Phase 3 Final Project Assessment **Name: Prajwal Diwakar**

Non-Functional Testing Using Postman, REST Assured, and JMeter.

Source Code:

Postman

Assignment 1:

Post Method: Create petID and PetName

```
pm.test("Status code is 200", function () {
   pm.response.to.have.status(200);
});

pm.test("Response body contains 'available'", function () {
   pm.expect(pm.response.json().status).to.eql('available');
});
```

Get Method: Validate petID:

```
pm.test("Status code is 200", function () {
   pm.response.to.have.status(200);
});
```

Delete Method:Delete petID:

```
pm.test("Status code is 200", function () {
   pm.response.to.have.status(200);
});
```

Assignment 2

Put Method: Validate status value

```
pm.test("Response to have 'id' property", function () {
   var jsonData = pm.response.json();
   pm.expect(pm.response.json()).to.have.property('id');
   pm.expect(jsonData.category.id).to.eql(20021);
});
```

```
pm.test("Status code to have 200", function(){
   pm.response.to.have.status(200);
});

pm.test("Validate status value in response", function () {
   pm.expect(pm.response.json()).to.have.property('status');
});
```

Assignment 3

find user by username

```
pm.test("Status code is 200", function () {
   pm.response.to.have.status(200);
});

pm.test("Validate username in response", function () {
   pm.expect(pm.response.json().username).to.eql(pm.environment.get("username"));
});

pm.test("Validate email in response", function () {
   pm.expect(pm.response.json().email).to.eql("Positive@Attitude.com");
});

pm.test("Validate userStatus in response", function () {
   pm.expect(pm.response.json().userStatus).to.eql(1);
});
```

Assignment 4

Find pet by status=available

```
// Validate id = 20021 in response

pm.test("Check id in response", function() {

pm.expect(pm.response.json().category.id).to.eql(20021);

});

// Validate response = 200

pm.test("Check response code", function() {
```

```
pm.response.to.have.status(200);

});

// Validate status value in JSON response based on environment

pm.test("Check status value in response", function() {

pm.expect(pm.response.json().status).to.eql(pm.environment.get("status"));

});
```

Find pet by status=pending

```
// Check if the response status code is 200

pm.test("Status code is 200", function () {

pm.response.to.have.status(200);

});

// Check if all pet details have status = pending

var jsonData = pm.response.json();

jsonData.forEach(function (pet) {

pm.expect(pet.status).to.equal("pending");

});
```

Find pet by status=sold

```
// Check if the response status code is 200

pm.test("Status code is 200", function () {

pm.response.to.have.status(200);

});
```

```
// Check if all pet details have status = sold

var jsonData = pm.response.json();

jsonData.forEach(function (pet) {

pm.expect(pet.status).to.equal("sold");

});
```

user Logout

```
// Check if the response status code is 200

pm.test("Status code is 200", function () {

pm.response.to.have.status(200);

});

// Parse the response JSON

var jsonData = pm.response.json();

// Check if the 'code' field in the response is equal to 200

pm.test("Code is 200", function () {

pm.expect(jsonData.code).to.equal(200);

});

// Parse the response JSON

var jsonData = pm.response.json();

// Check if the 'message' field in the response is equal to "OK"

pm.test("Message is OK", function () {

pm.expect(jsonData.message).to.equal("ok");

});
```

RestAssured:

Phase3.Assignment01_RestAssured.PostRequest

```
package Phase3.Assignment01_RestAssured;

import static org.hamcrest.CoreMatchers.equalTo;
import io.restassured.RestAssured;
import io.restassured.http.ContentType;
import org.testng.annotations.Test;
import org.apache.log4j.Logger;


public class PostRequest {

        private static final String Base_Url ="https://petstore.swagger.io/v2";
        static final Logger logger=Logger.getLogger(PostRequest.class);

          @Test(description = "Post request method with the rest assured")

          public void testPostPet() {
             // Base URL for the API

                logger.info("START:: POST method to create the PET details");
             // JSON Body
        String requestBody = "{ \"id\": 344, \"category\": { \"id\": 0, \"name\": \"string\" },
\"name\": \"Doggie\", " +
                "\"photoUrls\": [ \"string\" ], \"tags\": [ { \"id\": 0, \"name\": \"string\" } ], " +
                "\"status\": \"available\" }";


            try {
                // Send POST request
                     RestAssured.given().baseUri(Base_Url)
                     .contentType(ContentType.JSON)
                     .body(requestBody)
                     .when()
                     .post("/pet")
                     .then().statusCode(200)
                     .and()
                     .assertThat()
                     .body("id",equalTo(344))
                     .and().body("status", equalTo("available"));
```

```
                    logger.info("Request Body is " +requestBody);
                }

                catch (Exception e) {
                                        logger.error("Exception Object :: " + e.toString());
                                        logger.error("End Exception :: "+e.getLocalizedMessage());

                  }

                String response=  RestAssured.given().baseUri(Base_Url)
                                   .contentType(ContentType.JSON)
                                   .body(requestBody)
                                   .when()
                                   .post("/pet").getBody().asPrettyString();

                logger.info("Response is" + response);
                logger.info("END:: POST method to create the PET details");




               }
             }
```

Phase3.Assignment01_RestAssured. GetRequest

```
package Phase3.Assignment01_RestAssured;

import io.restassured.RestAssured;
import io.restassured.http.ContentType;

import org.testng.annotations.Test;

import org.apache.log4j.Logger;


public class GetRequest {

        private static final String Base_Url ="https://petstore.swagger.io/v2";
        static final Logger logger= Logger.getLogger(GetRequest.class);

        @Test(description="Get request method to get PET details using the petID")
```

```
	public void testGetPet() {
		int petID= 344;
		logger.info("START::GET method for the PET test");
		logger.info("POST: URL" +Base_Url+ "/pet/" +petID);

		try {
			RestAssured.given().baseUri(Base_Url)
			.contentType(ContentType.JSON)
			.when().get("/pet/" +petID)
			.then().assertThat().statusCode(200);
		}
		catch(Exception e) {
				logger.error("Exception Object :: " + e.toString());
				logger.error("End Exception :: "+e.getLocalizedMessage());
		}

		String response= RestAssured.given().baseUri(Base_Url)
				.when().get("/pet/" +petID).getBody().asPrettyString();
		logger.info("Response is " +response);
		logger.info("END:: GET method for the PET test");
	}

}
```

Phase3.Assignment01_RestAssured.DeleteRequest

```
package Phase3.Assignment01_RestAssured;

import io.restassured.RestAssured;
import org.testng.annotations.Test;
import org.apache.log4j.Logger;


public class DeleteRequest {

	private static final String Base_Url ="https://petstore.swagger.io/v2";
	static final Logger logger= Logger.getLogger(DeleteRequest.class);

	@Test(description="Delete request method for the PET")

	public void testGetPet() {

		logger.info("START::Delete method for the PET test");
		int petID= 344;
		logger.info("DELETE: URL" +Base_Url+ "/pet/" +petID);
```

```
                try {
                        RestAssured.given().baseUri(Base_Url)
                        .when().delete("/pet/" +petID)
                        .then().statusCode(200);
                }
                catch(Exception e) {
                        logger.error("Exception Object :: " + e.toString());
                        logger.error("End Exception :: "+e.getLocalizedMessage());

                }

//              String response = RestAssured.given().baseUri(Base_Url)
//                              .when().delete("/pet/" +petID)
//                              .getBody().asPrettyString();

                String response= RestAssured.given().baseUri(Base_Url)
                                .when().delete("/pet/" +petID).getBody().asPrettyString();

        }


}
```

Phase3.Assignment02_RestAssured.putRequestMethod1

```
package Phase3.Assignment02_RestAssured;

import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;
import static org.hamcrest.CoreMatchers.equalTo;
import org.apache.log4j.Logger;
import org.testng.annotations.DataProvider;
import org.testng.annotations.Test;
import io.restassured.RestAssured;
import io.restassured.http.ContentType;


public class putRequestMethod1 {

        private static final String BASE_URL = "https://petstore.swagger.io/v2";
        static final Logger logger= Logger.getLogger(putRequestMethod1.class);
```

```
@DataProvider(name = "statusValues")
        public Iterator<Object[]> statusValues() {
                final List<Object[]> statusValues = new ArrayList<Object[]> ();
                statusValues.add(new Object[] {"available_DEV"});
                statusValues.add(new Object[] {"available_QA"});
                statusValues.add(new Object[] {"available_PROD"});

                return statusValues.iterator();
        }

 @Test(description="put request method ",dataProvider = "statusValues")

  public void putCallTesting(final String statusValue) {

        logger.info("START::PUT method for the PET test");
                logger.info("POST: URL" +BASE_URL+ "/pet/");

     // Prepare JSON request body with dynamic status field
     String requestBody = "{ " +
          "\"id\": 9223372016900013000, " +
          "\"category\": {\"id\": 20021, \"name\": \"string\"}, " +
          "\"name\": \"doggie\", " +
          "\"photoUrls\": [\"string\"], " +
          "\"tags\": [{\"id\": 0, \"name\": \"string\"}], " +
          "\"status\": \"" + statusValue + "\" " +
          "}";
     try {
         logger.info("Request body is" +requestBody);
         RestAssured.given().baseUri(BASE_URL)
              .contentType(ContentType.JSON)
              .body(requestBody)
              .when().put("/pet")      // Send PUT request
              .then().statusCode(200)    // Validate response code
              .and().assertThat()
        .body("category.id", equalTo(20021))                  // Validate id in response
              .and().body("status", equalTo(statusValue));   // Validate status value in
response

     }
     catch(Exception e) {

         logger.error("Exception Object :: " + e.toString());
                         logger.error("End Exception :: "+e.getLocalizedMessage());
     }
```

```
            String response= RestAssured.given().baseUri(BASE_URL)
                          .contentType(ContentType.JSON)
                          .body(requestBody)
                          .when().put("/pet").getBody().asPrettyString();
                    logger.info("Response is " +response);
                    logger.info("END:: PUT method for the PET test");


          }


}
```

Phase3.Assignment02_RestAssured.putRequestMethod2

```
package Phase3.Assignment02_RestAssured;

import static org.hamcrest.CoreMatchers.equalTo;
import java.util.HashMap;
import org.apache.log4j.Logger;
import org.testng.annotations.Parameters;
import org.testng.annotations.Test;
import io.restassured.RestAssured;
import io.restassured.http.ContentType;


public class putRequestMethod2 {

        private static final String BASE_URL = "https://petstore.swagger.io/v2";
        static final Logger logger= Logger.getLogger(putRequestMethod2.class);

        private static final HashMap<String, String> ENVIRONMENT_VALUES = new
HashMap<>();

          static {
            ENVIRONMENT_VALUES.put("DEV", "available_DEV");
            ENVIRONMENT_VALUES.put("QA", "available_QA");
            ENVIRONMENT_VALUES.put("PROD", "available_PROD");
          }

          @Parameters("environment")

          @Test           // (description="put request method ",dataProvider = "statusValues")
```

```
public void putCallTesting(String environment) {

        logger.info("START::PUT method for the PET test");
                logger.info("POST: URL" +BASE_URL+ "/pet/");

                 String statusValue = ENVIRONMENT_VALUES.get(environment);

    // Prepare JSON request body with dynamic status field
    String requestBody = "{ " +
            "\"id\": 9223372016900013000, " +
            "\"category\": {\"id\": 20021, \"name\": \"string\"}, " +
            "\"name\": \"doggie\", " +
            "\"photoUrls\": [\"string\"], " +
            "\"tags\": [{\"id\": 0, \"name\": \"string\"}], " +
            "\"status\": \"" + statusValue + "\" " +
            "}";
    try {
        logger.info("Request body is" +requestBody);
        RestAssured.given().baseUri(BASE_URL)
                .contentType(ContentType.JSON)
                .body(requestBody)
                .when().put("/pet")       // Send PUT request
                .then().statusCode(200)    // Validate response code
                .and().assertThat()
       .body("category.id", equalTo(20021))                  // Validate id in response
                .and().body("status", equalTo(statusValue));   // Validate status value in
response

    }
    catch(Exception e) {

        logger.error("Exception Object :: " + e.toString());
                             logger.error("End Exception :: "+e.getLocalizedMessage());
    }
    String response= RestAssured.given().baseUri(BASE_URL)
                            .contentType(ContentType.JSON)
                            .body(requestBody)
                            .when().put("/pet").getBody().asPrettyString();
                logger.info("Response is " +response);
                logger.info("END:: PUT method for the PET test");



}
```

```
}
```

Phase3.Assignment03_RestAssured.GetRequestUserName

```
package Phase3.Assignment03_RestAssured;

import org.testng.annotations.Test;
import io.restassured.RestAssured;
import org.apache.log4j.Logger;
import static org.hamcrest.CoreMatchers.equalTo;


public class GetRequestUserName {
        private static final String base_url ="https://petstore.swagger.io/v2/user/Uname001";
         static final Logger logger= Logger.getLogger(GetRequestUserName.class);


        @Test(description=" Get request method with rest assured")


        public void TestGetMethod() {

                logger.info("START::GET method for the PET test");
                logger.info("POST: URL" +base_url);
                try {

                        RestAssured.given()
                        .when().get(base_url)
                        .then().statusCode(200);
                        .and().assertThat()
                        .body("username", equalTo("Uname001"))
                        .and().body("email", equalTo("Positive@Attitude.com"))
                        .and().body("userStatus", equalTo(1));
                }
                catch(Exception e) {

        logger.error("Exception Object :: " + e.toString());
                        logger.error("End Exception :: "+e.getLocalizedMessage());
    }



                        String response= RestAssured.given().baseUri(base_url)
```

```
                    .when().get("/user/Uname001")
                .getBody().asPrettyString();

            logger.info("Response is " +response);
            logger.info("END:: GET method for the PET test");

    }

}
```

Phase3.Assignment04_RestAssured.GetRequestAuthentication

```
package Phase3.Assignment04_RestAssured;

import static org.hamcrest.CoreMatchers.notNullValue;
import org.apache.log4j.Logger;
import org.testng.annotations.Test;
import Phase3.Assignment03_RestAssured.GetRequestUserName;
import io.restassured.RestAssured;
import io.restassured.http.ContentType;


public class GetRequestAuthentication {


        private static final String BASE_URL = "https://petstore.swagger.io/v2";
         static final Logger logger= Logger.getLogger(GetRequestUserName.class);

        @Test(description = "Test Authentication with rest assured")

        public void testAuthenticationToken() {

            logger.info("START::GET method for the PET Authenticaton Login");
            logger.info("POST: URL" +BASE_URL);
            // create user post data
            User user = new User("Uname001" ," @tt!tude");

            logger.info("user object is" +user );
            try {
                RestAssured.given().baseUri(BASE_URL).when()
                .contentType(ContentType.JSON)
                .body(user)
                .log().uri()  // request logs
                .get("/user/login").then()
```

```
                        .log().body()  // response logs
                        .assertThat().statusCode(200).and()
                        .assertThat().body ("code", notNullValue()).and()
                        .assertThat().body ("type", notNullValue()).and()
                        .assertThat().body ("message", notNullValue());
                }
                catch(Exception e) {

            logger.error("Exception Object :: " + e.toString());
                        logger.error("End Exception :: "+e.getLocalizedMessage());
        }



                String response = RestAssured.given().baseUri(BASE_URL).when()
                .contentType(ContentType.JSON)
                .body(user)
                .get("/user/login").getBody().asString();

                logger.info("Response is " +response);
                logger.info("END::GET method for the PET Authenticaton Login");


            }
    }

    class User {

            public String username;
            public String password;

            public User(String username, String password) {
                super();
                this.username = username;
                this.password = password;
            }
    }
```

Phase3.Assignment05.RestAssured.TestPetStoreFindByStatus

```
package Phase3.Assignment05.RestAssured;

import io.restassured.RestAssured;
import io.restassured.http.ContentType;
import org.testng.annotations.Test;
import static org.hamcrest.Matchers.*;
import org.apache.log4j.Logger;


            public class TestPetStoreFindByStatus {

                private static final String base_url =
"https://petstore.swagger.io/v2/pet/findByStatus";
                static final Logger logger= Logger.getLogger(TestPetStoreFindByStatus.class);

                @Test

                public void findPetsByAvailableStatus() {

                        logger.info("START::GET method for the find PET by status=available");
                                    logger.info("GET: URL" +base_url);
                     // Make GET call with status=available
                   RestAssured.given()
                          .contentType(ContentType.JSON)
                          .queryParam("status", "available")
                      .when()
                          .get(base_url)
                      .then()
                          .statusCode(200)
                          .body("status", everyItem(equalTo("available")));

                   String response= RestAssured.given()
                          .contentType(ContentType.JSON)
                          .queryParam("status", "pending")
                      .when()
                          .get(base_url).getBody().asPrettyString();

                   logger.info("Response is " +response);
                                    logger.info("END:: GET method for the find PET by status=available");
                }

                @Test
                public void findPetsByPendingStatus() {
                        logger.info("START::GET method for the find PET by status=pending");
                                    logger.info("GET: URL" +base_url);
                   // Make GET call with status=pending
                   RestAssured.given()
                          .contentType(ContentType.JSON)
                .queryParam("status", "pending")
             .when()
                .get(base_url)
             .then()
                .statusCode(200)
                .body("status", everyItem(equalTo("pending")));

          String response= RestAssured.given()
                .contentType(ContentType.JSON)
                .queryParam("status", "pending")
             .when()
                .get(base_url).getBody().asPrettyString();

          logger.info("Response is " +response);
                    logger.info("END:: GET method for the find PET by status=pending");
      }

      @Test
      public void findPetsBySoldStatus() {
            logger.info("START::GET method for the find PET by status=sold");
                    logger.info("GET: URL" +base_url);
          // Make GET call with status=sold
          RestAssured.given()
                .contentType(ContentType.JSON)
                .queryParam("status", "sold")
             .when()
                .get(base_url)
             .then()
                .statusCode(200)
                .body("status", everyItem(equalTo("sold")));

          String response= RestAssured.given()
                .contentType(ContentType.JSON)
                .queryParam("status", "pending")
             .when()
                .get(base_url).getBody().asPrettyString();
          logger.info("Response is " +response);
                    logger.info("END:: GET method for the find PET by status=sold");
      }

}
```

Phase3.Assignment06.RestAssured.UserLogout

```
package Phase3.Assignment06.RestAssured;

import org.testng.annotations.Test;

import io.restassured.RestAssured;
import io.restassured.http.ContentType;

import static org.hamcrest.CoreMatchers.notNullValue;

import org.apache.log4j.Logger;


public class UserLogout {
		private static final String Base_url= "https://petstore.swagger.io";
		static final Logger logger= Logger.getLogger(UserLogout.class);


		@Test(description=" userLogout get method with the rest assured")

		public void TestUserLogout() {
			logger.info("START::GET method for the PET test");
			logger.info("POST: URL" +Base_url);
			try {
				RestAssured.given().baseUri(Base_url)
				.contentType(ContentType.JSON)
				.when().get("/v2/user/logout")
				.then().assertThat().statusCode(200)
				.and().assertThat().body("code", notNullValue())
				.and().assertThat().body("message", notNullValue());
			}


			catch(Exception e) {

		logger.error("Exception Object :: " + e.toString());
				logger.error("End Exception :: "+e.getLocalizedMessage());
	}

			String response =RestAssured.given().baseUri(Base_url)
					.contentType(ContentType.JSON)
					.when().get("/v2/user/logout").getBody().asPrettyString();

			logger.info("The response is" +response);
			logger.info("END::GET method for the PET test");
```

```
                        }
}
```

JMeter.

Phase3.FinalProject.JMeter

```
<?xml version="1.0" encoding="UTF-8"?>

<jmeterTestPlan version="1.2" properties="5.0" jmeter="5.6.2">

  <hashTree>

    <TestPlan guiclass="TestPlanGui" testclass="TestPlan" testname="Test Plan" enabled="true">

      <boolProp name="TestPlan.functional_mode">false</boolProp>

      <boolProp name="TestPlan.tearDown_on_shutdown">false</boolProp>

      <boolProp name="TestPlan.serialize_threadgroups">false</boolProp>

      <elementProp name="TestPlan.user_defined_variables" elementType="Arguments"
guiclass="ArgumentsPanel" testclass="Arguments" testname="User Defined Variables"
enabled="true">

        <collectionProp name="Arguments.arguments"/>

      </elementProp>

    </TestPlan>

    <hashTree>

      <ThreadGroup guiclass="ThreadGroupGui" testclass="ThreadGroup"
testname="Phase3.Final Project Thread Group " enabled="true">

        <stringProp name="ThreadGroup.on_sample_error">continue</stringProp>

        <elementProp name="ThreadGroup.main_controller" elementType="LoopController"
guiclass="LoopControlPanel" testclass="LoopController" testname="Loop Controller"
enabled="true">

          <stringProp name="LoopController.loops">1</stringProp>
          <boolProp name="LoopController.continue_forever">false</boolProp>

        </elementProp>

        <stringProp name="ThreadGroup.num_threads">10</stringProp>

        <stringProp name="ThreadGroup.ramp_time">1</stringProp>

        <boolProp name="ThreadGroup.delayedStart">false</boolProp>

        <boolProp name="ThreadGroup.scheduler">false</boolProp>

        <stringProp name="ThreadGroup.duration"></stringProp>

        <stringProp name="ThreadGroup.delay"></stringProp>

        <boolProp name="ThreadGroup.same_user_on_next_iteration">true</boolProp>

      </ThreadGroup>

      <hashTree>

        <HTTPSamplerProxy guiclass="HttpTestSampleGui" testclass="HTTPSamplerProxy"
testname="Authentication HTTP Request" enabled="true">

          <boolProp name="HTTPSampler.postBodyRaw">false</boolProp>

          <elementProp name="HTTPsampler.Arguments" elementType="Arguments"
guiclass="HTTPArgumentsPanel" testclass="Arguments" testname="User Defined Variables"
enabled="true">

            <collectionProp name="Arguments.arguments"/>

          </elementProp>

          <stringProp name="HTTPSampler.domain">httpbin.org</stringProp>

          <stringProp name="HTTPSampler.protocol">http</stringProp>

          <stringProp name="HTTPSampler.path">/basic-auth/user/passwd</stringProp>

          <stringProp name="HTTPSampler.method">GET</stringProp>

          <boolProp name="HTTPSampler.follow_redirects">true</boolProp>
          <boolProp name="HTTPSampler.auto_redirects">false</boolProp>

          <boolProp name="HTTPSampler.use_keepalive">true</boolProp>

          <boolProp name="HTTPSampler.DO_MULTIPART_POST">false</boolProp>

          <boolProp name="HTTPSampler.BROWSER_COMPATIBLE_MULTIPART">false</boolProp>

          <boolProp name="HTTPSampler.image_parser">false</boolProp>

          <boolProp name="HTTPSampler.concurrentDwn">false</boolProp>

          <stringProp name="HTTPSampler.concurrentPool">6</stringProp>

          <boolProp name="HTTPSampler.md5">false</boolProp>

          <intProp name="HTTPSampler.ipSourceType">0</intProp>

        </HTTPSamplerProxy>

        <hashTree>

          <JSONPathAssertion guiclass="JSONPathAssertionGui" testclass="JSONPathAssertion"
testname="JSON Assertion" enabled="true">

            <stringProp name="JSON_PATH">$.authenticated</stringProp>

            <stringProp name="EXPECTED_VALUE">true</stringProp>

            <boolProp name="JSONVALIDATION">true</boolProp>

            <boolProp name="EXPECT_NULL">false</boolProp>

            <boolProp name="INVERT">false</boolProp>

            <boolProp name="ISREGEX">true</boolProp>

          </JSONPathAssertion>

          <hashTree/>

        </hashTree>

        <AuthManager guiclass="AuthPanel" testclass="AuthManager" testname="HTTP
Authorization Manager" enabled="true">
```

```
          <collectionProp name="AuthManager.auth_list">
            <elementProp name="" elementType="Authorization">
              <stringProp name="Authorization.url"></stringProp>
              <stringProp name="Authorization.username">user</stringProp>
              <stringProp name="Authorization.password">passwd</stringProp>
              <stringProp name="Authorization.domain"></stringProp>
              <stringProp name="Authorization.realm"></stringProp>
            </elementProp>
          </collectionProp>
          <boolProp name="AuthManager.controlledByThreadGroup">false</boolProp>
        </AuthManager>
        <hashTree/>
        <HTTPSamplerProxy guiclass="HttpTestSampleGui" testclass="HTTPSamplerProxy"
testname="HTTP Request" enabled="true">
          <boolProp name="HTTPSampler.postBodyRaw">false</boolProp>
          <elementProp name="HTTPsampler.Arguments" elementType="Arguments"
guiclass="HTTPArgumentsPanel" testclass="Arguments" testname="User Defined Variables"
enabled="true">
            <collectionProp name="Arguments.arguments"/>
          </elementProp>
          <stringProp name="HTTPSampler.domain">www.simplilearn.com</stringProp>
          <stringProp name="HTTPSampler.protocol">http</stringProp>
          <stringProp name="HTTPSampler.path">/</stringProp>
          <stringProp name="HTTPSampler.method">GET</stringProp>
          <boolProp name="HTTPSampler.follow_redirects">true</boolProp>

          <boolProp name="HTTPSampler.auto_redirects">false</boolProp>

          <boolProp name="HTTPSampler.use_keepalive">true</boolProp>

          <boolProp name="HTTPSampler.DO_MULTIPART_POST">false</boolProp>

          <boolProp name="HTTPSampler.BROWSER_COMPATIBLE_MULTIPART">false</boolProp>

          <boolProp name="HTTPSampler.image_parser">false</boolProp>

          <boolProp name="HTTPSampler.concurrentDwn">false</boolProp>

          <stringProp name="HTTPSampler.concurrentPool">6</stringProp>

          <boolProp name="HTTPSampler.md5">false</boolProp>

          <intProp name="HTTPSampler.ipSourceType">0</intProp>

        </HTTPSamplerProxy>

        <hashTree>

          <XPathAssertion guiclass="XPathAssertionGui" testclass="XPathAssertion"
testname="XPath Assertion" enabled="true">

            <boolProp name="XPath.negate">false</boolProp>

            <stringProp name="XPath.xpath">//img[@title=&apos;Simplilearn - Online Certification
Training Course Provider&apos;]</stringProp>

            <boolProp name="XPath.validate">false</boolProp>

            <boolProp name="XPath.whitespace">false</boolProp>

            <boolProp name="XPath.tolerant">true</boolProp>

            <boolProp name="XPath.namespace">false</boolProp>

            <boolProp name="XPath.quiet">false</boolProp>

          </XPathAssertion>

          <hashTree/>
```

```
    </hashTree>

    <ResultCollector guiclass="ViewResultsFullVisualizer" testclass="ResultCollector"
testname="View Results Tree" enabled="true">

      <boolProp name="ResultCollector.error_logging">false</boolProp>

      <objProp>

        <name>saveConfig</name>

        <value class="SampleSaveConfiguration">

          <time>true</time>

          <latency>true</latency>

          <timestamp>true</timestamp>

          <success>true</success>

          <label>true</label>

          <code>true</code>

          <message>true</message>

          <threadName>true</threadName>

          <dataType>true</dataType>

          <encoding>false</encoding>

          <assertions>true</assertions>

          <subresults>true</subresults>

          <responseData>false</responseData>

          <samplerData>false</samplerData>

          <xml>false</xml>

          <fieldNames>true</fieldNames>
```

```xml
       <responseHeaders>false</responseHeaders>
       <requestHeaders>false</requestHeaders>
       <responseDataOnError>false</responseDataOnError>
       <saveAssertionResultsFailureMessage>true</saveAssertionResultsFailureMessage>
       <assertionsResultsToSave>0</assertionsResultsToSave>
       <bytes>true</bytes>
       <sentBytes>true</sentBytes>
       <url>true</url>
       <threadCounts>true</threadCounts>
       <idleTime>true</idleTime>
       <connectTime>true</connectTime>
      </value>
     </objProp>
     <stringProp name="filename"></stringProp>
    </ResultCollector>
    <hashTree/>
   </hashTree>
  </hashTree>
 </hashTree>
</jmeterTestPlan>
```