[PAINLESS]

1. Extensible Security implementation for restricting anonymous BOT hit for APIs and Form Submitting events in your web application.

**Answer:**

For Form Validation, we have implemented Google **recaptcha V2** in the Login form for OCR

Application (Question #7). Our captcha validation implementation is integrated with OCR web app.

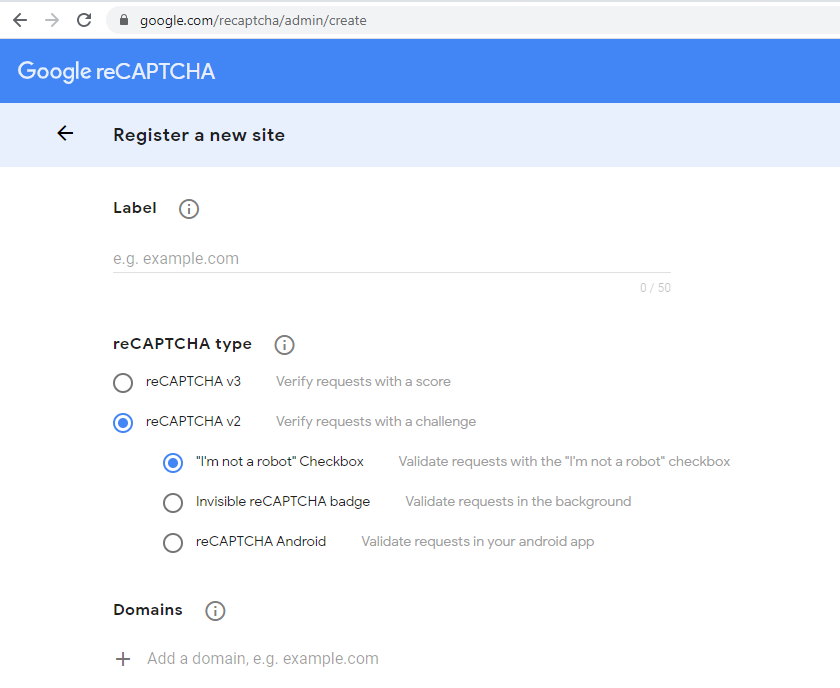
**Source Code GIT Repository:**

[**https://github.com/Tech-Pirates-Aspiring-Minds/springboot-ocr-data-extraction**](https://github.com/Tech-Pirates-Aspiring-Minds/springboot-ocr-data-extraction)

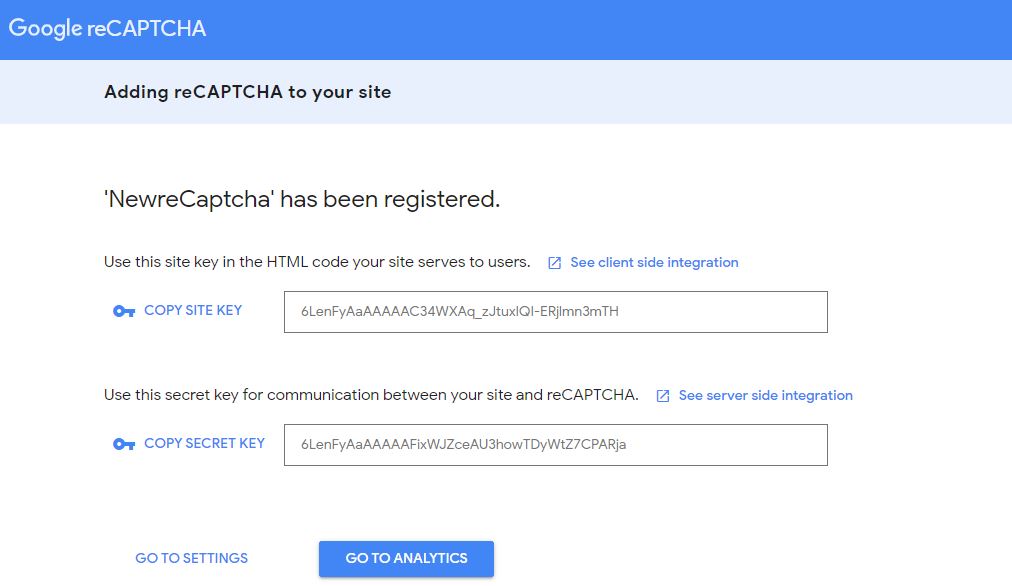
Captcha poses a digital challenge that is usually easily solvable by human beings and difficult for computers to perform, thereby mitigating the risk of bots automatically filling up a form.

**Recaptcha Key creation:**

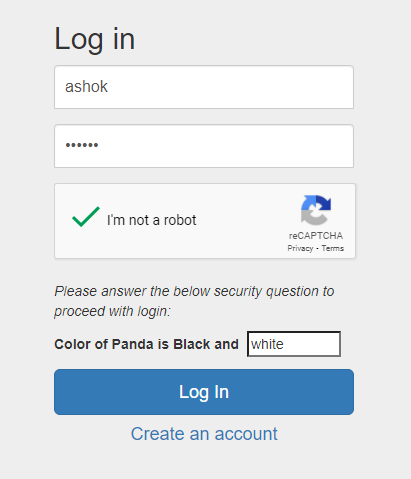
**STEP 1:** <https://www.google.com/recaptcha/admin/create>



**STEP 2:** Obtain the keys generated to use in form validation



**Form Validation (client side):**



Login Form includes validation for captcha and a challenge question for two steps security.

**Client side validations implemented are (in JS):**

1. Username is mandatory
2. Password is mandatory
3. Captcha is mandatory
4. Challenge answer is mandatory

P.S: The challenge question is not dynamic.

This is a basic level challenge to enter answer in a free text, which a robot cannot predict / fill easily.

**Server side validations implemented are:**

1. User Name and Password Validation - using Spring security
2. Captcha validation - using Recaptcha API response

Invoke captcha validation only after user name and password validation is successful.

I.e. logically we don’t have to invoke captcha API if either username or password or both are incorrect.

**Related files:**

CaptchaService.java

CaptchaServiceImpl.java

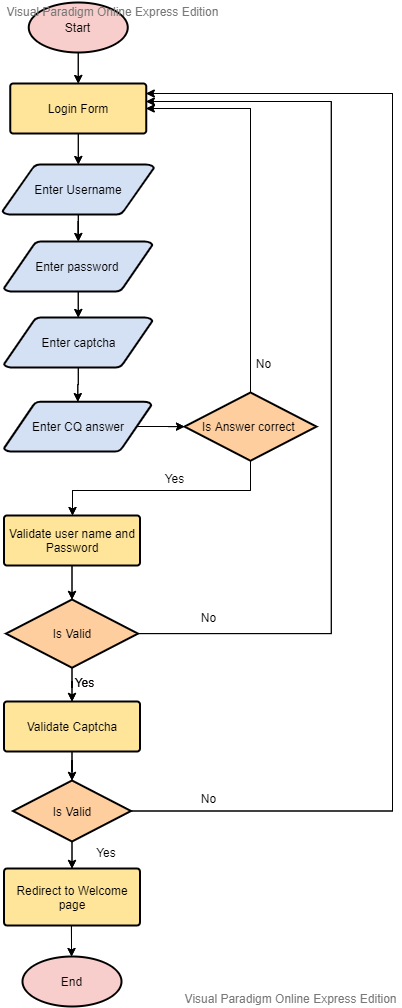
CaptchaValidator.java

CaptchaResponse.java

**Test Cases:**

****

**Flow diagram:**

****