SOURCE CODE:

Postman Assignment:

```
{
	"info": {
		"_postman_id": "bd9cb61a-d7c7-40a5-9c4c-b3e390b91aaf",
		"name": "Pet_ID_Testing",
		"schema":
"https://schema.getpostman.com/json/collection/v2.1.0/collection.json",
		"_exporter_id": "31715036"
	},
	"item": [
		{
			"name": "CreatePetID",
			"event": [
				{
					"listen": "test",
					"script": {
						"exec": [
							"pm.test(\"Status code is 200\", function
() {\r",
							"    pm.response.to.have.status(200);\r",
							"});\r",
							"\r",
							"pm.test(\"Body matches string\",
function () {\r",
							"
pm.expect(pm.response.text()).to.include(\"available\");\r",
							"});"
						],
						"type": "text/javascript"
```

```
                            }
                        },
                        {
                            "listen": "prerequest",
                            "script": {
                                "exec": [
                                    ""
                                ],
                                "type": "text/javascript"
                            }
                        }
                    ],
                    "request": {
                        "method": "POST",
                        "header": [],
                        "body": {
                            "mode": "raw",
                            "raw": "{\r\n    \"id\": {{petID}},\r\n    \"category\":
{\r\n        \"id\": 0,\r\n        \"name\": \"string\"\r\n    },\r\n    \"name\": \"{{petName}}\",\r\n
\"photoUrls\": [\r\n        \"string\"\r\n    ],\r\n    \"tags\": [\r\n        {\r\n            \"id\": 0,\r\n
\"name\": \"string\"\r\n        }\r\n    ],\r\n    \"status\": \"available\"\r\n}",
                            "options": {
                                "raw": {
                                    "language": "json"
                                }
                            }
                        },
                        "url": {
                            "raw": "https://petstore.swagger.io/v2/pet",
                            "protocol": "https",
                    "host": [
                        "petstore",
                        "swagger",
                        "io"
                    ],
                    "path": [
                        "v2",
                        "pet"
                    ]
                }
            },
            "response": []
        },
        {
            "name": "GetPetID",
            "event": [
                {
                    "listen": "test",
                    "script": {
                        "exec": [
                            "pm.test(\"Status code is 200\", function
() {\r",
                            "    pm.response.to.have.status(200);\r",
                            "});\r",
                            "pm.test(\"Body matches string\",
function () {\r",
                            "
pm.expect(pm.response.text()).to.include(\"available\");\r",
                            "});"
                        ],
                "type": "text/javascript"
            }
        }
    ],
    "request": {
        "method": "GET",
        "header": [
            {
                "key": "accept",
                "value": "application/json"
            },
            {
                "key": "api_key",
                "value": "12345"
            }
        ],
        "url": {
            "raw": "https://petstore.swagger.io/v2/pet/{{petID}}",
            "protocol": "https",
            "host": [
                "petstore",
                "swagger",
                "io"
            ],
            "path": [
                "v2",
                "pet",
                "{{petID}}"
            ]
```

```
                }
            },
            "response": []
        },
        {
            "name": "DeletePet",
            "event": [
                {
                    "listen": "test",
                    "script": {
                        "exec": [
                            "pm.test(\"Status code is 200\", function
() {\r",
                            "    pm.response.to.have.status(200);\r",
                            "});\r",
                            "pm.test(\"Body matches string\",
function () {\r",
                            "
pm.expect(pm.response.text()).to.include(\"unknown\");\r",
                            "});"
                        ],
                        "type": "text/javascript"
                    }
                }
            ],
            "request": {
                "method": "DELETE",
                "header": [
                    {
                        "key": "accept",
                                "value": "application/json"
                        },
                        {
                                "key": "api_key",
                                "value": "12345"
                        }
                ],
                "url": {
                        "raw": "https://petstore.swagger.io/v2/pet/{{petID}}",
                        "protocol": "https",
                        "host": [
                                "petstore",
                                "swagger",
                                "io"
                        ],
                        "path": [
                                "v2",
                                "pet",
                                "{{petID}}"
                        ]
                }
        },
        "response": []
},
{
        "name": "Assignmen002",
        "request": {
                "method": "PUT",
                "header": [],
					"body": {
						"mode": "raw",
						"raw": "{\r\n\"id\": 9223372016900013000,
\"category\": {\r\n\"id\": 20021,\r\n\"name\": \"string\" },\r\n\"name\": \"doggie\",
\"photoUrls\": [\r\n\"string\"\r\n], \"tags\": [\r\n{\r\n\"id\": 0,\r\n\"name\":
\"string\"\r\n}\r\n],\r\n\"status\": \"{{status}}\" \r\n}",
						"options": {
							"raw": {
								"language": "json"
							}
						}
					},
					"url": {
						"raw": "{{testUrl}}",
						"host": [
							"{{testUrl}}"
						]
					}
				},
				"response": []
			},
			{
				"name": "Assignment003",
				"event": [
					{
						"listen": "test",
						"script": {
							"exec": [
								"pm.test(\"Status code is 200\", function
() {\r",
								"    pm.response.to.have.status(200);\r",
                                        "});\r",
                                        "pm.test(\" Validate UserName\",
function () {\r",
                                        "    var jsonData =
pm.response.json();\r",
                                        "
pm.expect(jsonData.username).to.eql(\"Uname001\");\r",
                                        "});\r",
                                        "pm.test(\" Validate Email \", function ()
{\r",
                                        "    var jsonData =
pm.response.json();\r",
                                        "
pm.expect(jsonData.email).to.eql(\"Positive@Attitude.com\");\r",
                                        "});\r",
                                        "pm.test(\"Your test name\", function ()
{\r",
                                        "    var jsonData =
pm.response.json();\r",
                                        "
pm.expect(jsonData.userStatus).to.eql(1);\r",
                                        "});"
                                    ],
                                    "type": "text/javascript"
                              }
                        }
                  ],
                  "request": {
                        "method": "GET",
                        "header": [],
                        "url": {
                              "raw":
"https://petstore.swagger.io/v2/user/{{UserName}}",
                    "protocol": "https",
                    "host": [
                        "petstore",
                        "swagger",
                        "io"
                    ],
                    "path": [
                        "v2",
                        "user",
                        "{{UserName}}"
                    ]
                }
            },
            "response": []
        },
        {
            "name": "Assignment004",
            "event": [
                {
                    "listen": "test",
                    "script": {
                        "exec": [
                            "pm.test(\"Status code is 200\", function
() {\r",
                            "    pm.response.to.have.status(200);\r",
                            "});\r",
                            "\r",
                            "pm.test(\"All pets are available\",
function () {\r",
                                                        "    let responseJson = 
pm.response.json();\r",
                                                        "    responseJson.forEach((pet) => {\r",
                                                        "        
pm.expect(pet.status).to.equal(\"available\");\r",
                                                        "    });\r",
                                                        "});"
                                                ],
                                                "type": "text/javascript"
                                        }
                                }
                        ],
                        "request": {
                                "method": "GET",
                                "header": [],
                                "url": {
                                        "raw": 
"https://petstore.swagger.io/v2/pet/findByStatus?status= Sold",
                                        "protocol": "https",
                                        "host": [
                                                "petstore",
                                                "swagger",
                                                "io"
                                        ],
                                        "path": [
                                                "v2",
                                                "pet",
                                                "findByStatus"
                                        ],
                                        "query": [
                {
                    "key": "status",
                    "value": " Sold"
                },
                {
                    "key": "status",
                    "value": " pending",
                    "disabled": true
                },
                {
                    "key": "status",
                    "value": " sold",
                    "disabled": true
                }
            ]
        }
    },
    "response": []
},
{
    "name": "Assignment005",
    "event": [
        {
            "listen": "test",
            "script": {
                "exec": [
                    "pm.test(\"Validate code\", function ()
{\r",
                    "    pm.response.to.have.status(200);\r",
                    "});\r",
                                                "pm.test(\"Validate message\", function
() {\r",
                                                "    var jsonData =
pm.response.json();\r",
                                                "
pm.expect(jsonData.message).to.eql(\"ok\");\r",
                                                "});\r",
                                                "pm.test(\"Validate message\", function
() {\r",
                                                "    var jsonData =
pm.response.json();\r",
                                                "
pm.expect(jsonData.code).to.eql(200);\r",
                                                "});"
                                        ],
                                        "type": "text/javascript"
                                }
                        }
                ],
                "request": {
                        "method": "GET",
                        "header": [],
                        "url": {
                                "raw": "https://petstore.swagger.io/v2/user/logout",
                                "protocol": "https",
                                "host": [
                                        "petstore",
                                        "swagger",
                                        "io"
                                ],
                                "path": [
                                                "v2",

                                                "user",

                                                "logout"

                                        ]

                                }

                        },

                        "response": []

                }

        ]

}
```

===

REST Assured Assignment:

```java
package courseEndProject;


import java.io.File;

import org.apache.logging.log4j.LogManager;
import org.apache.logging.log4j.Logger;
import org.hamcrest.Matchers;
import org.testng.annotations.Test;

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


public class Assignment001 {

	Logger logger = LogManager.getLogger(Assignment001.class);

	@Test(priority='1')

	public void Assignment001Post()  {


		logger.info("Course End Project - Assignment001 - Post 
Request");
		File file = new File("C:\\Users\\DELL\\eclipse-
workspace\\SL_SeleniumDemo_Workspace\\Phase-3-
EndProject\\src\\main\\resource\\data.json");
		int id = RestAssured.given()
				.baseUri("https://petstore.swagger.io/v2/pet")
				.contentType(ContentType.JSON)
```

```
                        .body(file)
                        .when().post()
                        .then()
                        .statusCode(200)
                        .log().all()
                        .body("name",
Matchers.equalTo("Doggie")).extract().path("id");
                        logger.trace("The status code is checked");
                        System.out.println(id);
                        logger.trace("ID has been captured and validated");



      }
      @Test(priority='2', dependsOnMethods="Assignment001Post")

      public void assignment001Get() {

            logger.info("Course End Project - Assignment001 - Get
Request");

            int id = RestAssured.given()
                        .baseUri("https://petstore.swagger.io/v2/pet/344")
                        .when().get()
                        .then().statusCode(200)
                        .log().all()
                        .body("status",
Matchers.equalTo("available")).extract().path("category.id");
            System.out.println(id);
            logger.trace("ID  and status has been captured and validated");
      }

      @Test(priority='3', dependsOnMethods="assignment001Get")

      public void assignment001Delete() {

            logger.info("Course End Project - Assignment001 - Delete
Request");

            RestAssured.given()
                        .baseUri("https://petstore.swagger.io/v2/pet/344")
                        .when().delete()
                        .then().statusCode(200)
                        .log().all()
                        .body("code", Matchers.equalTo(200))
                        .body("type", Matchers.equalTo("unknown"))
                        .body("message", Matchers.equalTo("344"));

      }




}
============================================================================
package courseEndProject;

import java.io.File;

import org.apache.logging.log4j.LogManager;
import org.apache.logging.log4j.Logger;
import org.hamcrest.Matchers;
import org.testng.annotations.Test;

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


public class Assignment002 {

	Logger logger = LogManager.getLogger(Assignment002.class);

	@Test(priority = '1')

	public void assignment002Post() {
		logger.info("Course End project - Assignment002 - POST
request");
		File file = new File("C:\\Users\\DELL\\eclipse-
workspace\\SL_SeleniumDemo_Workspace\\Phase-3-
EndProject\\src\\main\\resource\\data.json");
		int id =
RestAssured.given().baseUri("https://petstore.swagger.io/v2/pet").contentTy
pe(ContentType.JSON)

	.body(file).when().post().then().statusCode(200).log().all().body("na
me", Matchers.equalTo("Doggie"))
				.extract().path("id");
		logger.trace("The status code is checked");

		System.out.println(id);

		logger.trace("Id has been captured and validated");

	}

	@Test(priority = '2', dependsOnMethods = "assignment002Post")

	public void assignment002Put() {
		File file = new File("C:\\Users\\DELL\\eclipse-
workspace\\SL_SeleniumDemo_Workspace\\Phase-3-
EndProject\\src\\main\\resource\\dataput.json");
		int id =
RestAssured.given().baseUri("https://petstore.swagger.io/v2/pet/").contentT
ype(ContentType.JSON)

	.body(file).when().put().then().statusCode(200).log().all()
				.body("status",
Matchers.equalTo("available_QA")).extract().path("id");

		System.out.println(id);

	}

}
===========================================================================
```

```java
package courseEndProject;

import org.hamcrest.Matchers;
import org.testng.annotations.Test;

import io.restassured.RestAssured;


public class Assignment003And004 {

	@Test(priority='1')

	public void assignment003User()
	{
		RestAssured.given()
		.baseUri("https://petstore.swagger.io/v2/user/Uname001")
		.when()
		.get()
		.then()
		.statusCode(200)
		.log().all()
		.body("username", Matchers.equalTo("Uname001"))
		.body("email", Matchers.equalTo("Positive@Attitude.com"))
		.body("userStatus", Matchers.equalTo(1))
		;



	}


	@Test(priority='2')

	public void assignment004login()
	{
		RestAssured.given()
		.baseUri("https://petstore.swagger.io/v2/user/login")
		.auth().preemptive().basic("Uname001", "@tt!tude")
		.when()
		.get()
		.then()
		.statusCode(200)
		.log().all()
		.body("message", Matchers.anything());



	}

}
```

===

```java
package courseEndProject;

import org.hamcrest.Matchers;
import org.testng.annotations.Test;

import io.restassured.RestAssured;


public class Assignment005And006 {


	@Test(priority='1')

	public void assignment005FindByStatus()
	{
		RestAssured.given()
		.baseUri("https://petstore.swagger.io/v2/pet/findByStatus")
		//.queryParam("status", "available")
		//.queryParam("status", "pending")
		.queryParam("status", "sold")
		.when()
		.get()
		.then()
		.statusCode(200)
		.log().all()

		;



	}



	@Test(priority='2')

	public void assignment006Logout()
	{
		RestAssured.given()
		.baseUri("https://petstore.swagger.io/v2/user/logout")
		.when()
		.get()
		.then()
		.statusCode(200)
		.log().all()
		.body("code", Matchers.equalTo(200))
		.body("type", Matchers.equalTo("unknown"))
		.body("message", Matchers.equalTo("ok"))
		;



	}


}
```
===

```
{
    "id": 344,
    "category": {
      "id": 0,
      "name": "string"
    },
    "name": "Doggie",
    "photoUrls": [
      "string"
    ],
    "tags": [
      {
        "id": 0,
        "name": "string"
      }
    ],
    "status": "available"
  }
```

==

```
{
    "id": 987,
    "category": {
      "id": 0,
      "name": "string"
    },
    "name": "duck",
    "photoUrls": [
      "string"
    ],
    "tags": [
      {
        "id": 0,
        "name": "string"
      }
    ],
    "status": "available_QA"
  }
```

==

Jmeter Assignment:

```
<?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="Jmeter-Assignment-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="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">1</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>
```

```xml
    <boolProp name="ThreadGroup.same_user_on_next_iteration">true</boolProp>
  </ThreadGroup>
  <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" enabled="true">
        <collectionProp name="Arguments.arguments"/>
      </elementProp>
      <stringProp name="HTTPSampler.domain">httpbin.org</stringProp>
      <stringProp name="HTTPSampler.protocol">https</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>
      <AuthManager guiclass="AuthPanel" testclass="AuthManager" testname="HTTP
Authorization Manager" enabled="true">
        <collectionProp name="AuthManager.auth_list">
          <elementProp name="" elementType="Authorization">
```

```
            <stringProp name="Authorization.url">https://httpbin.org/</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/>
    </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>

            <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>

    <ThreadGroup guiclass="ThreadGroupGui" testclass="ThreadGroup" testname="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">1</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="HTTP Request" enabled="true">
          <boolProp name="HTTPSampler.postBodyRaw">false</boolProp>
          <elementProp name="HTTPsampler.Arguments" elementType="Arguments"
guiclass="HTTPArgumentsPanel" testclass="Arguments" enabled="true">
            <collectionProp name="Arguments.arguments"/>
          </elementProp>
          <stringProp name="HTTPSampler.domain">www.simplilearn.com</stringProp>
          <stringProp name="HTTPSampler.protocol">https</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">false</boolProp>
            <boolProp name="XPath.namespace">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>
              <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>
```