# Selenium API

It’s Selenium API, you can test simple call one API, for all end-2-end test case only deploying on your server API.
**IMPORTANT**. If you want to deploy on cloud you must configure properly selenium as well browser driver.
I realize a proof of concept for Chrome on AWS Ec2. It’s work like a charm!

## Requirements

**REST API** /test accept json request for simulate user UI action:

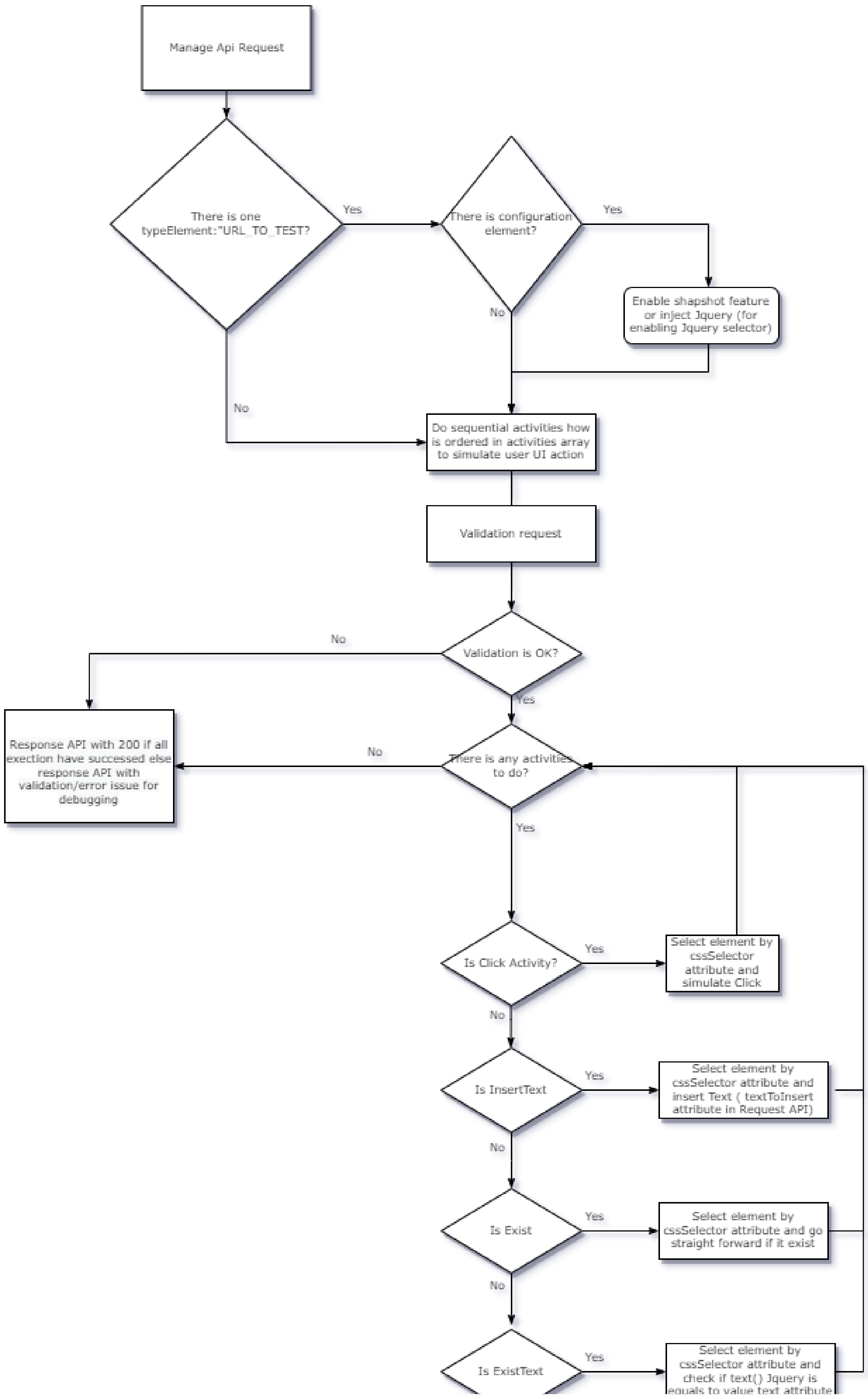
- **Click** on anchor and button
- **Insert text** in texbox
- **Perform** some check existing text in specific DOM element
- **Check** if exist specific DOM element that will be recognize from css selector

Request will be validate by check if exist all required input and css selector find minimum one DOM element.
If there are some validation errors that will be showed in response.

# API Selenium

GET IP:PORT/test

REQUEST BODY: json structure

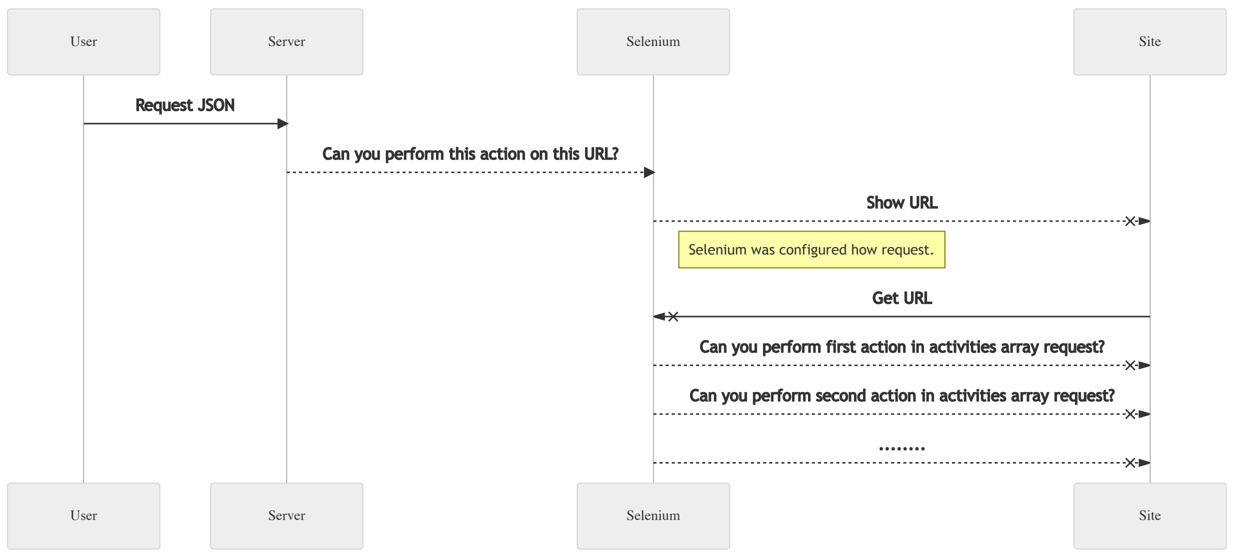

Manage Api Request
There is one typeElement:"URL_TO_TEST?
Yes
There is configuration element?
Yes
Enable shapshot feature or inject Jquery (for enabling Jquery selector)
No
No
Do sequential activities how is ordered in activities array to simulate user UI action
Validation request
No
Validation is OK?
Yes
Response API with 200 if all exection have successed else response API with validation/error issue for debugging
No
There is any activities to do?
Yes
Is Click Activity?
Yes
Select element by cssSelector attribute and simulate Click
No
Is InsertText
Yes
Select element by cssSelector attribute and insert Text ( textToInsert attribute in Request API)
No
Is Exist
Yes
Select element by cssSelector attribute and go straight forward if it exist
No
Is ExistText
Yes
Select element by cssSelector attribute and check if text() Jquery is equals to value text attribute

## Technical requirement

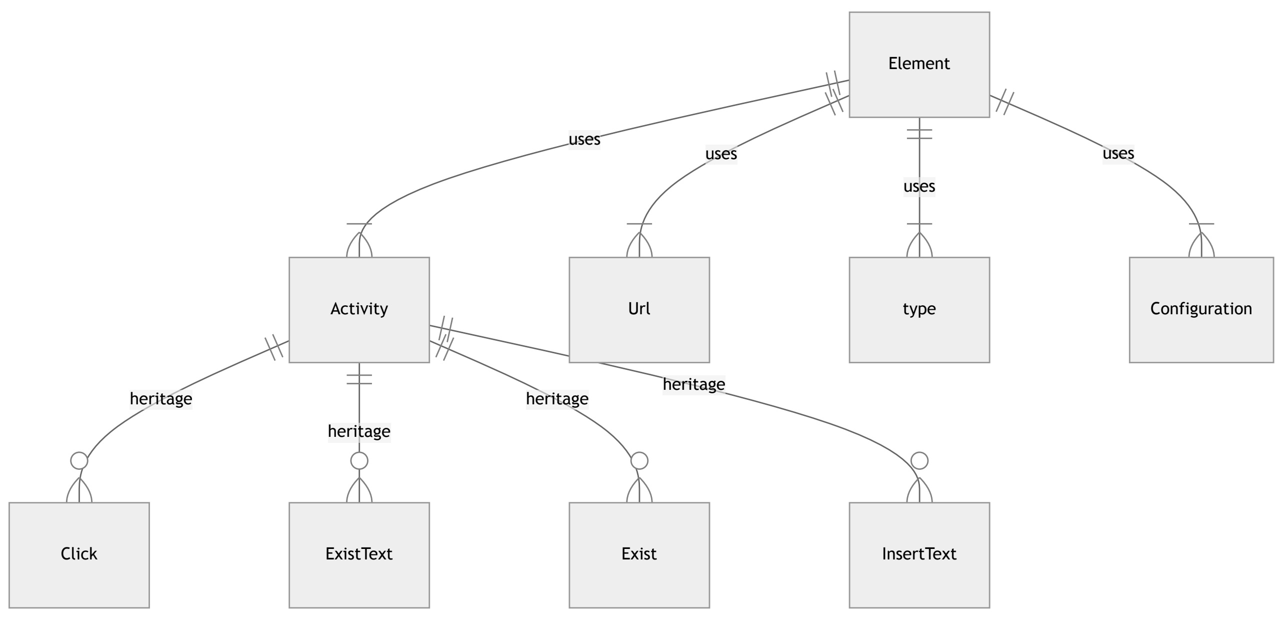
Spring Boot application expose REST API /test accept request for run headless Selenium instance.

### Diagrams

What’s do it succed when you hit on REST API?
We discuss about parts are envolved when you running one test.


User
Server
Selenium
Site
Request JSON
Can you perform this action on this URL?
Show URL
Selenium was configured how request.
Get URL
Can you perform first action in activities array request?
Can you perform second action in activities array request?
........
User
Server
Selenium
Site

When you send request you must define this elements:



## Actions performed

All actions extends Activity class and was distinguee by type field.

| Value type field | Action |
|---|---|
| ExistText | `After item DOM selected check if text setter in 'text' attribute is present` |
| Exist | `If item DOM selected go forward` |
| Click | `After item DOM selected click on it` |
| InsertText | `After item DOM selected set value set on 'textToInsert attribute'` |

## API REST: POST /test

Request is formed by:

- **Url**: is url to start test case

- **Configuration**: some extra configuration

    - **useJQuery**: Inject JQuery script
    - **makeSnapshot**: make Snapshot
    - **headless**: Enable headless mode (enable that for server mode)
    - **remoteDebuggingPort**: could be necessary in some case for ec2 instance
    - **browserVersion**: you can define which version browser is installed on system environment.
    - **driverVersion**: you can define which version driver is installed on system environment.
    - **enableThirdPartyCookies**: you can define true, if you want enable third Party cookies management browser
    - **userAgent**: you can customize userAgent
    - **singleProcess**: you can add option
    - **extraConfiguration**: you can add extra options will be add (all options must be separate ; symbol)

- **Activities**: Actions will be performed sequentially. Think about: if you click on ahref, the page changed url and there are some DOM element different then you check if exist element and click on it after.
  Change order action executing change result, so make sure ordering actions are the same you performe like user.

JSON request to API running different actions: before click on button with redirection to new page, after fill form and insert values submitted.
After check if exist “OK” phrase in the page for checking if exist the address we find.

```
{
 "url": "https://www.eolo.it/home/casa/eolo/eolo-piu.html",
"configuration":{
"useJQuery":false,
"makeSnapshot":false,
"headless":false,
"singleProcess":false,
"remoteDebuggingPort":false,
"driverVersion":"105.0.5195.52",
"browserVersion": "105.0.5195.102",
"userAgent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTM
"extraConfiguration": "--disable-dev-shm-usage"
},
"activities": [
{
        "type": _"Click"_,
        "selector":{
                "cssSelector": ".eolo-offerta-rectangle .btn.eolo-orange-button.
                }
},
{
        "type": "InsertText",
        "textToInsert": "GALLARATE",
        "selector":{
                "cssSelector": ".form-field.field-example #city_label"
                }
        },
        {
                "type": "Click",
                "selector":{
                        "cssSelector": "#ui-id-1 .ui-menu-item"
                }
        },
        {
                "type": "InsertText",
                "textToInsert":"Via Roma",
                "selector":{
                        "cssSelector": ".form-field.field-example #address"
                }
        },
        {
                "type": "Click",
                "selector":{
                        "cssSelector": ".ui-autocomplete .ui-menu-item.ui-menu-i
                }
        },
        {
                "type": "InsertText",
                "textToInsert":"1",
                "selector":{
                "cssSelector": ".form-field.field-example #number"
                }
        },
        {
                "type": "Click",
                "selector":{
                "cssSelector": "#copertura-eolo-2-form button.eolo-orange-button
                }
        },
        {
                "type": "Exist",
                "selector":{
                "cssSelector": ".new-funnel-2021--address-content-info img.new-f
                }
        },
        {
                "type": "ExistText",
                "text": "La tua zona è coperta!",
                "selector":{
                "cssSelector": "#flow-form .new-funnel-2021--configuration.new-f
                }
        }
]
}
```

## Response API

How we discuss before, API can have 2 response status:

- **500**: if something go wrong, you can have a similar response

```
{
        "requestId": null,
        "errors": {
                "error": {
                                "text": "Error configuration test",
                                "className": "configuration",
                                "message": null
                }
        },
"result": "Test non avvenuto con successo"
}
```

- **200**: if is ok, we have in response one id and message for showing message success
  {
  “requestId”: “5a730e98-d5c3-4959-bfa0-d6daf398e791”,

“errors”: null,
“result”: “Test avvenuto con successo”
}

### How we can test call API?

It easy to create one test **JUnit** for call localhost API and create correct request JSON.
If you want you can send raw json file to body as well, use this code:

```
@Test
public void PageTestingRest_existText() throws Exception {
        Click click = new Click();
        click.getSelector().setCssSelector("input[value='Google Search']");
        Configuration configuration = BuilderConfiguration._create_().browserVer
        TestCase test = BuilderTestCase._create_().url(testProperty.getUrl()).ad
        mvc.perform(_get_(testProperty.getApiTest()).content(JsonUtils._convertO
}
```

## Configuration Ec2 Amazon Linux

If you want Selenium API in Ec2 Aws you must install Chromium and Chrome Browser for permit to selenium running Chrome instance.
Actually we must install version major than 104, because selenium driver dependency manage only 104 or major.
It necessary to install properly **Chrome Driver**

```
wget https://chromedriver.storage.googleapis.com/104.0.5112.20/chromedriver_linu
sudo unzip chromedriver_linux64.zip
sudo mv chromedriver /usr/bin/chromedriver
chromedriver --version
```

For the **Google Chrome** to work we also need to install Google Chrome.

```
curl https://intoli.com/install-google-chrome.sh | bash
sudo mv /usr/bin/google-chrome-stable /usr/bin/google-chrome
```

```
google-chrome --version && which google-chrome
```

In general can be useful install chromium browser but it depend to kernel and server you choose:

```
sudo yum install chromium browser
sudo amazon-linux-extras install epel -y
sudo yum install -y chromium
chromium -version
```

## Troubleshooting

If you have this error when you try to run spring boot application:

```
The JAVA_HOME environment variable is not defined correctly, this environment va
```

If you have correct Java and Maven configuration try to refresh it with

```
source /etc/profile.d/maven.sh
```

## Run application

Application can be simple runned with

```
cd /home/ec2-user/actions-runner/_work/PROJECT_NAME
mvn spring-boot:run
```

Where PROJECT_NAME is correct directory which project is saved.
You can run in different way standalone application, java application etc.

## Error in response API

If you have some error in response API link to Selenium, could be a version mismatching from **version environment** and **version software** setup (java code).

Version driver can work with different version browser, for example:

```
 "driverVersion":"104.0.5112.20"
```

works correctly with

```
  "browserVersion":"104.0.5112.79"
```

(on environment we have driver version **104.0.5112.20** and browser version **104.0.5112.79** so we setup in api this configuration).
But occasionally auto update browser on environment can require changes in request API regarding driverVersion and browserVersion.
Actually API implements Selenium for Chromium so we refer this browser.

To stop Chrome browser auto-updating, take one of the following actions:

- Create an empty repository before installing Chrome browser:
  `$ sudo touch /etc/default/google-chrome`
- Add the following line to **/etc/default/google-chrome**:
  `repo_add_once=false`

## Future improvement

1. Design a **microservices architecture** can improve reliability and
   number of request management with more microservice API back to one **Gateway API** hide complexity.
2. Uploading batches on **S3 bucket** all snapshot for consulting necessary
3. **Dashboard** with all request api and result (SUCCESS or FAILURE)
4. Implements **Kafka integration** for **asynchrous batch** and use Kafka streams for stats real time. Could be consumed for showing on dashboard.
5. Use **Spring Batch** for plan testing in specific time (example in hour when environment not be so loaded).

## Documentation API

Documentation REST API is automagically with Swagger library.
If you go at ./swagger-ui/index.html you can dinamically see request and response API structure.