Develop Your Own Performance Tool Integrated With Selenium

Saar Rachamim, Gett

About Me

Saar Rachamim

Gett (Israel)

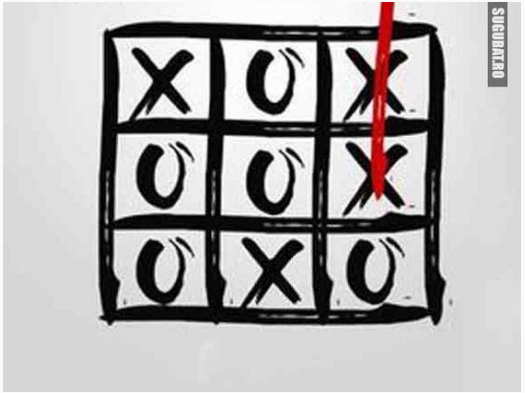
Automation Tech Lead

saarr@gett.com

About Me

Developing automation tools

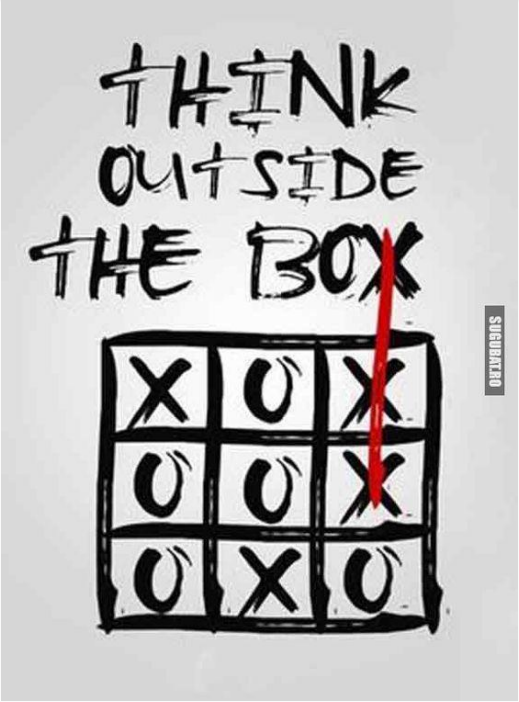
Innovation



About Me

Developing automation tools

Innovation



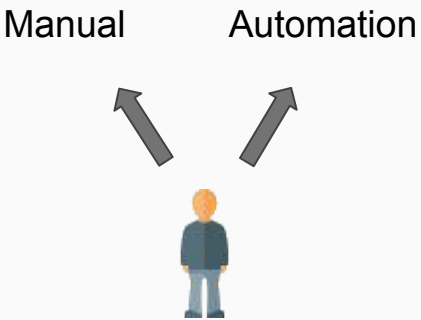
About Me

App Stability Problem

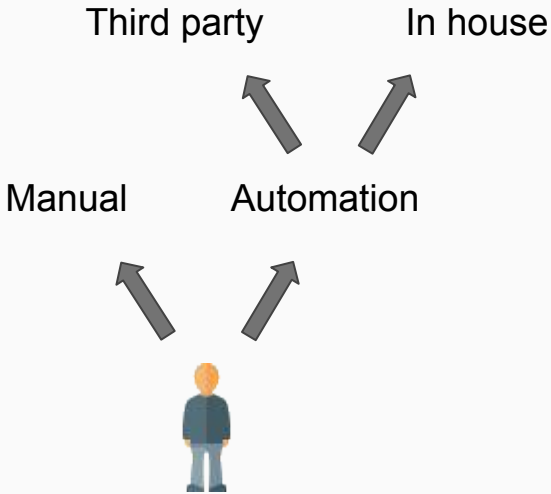
App Stability

Solutions

Solutions



Solutions

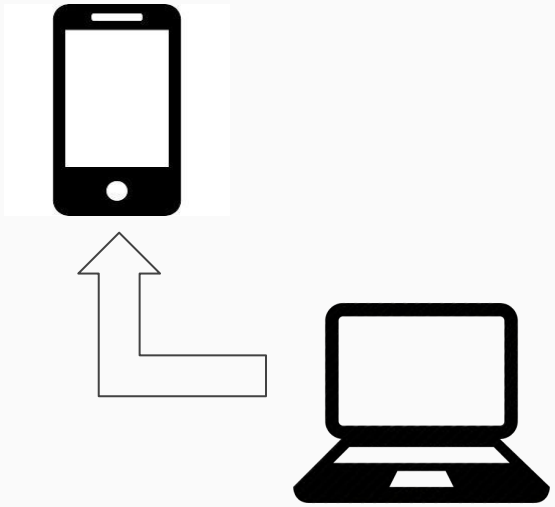


Selected Solution: Developing in House Tool

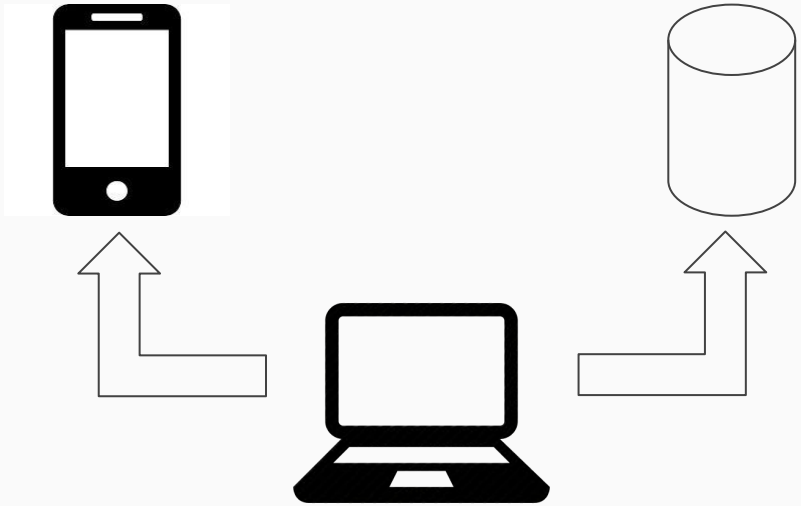
Developing your own tool

- Questions in mind
  - What language to choose.
  - What platform - mobile, desktop or web.
  - DB: Sql, NoSql.
  - AWS, Google services or any other.

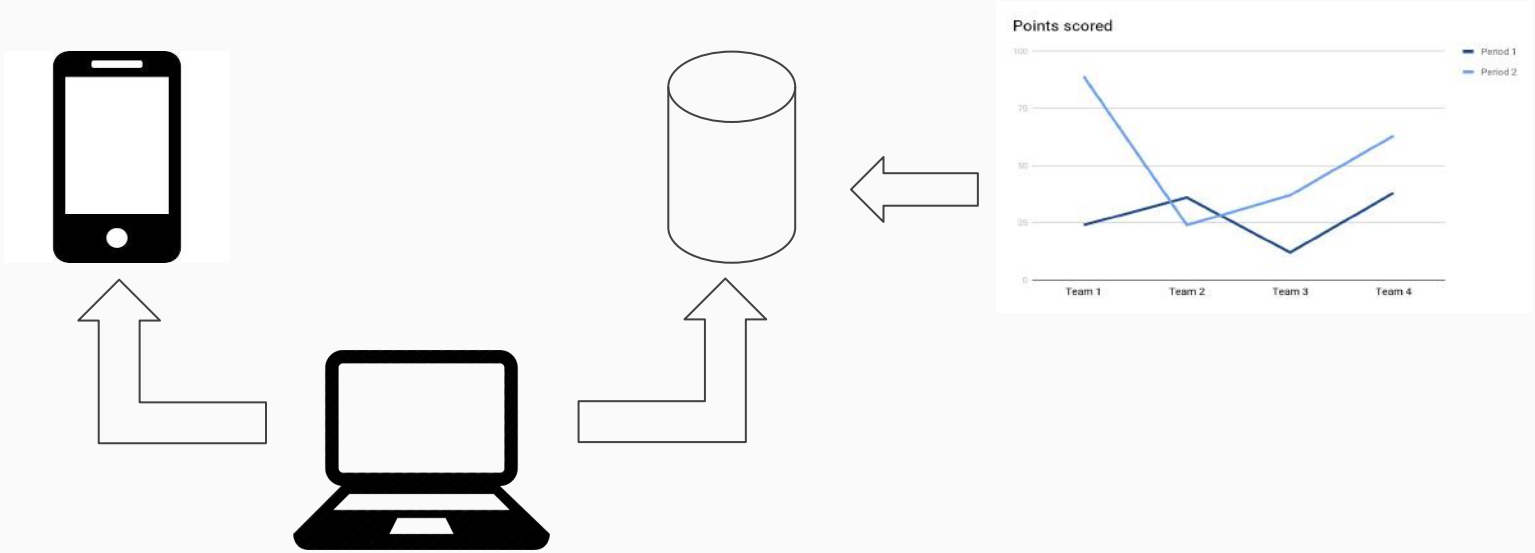
Developing your own tool



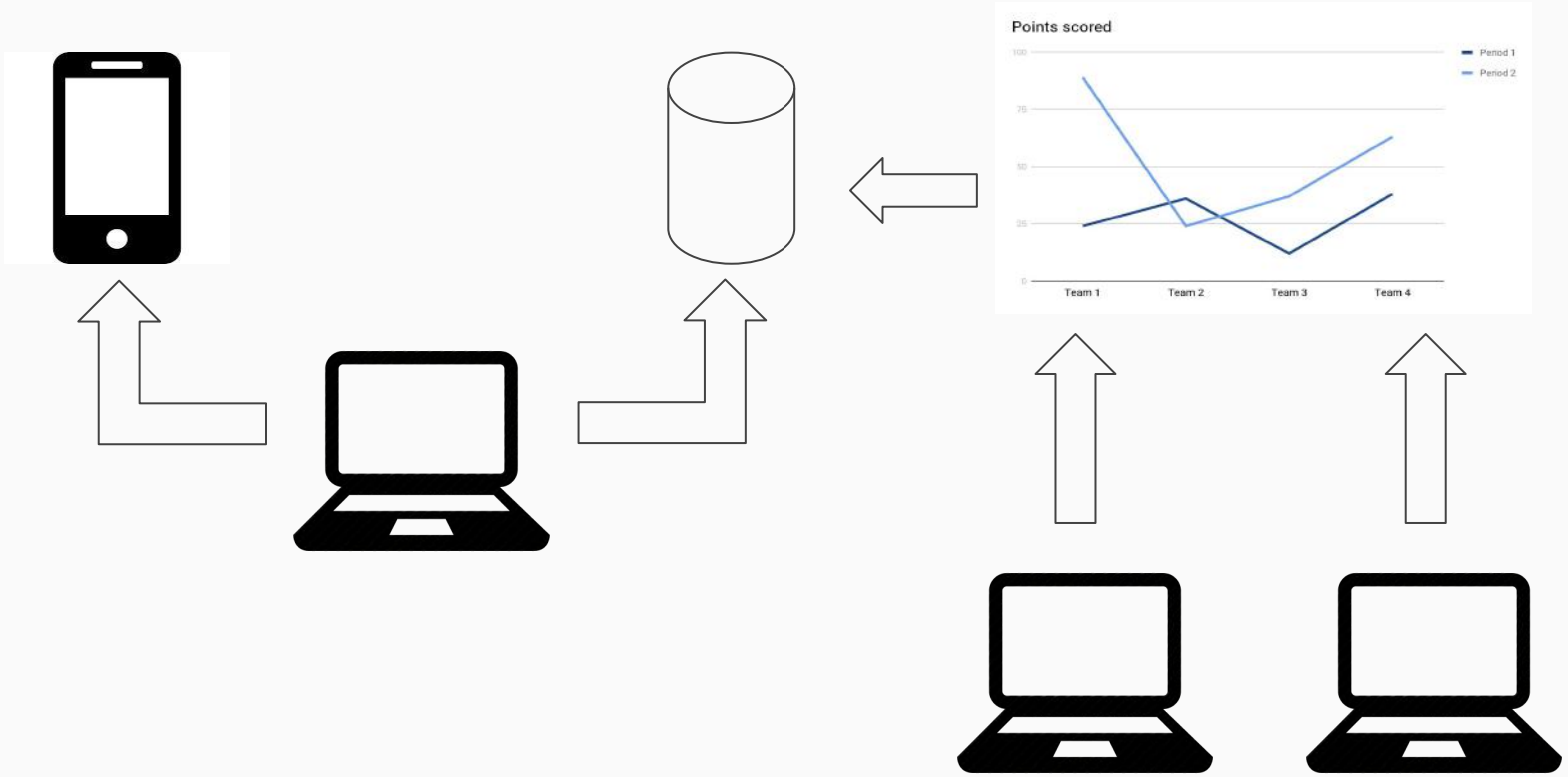
Developing your own tool



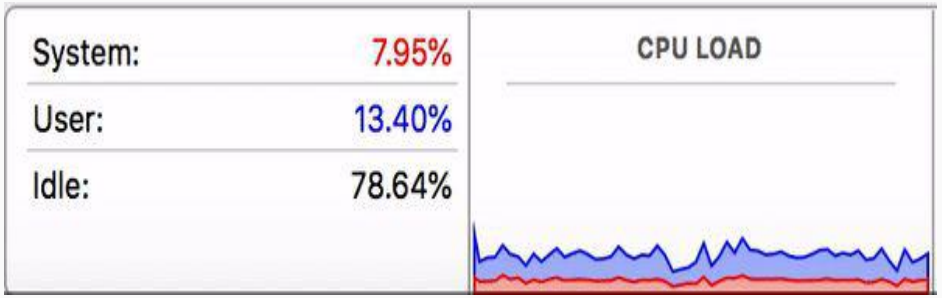
Developing your own tool

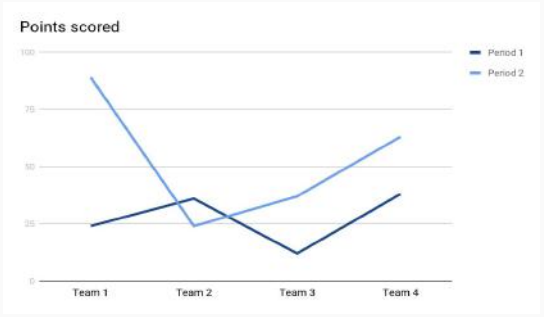


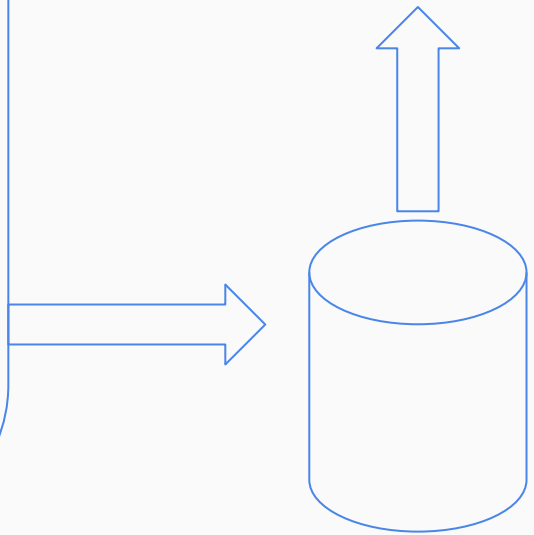
Developing your own tool



Developing your own tool







The Tool Code

Test Code

```
public class SeleniumTest {

    @Test
    public void test(){
        String testName = Thread.currentThread().getStackTrace()[1].getMethodName();
        new ThreadExecuter(testName).execute();
        new TestExecuter().execute();
    }
}
```

Test Code

```
public class SeleniumTest {

    @Test
    public void test(){
        String testName = Thread.currentThread().getStackTrace()[1].getMethodName();
        new ThreadExecuter(testName).execute();
        new TestExecuter().execute();
    }
}
```

Test Code

```
public class SeleniumTest {

    @Test
    public void test(){
        String testName = Thread.currentThread().getStackTrace()[1].getMethodName();
        new ThreadExecuter(testName).execute();
        new TestExecuter().execute();
    }
}
```

Test Code

```
public class SeleniumTest {

  @Test
  public void test(){
     String testName = Thread.currentThread().getStackTrace()[1].getMethodName();
     new ThreadExecuter(testName).execute();
     new TestExecuter().execute();
  }
}
```

Test Code

```
public class TestExecuter {

  public void execute(){
     WebDriver driver = new ChromeDriver();
     driver.manage().timeouts().implicitlyWait(30, TimeUnit.SECONDS);
     boolean stop = false;

     while(stop){
        try{
           driver.get(Constants.GOOGLE);
           WebElement searchField = driver.findElement(By.xpath(Constants.SEARCH_FIELD));
           searchField.sendKeys(new TextGenerator().getTextToSearch());
           WebElement searchButton = driver.findElement(By.xpath(Constants.SEARCH_BUTTON));
           searchButton.click();
           Thread.sleep(Constants.FIVE_SECONDS);
        }
        catch(Exception e){}
     }
  }
}
```

Test Code

```
public class TestExecuter {

  public void execute(){
     WebDriver driver = new ChromeDriver();
     driver.manage().timeouts().implicitlyWait(30, TimeUnit.SECONDS);
     boolean stop = false;

     while(stop){
        try{
           driver.get(Constants.GOOGLE);
           WebElement searchField = driver.findElement(By.xpath(Constants.SEARCH_FIELD));
           searchField.sendKeys(new TextGenerator().getTextToSearch());
           WebElement searchButton = driver.findElement(By.xpath(Constants.SEARCH_BUTTON));
           searchButton.click();
           Thread.sleep(Constants.FIVE_SECONDS);
        }
        catch(Exception e){}
     }
  }
}
```

Process Handler Code

```
public String getCpuUsageByProcessName(String name) throws IOException {
   String[] inputArr = {"ps", "-em", "-o", "%cpu,command"};
   String retVal = executeTerminalCommands(inputArr, name);
   return returnSubStringAccordingStr(retVal, " /").replace(" ", "");
}

private String executeTerminalCommands(String[] inputArr, String commandPart) throws IOException {
   String line;
   Process process = Runtime.getRuntime().exec(inputArr);
   BufferedReader input = new BufferedReader(new InputStreamReader(process.getInputStream()));
   while ((line = input.readLine()) != null) {
      if (line.contains(commandPart))
      {
         return line;
      }
   }
   return "-1";
}
```

Process Handler Code

```
public String getCpuUsageByProcessName(String name) throws IOException {
   String[] inputArr = {"ps", "-em", "-o", "%cpu,command"};
   String retVal = executeTerminalCommands(inputArr, name);
   return returnSubStringAccordingStr(retVal, " /").replace(" ", "");
}

private String executeTerminalCommands(String[] inputArr, String commandPart) throws IOException {
   String line;
   Process process = Runtime.getRuntime().exec(inputArr);
   BufferedReader input = new BufferedReader(new InputStreamReader(process.getInputStream()));
   while ((line = input.readLine()) != null) {
      if (line.contains(commandPart))
      {
         return line;
      }
   }
   return "-1";
}
```

Process Handler Code

```
public String getCpuUsageByProcessName(String name) throws IOException {
   String[] inputArr = {"ps", "-em", "-o", "%cpu,command"};
   String retVal = executeTerminalCommands(inputArr, name);
   return returnSubStringAccordingStr(retVal, " /").replace(" ", "");
}

private String executeTerminalCommands(String[] inputArr, String commandPart) throws IOException {
   String line;
   Process process = Runtime.getRuntime().exec(inputArr);
   BufferedReader input = new BufferedReader(new InputStreamReader(process.getInputStream()));
   while ((line = input.readLine()) != null) {
      if (line.contains(commandPart))
      {
         return line;
      }
   }
   return "-1";
}
```

Process Handler Code

```
public String getCpuUsageByProcessName(String name) throws IOException {
   String[] inputArr = {"ps", "-em", "-o", "%cpu,command"};
   String retVal = executeTerminalCommands(inputArr, name);
   return returnSubStringAccordingStr(retVal, " /").replace(" ", "");
}

private String executeTerminalCommands(String[] inputArr, String commandPart) throws IOException {
   String line;
   Process process = Runtime.getRuntime().exec(inputArr);
   BufferedReader input = new BufferedReader(new InputStreamReader(process.getInputStream()));
   while ((line = input.readLine()) != null) {
      if (line.contains(commandPart))
      {
         return line;
      }
   }
   return "-1";
}
```

Requests Handler Code

```
ProcessHandler ph = new ProcessHandler();
RequestHandler requestHandler = new RequestHandler(Constants.FB_URL);
String retKey = requestHandler.postHttp("", new String[]{"Content-Type", "Authorization"}, new
String[]{"application/x-www-form-urlencoded", "key=" + Constants.SECRET}, "{\"time\":\"" +
Calendar.getInstance().getTime() + "\", \"testName\":\"" + testName + "\", \"browser\":\"Chrome\"}");

while (true) {
  try {
    String cpu = ph.getCpuUsageByProcessName(Constants.PROCESS_NAME);
    String mem = ph.getMemoryUsageByProcessName(Constants.PROCESS_NAME);
    requestHandler.postHttp(retKey, new String[]{"Content-Type", "Authorization"}, new
String[]{"application/x-www-form-urlencoded", "key=" + Constants.SECRET}, "{\"mem\":\"" + mem + "\",\"cpu\":\"" +
cpu + "\", \"time\":\"" + Calendar.getInstance().getTime() + "\"}");
  } catch (Exception e) {
    System.out.println(e);
  }
}
```

Requests Handler Code

```
ProcessHandler ph = new ProcessHandler();
RequestHandler requestHandler = new RequestHandler(Constants.FB_URL);
String retKey = requestHandler.postHttp("", new String[]{"Content-Type", "Authorization"}, new
String[]{"application/x-www-form-urlencoded", "key=" + Constants.SECRET}, "{\"time\":\"" +
Calendar.getInstance().getTime() + "\", \"testName\":\"" + testName + "\", \"browser\":\"Chrome\"}");

while (true) {
  try {
    String cpu = ph.getCpuUsageByProcessName(Constants.PROCESS_NAME);
    String mem = ph.getMemoryUsageByProcessName(Constants.PROCESS_NAME);
    requestHandler.postHttp(retKey, new String[]{"Content-Type", "Authorization"}, new
String[]{"application/x-www-form-urlencoded", "key=" + Constants.SECRET}, "{\"mem\":\"" + mem + "\",\"cpu\":\"" +
cpu + "\", \"time\":\"" + Calendar.getInstance().getTime() + "\"}");
  } catch (Exception e) {
    System.out.println(e);
  }
}
```

Requests Handler Code

```
ProcessHandler ph = new ProcessHandler();
RequestHandler requestHandler = new RequestHandler(Constants.FB_URL);
String retKey = requestHandler.postHttp("", new String[]{"Content-Type", "Authorization"}, new
String[]{"application/x-www-form-urlencoded", "key=" + Constants.SECRET}, "{\"time\":\"" +
Calendar.getInstance().getTime() + "\", \"testName\":\"" + testName + "\", \"browser\":\"Chrome\"}");

while (true) {
  try {
    String cpu = ph.getCpuUsageByProcessName(Constants.PROCESS_NAME);
    String mem = ph.getMemoryUsageByProcessName(Constants.PROCESS_NAME);
    requestHandler.postHttp(retKey, new String[]{"Content-Type", "Authorization"}, new
String[]{"application/x-www-form-urlencoded", "key=" + Constants.SECRET}, "{\"mem\":\"" + mem + "\",\"cpu\":\"" +
cpu + "\", \"time\":\"" + Calendar.getInstance().getTime() + "\"}");
  } catch (Exception e) {
    System.out.println(e);
  }
}
```

Html Code

```
<div class="container">
                <div class="jumbotron">
                   <div class="panel-heading"></div>
                   <div ng-controller="MyController">
                    <div class="text-center">
                      <select ng-options="testName as testName for testName in testsNames" ng-model="selectedTest"></select>
                      <select ng-options="testTime as testTime for testTime in testsTime" ng-model="timeSelected"></select>
                      <select ng-options="measurement as measurement for measurement in measurements"
ng-model="measurementSelected"></select>
                      <button type="button" class="btn btn-primary my-2" ng-click="createGraph(selectedTest, timeSelected,
measurementSelected)">Execute</button>
                    </div>
                    <div class="text-left" style="margin-top:30px">
```

Html Code

```
<div class="container">
                <div class="jumbotron">
                  <div class="panel-heading"></div>
                  <div ng-controller="MyController">
                    <div class="text-center">
                      <select ng-options="testName as testName for testName in testsNames" ng-model="selectedTest"></select>
                      <select ng-options="testTime as testTime for testTime in testsTime" ng-model="timeSelected"></select>
                      <select ng-options="measurement as measurement for measurement in measurements"
ng-model="measurementSelected"></select>
                      <button type="button" class="btn btn-primary my-2" ng-click="createGraph(selectedTest, timeSelected,
measurementSelected)">Execute</button>
                    </div>
                    <div class="text-left" style="margin-top:30px">
```

Angular Code

```
$scope.createGraph = function (selectedTest, timeSelected, measurementSelected) {
...
        angular.forEach(array, function (test) {
            if (test.time == timeSelected && test.testName == selectedTest) {
                $scope.currentData = new Array();
                for (var key in test) {
                    var specimentRecord = test[key];

                    if (specimentRecord != null && typeof specimentRecord !== "undefined") {
                        $scope.currentData.push(specimentRecord);
                        $scope.currentlabels.push(specimentRecord.time);
                    }
                }
                $scope.currentData = $filter('orderBy')($scope.currentData, "time");
                $scope.data = new Array();
                $scope.labels = new Array();
                for (var key in $scope.currentData) {
                    if (measurementSelected == "cpu" && typeof $scope.currentData[key].cpu !== "undefined")
                        $scope.data.push($scope.currentData[key].cpu);
                    if (measurementSelected == "memory" && typeof $scope.currentData[key].mem !== "undefined")
                        $scope.data.push($scope.currentData[key].mem);
                    if (typeof $scope.currentData[key].cpu !== "undefined" || typeof $scope.currentData[key].mem !== "undefined")
                        $scope.labels.push($scope.currentData[key].time);
                }
```

Angular Code

```
$scope.createGraph = function (selectedTest, timeSelected, measurementSelected) {
...
        angular.forEach(array, function (test) {
            if (test.time == timeSelected && test.testName == selectedTest) {
                $scope.currentData = new Array();
                for (var key in test) {
                    var specimentRecord = test[key];

                    if (specimentRecord != null && typeof specimentRecord !== "undefined") {
                        $scope.currentData.push(specimentRecord);
                        $scope.currentlabels.push(specimentRecord.time);
                    }
                }
                $scope.currentData = $filter('orderBy')($scope.currentData, "time");
                $scope.data = new Array();
                $scope.labels = new Array();
                for (var key in $scope.currentData) {
                    if (measurementSelected == "cpu" && typeof $scope.currentData[key].cpu !== "undefined")
                        $scope.data.push($scope.currentData[key].cpu);
                    if (measurementSelected == "memory" && typeof $scope.currentData[key].mem !== "undefined")
                        $scope.data.push($scope.currentData[key].mem);
                    if (typeof $scope.currentData[key].cpu !== "undefined" || typeof $scope.currentData[key].mem !== "undefined")
                        $scope.labels.push($scope.currentData[key].time);
                }
```

Angular Code

```
$scope.createGraph = function (selectedTest, timeSelected, measurementSelected) {
...
        angular.forEach(array, function (test) {
            if (test.time == timeSelected && test.testName == selectedTest) {
                $scope.currentData = new Array();
                for (var key in test) {
                    var specimentRecord = test[key];

                    if (specimentRecord != null && typeof specimentRecord !== "undefined") {
                        $scope.currentData.push(specimentRecord);
                        $scope.currentlabels.push(specimentRecord.time);
                    }
                }
                $scope.currentData = $filter('orderBy')($scope.currentData, "time");
                $scope.data = new Array();
                $scope.labels = new Array();
                for (var key in $scope.currentData) {
                    if (measurementSelected == "cpu" && typeof $scope.currentData[key].cpu !== "undefined")
                        $scope.data.push($scope.currentData[key].cpu);
                    if (measurementSelected == "memory" && typeof $scope.currentData[key].mem !== "undefined")
                        $scope.data.push($scope.currentData[key].mem);
                    if (typeof $scope.currentData[key].cpu !== "undefined" || typeof $scope.currentData[key].mem !== "undefined")
                        $scope.labels.push($scope.currentData[key].time);
                }
```

Angular Code

```
$scope.createGraph = function (selectedTest, timeSelected, measurementSelected) {
...
        angular.forEach(array, function (test) {
            if (test.time == timeSelected && test.testName == selectedTest) {
                $scope.currentData = new Array();
                for (var key in test) {
                    var specimentRecord = test[key];

                    if (specimentRecord != null && typeof specimentRecord !== "undefined") {
                        $scope.currentData.push(specimentRecord);
                        $scope.currentlabels.push(specimentRecord.time);
                    }
                }
                $scope.currentData = $filter('orderBy')($scope.currentData, "time");
                $scope.data = new Array();
                $scope.labels = new Array();
                for (var key in $scope.currentData) {
                    if (measurementSelected == "cpu" && typeof $scope.currentData[key].cpu !== "undefined")
                        $scope.data.push($scope.currentData[key].cpu);
                    if (measurementSelected == "memory" && typeof $scope.currentData[key].mem !== "undefined")
                        $scope.data.push($scope.currentData[key].mem);
                    if (typeof $scope.currentData[key].cpu !== "undefined" || typeof $scope.currentData[key].mem !== "undefined")
                        $scope.labels.push($scope.currentData[key].time);
                }
```

Angular Code

```
$scope.createGraph = function (selectedTest, timeSelected, measurementSelected) {
...
        angular.forEach(array, function (test) {
            if (test.time == timeSelected && test.testName == selectedTest) {
                $scope.currentData = new Array();
                for (var key in test) {
                    var specimentRecord = test[key];

                    if (specimentRecord != null && typeof specimentRecord !== "undefined") {
                        $scope.currentData.push(specimentRecord);
                        $scope.currentlabels.push(specimentRecord.time);
                    }
                }
                $scope.currentData = $filter('orderBy')($scope.currentData, "time");
                $scope.data = new Array();
                $scope.labels = new Array();
                for (var key in $scope.currentData) {
                    if (measurementSelected == "cpu" && typeof $scope.currentData[key].cpu !== "undefined")
                        $scope.data.push($scope.currentData[key].cpu);
                    if (measurementSelected == "memory" && typeof $scope.currentData[key].mem !== "undefined")
                        $scope.data.push($scope.currentData[key].mem);
                    if (typeof $scope.currentData[key].cpu !== "undefined" || typeof $scope.currentData[key].mem !== "undefined")
                        $scope.labels.push($scope.currentData[key].time);
                }
```

Angular Code

```
$scope.createGraph = function (selectedTest, timeSelected, measurementSelected) {
...
        angular.forEach(array, function (test) {
            if (test.time == timeSelected && test.testName == selectedTest) {
                $scope.currentData = new Array();
                for (var key in test) {
                    var specimentRecord = test[key];

                    if (specimentRecord != null && typeof specimentRecord !== "undefined") {
                        $scope.currentData.push(specimentRecord);
                        $scope.currentlabels.push(specimentRecord.time);
                    }
                }
                $scope.currentData = $filter('orderBy')($scope.currentData, "time");
                $scope.data = new Array();
                $scope.labels = new Array();
                for (var key in $scope.currentData) {
                    if (measurementSelected == "cpu" && typeof $scope.currentData[key].cpu !== "undefined")
                        $scope.data.push($scope.currentData[key].cpu);
                    if (measurementSelected == "memory" && typeof $scope.currentData[key].mem !== "undefined")
                        $scope.data.push($scope.currentData[key].mem);
                    if (typeof $scope.currentData[key].cpu !== "undefined" || typeof $scope.currentData[key].mem !== "undefined")
                        $scope.labels.push($scope.currentData[key].time);
                }
```

Angular Code

```
$scope.createGraph = function (selectedTest, timeSelected, measurementSelected) {
...
        angular.forEach(array, function (test) {
            if (test.time == timeSelected && test.testName == selectedTest) {
                $scope.currentData = new Array();
                for (var key in test) {
                    var specimentRecord = test[key];

                    if (specimentRecord != null && typeof specimentRecord !== "undefined") {
                        $scope.currentData.push(specimentRecord);
                        $scope.currentlabels.push(specimentRecord.time);
                    }
                }
                $scope.currentData = $filter('orderBy')($scope.currentData, "time");
                $scope.data = new Array();
                $scope.labels = new Array();
                for (var key in $scope.currentData) {
                    if (measurementSelected == "cpu" && typeof $scope.currentData[key].cpu !== "undefined")
                        $scope.data.push($scope.currentData[key].cpu);
                    if (measurementSelected == "memory" && typeof $scope.currentData[key].mem !== "undefined")
                        $scope.data.push($scope.currentData[key].mem);
                    if (typeof $scope.currentData[key].cpu !== "undefined" || typeof $scope.currentData[key].mem !== "undefined")
                        $scope.labels.push($scope.currentData[key].time);
                }
```

Angular Code

```
$scope.createGraph = function (selectedTest, timeSelected, measurementSelected) {
...
        angular.forEach(array, function (test) {
            if (test.time == timeSelected && test.testName == selectedTest) {
                $scope.currentData = new Array();
                for (var key in test) {
                    var specimentRecord = test[key];

                    if (specimentRecord != null && typeof specimentRecord !== "undefined") {
                        $scope.currentData.push(specimentRecord);
                        $scope.currentlabels.push(specimentRecord.time);
                    }
                }
                $scope.currentData = $filter('orderBy')($scope.currentData, "time");
                $scope.data = new Array();
                $scope.labels = new Array();
                for (var key in $scope.currentData) {
                    if (measurementSelected == "cpu" && typeof $scope.currentData[key].cpu !== "undefined")
                        $scope.data.push($scope.currentData[key].cpu);
                    if (measurementSelected == "memory" && typeof $scope.currentData[key].mem !== "undefined")
                        $scope.data.push($scope.currentData[key].mem);
                    if (typeof $scope.currentData[key].cpu !== "undefined" || typeof $scope.currentData[key].mem !== "undefined")
                        $scope.labels.push($scope.currentData[key].time);
                }
```

Demo

Summary

Questions?