**ТЕСТОВЫЕ ЗАДАНИЯ ДЛЯ СЕРВЕРНЫХ РАЗРАБОТЧИКОВ**

**Введение**

Задания предназначены для проверки знаний Selenium и Ruby (фреймворк Ruby on Rails) в качестве языка разработки.

Предполагаемый срок исполнения задания - не более 1 дня.

Можно выполнить 1 из 4 заданий.

**Общие указания к заданиям**

Необходимо продемонстрировать не только умение решать поставленную задачу, но и умение качественно оформлять код:

1. выбор содержательных названий для классов, методов и переменных
2. разумные комментарии к каждому классу и методу, указание типа и описание для каждого параметра.
3. разумные размеры (по количеству строк) методов и классов
4. использование констант с понятными названиями вместо «магических чисел».

Необходимо обдумать не только «позитивный» случай, но и возможные ошибки в процессе обработки (на практике они более чем возможны), в том числе случай, когда меняется структура страниц и алгоритм в принципе перестаёт функционировать корректно. Все такие ошибки должны корректно и, по возможности, унифицированно обрабатываться.

**Задания**

***Задание 1. Реализуйте автоматизированную проверку наличия новых личных сообщений на Вашем любимом форуме с помощью Selenium на ruby.***

Предполагаемый алгоритм решения задачи:

1. Залогиниться на форум, используя предоставленные логин и пароль аккаунта.
2. Перейти в раздел личных сообщений
3. Попытаться найти элемент, содержащий число новых сообщений
4. В случае его наличия - прочитать innerText / value.

Задачу нужно реализовать на Ruby On Rails в виде API-вызова, где логин и пароль аккаунта являются GET-параметрами, а ответ - JSON-объект с единственным полем "unread\_messages\_count".

Рекомендуется продумать архитектуру решения: желательно выделить "низкий уровень" элементарных запросов и "высокий уровень" управление навигацией и обработки ошибок навигации, обязательно использовать принцип "тонких контроллеров".

***Задание 2. Реализуйте автоматизированный сбор списка самых активных участников Вашего любимого форума (или любого форума, где есть список пользователей, отсортированный по числу сообщений)***

Предполагаемый алгоритм решения задачи:

1. Залогиниться на форум, используя предоставленные логин и пароль аккаунта, если это требуется для получения списка пользователей.
2. Получить список пользователей, отсортированных по числу сообщений.
3. Собрать все username аккаунтов (и число их сообщений), у которых больше N сообщений, пролистывая страницы списка.
4. Занести информацию в таблицу forum\_users базы данных с полями { VARCHAR username, INT messages\_count, DATE scraped\_date }, где username + scraped\_date (дата сбора данных) имеют ограничение на уникальность (unique constraint). В случае наличия записи - не обновлять данные.

Задачу нужно реализовать на Ruby On Rails в виде периодической задачи Resque + Resque Scheduler, где логин и пароль аккаунта, а также расписание выполнения хранятся в конфигурационном файле.

Рекомендуется продумать архитектуру решения: желательно выделить "низкий уровень" элементарных запросов и "высокий уровень" управление навигацией и обработки ошибок навигации, обязательно использовать принцип "тонких контроллеров" для resque jobs.

***Задание 3. Реализуйте автоматизированный сбор данных о пользователях Вашего любимого форума.***

Предполагаемый алгоритм решения задачи:

1. Залогиниться на форум, используя предоставленные логин и пароль аккаунта, если это требуется для получения списка пользователей.
2. Получить список пользователей, отсортированных по алфавиту.
3. Собрать все username аккаунтов, и для каждого получить аватар (именно картинку, а не просто URL) со страницы профиля.
4. Занести информацию в таблицу forum\_users базы данных с полями { VARCHAR username, VARCHAR profile\_url, VARCHAR avatar\_image, DATE last\_scraped\_date }, если запись уже существует - обновить данные.

Задачу нужно реализовать на Ruby On Rails в виде периодической задачи Resque + Resque Scheduler, где логин и пароль аккаунта, а также расписание выполнения хранятся в конфигурационном файле.

Работу с аватарами реализовать с использованием гема carrierwave.

Рекомендуется продумать архитектуру решения: желательно выделить "низкий уровень" элементарных запросов и "высокий уровень" управление навигацией и обработки ошибок навигации, обязательно использовать принцип "тонких контроллеров" для resque jobs.

***Задание 4. Реализуйте автоматизированный поиск тем по ключевым словам на Вашем любимом форуме.***

Предполагаемый алгоритм решения задачи:

1. Залогиниться на форум, используя предоставленные логин и пароль аккаунта, если это требуется для получения списка пользователей.
2. Получить список тем с ключевыми словами, используя поиск по форуму.
3. Собрать все названия и URL тем из списка результатов, а также количество сообщений в каждой из них. Желательно также получить стартовое сообщение темы.

Задачу нужно реализовать на Ruby On Rails в виде API-вызова, где логин и пароль аккаунта, а также поисковая фраза ("query") являются GET-параметрами, а ответ - JSON-объект с полями "success": true и "data" - списком объектов с полями "title", "body", "replies\_count", "url", описывающих найденные темы.

Рекомендуется продумать архитектуру решения: желательно выделить "низкий уровень" элементарных запросов и "высокий уровень" управление навигацией и обработки ошибок навигации, обязательно использовать принцип "тонких контроллеров".

Сontacts

[Evgeni Zhdanovskih](https://ru.linkedin.com/in/evgenii-zhdanovskikh-31888736)

whatsapp, telegram, etc.: +79153407008,

skype: zevusa

evgeni@smuva.com