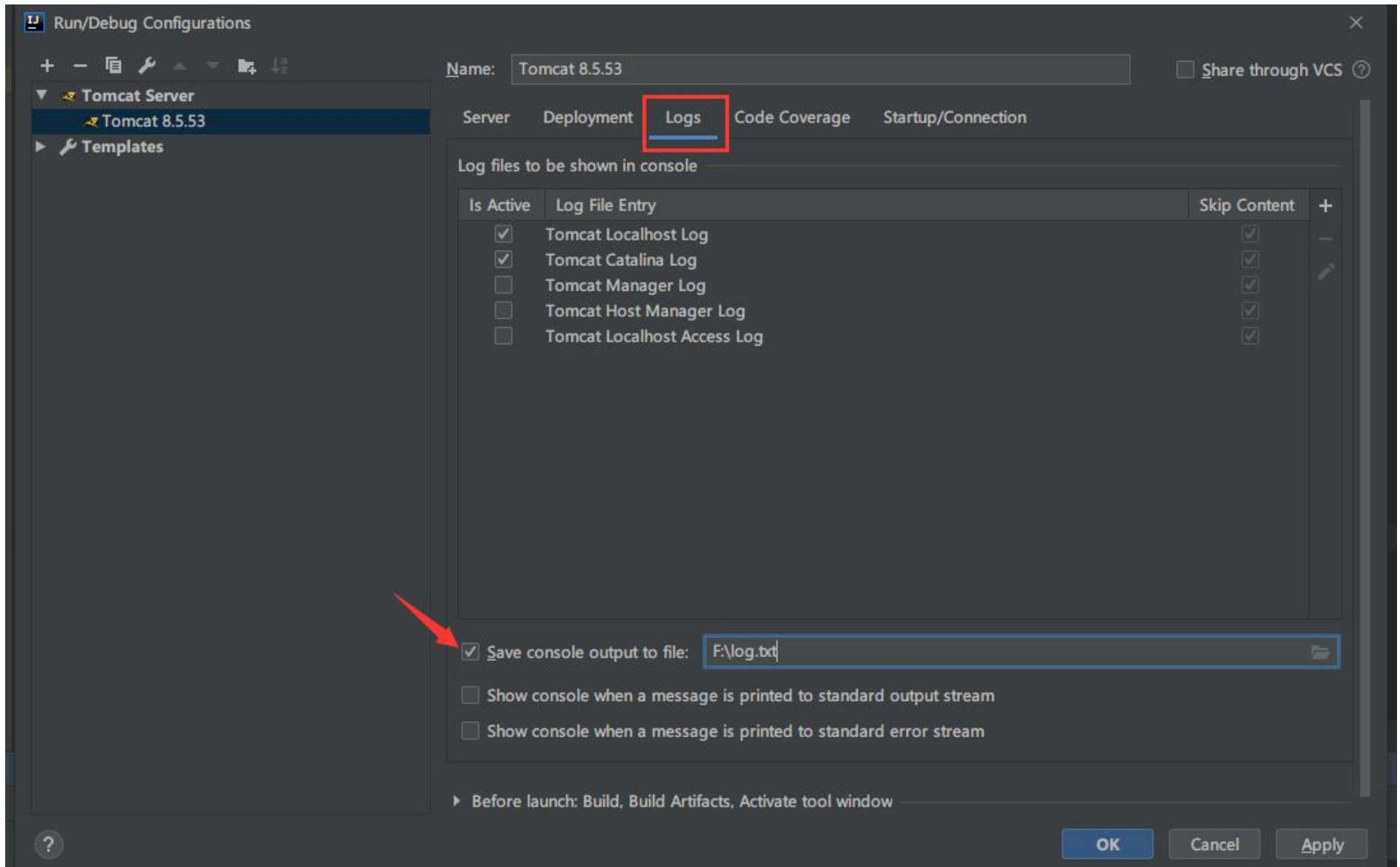
▼ Tomcat Logging in IntelliJ

- After IntelliJ starts, click "**Edit Configuration**" in the right upper corner.



- Click the **Log** section. The log file content is displayed on the console by default.

- You can also output the log into a file.




▼ Setup Maven

Maven is a tool to manage project dependencies and automate the build process.

Maven is built-in with IntelliJ IDEA. But to run it on the command line, you must install maven on your development machine.

To install Maven:

- Windows: follow [this tutorial](https://www.google.com/search?q=install+maven+on+windows&oq=install+maven+on+windows&aqs=chrome..69i57j0l5.3082j0j4&sourceid=chrome&ie=UTF-8)  (<https://www.google.com/search?q=install+maven+on+windows&oq=install+maven+on+windows&aqs=chrome..69i57j0l5.3082j0j4&sourceid=chrome&ie=UTF-8>)


- Mac: Run


```
brew install maven
```

- Ubuntu: Run

```
sudo apt-get install maven
```

▼ Setup .gitignore file

When putting your project code in the Git repository, a good practice of using Git is only to put the source code files in the Git repository and ignore all the compiled files (such as .class files, executables) and other irrelevant files (such as files generated by Eclipse). The purpose of the `.gitignore` file is to tell git to ignore specific files according to some patterns. You can learn more about gitignore 

(https://medium.com/@haydar_ai/learning-how-to-git-ignoring-files-and-folders-using-gitignore-177556afdbe3) [here](https://medium.com/@haydar_ai/learning-how-to-git-ignoring-files-and-folders-using-gitignore-177556afdbe3) 
(<https://www.atlassian.com/git/tutorials/saving-changes/gitignore>).

We provide a sample [gitignore](https://canvas.eee.uci.edu/courses/54347/files/22162673/download?wrap=1) (<https://canvas.eee.uci.edu/courses/54347/files/22162673/download?wrap=1>)  https://canvas.eee.uci.edu/courses/54347/files/22162673/download?download_frd=1) file you can use directly in your project. Download the file and put the file into your project git folder. Then rename the file from "gitignore" to ".gitignore" (notice we **add a dot "."** in the beginning). For Mac and Linux users, the file then becomes hidden, don't worry, this is the correct behavior. Go to the terminal and type "git status" to see the ".gitignore" file. For Windows users, see this [StackOverflow](https://stackoverflow.com/questions/10744305/how-to-create-gitignore-file) 
(<https://stackoverflow.com/questions/10744305/how-to-create-gitignore-file>) question to create the ".gitignore" file. Git will then automatically read the ".gitignore" file and ignore the unnecessary files.

▼ Task 5: Create a MySQL Database

Create a database called "moviedb" with the following tables:

Table Name	Attributes	Notes
------------	------------	-------

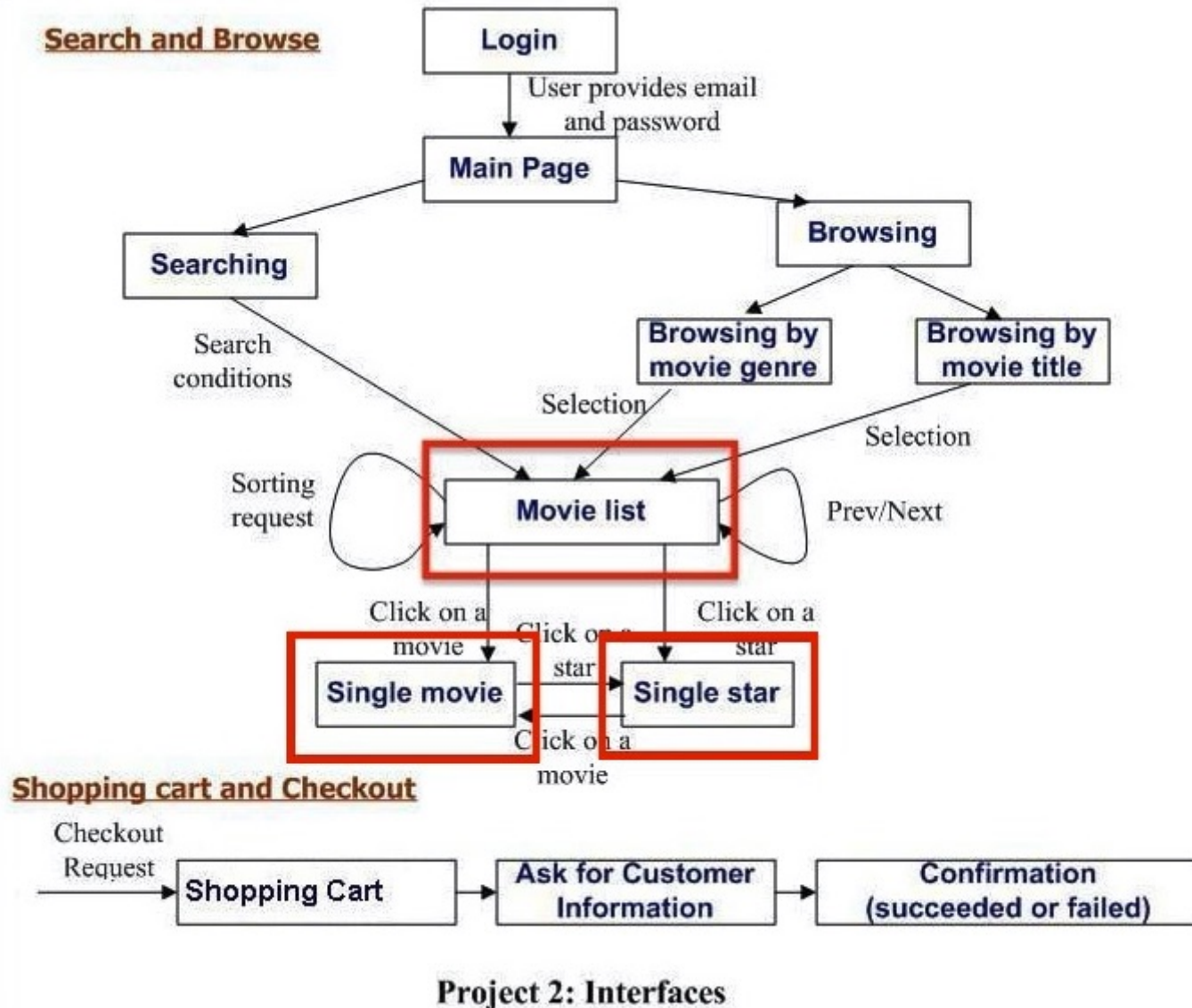
movies	id:varchar(10) (primary key) title:varchar(100) year:integer director:varchar(100)	required required required required
stars	id:varchar(10) (primary key) name:varchar(100) birthYear:integer	required required not required
stars_in_movies	starId:varchar(10), referencing stars.id movieId:varchar(10), referencing movies.id	all attributes required
genres	id:integer (primary key) name:varchar(32)	all attributes required; "id" should be "AUTO_INCREMENT"
genres_in_movies	genreId:integer, referencing genres.id movieId:varchar(10)], referencing movies.id	all attributes required
customers	id:integer (primary key) firstName:varchar(50) lastName:varchar(50) ccId:varchar(20), referencing creditcards.id address:varchar(200) email:varchar(50) password:varchar(20)	all attributes required; "id" should be "AUTO_INCREMENT"
sales	id:integer (primary key) customerId:integer, referencing customers.id movieId:varchar(10), referencing movies.id saleDate:date	all attributes required; "id" should be "AUTO_INCREMENT"
creditcards	id:varchar(20), (primary key) firstName:varchar(50) lastName:varchar(50) expiration:date	all attributes required

ratings	movieId:varchar(10), referencing movies.id rating:float numVotes:integer	all attributes required
---------	--	-------------------------

The table-creation SQL statements should be written in a **createtable.sql** file. All varchar() fields for which there is no data (i.e., the fields contents are missing or unknown) are the empty string (""); other non-required fields which have no data are null. Required fields have the constraint that they are not null.

Use the provided [movie-data.sql](https://canvas.eee.uci.edu/courses/54347/files/22162680/download) (<https://canvas.eee.uci.edu/courses/54347/files/22162680/download>) file to populate the tables.

▼ Task 6: Implement Movie List Page, Single Movie Page, Single Star Page



For Project 1, you only need to implement the Movie List, the Single Movie, and the Single Star page of the Fabflix Application. You'll implement other parts in Project 2. You do **NOT** need to implement other sorting and prev/next features. You must not:

- Implement any functionality in the web app using frameworks or scaffolding tools, such as DataTables.
- Store data directly in the frontend instead of the database.

You must code from the ground up.

Movie List Page

In Project 1, the Movie list Page shows the top 20 rated movies, sorted by the rating. You don't need to show all the movies. Each movie needs to contain the following information:

- title;
- year;
- director;
- first three genres (order does not matter) ;
- first three stars (order does not matter);
- rating.

Single Movie Page

From Movie List Page or Single Star Page, if the user clicks on a movie (hyperlinked), the corresponding Single Movie page displays all the information about the movie, including:

- title;
- year;
- director;
- all of the genres;
- all of the stars (hyperlinked);
- rating.

Single Star Page

From the Movie List Page or Single Movie Page, if the user clicks on a star (hyperlinked), the corresponding Single Star Page displays all the information about this star, including:

- name;
- year of birth (N/A if not available);
- all movies (hyperlinked) in which the star acted.

Jump Requirement

1. The user can jump between Single Movie Page and Single Star Page following hyperlinks.

2. The user can jump back to Movie List Page from any single page.

▼ Examples and Resources

1. **JDBC Example: Connection to MySQL**  <https://github.com/UCI-Chenli-teaching/cs122b-project1-jdbc-example>

Your Java Servlet needs to talk to MySQL using JDBC. This example has two sample JDBC programs that use Maven to manage JDBC. Please see README for instructions on creating the required database/tables and executing the sample code.

2. **Star Example: Tomcat Servlet**  <https://github.com/UCI-Chenli-teaching/cs122b-project1-star-example>

This example shows how to create a Tomcat Servlet. Servlet could serve HTML directly (Server Generated HTML) or data in JSON format (Micro-Service-based). This example shows you how to serve HTML directly. You need to deploy the example:

- You can run the project directly from IntelliJ if you have finished the IntelliJ Tomcat Integration steps. Follow the instruction in README.
- To directly deploy the WAR file to tomcat, use the tomcat manager webpage or the command line.
 - To deploy the WAR file using the Tomcat manager webpage:
 1. Start the Tomcat server. Go to <http://localhost:8080/> (<http://localhost:8080/>); this is your local web page. The tomcat welcome page should appear.
 2. Click "Tomcat Manager", enter your tomcat admin username and password.
 3. Under "Deploy directory or WAR file located on server" you should see a deploy button. You need not specify a path. If you click the deploy button, Tomcat will automatically refresh its list of applications.
 4. the new project should appear in the list. Click on the name of the project, and it should take you to the index.html page
 5. To see the star servlet, go to "localhost:8080/cs122b-fall21-project1-star-example/stars", you should see a table containing star information.
 - To deploy the WAR file in the command line:
 1. generate the WAR file using "mvn clean package". The war file is in the "target" folder.
 2. copy the WAR file to "tomcat_directory/webapps". For example "cp target/project1-star-example.war tomcat_directory/webapps/"

3. **API Example: Frontend-Backend separation**  <https://github.com/UCI-Chenli-teaching/cs122b-project1-api-example>

An "old" way to develop Web sites is to use Java Servlets to generate HTML pages. This approach is gradually becoming outdated and no longer considered a good practice. We require you to separate the frontend and backend. The backend will be a microservice that provides API in a JSON format to the frontend. The frontend fetches the data by sending HTTP requests to the backend and then displays the returned data. The frontend must be written in HTML, CSS, and JS files.

This example uses frontend-backend separation, similar to the project1-star-example we provided. Starting with this example, we use DataSource managed by Tomcat. Thus we don't have to set database information in each servlet. To config, a DataSource, follow the tutorial here <https://tomcat.apache.org/tomcat-9.0-doc/jndi-datasource-examples-howto.html> (<https://tomcat.apache.org/tomcat-9.0-doc/jndi-datasource-examples-howto.html>). Follow the instruction on README.md to deploy and test the examples on tomcat.

Other Online Resources

Here are some relevant tutorials. You can start the project after learning the basics. Only spend a little bit of time learning the advanced content. You will use them in Project 2.

- HTML and CSS tutorial: <http://html5dog.com/guides/html/beginner/> (<http://html5dog.com/guides/html/beginner/>)
- An interactive HTML tutorial: <https://www.codecademy.com/learn/learn-html> (<https://www.codecademy.com/learn/learn-html>)
- A good Java Servlet Tutorial: <http://tutorials.jenkov.com/java-servlets/index.html> (<http://tutorials.jenkov.com/java-servlets/index.html>)
- Another good Java Servlet Tutorial: <http://o7planning.org/en/10169/java-servlet-tutorial> (<http://o7planning.org/en/10169/java-servlet-tutorial>)

▼ Rubric

General requirements:

- Response time matters for web applications. Each request should be handled within approximately **500 ms** unless specified otherwise.
 - Please note this is an approximate time range, as response time will depend on many factors, especially network conditions.
Graders will not use a timer when grading but more of a sense of speed.
 - If your pages need multiple seconds to load, you must optimize your requisition, query, or database structure.
- The frontend should not cache more than **100** records.

	Entry	Points
AWS	AWS instance is set up properly	10

	Access Tomcat app manager on AWS	5
Git/GitHub	Git repository is IDE free and not leaking sensitive information: does not contain IntelliJ/Eclipse files (.idea, .project, .iml, etc), build targets, datafiles.	5
	Git repository contains all the source code for project 1, not using the GitHub upload file feature.	5
Maven	Web project is set up as a maven project correctly (or equivalent tool if used other frameworks)	5
Database	Can create tables using your create_table.sql on MySQL hosted on AWS	5
	Can populate tables provided using movie-data.sql on MySQL hosted on AWS	10
Movie List Page	For each movie, show all required information (title, year, director, 3 genres, 3 stars, rating)	15
	Hyperlink each movie to the Single Movie Page	2.5
	Hyperlink each star to the Singer Star Page	2.5
	Only show top 20 rated movies, sorted correctly	10
Single Movie Page	Display title, year, director, all genres, all stars, rating	5
	Hyperlink each star to Singer Star Page	2.5
Single Star Page	Show name, year of birth, all movies in which the star performed	5
	Hyperlink each movie to the Single Movie Page	2.5
Jump Functionality	Jump back to Movie List Page from Single Movie Page and Single Star Page (not browser history)	5
Report/README	<p>Include a short README.md with</p> <ol style="list-style-type: none"> 1) Demo video URL. 2) Each member's contribution to the project. 	5

Extra Credit	Show efforts to beautify the page and table using CSS	0~5
Total		100 + 5

▼ Demonstration

In summary, we require 2 things for the demo:

1. A screen recording demo video shows your web application working on AWS.
2. Leave an AWS instance running with the web application deployed for a week.

Screen recording demo video

- The entire demo should be on AWS. **NO** local demo allowed.
- You need a terminal that connects to AWS and a web browser open to the tomcat manager page on AWS (<http://<AWS public IP>:8080/manager/html>).
- Use screen recording tools (e.g, QuickTime Player for macOS and Open Broadcaster Software for Windows, or other equivalents)
- Please keep the length of the video within 15 minutes. However, do **NOT** edit the video in any way. Otherwise, it could be treated as cheating.
- **Upload the video to Youtube** or other public media hosting websites, make sure the URL is publicly accessible, then **include your video URL in README.md in your github repository**.
- Preparation before the demo:
 - ssh onto AWS instance.
 - on the AWS instance, drop the MySQL database `moviedb`.
 - remove your git repo from the AWS instance. The demo should start with a freshly cloned repo.
 - prepare movie-data.sql (you could modify it, but you are NOT allowed to remove any of the data we provided) and store it under `/home/ubuntu/`.
 - make sure MySQL and Tomcat are running on an AWS instance.
- Demo on AWS instance:
 - **Commit check:**

1. Clone your git repo to AWS instance: ``git clone <repo url>``. This step is to make sure you have a clean repo.
2. ``cd`` into your repo and run ``git log``, show your latest commit.
3. Failure to show commit check will result in an instant 0 of the project.
- **Build a MySQL database and populate data on AWS MySQL:**
 1. show all databases, should not have ``moviedb`` now:

```
mysql -u mytestuser -p -e "show databases;"
```
 2. build your ``moviedb``:

```
mysql -u mytestuser -p < create_table.sql
```
 4. populate the data:

```
mysql -u mytestuser -p --database=moviedb < /home/ubuntu/movie-data.sql
```
 5. show your data counts:

```
mysql -u mytestuser -p -e "use moviedb;select count(*) from stars;select count(*) from movies;"
```
- **Deploy your web application on AWS instance:**
 1. inside your repo, where the pom.xml file locates, build the war file:

```
mvn package
```
 2. show tomcat web apps, it should NOT have the war file yet:

```
ls -lah /var/lib/tomcat/webapps/
```
 3. copy your newly built war file:

```
cp ./target/*.war /var/lib/tomcat/webapps/
```
 4. show tomcat web apps, it should now have the new war file:


```
ls -lah /var/lib/tomcat/webapps/
```
- **Show your website in your web browser:**
 1. Refresh the tomcat manager page. You should see a new project (just deployed): project 1.
 2. Click the project link, which goes to your website's landing page (could be the Movie List Page).
 3. Navigate to the Movie List Page. Scroll up and down to show all 20 movies if needed.
 4. Click on a movie title hyperlink on the Movie List Page. Should jump to Single Movie Page. Show all information on the Single Movie Page.
 5. Click on a star name hyperlink on the Single Movie Page. Should jump to Single Star Page. Show all information on Single Star Page.
 6. On the Single Star Page, return to Movie List Page without using browser history. You can do this by clicking a button or link.

7. Repeat steps 4 - 6: this time, click a star name (to Single Star Page), then click a movie played by that star (to Single Movie Page), and return to Movie List Page.

- **Done.**
 - Here is an [example](https://youtu.be/tBf7ZrwW1lg)  [.\(https://youtu.be/tBf7ZrwW1lg\)](https://youtu.be/tBf7ZrwW1lg) demo video.
-

▼ Submission

First, submit on **GitHub** by pushing to the `main` branch.

- make sure your git repository contains the following:
 - all source code, including Java, SQL, HTML, CSS, etc...
 - .gitignore file to exclude those sensitive files (IDE files, certs, etc.) and built targets (WAR files, binaries, etc.).
 - README.md (contains your demo video URL and contributions for each member)
- Update your web application URL in [this sheet](https://docs.google.com/spreadsheets/d/1Dzulxj1BqqPxn2nkCwvsS5-hoorW948n-eJ0dZTAZDc/edit?usp=sharing).  [.\(https://docs.google.com/spreadsheets/d/1Dzulxj1BqqPxn2nkCwvsS5-hoorW948n-eJ0dZTAZDc/edit?usp=sharing\)](https://docs.google.com/spreadsheets/d/1Dzulxj1BqqPxn2nkCwvsS5-hoorW948n-eJ0dZTAZDc/edit?usp=sharing)
- AWS IP address can differ after submission if you restart the AWS instance. Update the URL on the google sheet.

Next, submit on **Gradescope** by selecting the `main` branch.

- Please add your teammate to your submission.
- Only one member of the team needs to submit the code.
- We will grade based on the submission on gradescope.
- You have a 24-hour grace period for submission, and the gradescope submission determines the usage of the grace period.