**Пространство имен против параметров инициализации contextConfigLocation в web.xml**

Я читаю документацию для Spring MVC, и у меня есть вопрос относительно параметров init. Я использую Spring 3.2, если это имеет значение. В чем разница между contextConfigLocation и пространством имен? Является ли contextConfigLocation значением только для указания папок, где класс контекста может найти определение XML и атрибут пространства имен, предназначенный для указания имени файла?

<servlet>

<servlet-name>AppServlet</servlet-name>

<servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>

<init-param>

<param-name>contextConfigLocation</param-name>

<param-value>WEB-INF</param-value>

</init-param>

<init-param>

<param-name>namespace</param-name>

<param-value>application-context.xml</param-value>

</init-param>

<load-on-startup>1</load-on-startup>

</servlet>

Это правильно? Должен ли он использовать /WEB -INF/application-context.xml? И должны ли вы указывать пути?

------------------------------------------------------------------------------------------------------------------------

**TL; DR**

Просто установите значения для contextConfigLocation, когда вам нужно указать настраиваемые файлы конфигурации. Таким образом вы укажете имена файлов конфигурации и их местоположения.

**namespace - это, по существу, альтернативный способ** сказать Spring контейнерный контейнер-загрузчик, какой конфигурационный файл использовать. Я никогда не беспокоюсь об этом, но просто использую contextConfigLocation всякий раз, когда мне нужно настраивать настраиваемые файлы конфигурации.

Вот пример из одного из моих предыдущих проектов Spring (некоторые из сокращений опущены для краткости):

<?xml version="1.0" encoding="UTF-8"?>

<web-app xmlns="http://java.sun.com/xml/ns/javaee"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://java.sun.com/xml/ns/javaee

http://java.sun.com/xml/ns/javaee/web-app\_3\_0.xsd">

<display-name>Spring Web Application example</display-name>

<!-- Configurations for the root application context (parent context) -->

<listener>

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

</listener>

<context-param>

<param-name>contextConfigLocation</param-name>

<param-value>

/WEB-INF/spring/jdbc/spring-jdbc.xml

/WEB-INF/spring/security/spring-security-context.xml

</param-value>

</context-param>

<!-- Configurations for the DispatcherServlet application context (child context) -->

<servlet>

<servlet-name>spring-mvc</servlet-name>

<servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>

<init-param>

<param-name>contextConfigLocation</param-name>

<param-value>

/WEB-INF/spring/mvc/spring-mvc-servlet.xml

</param-value>

</init-param>

</servlet>

<servlet-mapping>

<servlet-name>spring-mvc</servlet-name>

<url-pattern>/admin/\*</url-pattern>

</servlet-mapping>

</web-app>

**Длинный ответ**

ОК, сначала дайте понять некоторые важные моменты. Есть два типа контекстов, с которыми мы имеем дело:

* **корневой контекст** (родительский)
* **индивидуальный контекст сервлета** (дочерний)

Цитата из Spring Framework API (версия 3.2.2 на момент написания этого) для [WebApplicationContext](http://docs.spring.io/spring/docs/3.2.2.RELEASE/javadoc-api/org/springframework/web/context/WebApplicationContext.html) (выделение мое ):

Как и общие контексты приложений, контексты веб-приложений иерархическая. **Существует один корневой контекст для каждого приложения, тогда как каждый сервлет в приложении (включая сервлет диспетчера в MVC) имеет свой собственный дочерний контекст**.

Также здесь: [Контекстные иерархии](http://docs.spring.io/spring/docs/3.2.2.RELEASE/spring-framework-reference/html/testing.html#testcontext-ctx-management-ctx-hierarchies):

Например, если вы разрабатываете веб-приложение Spring MVC, вы обычно будет иметь **корневой WebApplicationContext, загруженный через SpringContextLoaderListener** и **дочерний WebApplicationContext, загруженный через Spring DispatcherServlet**. Это приводит к контексту parent-child иерархии, где общие компоненты и конфигурация инфраструктуры объявленный в корневом контексте и потребляемый в веб-компоненты.

И здесь: [17.2 DispatcherServlet](http://docs.spring.io/spring/docs/3.2.2.RELEASE/spring-framework-reference/html/mvc.html#mvc-servlet):

Экземпляры ApplicationContext в Spring могут быть скопированы. В веб-MVC framework, **каждый DispatcherServlet имеет свой собственный WebApplicationContext, который наследует все beans, уже определенные в корне WebApplicationContext**. Эти унаследованные beans могут быть переопределены в сервлета, и вы можете определить новую область beansлокальный для данного экземпляра Servlet.

Теперь рассмотрим конфигурацию **корневого контекста приложения**. Вот пример:

web.xml

<?xml version="1.0" encoding="UTF-8"?>

<web-app xmlns="http://java.sun.com/xml/ns/javaee"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://java.sun.com/xml/ns/javaee

http://java.sun.com/xml/ns/javaee/web-app\_3\_0.xsd">

<listener>

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

</listener>

<context-param>

<param-name>contextConfigLocation</param-name>

<param-value>

/WEB-INF/spring/daoContext.xml

/WEB-INF/spring/applicationContext.xml

</param-value>

</context-param>

</web-app>

Из официальной документации Spring (выделение мое):  
[5.14.4 Удобный экземпляр ApplicationContext для веб-приложений](http://docs.spring.io/spring/docs/3.2.2.RELEASE/spring-framework-reference/html/beans.html#context-create):

**Вы можете создавать экземпляры ApplicationContext декларативно**, используя, например, ContextLoader. Конечно, вы также можете создать Примеры ApplicationContext программно, используя один из Реализации ApplicationContext.

Вы можете **зарегистрировать ApplicationContext с помощью ContextLoaderListener**(см. пример выше)

Слушатель проверяет параметр contextConfigLocation. **Если параметр не существует, слушатель использует /WEB -INF/applicationContext.xml по умолчанию**. Когда параметр существует, слушатель отделяет строку, используя предопределенные разделители (запятая, точка с запятой и пробелы) и использует значения как где будут выполняться поисковые запросы. Ant стиль шаблоны также поддерживаются. Примерами являются /WEB -INF/\*Context.xml для все файлы с именами, заканчивающимися на "Context.xml", находящиеся в Каталог "WEB-INF" и /WEB -INF/\*\*/\*Context.xml, для всех таких файлов в любом подкаталоге "WEB-INF".

Довольно часто конфигурация Spring разделяется на несколько файлов. Это более логично и удобно, особенно в крупномасштабных проектах. В нашем примере мы явно определили два файла конфигурации XML: daoContext.xml и applicationContext.xml в настраиваемом месте: /WEB-INF/spring/. Опять же, если бы мы не определяли contextConfigLocation, ContextLoaderListener попытался найти файл конфигурации по умолчанию: /WEB-INF/applicationContext.xml.

Примечание:  
Корневой контекст **не является обязательным**. Также см. Этот ответ: <https://stackoverflow.com/questions/7451325/spring-mvc-what-are-a-context-and-namespace>

Итак, если файл конфигурации по умолчанию /WEB-INF/applicationContext.xml не подходит вашим потребностям, используйте ContextLoaderListener вместе с <context-param> contextConfigLocation, где вы можете определить настраиваемые файлы конфигурации **для определения корневого контекста приложения**.

Далее рассмотрим **индивидуальный (дочерний) контекст приложения**. Из официальной документации Spring (выделение мое):  
[17.2 DispatcherServlet](http://docs.spring.io/spring/docs/3.2.2.RELEASE/spring-framework-reference/html/mvc.html#mvc-servlet)

**После инициализации DispatcherServlet, Spring MVC ищет файл с именем  
[servlet-name] -servlet.xml в каталоге WEB-INF** вашего веб-приложение и создает там beans, переопределяя определения любого beans, определенного с тем же именем в глобальном сфера.

Рассмотрим следующую конфигурацию сервлетов DispatcherServlet (в файл web.xml):

<web-app>

<servlet>

<servlet-name>golfing</servlet-name>

<servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>

<load-on-startup>1</load-on-startup>

</servlet>

<servlet-mapping>

<servlet-name>golfing</servlet-name>

<url-pattern>/golfing/\*</url-pattern>

</servlet-mapping>

</web-app>

**О контекстеConfigLocation и пространстве имен**

Из документации (акцент мой):

С приведенной выше конфигурацией сервлета вам понадобится файл с именем  
/WEB-INF/golfing-servlet.xml в вашем приложении; это файл будет содержать все ваши компоненты Spring Web MVC (beans). Вы можете изменить точное местоположение этого файла конфигурации через параметр инициализации сервлета (подробнее см. ниже).  
...  
**Вы можете настроить отдельные экземпляры DispatcherServlet**, добавив Параметры инициализации сервлета (элементы init-param) для сервлета объявление в файле web.xml. См. Следующую таблицу для списка поддерживаемых параметров.

* **contextClass**: класс, реализующий WebApplicationContext, который создает контекст, используемый этим сервлетом. По умолчанию используется XmlWebApplicationContext.
* **contextConfigLocation**: строка, которая передается в экземпляр контекста (указанный contextClass), чтобы указать, где можно найти контекст (ы). Строка состоит из нескольких строк (с использованием запятой в качестве разделителя) для поддержки нескольких контекстов. В случае нескольких контекстных местоположений с beans, которые определены дважды, последнее место имеет приоритет.
* **пространство имен**. Пространство имен WebApplicationContext. По умолчанию используется [сервлет-имя] -сервера.

Теперь давайте изучим документацию API для связанных классов. Класс [DispatcherServlet](http://docs.spring.io/spring/docs/3.2.2.RELEASE/javadoc-api/org/springframework/web/servlet/DispatcherServlet.html) расширяет абстрактный класс [FrameworkServlet](http://docs.spring.io/spring/docs/3.2.2.RELEASE/javadoc-api/org/springframework/web/servlet/FrameworkServlet.html). В документах API FrameworkServlet (основное внимание):

Пропускает init-param сервлета "contextConfigLocation" в контекст экземпляр, разбор его в потенциально несколько путей к файлу, которые могут быть разделенных любым количеством запятых и пробелов, например, "Тест-servlet.xml, myServlet.xml". **Если явно не указано, контекст реализация должна создать местоположение по умолчанию из пространство имен сервлета**.

По умолчанию используется пространство имен "servlet-name'-servlet", например. "Тест-сервлет" для сервлета "test" (приводящий к "/WEB-INF/test-servlet.xml", местоположение по умолчанию с помощью XmlWebApplicationContext). **Пространство имен может также устанавливаются явно через "namespace" servlet init-param**.

Это выдержка из исходного кода FrameworkServlet:  
[FrameworkServlet.java](http://grepcode.com/file/repository.springsource.com/org.springframework/org.springframework.web.servlet/3.2.2/org/springframework/web/servlet/FrameworkServlet.java#FrameworkServlet)

....

/\*\*

\* Suffix for WebApplicationContext namespaces. If a servlet of this class is

\* given the name "test" in a context, the namespace used by the servlet will

\* resolve to "test-servlet".

\*/

public static final String DEFAULT\_NAMESPACE\_SUFFIX = "-servlet";

....

Класс контекста по умолчанию для FrameworkServlet - [XmlWebApplicationContext](http://docs.spring.io/spring/docs/3.2.2.RELEASE/javadoc-api/org/springframework/web/context/support/XmlWebApplicationContext.html). Из документов XmlWebApplicationContext API (основное внимание):

По умолчанию конфигурация будет взята из "/WEB-INF/applicationContext.xml" для корневого контекста и "/WEB-INF/test-servlet.xml" для контекста с пространством имен "test-servlet" (например, для экземпляра DispatcherServlet с servlet-name "test" ).

**Настройки по умолчанию для конфигурации можно переопределить с помощью контекст-параметр contextConfigLocation контекста ContextLoader и сервлета init-param для FrameworkServlet**. Конфигурационные местоположения могут либо обозначать конкретные файлы типа "/WEB-INF/context.xml" или Ant -образные шаблоны, такие как "/WEB-INF/\*-context.xml" (см. PathMatcher javadoc для шаблона подробнее).

Переопределение местоположений конфигурации по умолчанию с использованием contextConfigLocation такое же, как в приведенном выше примере для контекста корневого приложения.

Что касается **переопределения пространства имен по умолчанию**, то есть несколько важных моментов. Когда вы устанавливаете новое пространство имен, **не добавляйте его с помощью /WEB-INF и не добавляйте .xml к нему**. Причина этого может быть обнаружена, если мы посмотрим в исходном файле для класса XmlWebApplicationContext:  
[XmlWebApplicationContext.java](http://grepcode.com/file/repository.springsource.com/org.springframework/org.springframework.web/3.2.2/org/springframework/web/context/support/XmlWebApplicationContext.java#XmlWebApplicationContext)

...

/\*\* Default config location for the root context \*/

public static final String DEFAULT\_CONFIG\_LOCATION = "/WEB-INF/applicationContext.xml";

/\*\* Default prefix for building a config location for a namespace \*/

public static final String DEFAULT\_CONFIG\_LOCATION\_PREFIX = "/WEB-INF/";

/\*\* Default suffix for building a config location for a namespace \*/

public static final String DEFAULT\_CONFIG\_LOCATION\_SUFFIX = ".xml";

...

/\*\*

\* The default location for the root context is "/WEB-INF/applicationContext.xml",

\* and "/WEB-INF/test-servlet.xml" for a context with the namespace "test-servlet"

\* (like for a DispatcherServlet instance with the servlet-name "test").

\*/

@Override

protected String[] getDefaultConfigLocations() {

if (getNamespace() != null) {

return new String[] {DEFAULT\_CONFIG\_LOCATION\_PREFIX + getNamespace() + DEFAULT\_CONFIG\_LOCATION\_SUFFIX};

}

else {

return new String[] {DEFAULT\_CONFIG\_LOCATION};

}

}

Как вы можете видеть, исходный код говорит все.

**Пример указания пользовательского пространства имен**

web.xml

<?xml version="1.0" encoding="UTF-8"?>

<web-app xmlns="http://java.sun.com/xml/ns/javaee"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://java.sun.com/xml/ns/javaee

http://java.sun.com/xml/ns/javaee/web-app\_3\_0.xsd">

<!-- Configurations for the DispatcherServlet application context (child context) -->

<servlet>

<servlet-name>spring-mvc</servlet-name>

<servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>

<init-param>

<param-name>namespace</param-name>

<param-value>spring/mvc/spring-mvc</param-value>

</init-param>

</servlet>

<servlet-mapping>

<servlet-name>spring-mvc</servlet-name>

<url-pattern>/\*</url-pattern>

</servlet-mapping>

</web-app>

В результате вместо использования пространства имен по умолчанию для построения пути к конфигурационному файлу, который в противном случае был бы /WEB-INF/spring-mvc-servlet.xml, контейнер будет искать /WEB-INF/spring/mvc/spring-mvc.xml.

Примечание:  
Приведенные выше объяснения, связанные с настройкой настраиваемого пространства имен, относятся к классу контекста [XmlWebApplicationContext](http://docs.spring.io/spring/docs/3.2.2.RELEASE/javadoc-api/org/springframework/web/context/support/XmlWebApplicationContext.html). Можно указать альтернативный класс, например [AnnotationConfigWebApplicationContext](http://docs.spring.io/spring/docs/3.2.2.RELEASE/javadoc-api/org/springframework/web/context/support/AnnotationConfigWebApplicationContext.html), поэтому для этого будут особые моменты.

**Заключение**

Это (IMHO) гораздо проще использовать параметр contextConfigLocation для определения настраиваемых файлов конфигурации, как для контекста корневого приложения, так и для отдельных контекстов. Единственное различие заключается в том, что для корневого контекста приложения вы используете **<context-param>** внутри элемента <web-app>, но НЕ внутри определенного сервлета (также не забывайте класс слушателя). А для дочернего контекста вы используете **<init-param>**, вложенный внутри элемента <servlet> **для каждого определенного сервлета**. См. Мои примеры конфигураций (web.xml) в самом начале этого сообщения.

Дополнительные источники (как если бы это было недостаточно: -)):

* [Spring Справочник по Framework Документация](http://docs.spring.io/spring/docs/current/spring-framework-reference/html/)
* [Spring Контекст MVC Загрузка](http://www.jroller.com/latenightdev/entry/spring_mvc_context_loading)

Также см. ответы:

* [Разница между applicationContext.xml и spring -servlet.xml в Spring](https://stackoverflow.com/questions/3652090/difference-between-applicationcontext-xml-and-spring-servlet-xml-in-spring-frame)
* [В чем разница между ApplicationContext и WebApplicationContext в Spring MVC?](https://stackoverflow.com/questions/11708967/what-is-the-difference-between-applicationcontext-and-webapplicationcontext-in-s)
* [Spring -MVC: что такое "контекст" и "пространство имен" ?](https://stackoverflow.com/questions/7451325/spring-mvc-what-are-a-context-and-namespace)