Инструкция.

Инструкция по созданию проекта для автоматизированного тестирования.

Автоматизированное тестирование WEB-приложений.

22 страниц

Дата: 02.03.2017

Версия: 0.1

Статус: Создание

# **Список изменений**

|  |  |  |  |
| --- | --- | --- | --- |
| Версия | Дата | Изменение | Кем сделано |
| 0.1 | 02.03.2017 | Создан | Нечкин С.С. |

Аннотация

В данном документе приведены сведения по установке и настройке среды для автоматизированного тестирования.

Содержание

[**Список изменений** 2](#_Toc476245916)

[1 ОБЩИЕ СВЕДЕНИЯ 5](#_Toc476245917)

[2 Создание проекта для автоматизированного тестирования. 6](#_Toc476245918)

[3 Установка и настройка Java 7](#_Toc476245919)

[2.1 Инсталляция Java 7](#_Toc476245920)

[2.2 Настройка переменной среды 7](#_Toc476245921)

[2.3. Проверка установки Java 8](#_Toc476245922)

[4 Установка среды разработки Eclipse 9](#_Toc476245923)

[3.1 Инсталяция Eclipse. 9](#_Toc476245924)

[5 Установка и настройка Gradle 10](#_Toc476245925)

[4.1. Инсталяция Gradle 10](#_Toc476245926)

[4.2. Установка переменой среды для Gredle. 10](#_Toc476245927)

[4.3. Проверка установки Gradle 10](#_Toc476245928)

[4.4. Установка плагина Gradle в среде разработки Eclips. 11](#_Toc476245929)

[4.5. Создание проекта Gradle в среде разработки Eclipse 11](#_Toc476245930)

[4.6. Настройка Gradle-проекта в среде разработки Eclipse 12](#_Toc476245931)

[4.7 Запуск Gradle проекта из Eclipse. 13](#_Toc476245932)

[6 Установка и настройка Selenide 14](#_Toc476245933)

[6.1. Добавление зависимости для Selenide и Junit в Gradle проекте. 14](#_Toc476245934)

[6.2. Создание простого теста на Selenide. 14](#_Toc476245935)

[6.3. Изменение запускаемого браузера на «Google Crome». 15](#_Toc476245936)

[6.4. Примеры вызова функция Selenide 16](#_Toc476245937)

[7 Установка и настройка Allure 17](#_Toc476245938)

[7.1 Инсталляция Allure 17](#_Toc476245939)

[7.2 Настройка переменной среды 17](#_Toc476245940)

[7.3 Подключение зависимости для Allure в Gradle-проекте 17](#_Toc476245941)

[7.3 Установка аннотаций для отчёта Allure в проекте c Selenide 17](#_Toc476245942)

[7.3.1 @Title 18](#_Toc476245943)

[7.3.2 @Description 18](#_Toc476245944)

[7.3.3 @Severity 18](#_Toc476245945)

[7.3.4 @Features и @Stories 19](#_Toc476245946)

[7.3.5 @Step 19](#_Toc476245947)

[7.3.6 @Attachment 20](#_Toc476245948)

1. ОБЩИЕ СВЕДЕНИЯ

Инструкция содержит рекомендации по созданию проекта для автоматизированного тестирования WEB-приложений.

В документе кратко описаны шаги по установке и настройке следующего ПО:

**Java SE** - Набор спецификаций и соответствующей документации для языка Java, описывающей архитектуру серверной платформы для задач средних и крупных предприятий.

**Eclipse** - Свободная интегрированная среда разработки модульных кроссплатформенных приложений. Развивается и поддерживается Eclipse Foundation.

**Gradle** - Система автоматической сборки, построенная на принципах Apache Ant и Apache Maven, но предоставляющая DSL на языке Groovy вместо традиционной XML-образной формы представления конфигурации проекта.

**Selenide** - Фреймворк для автоматизированного тестирования веб-приложений на основе Selenium WebDriver

**Allure** - Фреймворк от Яндекса для создания простых и понятных отчётов по автоматизированному тестированию.

1. Создание проекта для автоматизированного тестирования.

Для создания проекта требуется выполнить пошаговую установку и настройку следующий программных продуктов:

1. Java SE
2. Eclipse
3. Gradle
4. Selenide
5. Allure

Шаги по установке и настройке перечисленных продуктов описаны ниже.

1. Установка и настройка Java

## 2.1 Инсталляция Java

С сайта ORACLE скачать инсталлятор Java SE по адресу:

<http://java.sun.com/javase/downloads>

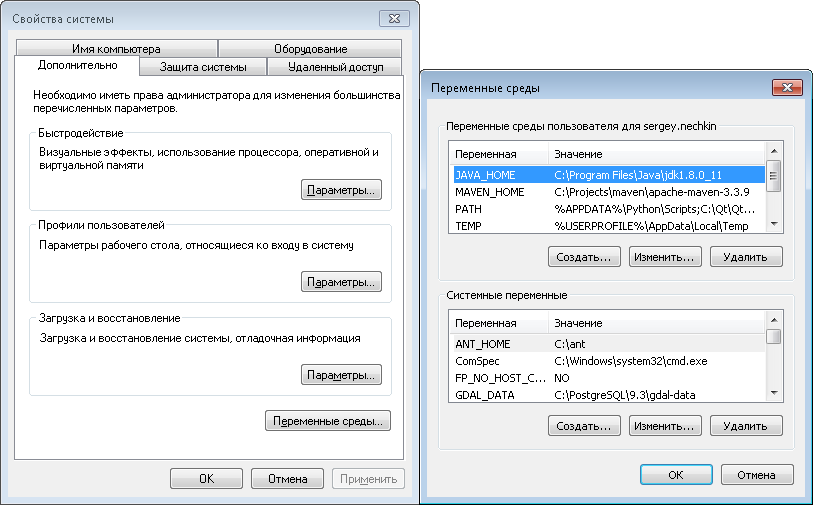
<http://www.oracle.com/technetwork/java/index-jsp-138363.html>

Выполнить установку.

## 2.2 Настройка переменной среды

Пример для Windows 7:

1. Открыть меню «Пуск»
2. Выбрать пункт «Панель управления»
3. В окне «Все элементы панели управления» кликнуть «Система»
4. В открывшемся окне кликнуть ссылку «Изменить параметры»
5. В окне «Свойства системы» открыть вкладку «Дополнительно»
6. Нажать на кнопку «Переменные среды»
7. В качестве имени переменной указать «JAVA\_HOME» (или «JRE\_HOME», если устанавливается JRE), а в качестве значения — путь к установленной JVM (Путь к установленной Java SE). Путь не должен обрамляться кавычками, даже если содержит пробелы.
8. В блоке «Системные переменные» изменить переменную Path – после точки с запятой добавить строку %JAVA\_HOME%\bin



## 2.3. Проверка установки Java

Открыть компаундную строку(CMD)

Выполнить команду java -version.

На экране должна отобразится информация о версии установленной JVM (Java машины), если этого не произошло, то необходимо проверить пути указанные в переменных среды.

1. Установка среды разработки Eclipse

## 3.1 Инсталяция Eclipse.

Скачать установщик Eclipse по адресу:

<http://www.eclipse.org/downloads>

Запустить и установить Eclipse в любую директорию.

Выбрать установку «Eclipse IDE for Java EE Developers»

1. Установка и настройка Gradle

## 4.1. Инсталяция Gradle

Скачать архив Gredle по адресу

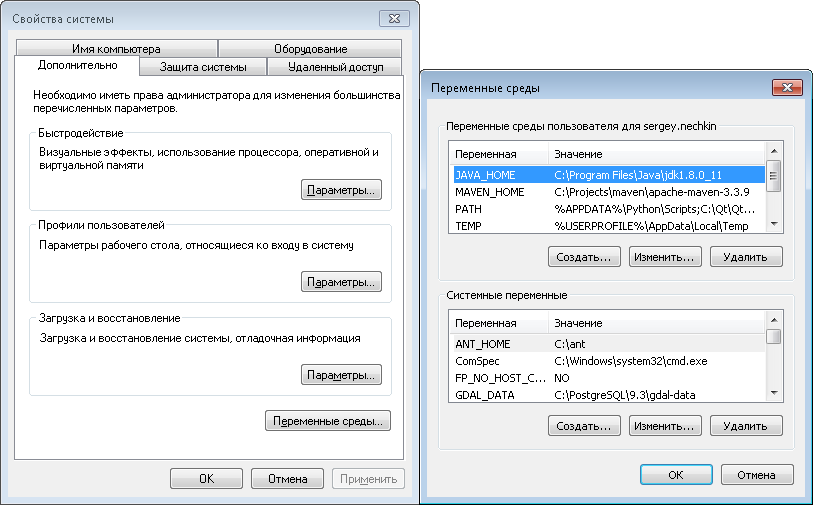
<https://gradle.org>

Извлечь файлы в любую директорию.

## 4.2. Установка переменой среды для Gredle.

Пример для Windows 7:

1. Открыть меню «Пуск»
2. Выбрать пункт «Панель управления»
3. В окне «Все элементы панели управления» кликнуть «Система»
4. В открывшемся окне кликнуть ссылку «Изменить параметры»
5. В окне «Свойства системы» открыть вкладку «Дополнительно»
6. Нажать на кнопку «Переменные среды»
7. В качестве имени переменной указать «GRADLE\_HOME», а в качестве значения — путь к директории с извлечёнными файлама Gredle. Путь не должен обрамляться кавычками, даже если содержит пробелы.
8. В блоке «Системные переменные» изменить переменную Path – после точки с запятой добавить строку %GRADLE\_HOME%\bin



## 4.3. Проверка установки Gradle

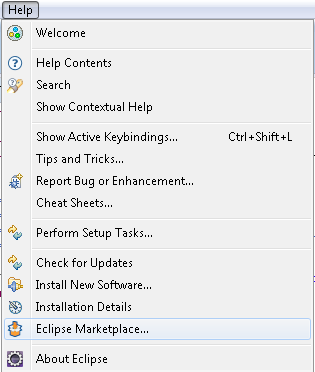
Открыть компаундную строку(CMD)

Выполнить команду gradle.

На экране должна отобразится строка "BUILD SUCCESSFUL", если этого не произошло, то необходимо проверить пути указанные в переменных среды.

## 4.4. Установка плагина Gradle в среде разработки Eclips.

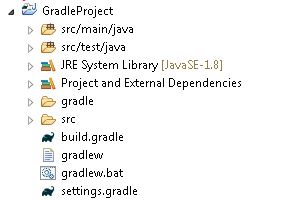
1. Запустить Eclipse.
2. Перейти в пункт меню Help
3. Выбрать «Eclipse Marketplace...»



1. Найти плагин "Buildship Gradle Integration"
2. Выполнить установку плагина.

## 4.5. Создание проекта Gradle в среде разработки Eclipse

1. Запустить Eclipse c установленным плагином "Buildship Gradle Integration".
2. Перейти в пункт меню «File»
3. Выбрать пункт «new»
4. Выбрать подпункт «Other...»
5. В открывшемся окне выбрать пункт «Gradle» => «Gradle Project»
6. Нажать кнопку «Next»
7. Заполнить поле «Project name»
8. Нажать кнопку «Finish»
9. Создан новый проект. В проекте создана структура файлов и каталогов как на картинке:



## 4.6. Настройка Gradle-проекта в среде разработки Eclipse

1. Запустить Eclipse c созданным Gradle проектом.
2. В проекте найти и открыть файл «Build.Gradle».
3. В файле добавить следующие строки, если отсутсвуют:

apply plugin: 'java'

apply plugin: 'eclipse'

apply plugin: 'application' // Добавляет зависимые библиотеки в Jar файл

mainClassName = 'MainTest' // Указатель на стартовый класс

// Задаёт кодировку для выполняемых файлов

compileJava.options.encoding = 'UTF-8'

tasks.withType(JavaCompile) {

options.encoding = 'UTF-8'

}

jar { // Имя и версия jar файла собранного проекта

baseName = 'hpsmautotest'

version = '0.1.0'

}

repositories { // Разрешать зависимости из Maven Central репозитория

mavenLocal()

mavenCentral()

jcenter()

}

dependencies { // Добавление зависимостей

// Пример подключения библиотеки: compile "joda-time:joda-time:2.2"

}

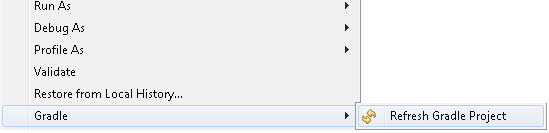
// Позволяют запускать сборку с Gradle без необходимости установки самого Gradle в систему где запускается проект Запустите следующую команду для загрузки и инициализации wrapper-скриптов: gradle wrapper Теперь каждый, кто клонирует ваш проект, сможет его собрать точно таким же способом ( gradlew build )

task wrapper(type: Wrapper) {

gradleVersion = '1.11'

}

1. Сохранить файл
2. Вызвать подменю на корне проекта (кликнуть правой кнопкой мыши).
3. Раскрыть пункт меню «Gradle» и выбрать пункт «Refresh Gradle Project»



1. Все подкаченные библиотеки будут загружены в проект

## 4.7 Запуск Gradle проекта из Eclipse.

На корневой папке проекта кликнуть правой кнопкой

Выбрать Run AS => Java Application

В Открывшемся окне выбрать build

Нажать кнопку "Ок"

Или запустить как JUnit Test

1. Установка и настройка Selenide

## 6.1. Добавление зависимости для Selenide и Junit в Gradle проекте.

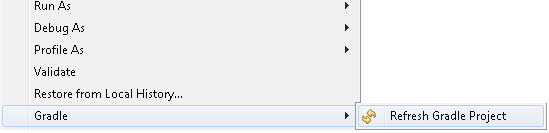
Junit - библиотека для модульного тестирования программного обеспечения на языке Java. Необходимо для работы selenide.

1. Запустить Eclipse c созданным Gradle проектом.
2. В проекте найти и открыть файл «Build.Gradle».
3. В блок dependencies добавить строку:

compile group: 'com.codeborne', name: 'selenide', version: '4.3'

compile group: 'junit', name: 'junit', version: '4.12'

1. Сохранить файл
2. Вызвать подменю на корне проекта (кликнуть правой кнопкой мыши).
3. Раскрыть пункт меню «Gradle» и выбрать пункт «Refresh Gradle Project»



1. Все подкаченные библиотеки будут загружены в проект

## 6.2. Создание простого теста на Selenide.

1. Запустить Eclipse c созданным Gradle проектом.
2. в каталоге проекта src\test\java создать новый пакет с названием, например тестируемой системы.
3. Внутри пакета создать класс 'MainTest'. Если название класса задано иное, то в файле «Build.Gradle» следует изменить значение для «mainClassName».
4. В файле созданного класса прописать две строки (подключить библиотеки):

import static com.codeborne.selenide.Selenide.\*;

import org.junit.Test;

1. В теле класса создать новую фецкцию. Напрмимер:

public void testOpenYandex() {

}

1. Внутри этой функции добавить вызов функции из Selenide.

Направмер: open("http://ya.ru");

Эта команда задаёт ссылку в адресной строке браузера и переходит по ней.

1. Перед созданной функцией «testOpenYandex» добавить аннотацию @Test.
2. Сохранить файл.
3. В итоге должен получится следующий код:

package hpsm;

import static com.codeborne.selenide.Selenide.\*;

import org.junit.Test;

public class MainTest {

@Test

public void testAuthorization() {

open("http://ya.ru");

}

}

1. После запуска будет открыто окно браузера FireFox и выполнен переход по ссылке «http://ya.ru»

## 6.3. Изменение запускаемого браузера на «Google Crome».

1. Загрузить драйвер для управления браузером «Crome» по адресу

<https://sites.google.com/a/chromium.org/chromedriver/downloads>

1. Запустить Eclipse c созданным Gradle проектом.
2. В проекте создать директорию "resources" в каталоге "src/test/java"
3. В директории "resources" создать директорию "web\_drivers"
4. Загруженный драйвер разархивировать в директорию "web\_drivers"
5. В проекте открыть класс и функцию которые будут запущены первыми при старте.
6. В функцию добавить конфигурацию:

final String pathToChromedriverExe = "src" + File.separator +

"test" + File.separator +

"java" + File.separator +

"resources" + File.separator +

"web\_drivers" + File.separator +

"chromedriver.exe";

Configuration.chromeSwitches = pathToChromedriverExe;

Configuration.browser = "chrome";

Configuration.browserSize = "800x600"; //задать размер окна браузера

//System.setProperty("webdriver.chrome.driver", pathToChromedriverExe); //альтернативный вариант

//System.setProperty("selenide.browser", "Chrome"); //альтернативный вариант

1. Сохранить файл. Теперь при запуске будет открываться браузер Chrome.

## 6.4. Примеры вызова функция Selenide

open("http://ya.ru");

$("#idname").setValue("Ввести этот текст в элемент").pressEnter();

// $$ - поиск во всей странице

// exactTexts - Проверяет количество и порядок ткстов в массиве элементов

$$(#idnamelist).shouldHave(exactTexts("Текст1", "Текст2"))

//Добавление новой команды к Selenide

public SeleniumElement doubleClick(SeleniumElement element) {

action().doubleClick(element).proform();

return element;

}

1. Установка и настройка Allure

## 7.1 Инсталляция Allure

Загрузить архив c Allure по адресу:

<https://github.com/allure-framework>

Распаковать архив в любую директорию.

Запуск сформированного отчёта выполняется через команду allure report open

## 7.2 Настройка переменной среды

Пример для Windows 7:

1. Открыть меню «Пуск»
2. Выбрать пункт «Панель управления»
3. В окне «Все элементы панели управления» кликнуть «Система»
4. В открывшемся окне кликнуть ссылку «Изменить параметры»
5. В окне «Свойства системы» открыть вкладку «Дополнительно»
6. Нажать на кнопку «Переменные среды»
7. В блоке «Системные переменные» изменить переменную Path – после точки с запятой добавить путь к директории с Allure

## 7.3 Подключение зависимости для Allure в Gradle-проекте

1. Запустить Eclipse c созданным Gradle проектом.
2. В проекте найти и открыть файл «Build.Gradle».
3. В блок dependencies добавить строку:

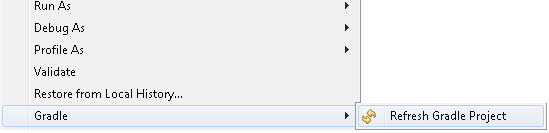
compile group: 'ru.yandex.qatools.allure', name: 'allure-maven-plugin', version: '2.5'

compile group: 'ru.yandex.qatools.allure', name: 'allure-junit-adaptor', version: '1.5.2'

compile group: 'junit', name: 'junit', version: '4.12'

compile group: 'org.aspectj', name: 'aspectjweaver', version: '1.8.10'

1. Сохранить файл
2. Вызвать подменю на корне проекта (кликнуть правой кнопкой мыши).
3. Раскрыть пункт меню «Gradle» и выбрать пункт «Refresh Gradle Project»



1. Все подкаченные библиотеки будут загружены в проект

## 7.3 Установка аннотаций для отчёта Allure в проекте c Selenide

Для отражения данных в отчете по выполненному автоматизированному тесту, следует прописать аннотации над выполняемыми функциями(методам) или классами.

Ниже приведены некоторые аннотации:

## 7.3.1 @Title

**@Title**

Задаёт имена сьютов и кейсов в Allure отчёте.

Класс или метод будет выглядеть следующим образом:

@Title("This is our cool test suite")

public class SimpleTest {

@Title("First cool check")

@Test

public void firstSimpleTest() {

assertTrue("Result not equals to 4", 2 \* 2 == 4);

}

}

## 7.3.2 @Description

**@Description**

Добавляет описание к кейсам и сьютам

Пример:

@Title("This is our cool test suite")

@Description("In this cool suite we will test only cool features")

public class SimpleTest {

@Title("First cool check")

@Description("In this cool test we will check cool thing")

@Test

public void firstSimpleTest() {

assertTrue("Result not equals to 4", 2 \* 2 == 4);

}

}

## 7.3.3 @Severity

**@Severity**

Позволяет задать насколько важен тест

Позволяет устанавливать одно из значений: BLOCKER, CRITICAL, NORMAL, MINOR, TRIVIAL.

Пример:

@Severity(SeverityLevel.BLOCKER)

## 7.3.4 @Features и @Stories

**@Features** и **@Stories**

Группирует сьюты и тесты в отчёте.

Например, тесты разделены на 2 разные функциональности. Если добавить @Features и @Stories к тестовому классу или тестовому методу, то при генерации отчета, перейдя на главной странице в боковом меню на таб Behaviors, можно будет увидеть небольшой график, разделенный по фичам или историям. При выборе любой из историй - будут показаны результаты замапленных на нее тестов. Каждая из фич может содержать в себе сколько угодно историй.

## 7.3.5 @Step

**@Step**

Задаёт наименование шага в тесте.

Эту аннотацию можно прописывать для любой выполняемой функции Selenide

Пример:

@Title("This is our cool test suite")

@Description("In this cool suite we will test only cool features")

public class SimpleTest {

@Severity(SeverityLevel.BLOCKER)

@Title("First cool check")

@Description("In this cool test we will check cool thing")

@Test

public void firstSimpleTest() {

checkResult(getDivisionResult(4, 2), 2);

}

@Step("Проверка результата вычислений")

private void checkResult(int actualResult, int expectedResult) {

assertTrue("Actual result("+ actualResult+") not equals to expected("+expectedResult+")", actualResult == expectedResult);

}

@Step("{0}")

private void log(String value) {

//empty method

}

@Step("Получение результата деления числа {0} на {1}")

private int getDivisionResult(int firstInt, int secondInt) {

checkNotZero(secondInt);

return firstInt / secondInt;

}

@Step("Проверка делителя")

private void checkNotZero(int intValue) {

log("Значение делителя: " + intValue);

assertTrue("Делитель равен 0", intValue != 0);

}

}

* checkResult - метод, проверяющий результат вычислений
* getDevisionResult - метод, возвращающий результат деления 2-х числел типа int
* checkNotZero - метод, проверяющий, что делитель не равен нулю(мы же не хотим увидеть наш тест в статусе BROKEN)
* log - это вспомогательный пустой метод, созданный для возможности логирования в отчете некоторых желаемых значений в виде шагов.
* Обозначения "{0}" и "{1}" внутри значения аннотации - это обращение к параметрам метода.

## 7.3.6 @Attachment

**@Attachment**

Позволяет добавить в отчёт картинки, тексты и html страницы.

Аннотация @Attachment имеет 2 параметра:

value - имя, с которым будет сохранено вложение. Тут так же возможно использование обращения к параметрам метода через индекс.

type - это mime type аттача.

Пример:

@Attachment(value = "{0}", type = "text/plain")

public static String saveTextLog(String attachName, String message) {

return message;

}

Теперь немного упростим тест(уменьшим кол-во шагов) и добавим прикрепление аттача:

@Title("This is our cool test suite")

@Description("In this cool suite we will test only cool features")

public class SimpleTest {

@Title("First cool check")

@Description("In this cool test we will check cool thing")

@Test

public void firstSimpleTest() {

checkResult(4 / 2, 2);

}

@Step("Проверка результата вычислений")

private void checkResult(int actualResult, int expectedResult) {

saveTextLog("Simple text attach", "This is sparta!!!");

assertTrue("Actual result(" + actualResult + ") not equals to expected(" + expectedResult + ")", actualResult == expectedResult);

}

@Attachment(value = "{0}", type = "text/plain")

public static String saveTextLog(String attachName, String message) {

return message;

}

}

**Пример приложения картинки:**

@Attachment(value = "{0}", type = "image/png")

public static byte[] saveImageAttach(String attachName) {

try {

URL defaultImage = SimpleTest.class.getResource("/allure.png");

File imageFile = new File(defaultImage.toURI());

return toByteArray(imageFile);

} catch (Exception e) {

e.printStackTrace();

}

return new byte[0];

}

private static byte[] toByteArray(File file) throws IOException {

return Files.readAllBytes(Paths.get(file.getPath()));

}

Для корректной работы с изображением – нужно вернуть массив байтов.

Пример:

@Test

public void firstSimpleTest() {

checkResult(4 / 2, 2);

}

@Step("Проверка результата вычислений")

private void checkResult(int actualResult, int expectedResult) {

saveImageAttach("Image attach");

assertTrue("Actual result(" + actualResult + ") not equals to expected(" + expectedResult + ")", actualResult == expectedResult);

}

**@Attachment с HTML страницей**

Пример:

@Attachment(value = "{0}", type = "text/html")  
public static byte[] saveHtmlAttach(String attachName) {  
 try {  
 URL defaultImage = SimpleTest.class.getResource("/test.html");  
 File imageFile = new File(defaultImage.toURI());  
 return toByteArray(imageFile);  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
 return new byte[0];  
}

Пример по аналогии с аттачем текста:

@Attachment(value = "{0}", type = "text/html")  
public static String saveHtmlAttach(String attachName, String html) {  
 return html;  
}