Spring @ContextConfiguration how to put the right location for the xml

In our project we are writting a test to check if the controller returns the right modelview

```
@Test
    public void controllerReturnsModelToOverzichtpage()
    {
        ModelAndView modelView = new ModelAndView();
        KlasoverzichtController controller = new KlasoverzichtController();
        modelView = controller.showOverzicht();

        assertEquals("Klasoverzichtcontroller returns the wrong view ",
modelView.getViewName(), "overzicht");
    }
```

This returns the exception null.

We are now configuring the @contextconfiguration but we don't know how to load the right xml who is located at src\main\webapp\root\WEB-INF\root-context.xml

```
@RunWith(SpringJUnit4ClassRunner.class)
@ContextConfiguration
public class TestOverzichtSenario{
....
```

This documentation isn't clear enough to understand

Any suggestion on how to make sure the contextannotation loads the right xml?

Edit v2

I copied the configuration .xml files from the webINF folder to

```
src\main\resources\be\..a bunch of folders..\configuration\*.xml
```

and changed the web.xml in webinf to

```
<param-name>contextConfigLocation</param-name>
<param-value>
            classpath*:configuration/root-context.xml
            classpath*:configuration/applicationContext-security.xml
        </param-value>
```

and now get the error

```
org.springframework.beans.factory.BeanDefinitionStoreException: IOException parsing XML
document from ServletContext resource [/WEB-INF/mvc-dispatcher-servlet.xml]; nested
exception is java.io.FileNotFoundException: Could not open ServletContext resource [/WEB-
INF/mvc-dispatcher-servlet.xml]

org.springframework.beans.factory.xml.XmlBeanDefinitionReader.loadBeanDefinitions(XmlBeanD


org.springframework.beans.factory.xml.XmlBeanDefinitionReader.loadBeanDefinitions(XmlBeanD


org.springframework.beans.factory.support.AbstractBeanDefinitionReader.loadBeanDefinitions


org.springframework.beans.factory.support.AbstractBeanDefinitionReader.loadBeanDefinitions


org.springframework.beans.factory.support.AbstractBeanDefinitionReader.loadBeanDefinitions


org.springframework.web.context.support.XmlWebApplicationContext.loadBeanDefinitions(XmlWel


org.springframework.web.context.support.XmlWebApplicationContext.loadBeanDefinitions(XmlWel


org.springframework.context.support.AbstractRefreshableApplicationContext.refreshBeanFacto


org.springframework.context.support.AbstractApplicationContext.obtainFreshBeanFactory(Abst
```

```
org.springframework.context.support.AbstractApplicationContext.refresh(AbstractApplicationC

org.springframework.web.servlet.FrameworkServlet.createWebApplicationContext(FrameworkServl

org.springframework.web.servlet.FrameworkServlet.createWebApplicationContext(FrameworkServl

org.springframework.web.servlet.FrameworkServlet.initWebApplicationContext(FrameworkServle

org.springframework.web.servlet.FrameworkServlet.initServletBean(FrameworkServlet.java:306

    org.springframework.web.servlet.HttpServletBean.init(HttpServletBean.java:127)
    javax.servlet.GenericServlet.init(GenericServlet.java:212)

com.springsource.insight.collection.tcserver.request.HttpRequestOperationCollectionValve.in

    org.apache.catalina.valves.ErrorReportValve.invoke(ErrorReportValve.java:102)
    org.apache.catalina.connector.CoyoteAdapter.service(CoyoteAdapter.java:293)
    org.apache.coyote.http11.Http11Processor.process(Http11Processor.java:849)

org.apache.coyote.http11.Http11Protocol$Http11ConnectionHandler.process(Http11Protocol.java

    org.apache.tomcat.util.net.JIoEndpoint$SocketProcessor.run(JIoEndpoint.java:379)
    java.util.concurrent.ThreadPoolExecutor$Worker.runTask(Unknown Source)
    java.util.concurrent.ThreadPoolExecutor$Worker.run(Unknown Source)
    java.lang.Thread.run(Unknown Source)
```

java unit-testing spring annotations maven

edited May 24 '16 at 0:28
abhi
1,081 1 10 25

asked Dec 7 '10 at 14:29
David
2,750 2 20 42

9 Answers

That's the reason not to put configuration into `webapp` .

As far as I know, there are no good ways to access files in `webapp` folder from the unit tests. You can put your configuration into `src/main/resources` instead, so that you can access it from your unit tests (as described in the docs), as well as from the webapp (using `classpath:` prefix in `contextConfigLocation`).

See also:

- 4.7 Application contexts and Resource paths

edited May 30 '12 at 17:17

answered Dec 7 '10 at 14:41
axtavt
194k 30 407 424

1 Do you have any documentation references what "classpath: " does? – David Dec 7 '10 at 17:51

@Dvd Prd: Updated – axtavt Dec 7 '10 at 18:20

thx , can you check my changes? – David Dec 7 '10 at 18:59

@Dvd: I guess you need `classpath:be/..a bunch of folders../configuration/root-context.xml` and `classpath:be/..a bunch of folders../configuration/applicationContext-security.xml` – axtavt Dec 7 '10 at 19:02

Our Tests look like this (using Maven and Spring 3.1):

```
@ContextConfiguration
(
  {
   "classpath:beans.xml",
   "file:src/main/webapp/WEB-INF/spring/applicationContext.xml",
   "file:src/main/webapp/WEB-INF/spring/dispatcher-data-servlet.xml",
   "file:src/main/webapp/WEB-INF/spring/dispatcher-servlet.xml"
  }
)
@RunWith(SpringJUnit4ClassRunner.class)
```

```
public class CCustomerCtrlTest
{
  @Resource private ApplicationContext m_oApplicationContext;
  @Autowired private RequestMappingHandlerAdapter m_oHandlerAdapter;
  @Autowired private RequestMappingHandlerMapping m_oHandlerMapping;
  private MockHttpServletRequest m_oRequest;
  private MockHttpServletResponse m_oResp;
  private CCustomerCtrl m_oCtrl;

// more code ....
}
```

answered Mar 17 '12 at 6:49

bernd

661 5 2

Thanks for the example – David Mar 17 '12 at 8:29

Thank you very much, it solves my problem! But do you think the approach: "file:path" is correct? – Andrew May 29 '13 at 11:37

This is a maven specific problem I think. Maven does not copy the files form `/src/main/resources` to the target-test folder. You will have to do this yourself by configuring the resources plugin, if you absolutely want to go this way.

An easier way is to instead put a test specific context definition in the `/src/test/resources` directory and load via:

```
@ContextConfiguration(locations = { "classpath:mycontext.xml" })
```

edited Jul 16 '13 at 12:23

answered Dec 7 '10 at 14:43

fasseg

12.9k 7 52 69

/src/main/resources should not be copied to the target/test-classes. Test classes – dan carter Jul 15 '13 at 20:58

4 /src/main/resources should **not** be copied to the target/test-classes folder. test-classes is for test resources, i.e. src/test/java and src/test/resources. src/main/resources gets copied to target/classes and target/classes is placed on the classpath when running tests, and thus any spring context files in src/main/resources can be loaded in unit tests as a classpath resource e.g. src/main/resources/spring/my-context.xml can be loaded as @ContextConfiguration({"/spring/my-context.xml"}) – dan carter Jul 15 '13 at 21:04

@Dan: I'd say, that if you can e.g. reuse (parts of) the config from `main` in the `test` context, it makes sense to let maven copy the existing files for the tests using the maven-resources-plugin. So saying "should **not**" be copied is a bit too harsh. Of course Maven does not do this by default, but if you have good reasons to do so yourself...why not? – fasseg Jul 16 '13 at 12:36

@fasseg: What @Dan meant is that you don't need to copy the config from `main` to the `test` context, because this config is already provided to the unit tests (because `target/classes` is placed on the classpath when running them). – ahmehri Sep 5 '14 at 8:45

Simple solution is

```
@ContextConfiguration(locations = { "file:src/main/webapp/WEB-INF/applicationContext.xml"
})
```

from spring forum

answered Jul 31 '13 at 7:19

Bala

1,904 5 18 29

We Use:

```
@ContextConfiguration(locations="file:WebContent/WEB-INF/spitterMVC-servlet.xml")
```

the project is a eclipse dynamic web project, then the path is:

```
{project name}/WebContent/WEB-INF/spitterMVC-servlet.xml
```

edited Mar 6 '14 at 10:19

Subhrajyoti Majumder

31.1k 10 52 86

answered Jan 16 '12 at 22:54

Grubhart

682 6 13

Suppose you are going to create a test-context.xml which is independent from app-context.xml for testing, put test-context.xml under /src/test/resources. In the test class, have the @ContextConfiguration annotation on top of the class definition.

```
@ContextConfiguration(locations = "/test-context.xml")
public class MyTests {
    ...
}
```

Spring document Context management

edited Aug 3 '13 at 4:16

answered Aug 3 '13 at 0:18

Dino Tw

1,412 2 18 35

Loading the file from: `{project}/src/main/webapp/WEB-INF/spring-dispatcher-servlet.xml`

```
@RunWith(SpringJUnit4ClassRunner.class)
@ContextConfiguration(locations = { "file:src/main/webapp/WEB-INF/spring-dispatcher-
servlet.xml" })
@WebAppConfiguration
public class TestClass {
    @Test
    public void test() {
         // test definition here..
    }
}
```

edited Oct 19 '15 at 6:37

answered Oct 17 '15 at 14:27

Arpit

8,742 5 37 64

@RunWith(SpringJUnit4ClassRunner.class)@ContextConfiguration(locations = {"/Beans.xml"}) public class DemoTest{}

answered Apr 16 '15 at 7:32

Deepu Surendran

187 1 3

Sometimes it might be something pretty simple like missing your resource file in test-classses folder due to some cleanups.

answered Dec 14 '16 at 15:36

gpushkas

9 2