

# Continuous integration of a Time tracking application

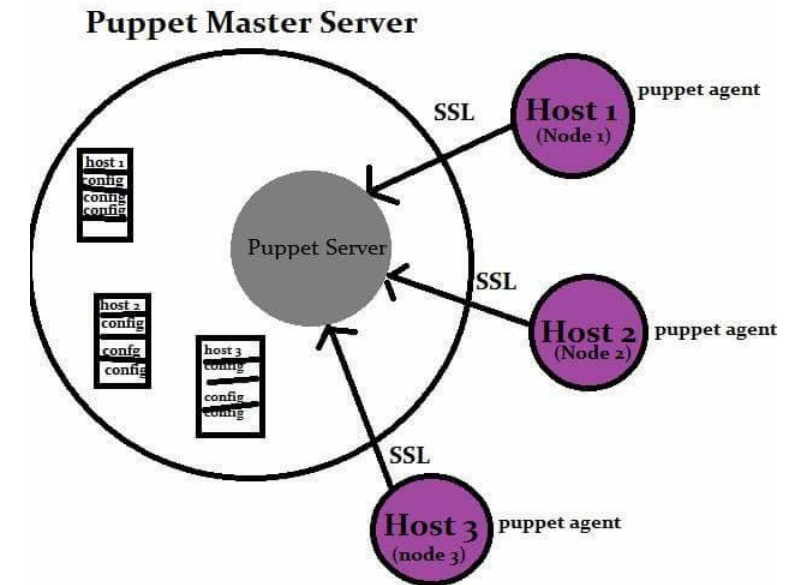
Team members.	Matrik.nr.	Roles
1. Y.Manoj reddy	210040	Operations engineer
2. Ikram Ul Haq	209372	Software engineer
3. P. Bala Krishna	210049	Test engineer
4. M. Gokul	206744	Software engineer

# Content

- Configuration with Puppet
- Continuous integration & continuous delivery
- Jenkins
- Maven with selenium
- Mockito

# Configuration of server with Puppet

- Puppet– An automation tool aid to configure a central server.
- Master less puppet environment
  - i) manifests and modules distributed to machines
- Ubuntu 14.04 is virtual private server.



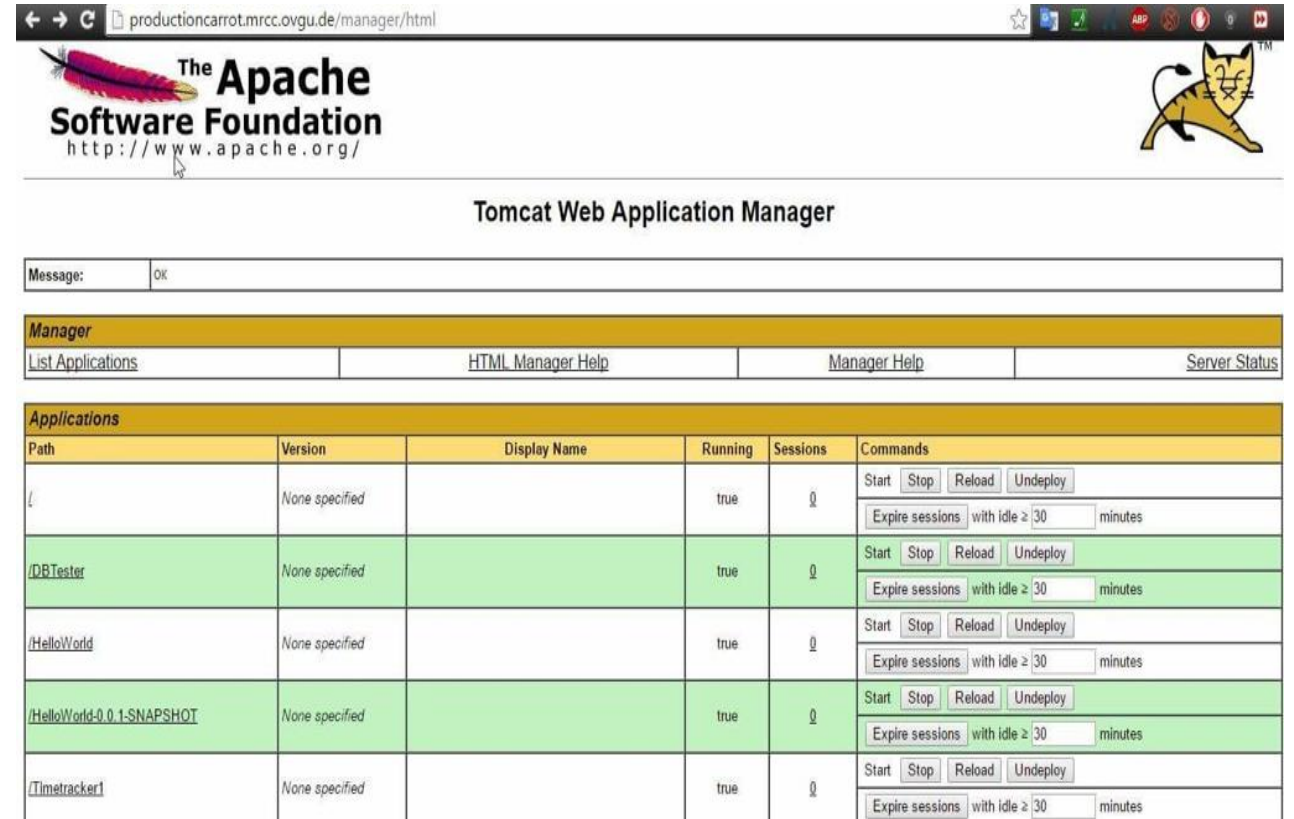
## Configuration server with Puppet/2

- Downloaded and installed puppet for Ubuntu 14.04
- Commands for installation

```
# wget http://apt.puppetlabs.com/puppetlabs-release-trusty.deb  
# dpkg -I /tmp/puppetlabs-release-trusty.deb
```
- A puppet manifest decides the puppet work structure.
- Manifests consists of following resource declarations  
Exec, Package, Service, File.

# Configuration server with Puppet/3

- Apache & tomcat provisioned to the server .
- Init.pp file and site.pp are the main manifests.
- Cnames of the server are  
productioncarrot.mrcc.ovgu.de  
testingcarrot.mrcc.ovgu.de

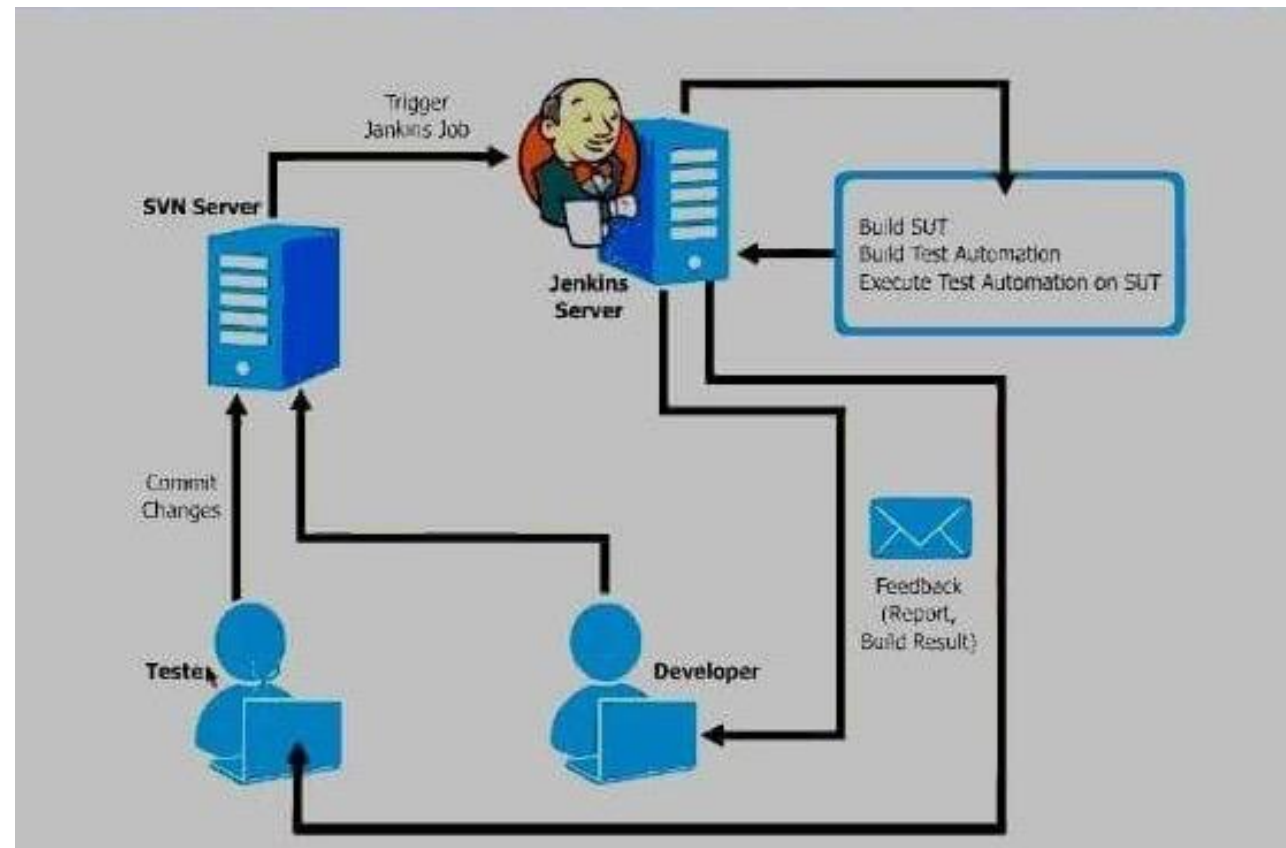


The screenshot shows the Tomcat Web Application Manager interface in a web browser. The browser address bar shows the URL: productioncarrot.mrcc.ovgu.de/manager/html. The page header includes the Apache Software Foundation logo and the text "The Apache Software Foundation http://www.apache.org/". The main title is "Tomcat Web Application Manager". Below the title, there is a message box showing "Message: OK". The interface is divided into two main sections: "Manager" and "Applications". The "Manager" section has links for "List Applications", "HTML Manager Help", "Manager Help", and "Server Status". The "Applications" section is a table with columns: Path, Version, Display Name, Running, Sessions, and Commands. The table lists several applications, including "/", /DBTester, /HelloWorld, /HelloWorld-0.0.1-SNAPSHOT, and /Timetracker1. Each application row has buttons for Start, Stop, Reload, and Undeploy, as well as an "Expire sessions" button with a dropdown menu set to "with idle ≥ 30 minutes".

Path	Version	Display Name	Running	Sessions	Commands
/	None specified		true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/DBTester	None specified		true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/HelloWorld	None specified		true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/HelloWorld-0.0.1-SNAPSHOT	None specified		true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/Timetracker1	None specified		true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes

## Continuous integration & continuous deployment

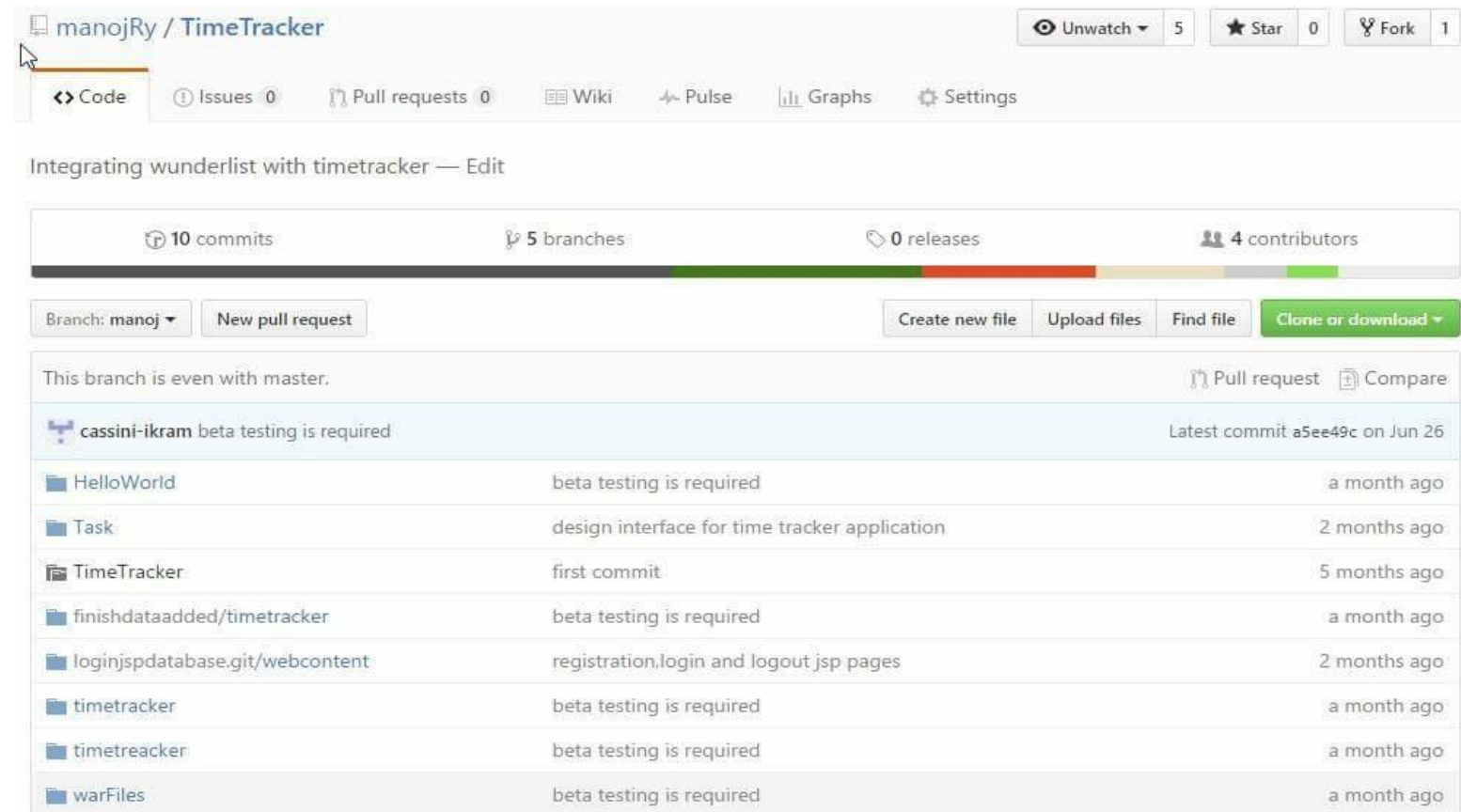
- Merging all developer working copies into a shared repository
- Fast development process





# Git hub

- GitHub as the shared repository for developers
- Source code is pulled in Jenkins after developer commit



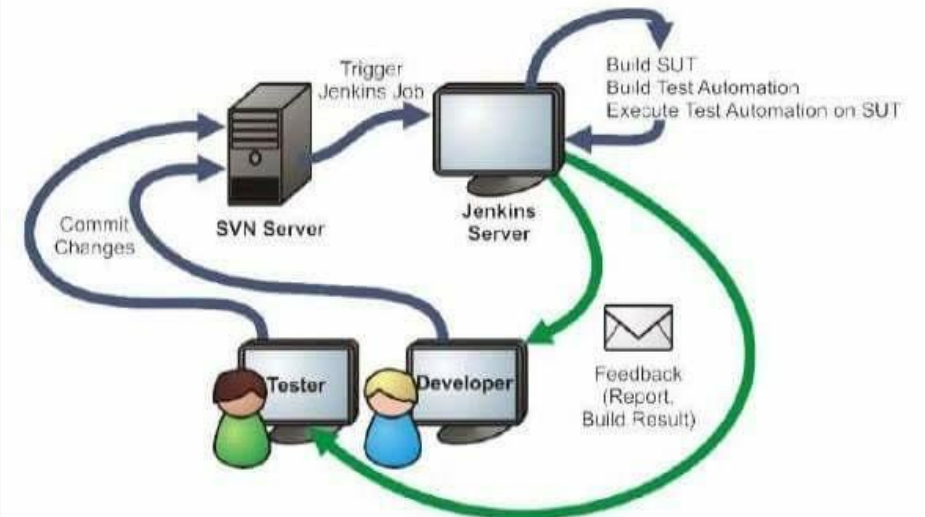
The screenshot shows the GitHub interface for the repository 'manojRy / TimeTracker'. At the top, there are navigation links: 'Code', 'Issues 0', 'Pull requests 0', 'Wiki', 'Pulse', 'Graphs', and 'Settings'. Below these, a progress bar indicates the repository's status: 10 commits, 5 branches, 0 releases, and 4 contributors. A 'Branch: manoj' dropdown and a 'New pull request' button are visible. The main content area shows a list of commits, with the most recent one being 'cassini-ikram beta testing is required' (commit a5ee49c on Jun 26). Below this, a table lists files and their commit history:

File	Commit Message	Time
HelloWorld	beta testing is required	a month ago
Task	design interface for time tracker application	2 months ago
TimeTracker	first commit	5 months ago
finishdataadded/timetracker	beta testing is required	a month ago
loginjspdatabase.git/webcontent	registration, login and logout.jsp pages	2 months ago
timetracker	beta testing is required	a month ago
timetracker	beta testing is required	a month ago
warFiles	beta testing is required	a month ago

# Jenkins

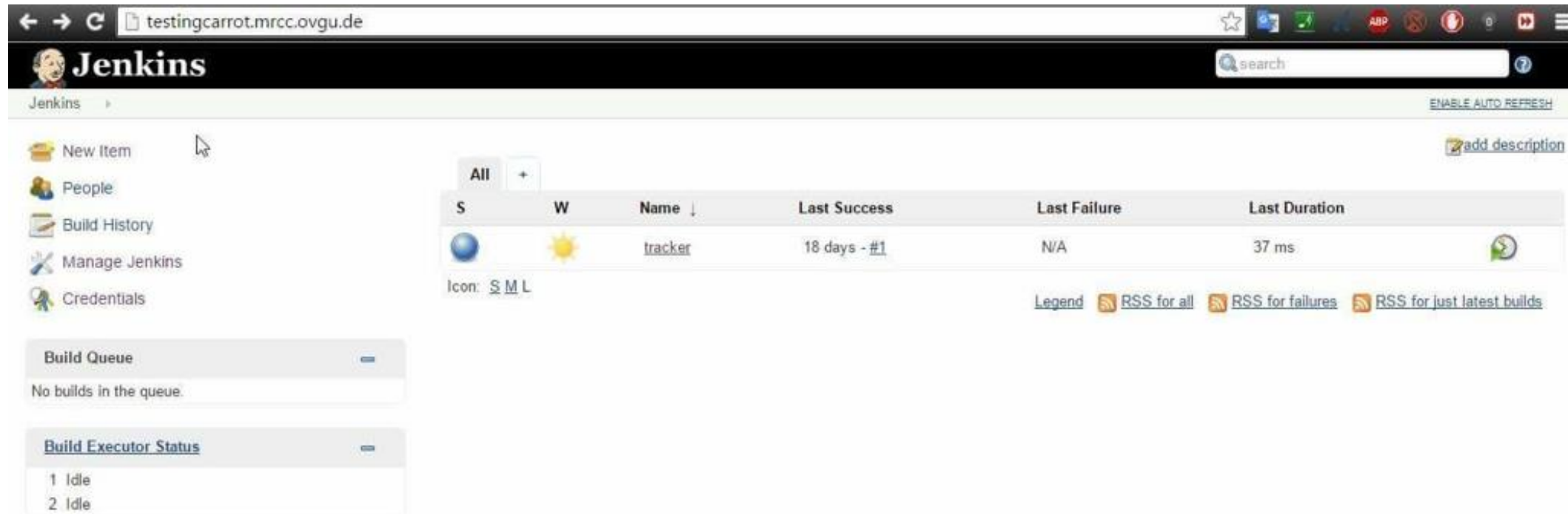
- Open source continuous integration tool
- Can perform unit test, integration test, determine build status ....
- Developers will receive a feedback of their build

## Workflow View










- Jenkins setup on testingcarrot.mrcc.ovgu.de



The screenshot shows the Jenkins web interface. The top navigation bar includes the Jenkins logo, a search bar, and a link to 'ENABLE AUTO REFRESH'. The left sidebar contains links for 'New Item', 'People', 'Build History', 'Manage Jenkins', and 'Credentials'. The main content area displays a table of build jobs. The table has columns for 'S' (Success), 'W' (Warning), 'Name', 'Last Success', 'Last Failure', and 'Last Duration'. A single job named 'tracker' is listed, showing a success status, a last success time of '18 days - #1', and a last duration of '37 ms'. Below the table, there is a legend for RSS feeds and a section for 'Build Queue' and 'Build Executor Status'.

S	W	Name ↓	Last Success	Last Failure	Last Duration
		<a href="#">tracker</a>	18 days - #1	N/A	37 ms

Icon: [S](#) [M](#) [L](#)

Legend  [RSS for all](#)  [RSS for failures](#)  [RSS for just latest builds](#)

**Build Queue**  
No builds in the queue.

**Build Executor Status**  
1 Idle  
2 Idle

## Console Output

```
Started by user anonymous
Building in workspace /var/lib/jenkins/workspace/tracker
Finished: SUCCESS
```

## Jenkins/3

- Installed Jenkins on Ubuntu 14.04 through repository

- Installation commands of Jenkins

```
# wget -q -O - https://Jenkins-ci.org/debian/jenkins-ci.org.key | sudo apt -key -add
```

```
# sh -e 'echo deb http://pkg.jenkins-ci.org/debianbinary />/etc/apt/sources.list.d/jenkins.list
```

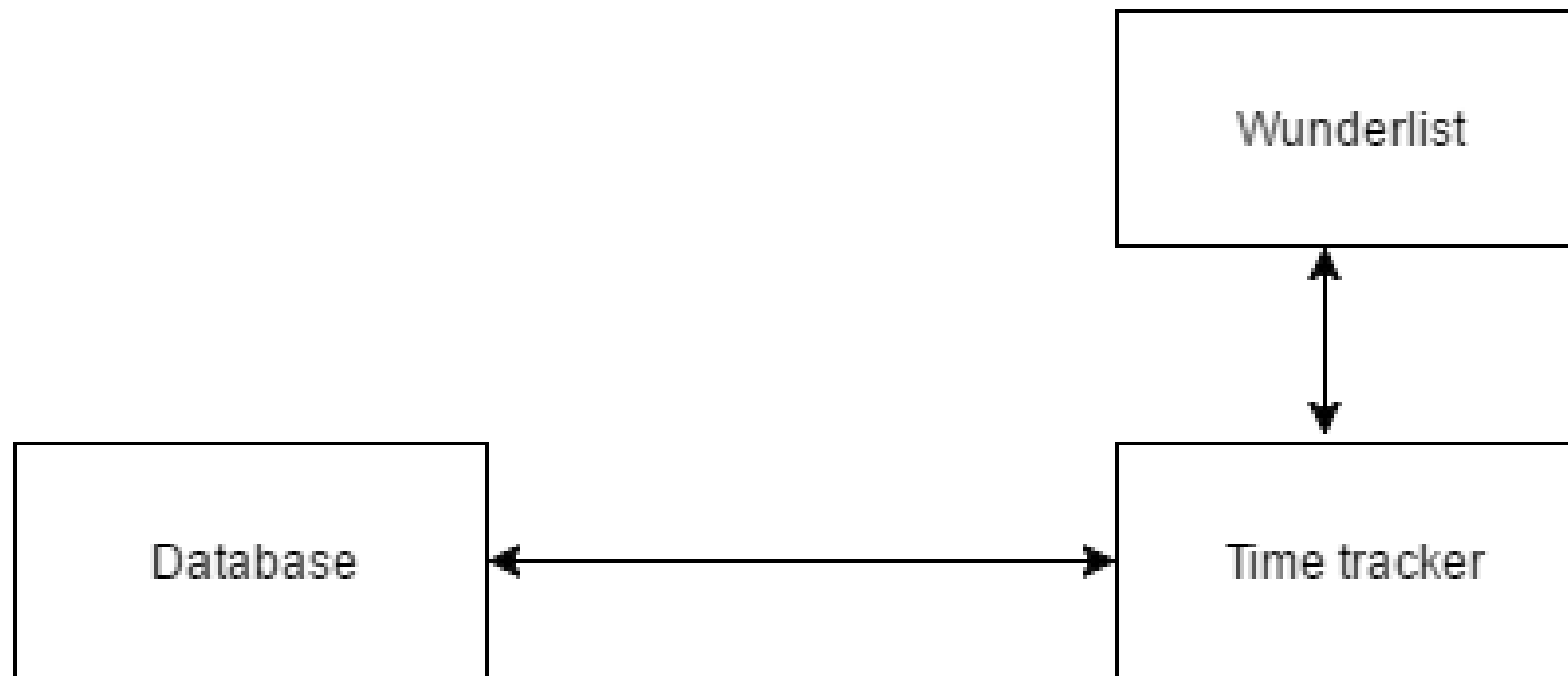
```
#apt -get install jenkins
```

- Puppet run is performed after installing Jenkins

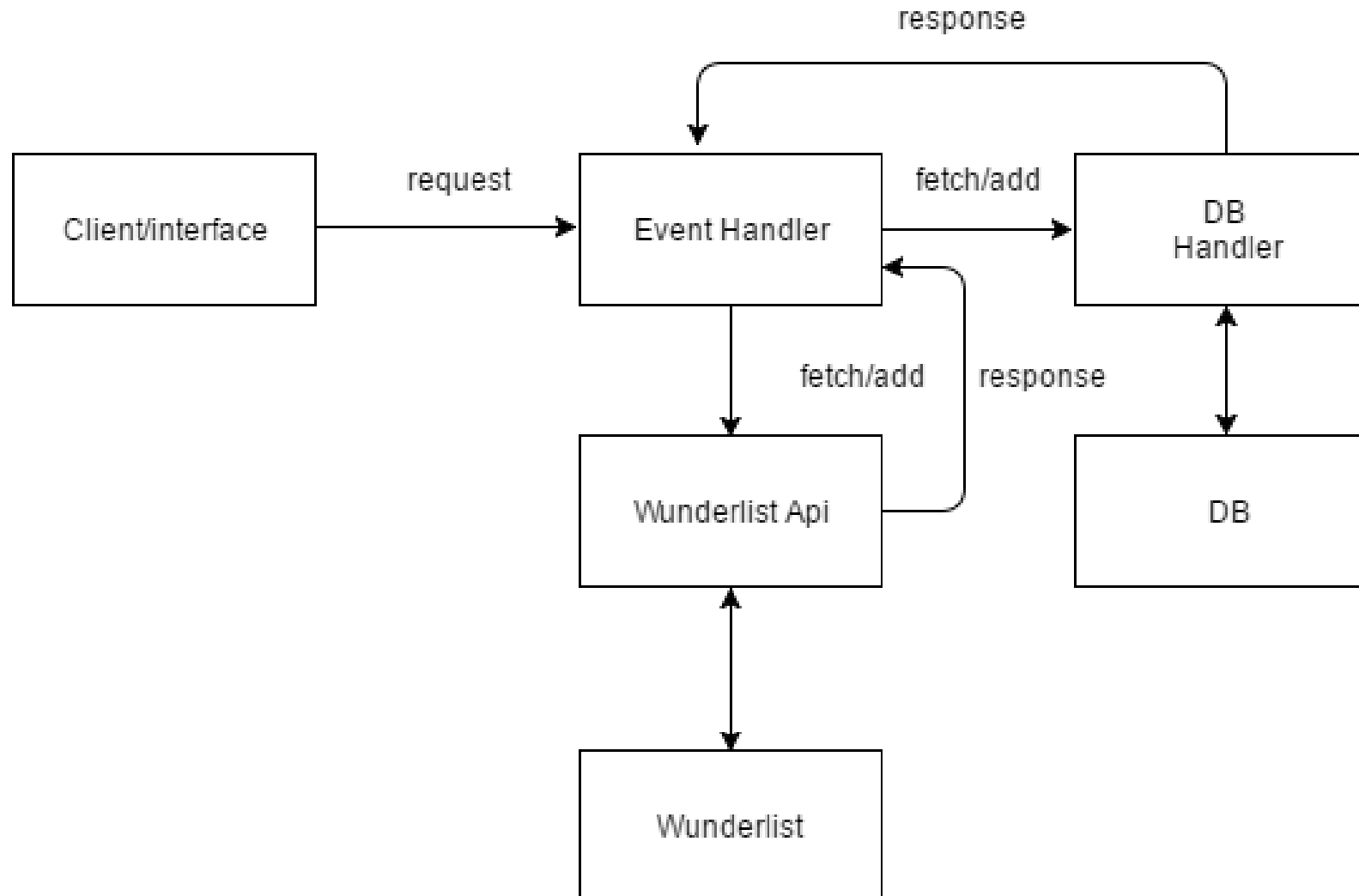
# Time Tracking Application

Synchronized with [wunderlit.com](https://wunderlit.com)

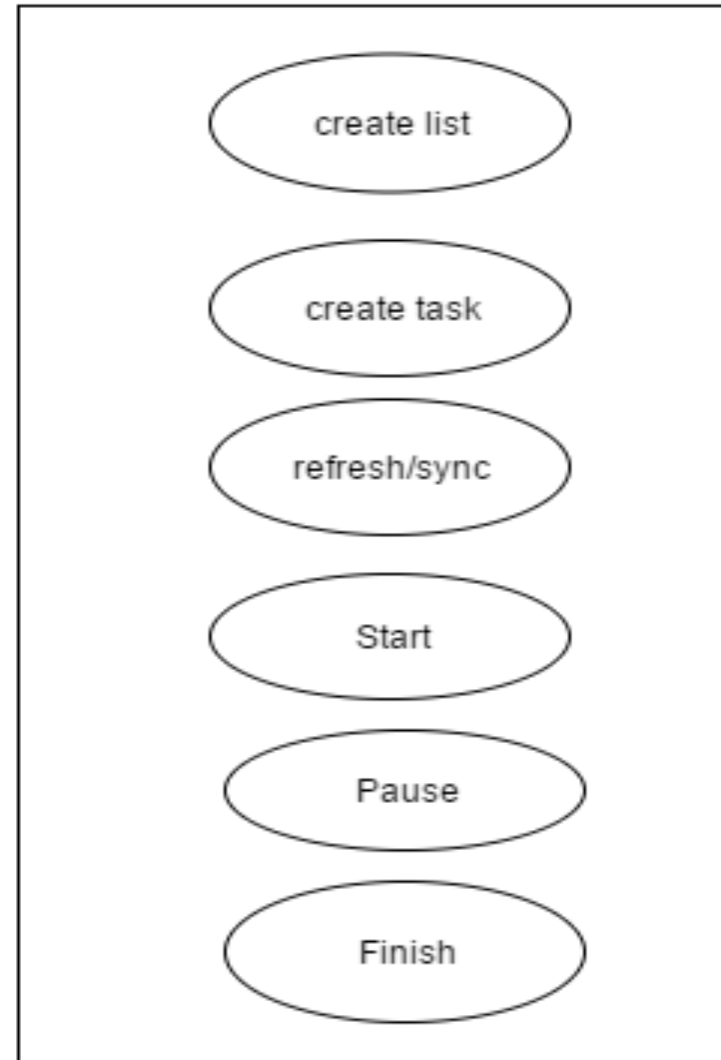
# Basic Architecture



# Request Processing

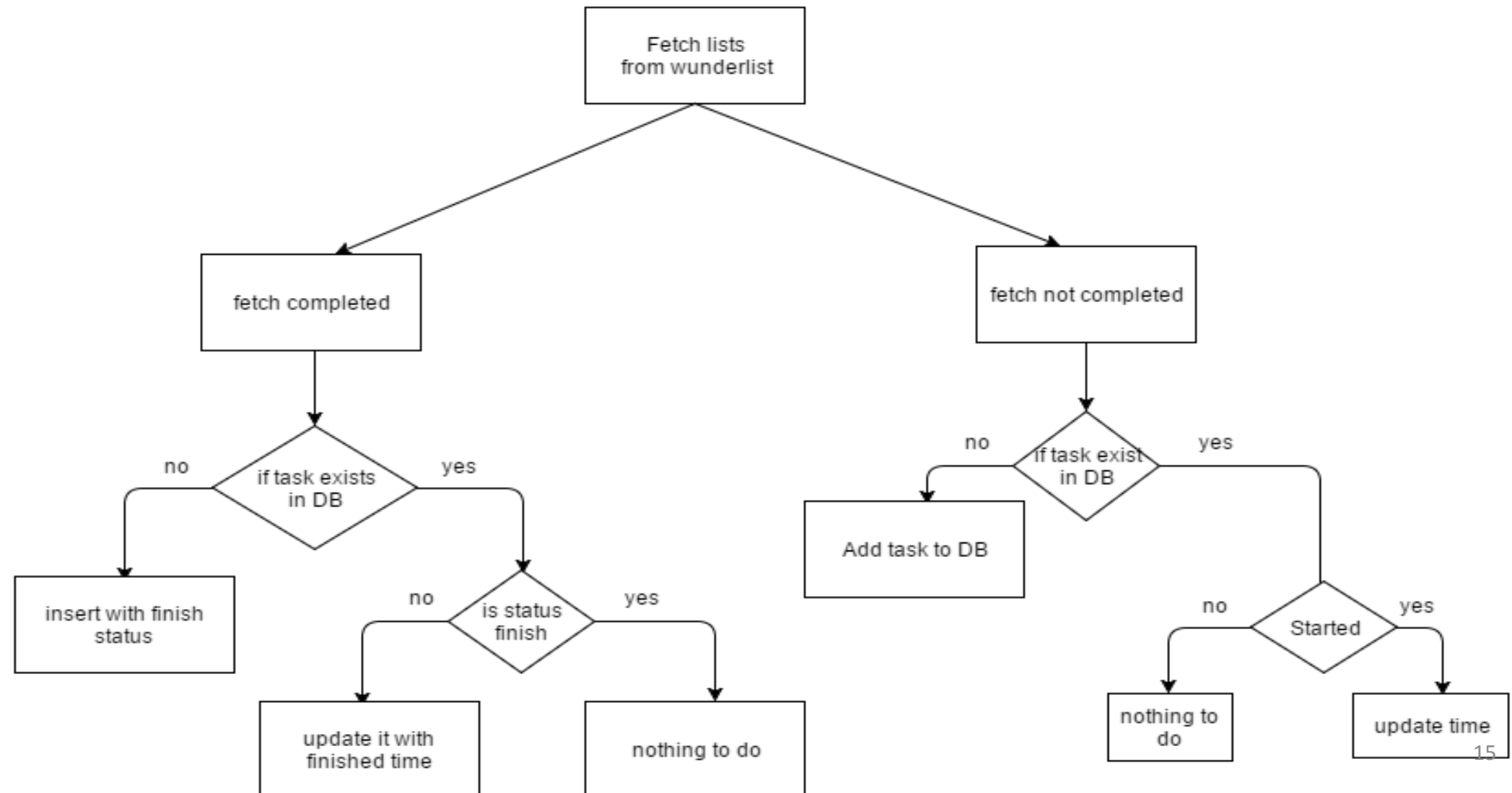


# Operations

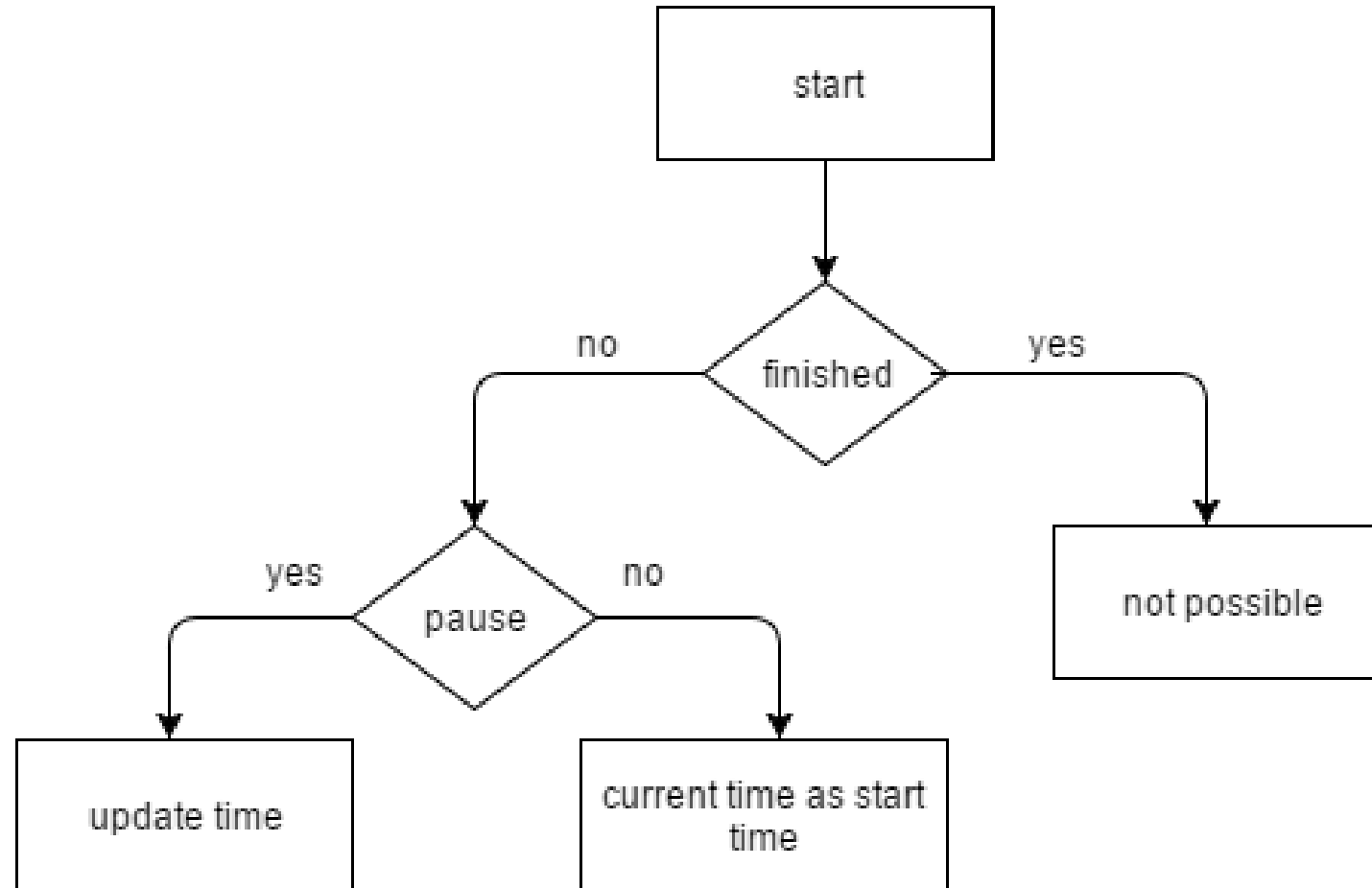




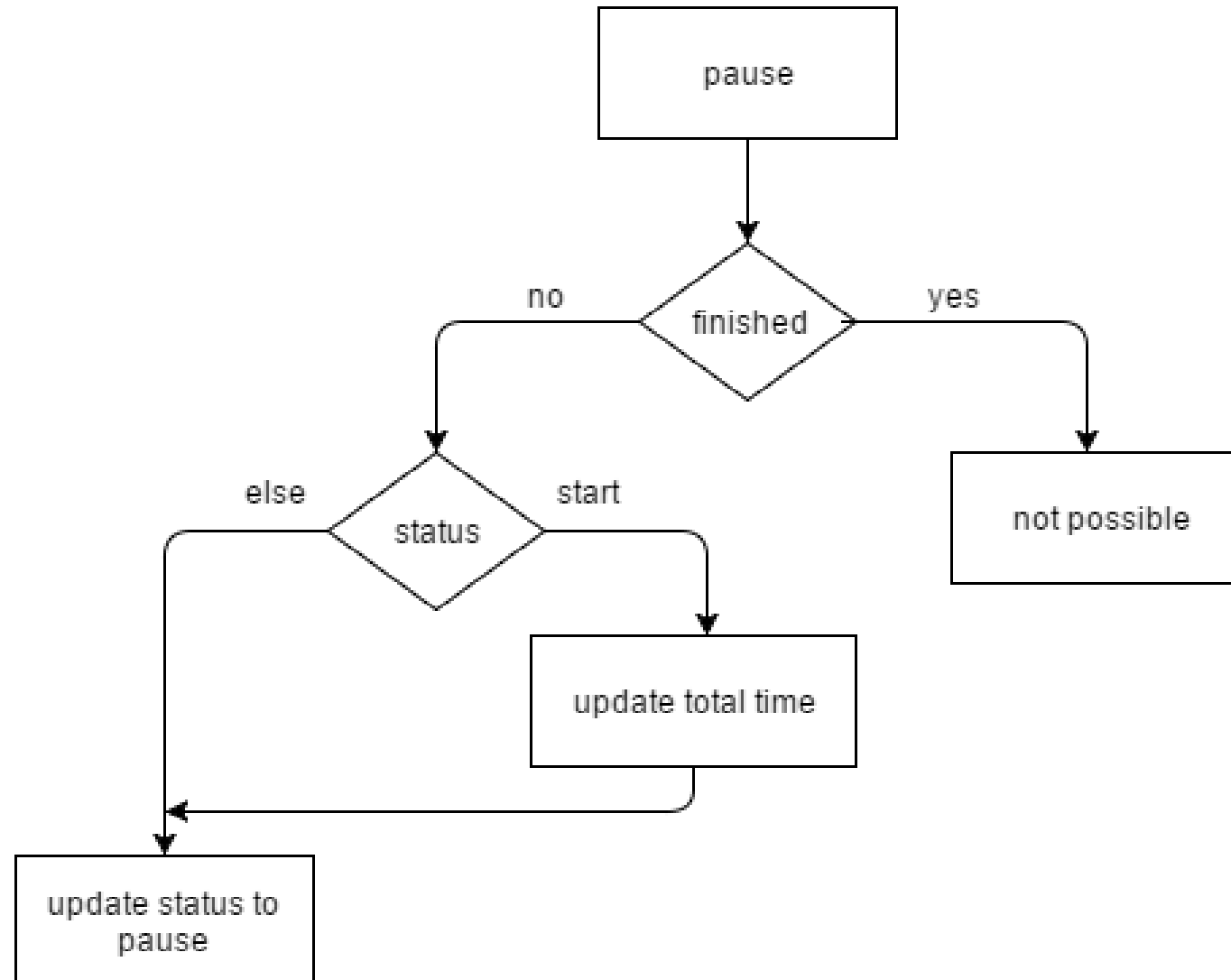
# Loading/Refresh



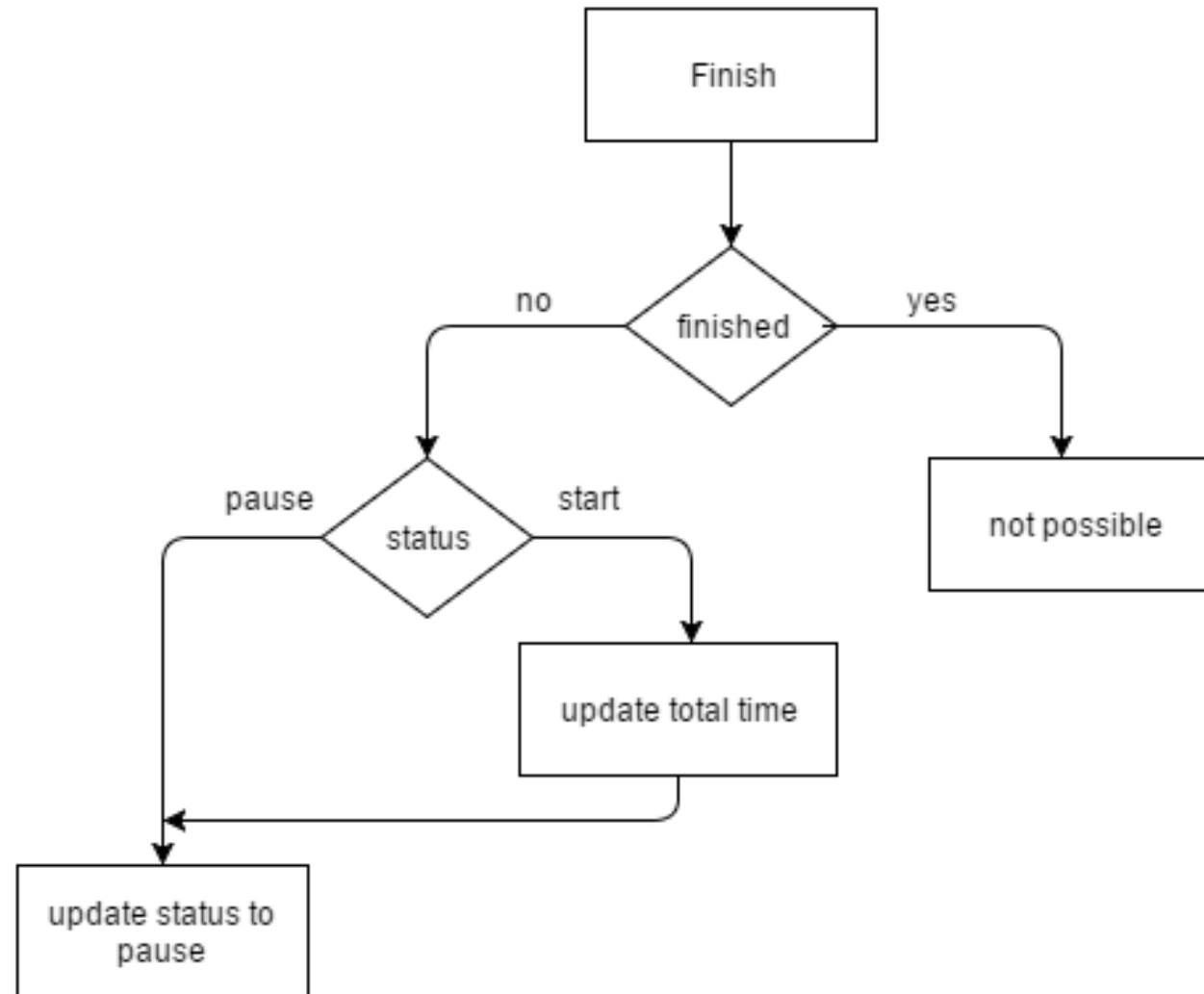
# Start Task



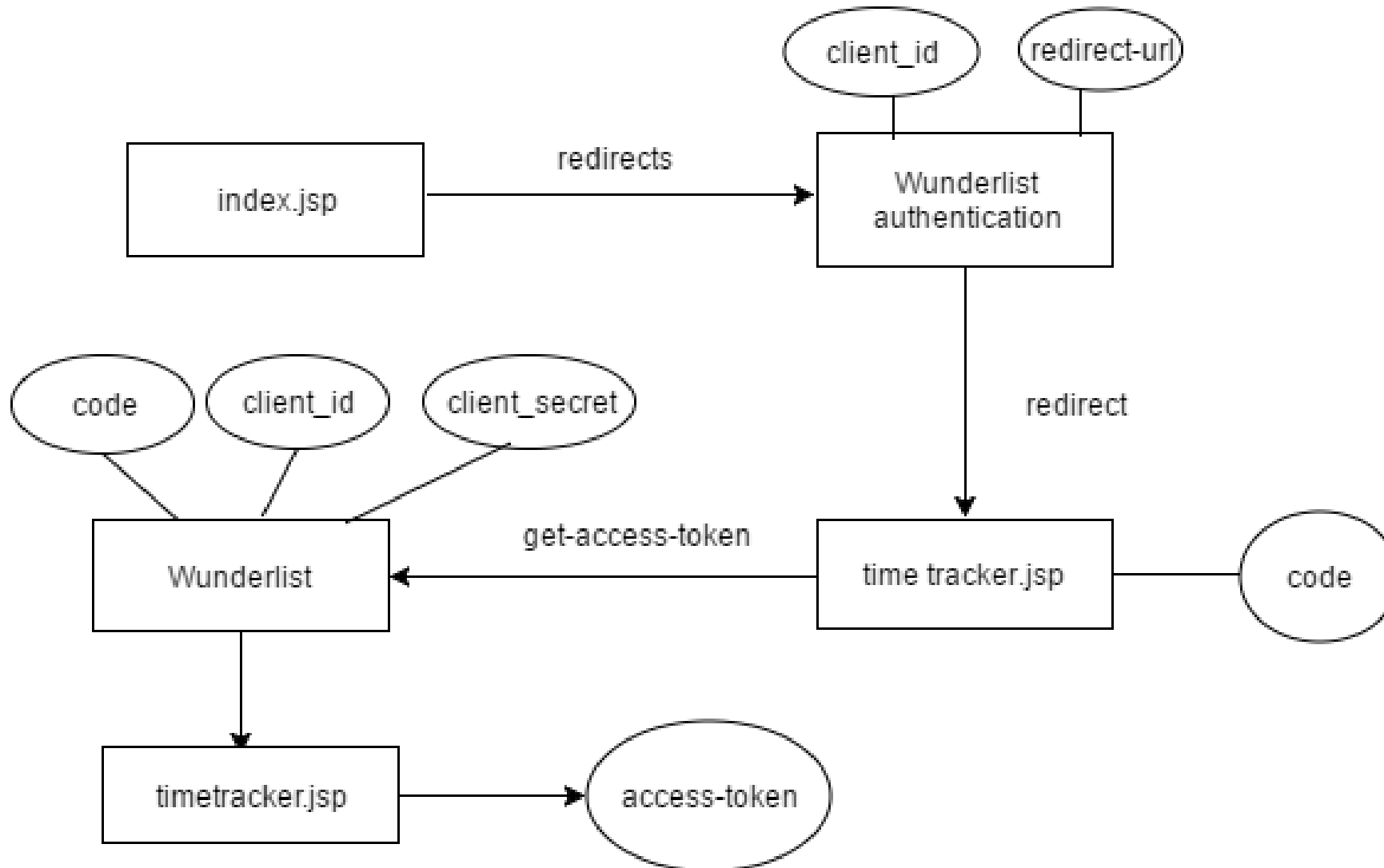
# Pause Task



# Finish Task



## Uniquely identifying user



# Scrum

- Scrum roles
  - Team members
  - Scrum Master
  - Product Owner
- Scrum board
  - <https://freedcamp.com>
- Scrum methodology
  - Biweekly sprints

Filter by choose ▾

Contains text 🔍

Save

	User created	
	User assigned to	
Ikram u.	Priority	
id-signup	<b>Progress</b>	No progress
-installati	Due Date	Completed
F	Date Created	In Progress
Completed on Feb 2, 2016		

Scrum board



# Black Box Testing

- Functionality testing without taking the actual implementation in concern
- Focuses on the outputs in response to selected inputs and execution conditions
- This program is considered as a "black box"

## Black Box/2

- Is conducted to evaluate the compliance of a system or application with specified functional requirements
- The design and structure of the code is not known to the tester



# Tools for Black Box

## Selenium:

- Selenium is an open source automated testing suite for web applications
- Selenium web driver is used to check the actual functional flow of time tracker application.



# Test cases

- Sets of test inputs, execution conditions and expected results developed for a particular functional requirement
- Identify test cases and identify the conditions that will cause it to execute

## Test cases/2

**Test Case ID:** TT\_02

**Test Priority (Low/Medium/High):**High

**Module Name :** Time Tracking Application.

**Test Title:** Create a task and monitor it.

**Pre Conditions:**User has a task to create .

Step	Test Steps	Test Data	Expected Result	Actual Result	Status (Pass/Fail)
1	Navigate to given URL	<a href="http://productioncarrot.mrcc.ovgu.de/timetracker/timetracker.jsp?state=RANDOM&amp;code=e6216e03560a3c39b352#">http://productioncarrot.mrcc.ovgu.de/timetracker/timetracker.jsp?state=RANDOM&amp;code=e6216e03560a3c39b352#</a>	User should be able to create a task and monitor it.	User is navigated to dashboard and create a new task and start it, pause it after some time and finish it.	Pass
2	Provide a valid task name	Movies			

# Maven

- Building management tool that allows a developer to comprehend the complete state of a development effort in the shortest period of time.

## Areas of concerns:

- Making the build process easy .
- Providing a uniform build system.
- Providing quality project information.
- Providing guidelines for best practices development.



# Maven in selenium

- Helps to manage our selenium project's build easily
- Create right project structure and manage .jar files in project build path

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>Test1</groupId>
  <artifactId>Timetrack</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <name>time</name>
  <dependencies>
    <dependency>
      <groupId>org.seleniumhq.selenium</groupId>
      <artifactId>selenium-firefox-driver</artifactId>
      <version>2.53.1</version>
    </dependency>
  </dependencies>
</project>
```

# Mockito

## Why?

- A unit test should test a class in isolation. Side effects from other classes or the system should be eliminated if possible.
- To eliminate these side effects you have to replace dependencies to other classes. This can be done using replacements for the real dependencies.

## How?

- A *dummy object* is passed around
- *Fake* objects have working implementations, but are usually simplified. For example, they use an in memory database and not a real database.
- A *stub* class is an partial implementation for an interface or class with the purpose of using an instance of this stub class during testing. Stubs may also record information about calls
- A *mock object* is a dummy implementation for an interface or a class in which you define the output of certain method calls

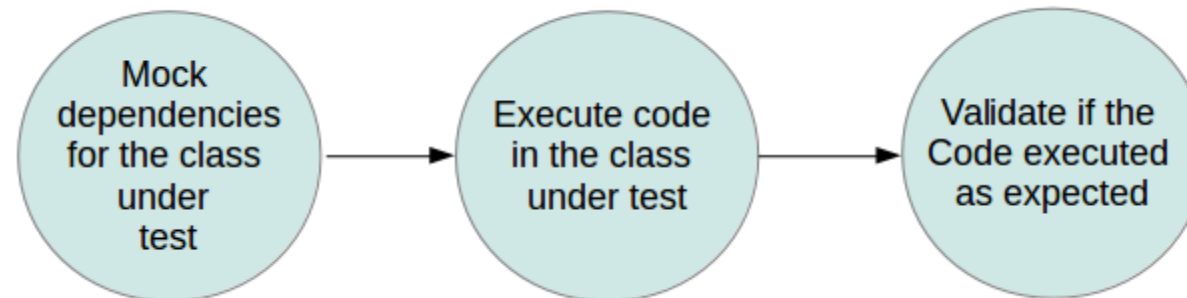
## What?

- *Mockito* is a popular mock framework which can be used in conjunction with JUnit. Mockito allows you to create and configure mock objects.
- Using Mockito simplifies the development of tests for classes with external dependencies significantly.

## Mockito(2)

If we use Mockito in tests you typically:

- Mock away external dependencies and insert the mocks into the code under test
- Execute the code under test
- Validate that the code executed correctly



## Mockito(3)

```
public class DB {

    String dbUrl = "jdbc:mysql://127.0.0.1:3306/timetracker";
    String dbClass = "com.mysql.jdbc.Driver";
    String username = "root";
    String password = "root";
    Connection connection;

    public String status = "";
    public long finishTask(String data) throws SQLException{

        String[] rows = data.split(";");
        String[] columns;
        String query = "", checkQuery = "";
        long totalTime = 0;

        for(int i = 0; i < rows.length; i++){

            columns = rows[i].split(",");
            query = "SELECT * FROM fact WHERE user=" + columns[0] + " AND list_id=" + columns[1] + " AND task_id=" + columns[2] + "";

            Statement statement = connection.createStatement();
            ResultSet resultSet = statement.executeQuery(query);

            while (resultSet.next()) {
                totalTime = resultSet.getInt("total_time");
                java.sql.Timestamp startTimeStamp = resultSet.getTimestamp("start_time");
                status = resultSet.getString("status");

                Date date = new Date();
                java.sql.Timestamp currentTimeStamp = new java.sql.Timestamp(date.getTime());

                if(status.equals("start")){
                    totalTime += (currentTimeStamp.getTime() - startTimeStamp.getTime());
                    java.sql.Timestamp updatedStamp = new java.sql.Timestamp(totalTime);
                }

                String updateQuery = "UPDATE fact SET status='finish', total_time=" + totalTime + ", start_time='" + currentTimeStamp + "';";

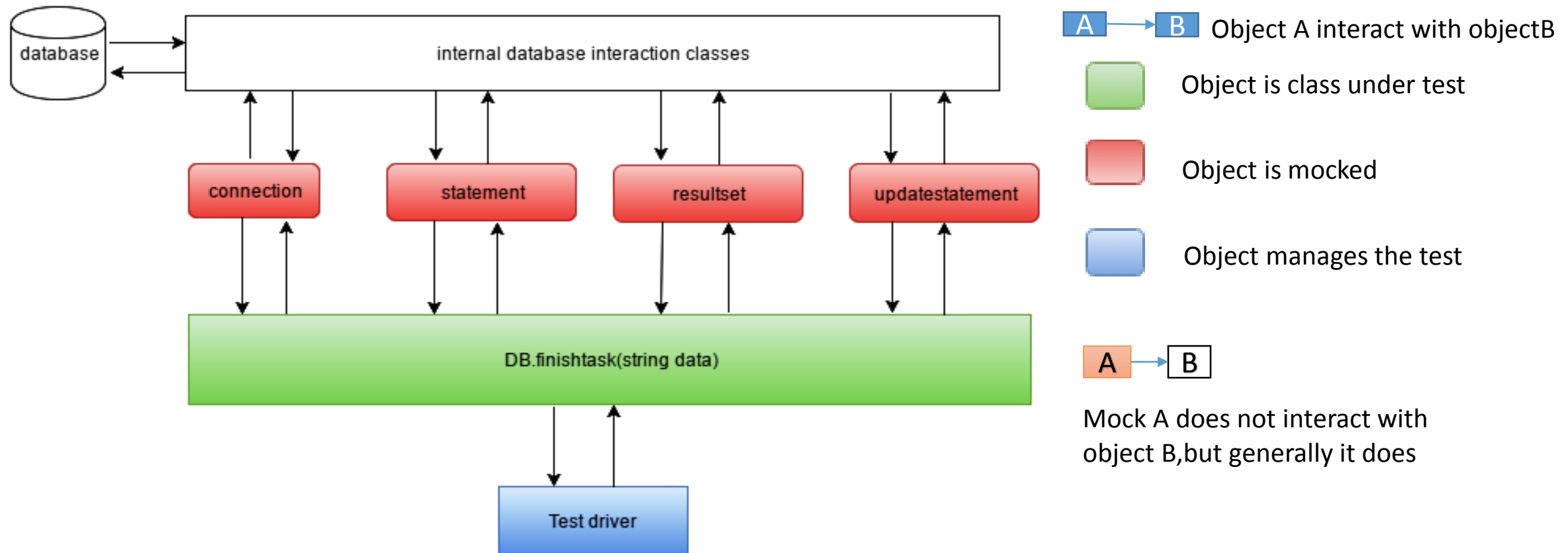
                Statement updateStatement = connection.createStatement();
                updateStatement.executeUpdate(updateQuery);
            }
        }
    }
}
```

## Mockito(4)

From (1)

- We desire to generate tests for finishtask method. As such we have to make the interactions with the external database deterministic
- As the database is interacted with through the Connection, Statement, Query, ResultSet, Updatestatement, Totaltimeinterfaces (Figure 1), to abstract it away we must mock these interfaces, and ensure that the finishtask dependencies are replaced with the mock objects.
- These mock objects are configured with a sequence of expected interactions and what it should respond for each of these interactions. They also record actual interactions locally, and can provide verification that specific interactions took place as expected after the method is run.

## Diagrammatic representation





## Generated test case

```
public class testDB
{

@Before
public void setUp()
{
DB db=new DB();//injects mock
}
@Test
public void testfinishtask()
{
@Mock
Connection connection;
Statement statement;
ResultSet resultset;
Totaltime totaltime;
Updatestatement updatestatement;
ResultSet resultset=statement.executeQuery(query);
//linking mocks
when(statement.executeQuery(query)).thenReturn(resultset);
when(resultset.next()).thenReturn(totaltime);
when(updatestatement.executeQuery(query)).thenReturn(totaltime);
long totaltime=db.finishtask(data);
assertTrue(totaltime=121314);
}
}
```

Generated test case

# Benefits of Mocking

- Create tests in advance
- Teams can work parallel
- Create demos
- Write test for resource which is not accessible
- Isolate systems

# References

- <http://www.slashroot.in/sites/default/files/Working%20of%20Puppet%20Configuration%20Mangement%20tool.png>
- [http://images.google.de/imgres?imgurl=http%3A%2F%2Fimage.slidesharecdn.com%2Fdevops-150627065118-lva1-app6892%2F95%2Fcontinuous-integration-with-jenkins-29-638.jpg%253Fcb%253D1435394546&imgrefurl=http%3A%2F%2Fwww.slideshare.net%2FEdurekaIN%2Fdevops-49899456&h=359&w=638&tbnid=1sG5xi301VmWqM%3A&docid=kC3vAWKJv7ajEM&ei=PqazV6ayLMyXgAaPlbeICg&tbn=isch&iact=rc&uact=3&dur=408&page=2&start=16&ndsp=24&ved=0ahUKEwimycrbj8fOAhXMC8AKHY\\_KDaEQMwhaKB0wHQ&bih=677&biw=136](http://images.google.de/imgres?imgurl=http%3A%2F%2Fimage.slidesharecdn.com%2Fdevops-150627065118-lva1-app6892%2F95%2Fcontinuous-integration-with-jenkins-29-638.jpg%253Fcb%253D1435394546&imgrefurl=http%3A%2F%2Fwww.slideshare.net%2FEdurekaIN%2Fdevops-49899456&h=359&w=638&tbnid=1sG5xi301VmWqM%3A&docid=kC3vAWKJv7ajEM&ei=PqazV6ayLMyXgAaPlbeICg&tbn=isch&iact=rc&uact=3&dur=408&page=2&start=16&ndsp=24&ved=0ahUKEwimycrbj8fOAhXMC8AKHY_KDaEQMwhaKB0wHQ&bih=677&biw=136)

# THANK YOU