DTF

Manual Técnico

Contenido

[I. Introducción 3](#_Toc393753951)

[Cómo leer el documento 3](#_Toc393753952)

[II. Bases de la solución 4](#_Toc393753953)

[Tecnologías empleadas en DTF 4](#_Toc393753954)

[Patrones de diseño implementados 4](#_Toc393753955)

[III. Descripción de la solución 6](#_Toc393753956)

[Descripción de módulos de DTF 6](#_Toc393753957)

[Diagrama de paquetes de dtf-api 8](#_Toc393753958)

[Diagrama de paquetes de dtf-maven-plugin 8](#_Toc393753959)

[Diagrama de paquetes general de DTF 8](#_Toc393753960)

[Diagramas de clases general de DTF 10](#_Toc393753961)

[Diagrama de clases de esquemas 10](#_Toc393753962)

[Diagramas de secuencia 12](#_Toc393753963)

[Limpieza de carpetas de archivos autogenerados 12](#_Toc393753964)

[Generación de suites de pruebas 12](#_Toc393753965)

[Generación de reportes 14](#_Toc393753966)

[IV. Diseño de interfaces de usuario 16](#_Toc393753967)

# Introducción

El presente manual técnico describirá de forma general los detalles de la implementación de DTF.

## Cómo leer el documento

Para leer y entender este manual técnico se debe contar con conocimientos de UML, patrones de diseño, Programación Orientada a Objetos y conocimientos básicos del funcionamiento de Apache Maven.

En primer lugar, el documento presenta los conceptos de diseño de sistemas de *software* involucrados en la construcción de DTF que describirán las bases de su implementación, esto involucra las decisiones de diseño arquitectónico del *software*. Y posteriormente se describirá técnicamente, a través de diagramas UML2, los módulos que componen al sistema, en donde se apreciará la implementación de los conceptos de diseño mencionados.

# Bases de la solución

La construcción de DTF se fundamente en un grupo de plataformas y guías bien definidos que en conjunto proporcionan robustez al *framework*. Estas guías corresponden a patrones de diseño y estándares de desarrollo que serán descritos en las siguientes secciones.

## Tecnologías empleadas en DTF

En seguida se presenta una tabla que enlista las tecnologías empleadas en la construcción de DTF y su uso dentro del *framework*.

|  |  |  |  |
| --- | --- | --- | --- |
| Tecnología | | Versión | Descripción |
| Java | 6+ | | Todos los sub-módulos de DTF están construidos usando el lenguaje de programación Java, usando el JDK6. Por lo tanto, el resto de tecnologías empleados funcionan sobre la plataforma Java. |
| Apache Maven | 3 | | Los comandos de DTF fueron construidos como tareas ejecutables de Maven. Además, el código fuente de cada módulo de DTF es un proyecto Maven. |
| Apache Ant |  | | En desarrollo, los módulos de DTF ejecutan algunas tareas de Ant para la construcción de los mismos. |
| Spring Framework | 3.0.5.RELEASE | | Se hace uso de algunos módulos de Spring en las configuraciones de conexión a Base de Datos. Estos módulos son *spring-beans* y *spring-context*. |
| Hibernate | 3.6.1.Final | | Para las operaciones de exportación y restauración de información en base de datos se utiliza Hibernate junto con un plugin de Maven especializado para llevar a cabo dichas tareas. |
| Selenium IDE |  | | Internamente, DTF funciona haciendo uso de Selenium IDE. |
| Javascript |  | | El plugin de DTF para Maven incluye un conjunto de código fuente de Selenium IDE, el cual está escrito en Javascript. |
| Rhino |  | | Se utiliza extensivamente esta implementación de Javascript para ejecutar las funciones de Selenium IDE escritas en este lenguaje. |
| XML/XSD/XSL |  | | Toda la configuración de DTF se escribe en archivos XML, los cuales se adhieren a la estricta validación de esquemas XSD. Además, la documentación de estos esquemas se genera haciendo uso de XSL. |
| HTML |  | | Su uso está relegado únicamente a casos de prueba generados automáticamente y a reportes de resultados generados de la misma forma. |

## Patrones de diseño implementados

En la siguiente tabla se enlistan los patrones de diseño implementados en la construcción de DTF y su uso dentro del *framework*.

|  |  |  |
| --- | --- | --- |
| Patrón o estándar | | Descripción |
| Composición | Es usado en conjunto con otros patrones y haciendo amplio uso de las características de la programación orientada a objetos. Se destaca sobre todo en la estructura de los elementos que componen una suite de pruebas. Se puede apreciar mejor en el diagrama de clases de la sección [Diagrama de clases de esquemas](#_Diagrama_de_clases). | |
| Contexto | Se podrá encontrar implementado este patrón de diseño en las pruebas unitarias generadas por DTF. Además de hacer uso también de la herencia, otra característica de la orientación a objetos. | |
| Fábrica de objetos | Será común encontrar estas fábricas en las clases generadas que mapean los esquemas de XML, tanto de *settings.xml* como de los casos de prueba. | |
| Utilerías estáticas | En los módulos de DTF se encontrarán clases de utilería que no guardan estado los cuales sirven para agrupar funcionalidades comunes a las clases que conforman el *framework*. | |
| Fachada | También se implementan fachadas. Las clases MOJO son un ejemplo de implementación de fachadas en DTF. | |
| Estrategia | El generador de suites de pruebas mantiene un registro de componentes, funciones y constructores de comandos, los cuales están especializados en crear comandos Selenium de DTF para cada tipo de acción configurada en un caso de pruebas. Estos constructores son estrategias de creación de comandos que DTF reconoce. | |

# Descripción de la solución

En esta sección se describirán los módulos que componen a DTF. Se detallarán los componentes desde una perspectiva lógica general, tanto de su estructura como funcionamiento. Para lograr este objetivo se presentará la estructura de carpetas de los módulos, diagramas UML de paquetes y clases, al igual que diagramas de secuencia.

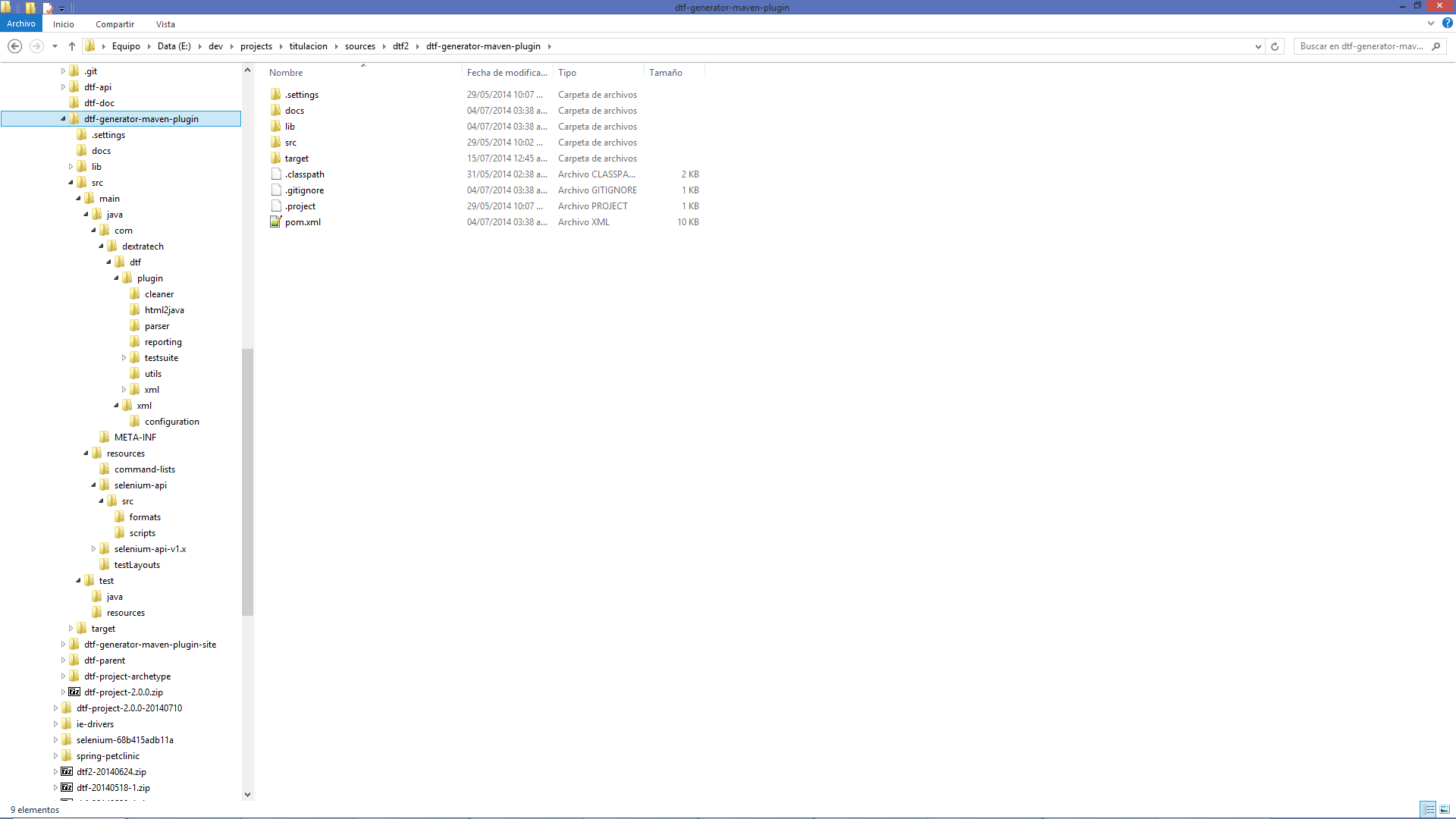
En los diagramas de clases se podrá apreciar la implementación de los patrones mencionados en la sección [Patrones de diseño implementados](#_Patrones_de_diseño).

## Descripción de módulos de DTF

El código fuente de DTF se compone de tres módulos que se describirán a continuación.

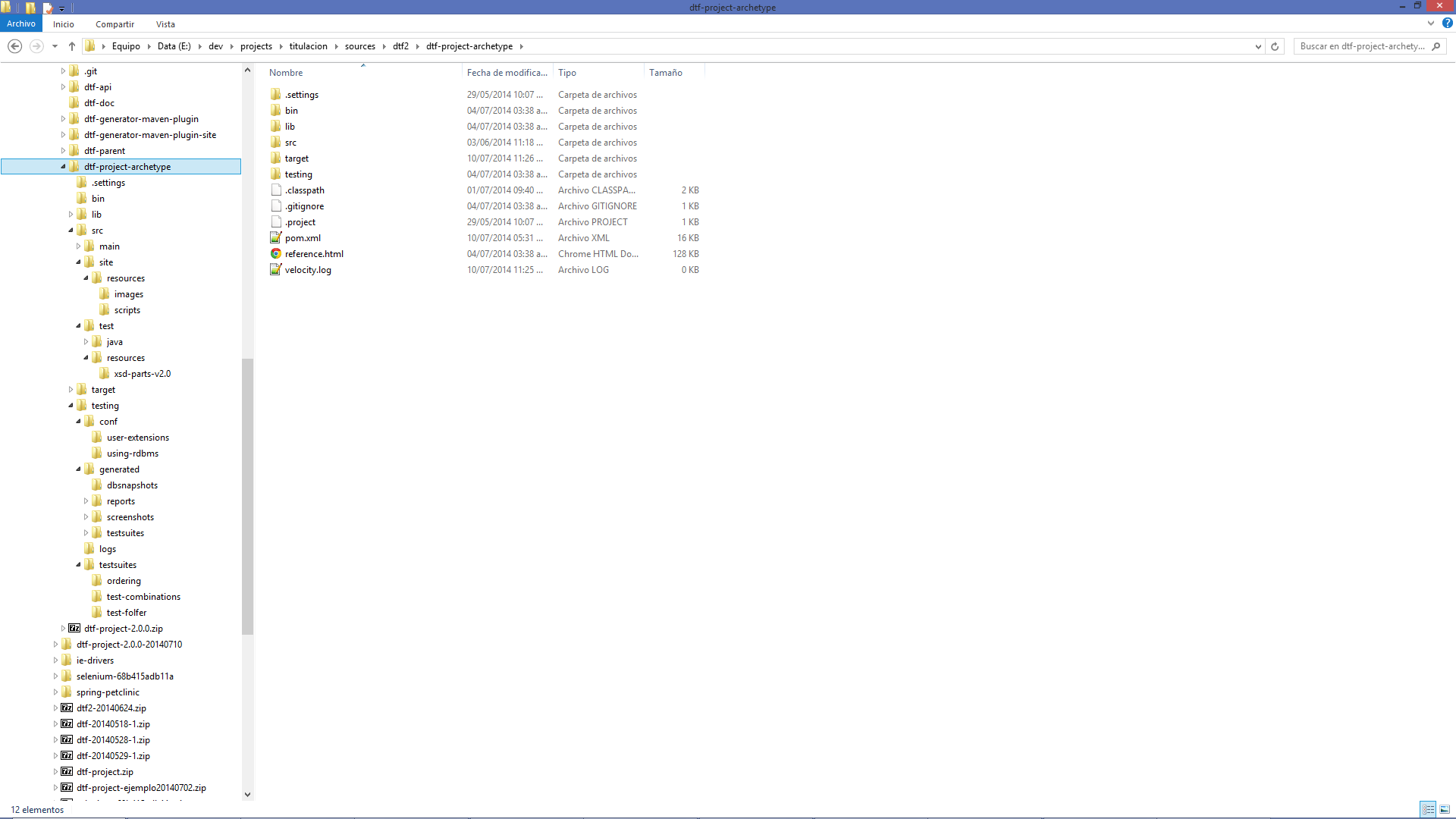
1. **dtf-parent**: Es el proyecto que engloba a los módulos que componen a DTF.
2. **dtf-maven-plugin**: Corresponde al plugin de DTF para Maven, el que contiene los comandos que serán ejecutados y lleva a cabo las funcionalidades principales del *framework*.

La siguiente imagen muestra la estructura de directorios del proyecto:



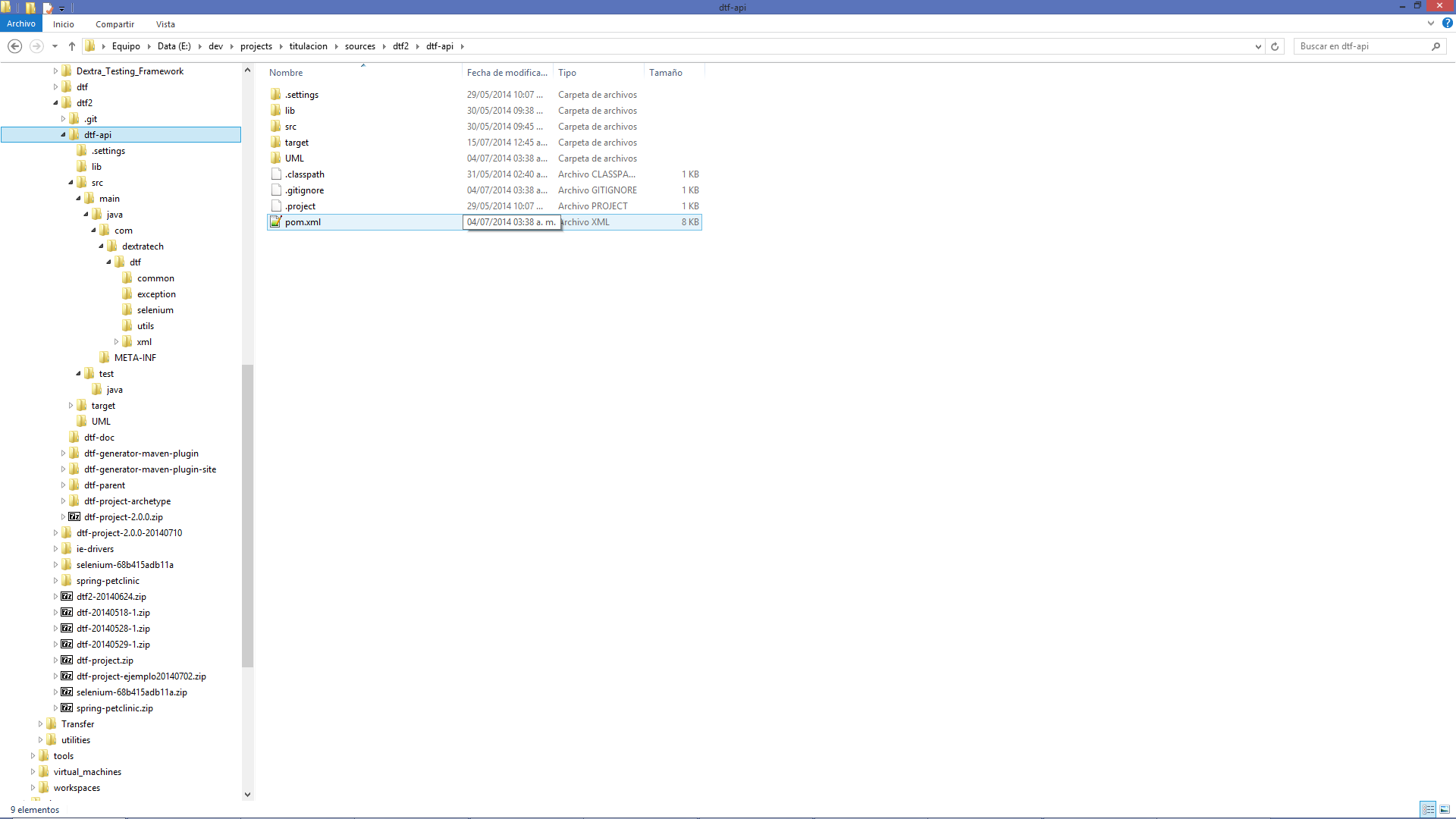
1. **dtf-project-archetype**: Inicialmente fue desarrollado como un arquetipo de Maven para generar nuevos proyectos de prueba, pero en la nueva versión de DTF esto ya no es necesario. Este módulo contiene las estructuras y archivos que corresponderán a un proyecto de pruebas, ya que con estos se generará el archivo comprimido que será usado para crear nuevos proyectos de prueba.

La siguiente imagen muestra la estructura de directorios del proyecto:



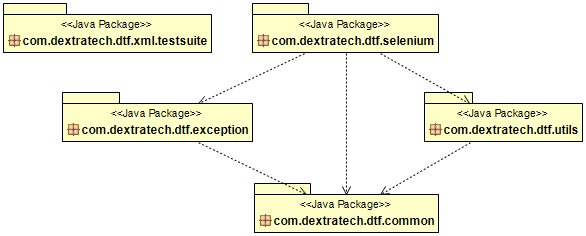
1. **dtf-api**: Este módulo está construido como un proyecto Maven. Contiene clases comunes a *dtf-maven-plugin* y al proyecto de pruebas.

La siguiente imagen muestra la estructura de directorios del proyecto:



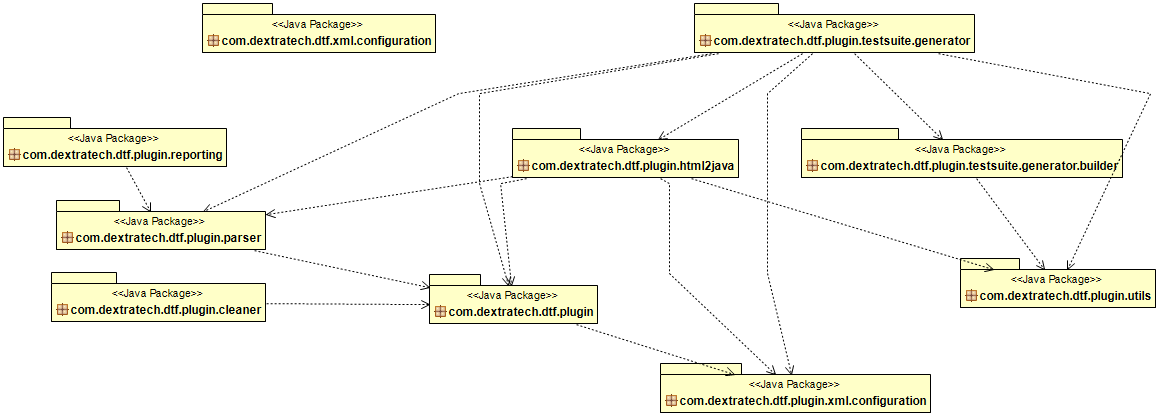
## Diagrama de paquetes de dtf-api

La siguiente imagen representa el diagrama de los paquetes de clases que componen el módulo *dtf-api*.



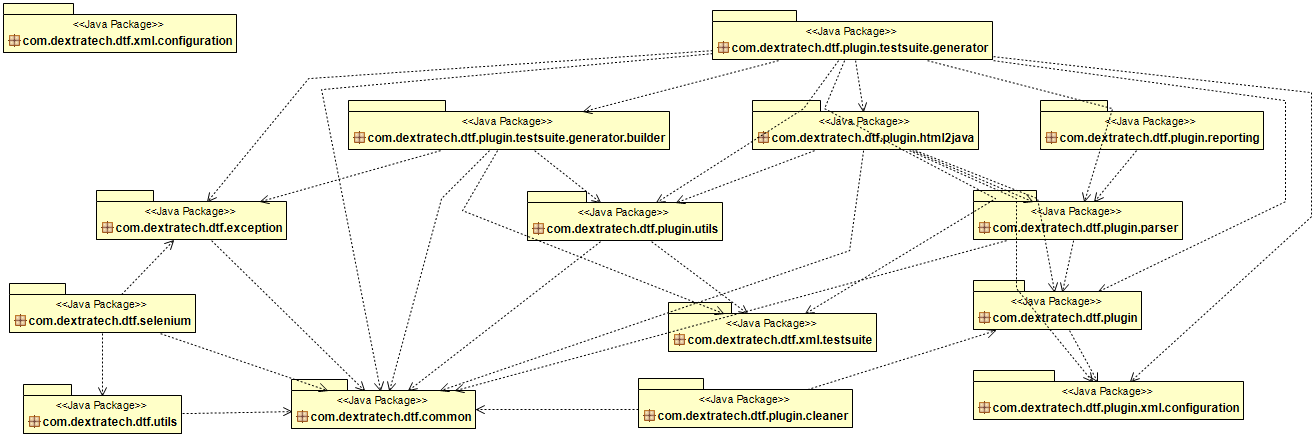
## Diagrama de paquetes de dtf-maven-plugin

La siguiente imagen representa el diagrama de los paquetes de clases que componen el módulo *dtf-maven-plugin*.



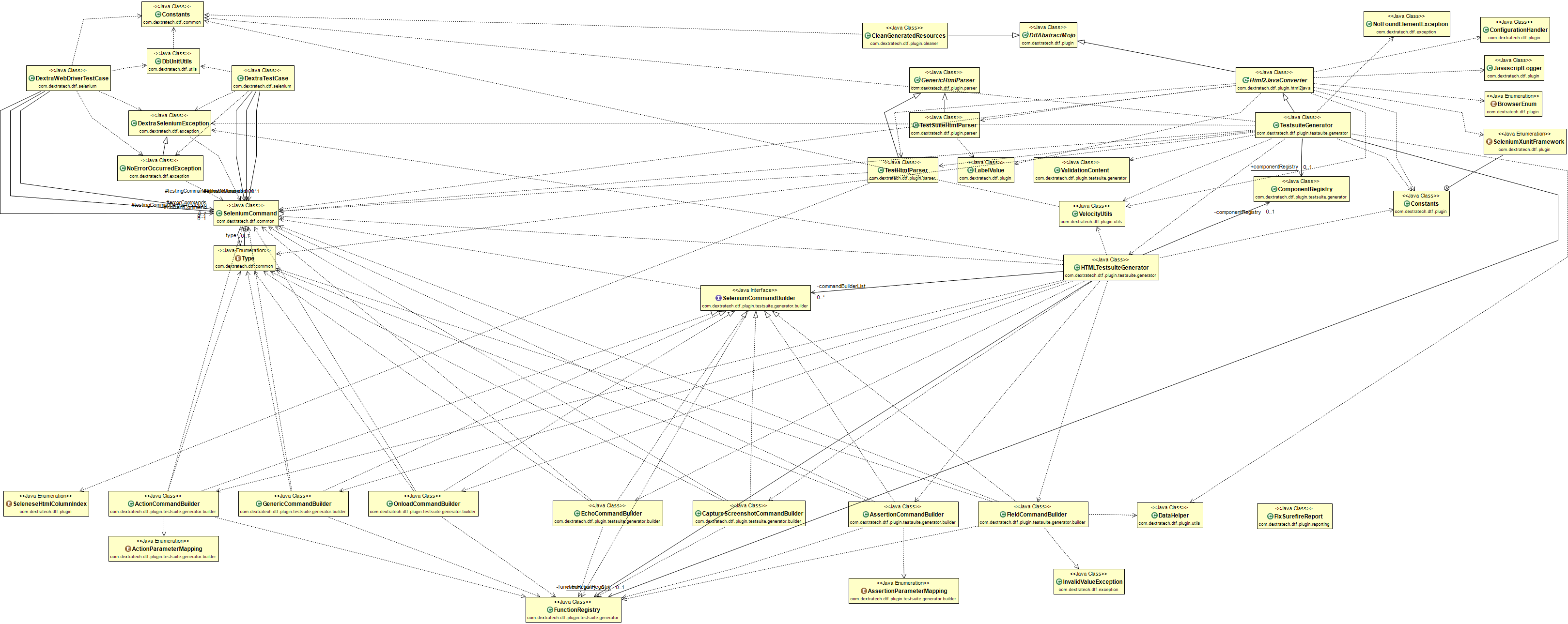
## Diagrama de paquetes general de DTF

El siguiente diagrama conjunta los paquetes de los dos módulos anteriores y su interdependencia. Permite una visión más amplia de los paquetes de clases que componen a la solución y que implementan los comandos que ejecutarán las tareas creadas para el *framework*.



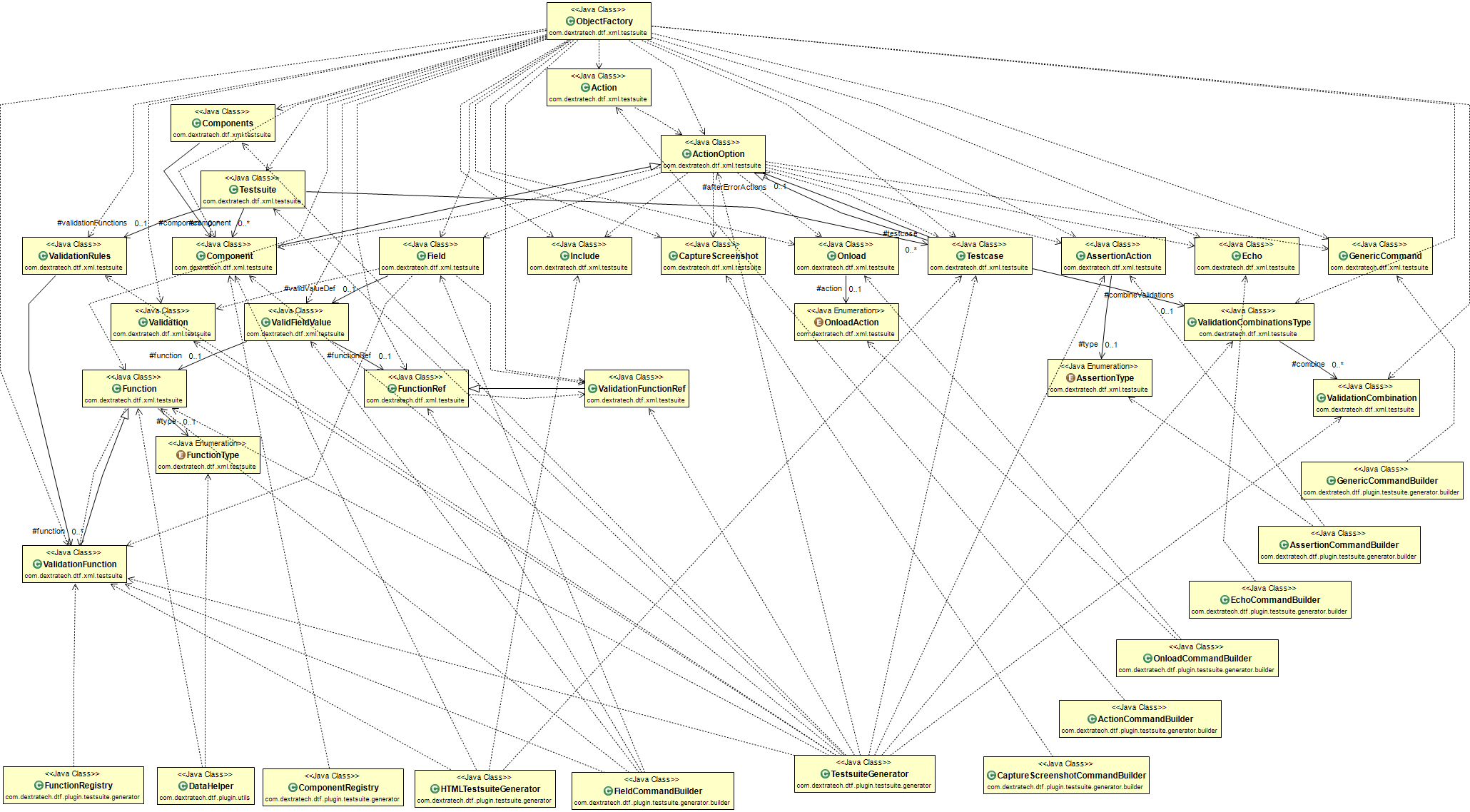
## Diagramas de clases general de DTF

El siguiente diagrama de clases involucra a las clases contenidas en los módulos *dtf-api* y *dtf-maven-plugin* del framework, pero no se han incluido las clases pertenecientes a los paquetes *com.dextratech.dtf.xml.testsuite* y *com.dextratech.dtf.plugin.xml.configuration*, ya que las clases de estos paquetes se expondrán en la próxima sección.



## Diagrama de clases de esquemas

Las clases que se presentan en el siguiente diagrama corresponden a los paquetes *com.dextratech.dtf.xml.testsuite* y *com.dextratech.dtf.plugin.xml.configuration*. Estas clases son generadas automáticamente usando JAXB y mapean a los archivos XSD de validación de los archivos XML usados en la configuración del *framework*. En este diagrama se agregaron también las clases con las que interactúan y que pertenecen a otros paquetes; esto se hace para tener una mejor perspectiva de las dependencias existentes con las clases del mapeo de esquemas.

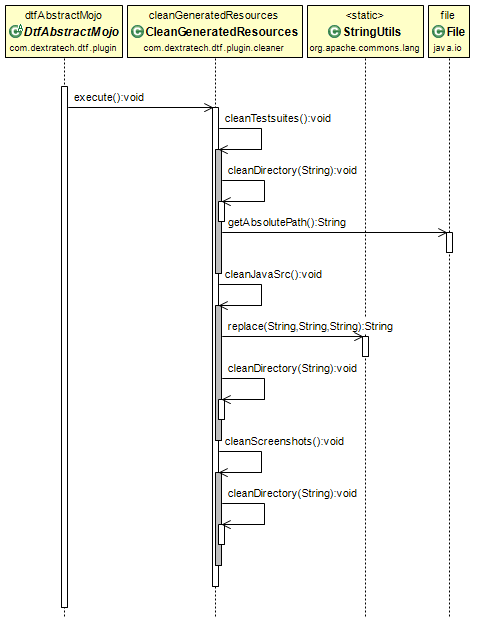


## Diagramas de secuencia

Esta sección describirá el flujo de invocaciones entre las principales clases involucradas en la ejecución de cada tarea de DTF.

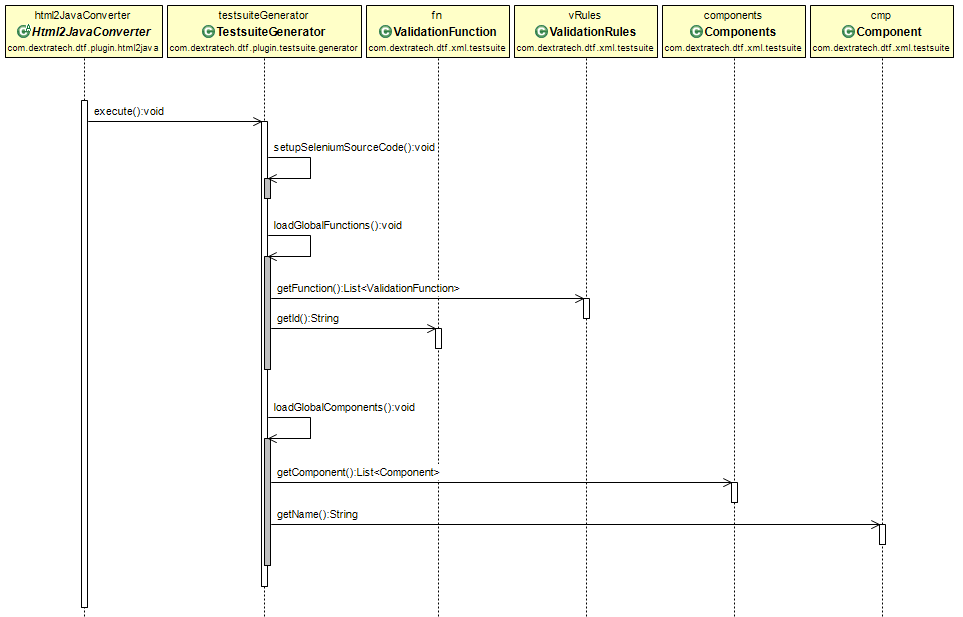
### Limpieza de carpetas de archivos autogenerados

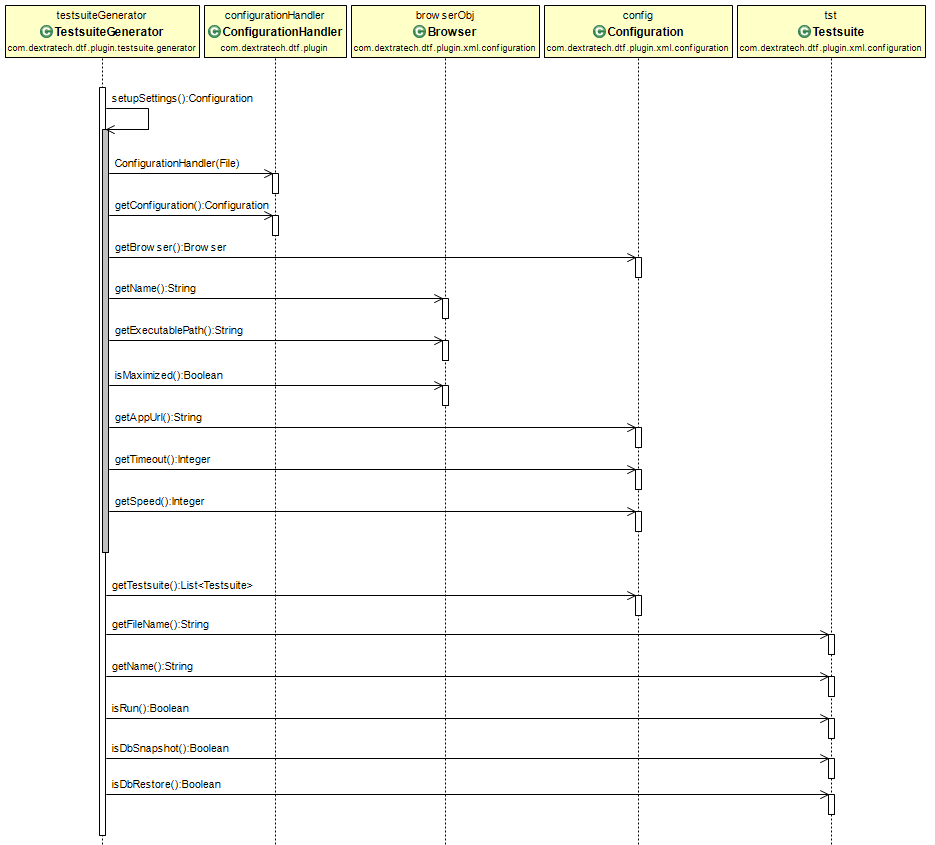
La tarea de limpieza de carpetas con contenido autogenerado es muy simple, como se muestra a continuación.

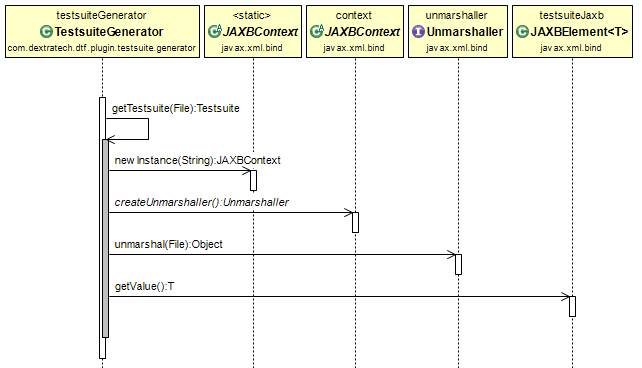


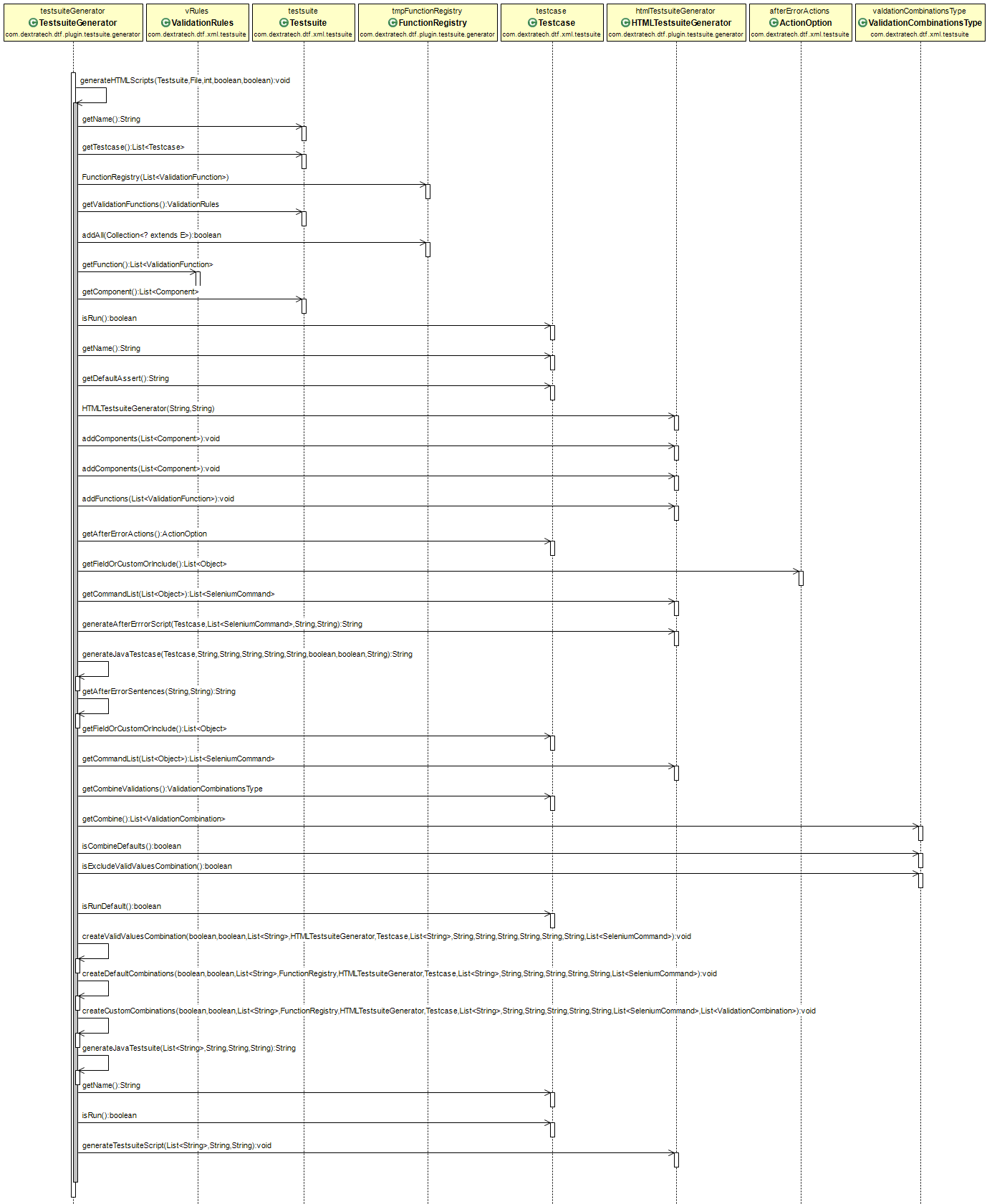
### Generación de suites de pruebas

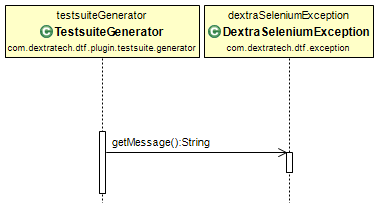
Esta tarea es la más grande y compleja del *framework* DTF, por lo tal se ha dividido en cinco partes consecutivas que permiten mostrar más detalladamente el flujo de invocaciones entre las principales clases involucradas en su ejecución.





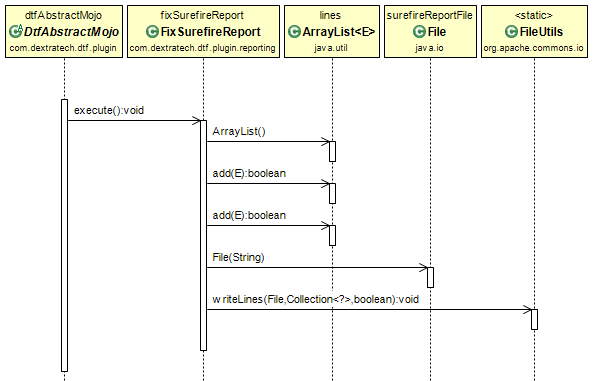






### Generación de reportes

Esta actividad es un complemento a la tarea de generación de reportes llevada a cabo durante la ejecución de las pruebas y durante la fase *site* de Maven. Se ejecuta para agregar *scripts* que permitirán desplegar correctamente la información adicional generada por DTF.



# Diseño de interfaces de usuario

DTF no cuenta con una interface gráfica de usuario sino que todas sus actividades son ejecutadas a través de línea de comandos, o usando un IDE que soporte la ejecución de tareas Maven, como Eclipse o Netbeans.