Введение

Цель вебинара:

- Понять ключевые различия между Selenium и Selenide.
- Оценить преимущества и недостатки каждого инструмента.
- Определить, какой из них лучше подходит для конкретных задач автоматизации.

Краткое описание:

- Обзор функциональных возможностей Selenium и Selenide.
- Примеры использования в реальных проектах.
- Основные критерии выбора инструмента в зависимости от задач и условий проекта.

Что такое Selenium?

Описание: Selenium – это набор инструментов для автоматизации веб-приложений. Он включает несколько компонентов:

Основные компоненты:

1. **Selenium WebDriver**:
    - Низкоуровневый API для взаимодействия с браузерами.
    - Позволяет управлять браузером на программном уровне, выполнять действия на странице, проверять состояния элементов.
    - Поддерживает работу с различными браузерами (Chrome, Firefox, Safari, Edge).
2. **Selenium IDE**:
    - Графический инструмент для записи и воспроизведения тестов.
    - Подходит для быстрого создания простых тестов без написания кода.
    - Удобен для начинающих и прототипирования тестов.
3. **Selenium Grid**:
    - Инструмент для параллельного выполнения тестов на различных машинах и браузерах.
    - Используется для распределенного тестирования, что ускоряет выполнение тестов и позволяет проверять совместимость на разных платформах.

Поддерживаемые языки программирования:

- Java
- Python

- C#
- Ruby
- JavaScript и другие

Архитектура:

- Использует модель клиент-сервер для взаимодействия с браузерами через WebDriver.
- Поддерживает работу с браузерами через локальные и удаленные драйверы.

Преимущества и недостатки Selenium

Преимущества:

- **Широкая совместимость:** Поддержка множества браузеров (Chrome, Firefox, Safari, Edge и др.) и операционных систем (Windows, macOS, Linux).
- **Гибкость:** Возможность написания сложных тестов с кастомной логикой и проверками.
- **Расширяемость:** Возможность интеграции с различными фреймворками и инструментами (JUnit, TestNG, Maven, Jenkins).
- **Сообщество:** Большое сообщество разработчиков, множество ресурсов и документации.

Недостатки:

- **Сложность управления ожиданиями:** Необходимость вручную настраивать ожидания (Implicit, Explicit Waits) для синхронизации с элементами.
- **Отсутствие встроенной поддержки скриншотов и логирования:** Требуется использование дополнительных библиотек.
- **Сложность работы с динамическими элементами:** Без дополнительных решений сложно обрабатывать всплывающие окна, фреймы и динамические элементы.

Что такое Selenide?

Описание: Selenide – это библиотека для автоматизированного тестирования веб-приложений, основанная на Selenium. Она предоставляет более высокоуровневый и удобный API для упрощения работы с веб-элементами и тестирования.

Основные особенности:

1. **Автоматические ожидания:**
   - Встроенные ожидания элементов: Selenide автоматически ждёт появления, исчезновения и изменения элементов, что значительно упрощает тесты.
   - Умное управление синхронизацией без необходимости ручной настройки.
2. **Простое и читаемое API:**
   - Методы, ориентированные на пользователя, такие как `$("#elementId").click()`, `$("#elementId").shouldHave(text("Hello"))`.
   - Поддержка CSS и XPath селекторов для поиска элементов.
3. **Работа с модальными окнами и фреймами:**
   - Поддержка работы с алертами, модальными окнами и фреймами прямо из коробки без дополнительной настройки.
4. **Интеграция с отчетностью и логированием:**
   - Легкая интеграция с Allure для генерации отчетов.
   - Автоматическое создание скриншотов и отчетов при ошибках.

Преимущества и недостатки Selenide

Преимущества:

- **Простота использования:** Упрощенный API и автоматические ожидания позволяют писать более короткие и понятные тесты.
- **Меньше кода:** Меньше строк кода для выполнения одинаковых задач по сравнению с Selenium.
- **Интеграция с отчетностью:** Поддержка Allure, автоматическое создание скриншотов при ошибках.
- **Управление браузером:** Selenide предоставляет встроенные методы для открытия, закрытия и перезагрузки браузера, что упрощает работу с тестами.

Недостатки:

- **Ограниченная поддержка браузеров:** Официальная поддержка только Chrome и Firefox. Для других браузеров могут потребоваться дополнительные настройки.
- **Меньшая гибкость:** Для более сложных сценариев, требующих специфической настройки, Selenide может оказаться менее гибким по сравнению с Selenium.
- **Зависимость от обновлений:** Так как Selenide использует WebDriver, его стабильная работа может зависеть от версий драйвера и браузера.

Технические особенности Selenium

1. Управление браузером:

- Selenium использует специфические драйверы для каждого браузера:
  - **ChromeDriver**: Управление Google Chrome.
  - **GeckoDriver**: Управление Mozilla Firefox.
  - **IEDriverServer**: Управление Internet Explorer.
  - **EdgeDriver**: Управление Microsoft Edge.
- Драйверы обеспечивают связь между тестами и браузерами через JSON Wire Protocol или W3C WebDriver Protocol.

2. Синхронизация:

- **Implicit Waits**: Задает глобальное время ожидания для поиска элементов.
- **Explicit Waits**: Условия ожидания, основанные на конкретных состояниях элементов (появление, исчезновение, кликабельность).
- **Fluent Waits**: Настраиваемые условия ожидания с возможностью указания времени, частоты и исключений.

3. Интеграция:

- **Инструменты CI/CD**: Поддержка интеграции с Jenkins, GitLab CI, TeamCity для автоматизированного выполнения тестов.
- **Тестовые фреймворки**: JUnit, TestNG, PyTest и другие, обеспечивающие удобное выполнение и управление тестами.

4. Архитектура:

- Модель клиент-сервер: WebDriver клиент посылает команды драйверу, который управляет браузером.
- **Selenium Grid**: Возможность распределенного выполнения тестов на различных машинах и операционных системах. Используется для параллельного тестирования и проверки кроссбраузерной совместимости.

Особенности Selenide

1. Автоматические ожидания:

- Selenide автоматически ждет, когда элемент станет видимым, кликабельным или исчезнет, что исключает необходимость ручного задания ожиданий.
- Умное управление синхронизацией позволяет избежать ложных срабатываний тестов из-за задержек в загрузке страницы.

2. Управление элементами:

- Selenide предлагает упрощенные методы для взаимодействия с элементами страницы:
  - `$("#elementId").click()` – клик по элементу.
  - `$("#elementId").setValue("text")` – ввод текста в поле.
  - `$("#elementId").shouldHave(text("Hello"))` – проверка наличия текста в элементе.
- Поддержка CSS и XPath селекторов, что позволяет гибко искать и обрабатывать элементы.

3. Скриншоты и отчёты:

- Автоматическое создание скриншотов при ошибках.
- Интеграция с Allure для генерации детализированных отчетов.
- Поддержка видеозаписи выполнения тестов (с помощью сторонних библиотек).

4. Архитектура:

- Selenide поддерживает работу в многопоточной среде, что позволяет запускать тесты параллельно.

- Возможность работы в headless-режиме, что ускоряет выполнение тестов и снижает нагрузку на систему.
- Интеграция с Gradle, Maven, JUnit и TestNG для удобного управления проектом и зависимостями.

Практическое сравнение кода

Пример на Selenium:

```
WebDriver driver = new ChromeDriver();
driver.get("https://example.com");
WebElement element = driver.findElement(By.id("elementI
element.click();
WebElement result = driver.findElement(By.id("resultId"
assertTrue(result.getText().contains("Success"));
driver.quit();

open("https://example.com");
$("#elementId").click();
$("#resultId").shouldHave(text("Success"));


#### Слайд 9: Как выбрать инструмент?
```markdown

<div style="page-break-before: always;"></div>

# Как выбрать инструмент?

**Критерии выбора:**

1. **Проект и его масштаб:**
   - Для небольших и средних проектов с типовыми сценар
   - Для крупных и сложных проектов, требующих гибкости

2. **Необходимость в гибкости:**
   - Если требуется тонкая настройка и кастомизация тес
```
```

```
   - Для более простых и прямолинейных сценариев – Sele

3. **Обучение и поддержка:**
   - Selenide легче изучить и освоить благодаря простом
   - Selenium предоставляет больше возможностей для обу

4. **Среда выполнения и инфраструктура:**
   - Если проект требует кроссбраузерного тестирования
   - Selenide подходит для однопоточных тестов и просто


<div style="page-break-before: always;"></div>

# Заключение и вопросы

**Основные выводы:**
- Selenium и Selenide имеют свои преимущества и недоста
- Выбор инструмента зависит от потребностей проекта, те
- Для быстрого старта и простых тестов Selenide являетс

**Рекомендации:**
- Использовать Selenide для небольших проектов с типовы
- Selenium – для крупных проектов, требующих максимальн

**Вопросы и ответы:**
- Ответы на вопросы участников вебинара.
- Обсуждение возможных кейсов и практического применени
```