Creating A Walking Skeleton

- Write and Automate First Acceptance Test
- Automate Build and Package
- Automate Deploy
- Implement Feature
- Automate Static Analysis
- Automate Code Coverage & Integration Test

1. Add an abstract JUnit test class called `AbstractLogin` to the common package:

```
import static org.junit.Assert.assertEquals;

import java.io.IOException;
import java.net.MalformedURLException;

import org.junit.After;
import org.junit.Before;
import org.junit.Test;

import com.gargoylesoftware.htmlunit.FailingHttpStatusCodeException;
import com.gargoylesoftware.htmlunit.WebClient;
import com.gargoylesoftware.htmlunit.html.HtmlPage;

public abstract class AbstractLogin
{
	private WebClient webClient;  // HTML Unit

	@Before
	public void setUp() throws Exception
	{
		webClient = new WebClient();
	}

	@After
	public void tearDown() throws Exception
	{
		webClient.closeAllWindows();
	}

	@Test
	public void
LoginPageIsDisplayedWhenAccessingTheApplicationForTheFirstTime()
			throws FailingHttpStatusCodeException,
				MalformedURLException, IOException
	{
		final HtmlPage page = webClient.getPage(getAppUrl());
		assertEquals("Login Page", page.getTitleText());
	}

	protected abstract String getAppUrl();
}
```

2. Add a new class call `LoginTest` to the `integration` package:

```
import walkingskeleton.common.AbstractLogin;

public class LoginTest extends AbstractLogin
{
	private final static String APP_URL =
"http://localhost:8080/WalkingSkeleton";

	@Override
	protected String getAppUrl()
	{
		return APP_URL;
	}
}
```

3. Add `compile-test` target to `build.xml`:

```
<import file="${buildfiles.dir}/compile.xml"/>
<target name = "compile-test" depends = "init">
		<antcall target="compile.compile-test"/>
</target>

<target name = "ci-build" depends = "clean, compile-test"/>
```

4. Add `acceptance-test` target to `build.xml`:

```
<!-- Acceptance Test -->
<import file="${buildfiles.dir}/test.xml"/>
<target name = "acceptance-test" depends = "compile-test">
	<antcall target="test.run-acceptance-test"/>
</target>

<target name = "ci-build" depends = "clean, acceptance-test"/>
```

5. Add a package called `walkingskeleton.sl.logging` to the `src` folder.

6. Add a class called `Log4jInitListener` to the logging package:

```
import javax.servlet.ServletContextEvent;
import javax.servlet.ServletContextListener;

import org.apache.log4j.Logger;
import org.apache.log4j.PropertyConfigurator;

public class Log4jInitListener implements ServletContextListener
{
    private static final Logger log = Logger.getLogger(Log4jInitListener.class);

    @Override
    public void contextInitialized(ServletContextEvent ce)
    {
        final String prefix = ce.getServletContext().getRealPath("/");
        final String file = ce.getServletContext().getInitParameter("log4j-init-
file");
        if ( file != null )
        {
            final String path = prefix + "/" + file;
            PropertyConfigurator.configure(path);
            log.info("Log4J configured by file " + path);
        }
    }

      @Override
      public void contextDestroyed(ServletContextEvent arg0)
      {     /* No op. */ }
}
```

7. Add `compile` & `compress` targets:

```
<import file="${buildfiles.dir}/compile.xml"/>
<target name = "compile" depends = "init">
      <antcall target="compile.compile"/>
</target>

<import file="${buildfiles.dir}/compress.xml"/>
<property name = "jar.file" value="${build.lib.dir}/walkingskeleton.jar"
/>
<target name = "compress" depends="compile">
      <antcall target="compress.compress"/>
</target>


<target name = "ci-build" depends = "clean, compress, run-acceptance-
test"/>
```

8. Add package target:

```
<import file="${buildfiles.dir}/package.xml"/>
<property name = "war.file" value="${build.dir}/${webapp.name}.war"/>
<target name = "package" depends="compress">
      <antcall target="package.package"/>
</target>
```

9. Add `deploy` target:

```
<import file="${buildfiles.dir}/deploy.xml"/>
<target name = "deploy" depends="package">
      <antcall target="deploy.deploy-webapp"/>
</target>
```

10. Update run acceptance test target:

```
<target name = "run-acceptance-test" depends = "deploy,compile-test">
```

And ci-built task:

```
<target name = "ci-build" depends = "undeploy-webapp, clean, run-
acceptance-test"/>
```

11. Add spring `web.xml` configuration:

```
<listener>
      <listener-class>
            org.springframework.web.context.ContextLoaderListener
      </listener-class>
</listener>

<context-param>
      <param-name>contextConfigLocation</param-name>
      <param-value>
            /WEB-INF/security-app-context.xml
      </param-value>
</context-param>

<filter>
      <filter-name>springSecurityFilterChain</filter-name>
      <filter-class>
            org.springframework.web.filter.DelegatingFilterProxy
      </filter-class>
</filter>

<filter-mapping>
      <filter-name>springSecurityFilterChain</filter-name>
      <url-pattern>/*</url-pattern>
</filter-mapping>
```

12. Copy `security-app-context.xml` to `WEB-INF direcotry`.

Run test and see it pass.

13. Add remaining tests to `AbstractLogin`:

```
import static org.junit.Assert.fail;
...

@Test
public void BadUsernameResultsInErrorMessage()
{
	fail("Not Implemented");
}

@Test
public void BadPasswordResultsInErrorMessage()
{
	fail("Not Implemented");
}

@Test
public void CorrectCredentialsDisplayHomePage()
{
	fail("Not Implemented");
}
```

14. Add test constants to `AbstractLogin`:

```
private static final String USERNAME           = "admin";
private static final String PASSWORD           = "admin";

private static final String BAD_USERNAME       = "bad username";
private static final String BAD_PASSWORD       = "bad password";

private static final String ERROR_MSG          = "Reason: Bad credentials";
```

15. Add `login` helper method to `AbstractLogin`:

```
private HtmlPage login(String username, String password)
{
    try
    {
      final HtmlPage page = webClient.getPage(getAppUrl());
      final HtmlForm form = page.getFormByName("f");

      if (username != null)
      {
            final HtmlTextInput usernameField =
form.getInputByName("j_username");
            usernameField.setValueAttribute(username);
      }

      if (password != null)
      {
            final HtmlPasswordInput passwordField =
form.getInputByName("j_password");
            passwordField.setValueAttribute(password);
      }

      final HtmlSubmitInput submitBtn = form.getInputByName("submit");
      return submitBtn.click();
    }
    catch(Exception e)
    {
      throw new RuntimeException(e);
    }
}
```

16. Add implementation for next test:

```
import static org.junit.Assert.assertThat;
import static org.junit.matchers.JUnitMatchers.containsString;
...

inal HtmlPage page = login(BAD_USERNAME, PASSWORD);
assertThat(page.asXml(), containsString(ERROR_MSG));
```

17. Add final test:

```
final HtmlPage page = login(USERNAME, PASSWORD);
assertEquals("Walking Skeleton", page.getTitleText());
assertThat(page.asXml(), containsString("Welcome!"));
```

18. Add static code analysis to `build.xml`:

```
<!-- Statis Analysis -->
<import file="${buildfiles.dir}/check.xml"/>

<target name = "ci-build" depends = "undeploy-webapp, clean, acceptance-
test, check"/>
```

19. Add:

```
@SuppressWarnings({"PMD.VariableNamingConventions"})
```

to `Login4JInitListener.java`. Add:

```
@SuppressWarnings({"PMD.TestClassWithoutTestCases"})
```

to `LoginTest.java`.

20. Fix Eclipse warnings:

Java Compiler->Errors / Warnings -> Enable Project Specific Settings->Unhandled token in @SuppressWarnings->Ignore.

21. Add integration `LoginTest.java`:

```
import net.purpletube.embeddedtomcat.EmbeddedTomcat;
import net.purpletube.embeddedtomcat.ServletContainer;
import net.purpletube.embeddedtomcat.StartStopTimer;
import walkingskeleton.common.AbstractLogin;

@SuppressWarnings({"PMD.TestClassWithoutTestCases"})
public class LoginTest extends AbstractLogin
{
	private final static int PORT 			= 8890;
	private final static String CONTEXT 	= "/WalkingSkeleton";
	private final static String WAR_PATH 	= "./war";

	private final ServletContainer server
			= new StartStopTimer(
					new EmbeddedTomcat(CONTEXT, WAR_PATH,
PORT));

	@Override
	public void setUp() throws Exception
	{
		server.start();
		super.setUp();
	}

	@Override
	public void tearDown() throws Exception
	{
		super.tearDown();
		server.stop();
	}

	@Override
	protected String getAppUrl()
	{
		return server.getUrl();
	}
}
```

22. Add code coverage task:

```
<!-- Coverage -->
<import file="${buildfiles.dir}/coverage.xml"/>
<target name = "coverage" depends = "compile-test">
	<antcall target="coverage.coverage"/>
</target>

<target name = "ci-build" depends = "undeploy-webapp, clean, acceptance-
test, check, coverage"/>
```

23. Add context parameter for log4j properties file and listener:

```
<context-param>
      <param-name>log4j-init-file</param-name>
      <param-value>
            WEB-INF/log4j.properties
      </param-value>
</context-param>

<listener>
      <listener-class>
            walkingskeleton.sl.logging.Log4jInitListener
      </listener-class>
</listener>
```