# How to write a forrest:contract?

*This How-To will explain how we wrote the contracts for views and hope afterwards you will be able to do the same.*

## Table of contents


1 Intended Audience....2
2 Purpose....2
3 Prerequisites....2
4 Steps....2
4.1 Enhance the maintainment....2
4.2 Explaining the blank forrest:contract....3
4.3 Create a new contract....4
4.4 Activating the contract....5
5 Further Reading....5
6 Feedback....5

## 1. Intended Audience

**Warning:**
"Views" are new functionality which is still in development phase. That is why it is in the "whiteboard" section of the Forrest distribution. This HowTo is a good start but still needs proof-reading.

Devs and skin developer that wants to get started with forrest:contract development. To really understand this how-to you need basic and sometimes advanced understanding of the "old fashion" skin development process.

## 2. Purpose

This setup guide will explain how to create a forrest:contract from scratch and how this forrest:contract work with the core parts of forrest.

## 3. Prerequisites

- You have a ready-to-go new seed (newSeed) based on views like described in Install views.
- This includes as well all additional plugins that are mentioned in Install views.
- Reading that how-to is as well a good idea to understand the used dir-structure in this how-to.

## 4. Steps

**Note:**
The following content is from many mails around the topic, this how-to tries to be the consolidation of this thread. It is mainly based on the [RT] Why using views - in comparison with "old fashion" skins - usecase i18n

By working on the i18n integration for "pelt" we crossed again the whys for using views. ;-) The maintainment problem was to change the captions of the skin features (contracts) to enable support for i18n. The case is that the `site2xhtml.xsl` has a lot of repeating code.

For example the "last-publish"-contract could be found 2 times in the code. This is not the only contract that was (is) double in the code. The problem with that is that we needed to search the code for each caption and senseless repeat the following maintainment step of adding the <i18n:text/>-tags.

```
- <script language="JavaScript"
- type="text/javascript">document.write("Published: " +
document.lastModified);</script>

+ <script type="text/javascript">document.write("<i18n:text >Last
+ Published:</i18n:text> " + document.lastModified);
```

### 4.1. Enhance the maintainment

Now we can enhance the maintainment for the future and we give this code snippets contracts names (based on their functionality). This naming enables us to keep the contract separate from the position

code itself. In xsl you would simply do:

1. replace the script by <xsl:call-template name="siteinfo-last-published"/>
2. and add:

```
<xsl:template name="siteinfo-last-published">
 <script type="text/javascript">
  document.write("<i18n:text >Last Published:</i18n:text> " +
document.lastModified);
 </script>
</xsl:template>
```

This allows us in a next maintainment just change the code of <xsl:template name="siteinfo-last-published"/> and apply it in any position where it is placed.

**Note:**
Now this refactoring of the site2xhtml.xsl is exactly what we doing in creating contracts for views.

## 4.2. Explaining the blank forrest:contract

To start a new forrest:contract you can copy the 'blank.ft' from `org.apache.forrest.plugin.output.viewHelper.xhtml/resources/templates`. The exact file system path can be looked up at `http://localhost:8888/ls.contracts.html`.

The 'blank.ft' is a simple xml file with the following code which you can use to base new contracts on:

```
<forrest:contract
  xmlns:forrest="http://apache.org/forrest/templates/1.0"
  name="blank" type="nugget">

  <!--NOTE:
    When using the blank template as c'n p master just search and replace 'blank'
by the {contract-name}!-->

  <description>
    {contract-name} will output {contract-funtion}. This is just a blank contract,
it will output *nothing*.
  </description>
  <usage><![CDATA[<forrest:contract name="blank"/>]]></usage>
  <forrest:template xmlns:forrest="http://apache.org/forrest/templates/1.0"
    format="xhtml" name="blank" inputFormat="xsl" body="false" head="false">
    <xsl:stylesheet version="1.1"
        xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
        <!--Add here the needed templates-->
    </xsl:stylesheet>
  </forrest:template>
</forrest:contract>
```

The most important is the name of the contract `<forrest:contract name="blank"/>`. This name is the same as the file name of the contract (without file extension) `blank.ft`.

**Note:**
This is a **naming convention** that you have to always met. All @name has to be file name of the contract without file extension!

The `<description/>` tag needs to be filled in with some information that is explaining the contract to the webdesigner. The better explained the more efficient for the webdesigner to pick the right

contract.

```
<description>
  siteinfo-last-published-howto will output the last published date of the site
with the help of jscript.
</description>
```

In the `<usage/>` tag we have to explain how the designer can use the contract in his view.

```
<usage><![CDATA[<forrest:contract
name="siteinfo-last-published-howto"/>]]></usage>
```

`<forrest:template name="blank" body="false" head="false">` That leads to the template attribute @body="true" and @head="false". In xhtml a contract can add content to the <body/> or/and <head/> part of <html/>. This values have to be change when adding an actual template. Besides this a xsl-template has to indicate this in the naming. A template that add content to the html body has to end with "`-body`"!!!

**Note:**

It is possible to use contracts in different in/output-formats. We are focusing for now on format="xhtml" as ouput and the inputFormat="xsl".

A <forrest:template /> has the son <xsl:stylesheet/> where we can create templates for the html-head and html-body. For adding content into the body of the final document change @body="true" and add:

```
<xsl:stylesheet version="1.1"
 xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
 <!--Add here the needed templates-->
 <xsl:template name="blank-body"/>
</xsl:stylesheet>
```

## 4.3. Create a new contract

**FIXME (thorsten):**

We need to explain basic naming convention for contracts. Like "naming do not say about layout position but functionality of the contract.".

Now lets pick up the example we started with and create a "siteinfo-last-published-howto" contract. Save the blank.ft to `{project.home}/src/documentation/resources/templates/siteinfo-last-publishe`

Now the maintainment optimized code (xpath="/html/body/*") was:

```
<xsl:template name="siteinfo-last-published">
 <script type="text/javascript">
  document.write("<i18n:text >Last Published:</i18n:text> " +
document.lastModified);
 </script>
</xsl:template>
```

In this code we have to do the following steps to use it in our contract:

- Search and replace "siteinfo-last-published" with "siteinfo-last-publish-howto-body"
- Add a "debug string - " to the template

The contract after this steps should look like:

```
<xsl:template name="siteinfo-last-publish-howto-body">
debug string -
```

```
 <script type="text/javascript">
  document.write("<i18n:text >Last Published:</i18n:text> " +
document.lastModified);
 </script>
</xsl:template>
```

Now we have to do some last steps in the siteinfo-last-publish-howto.ft

- Search and replace "blank" with "siteinfo-last-publish-howto"
- Add description and usage of the contract
- Set @body="true"
- Copy the maintainment optimized code to the contract.

As the result your code should look like this:

```
<forrest:contract xmlns:forrest="http://apache.org/forrest/templates/1.0"
  name="siteinfo-last-published-howto" type="nugget">
  <description>
    siteinfo-last-published-howto will output the last published date of the site
with the help of jscript.
  </description>
  <usage><![CDATA[<forrest:contract
name="siteinfo-last-published-howto"/>]]></usage>
  <forrest:template
    xmlns:i18n="http://apache.org/cocoon/i18n/2.1"
    xmlns:forrest="http://apache.org/forrest/templates/1.0"
    format="xhtml" name="siteinfo-last-published-howto" inputFormat="xsl"
body="true" head="false">
    <xsl:stylesheet version="1.1"
        xmlns:i18n="http://apache.org/cocoon/i18n/2.1"
        xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
        <xsl:template name="siteinfo-last-published-howto-body">
          debug string -
          <script type="text/javascript">document.write("<i18n:text >Last
Published:</i18n:text> " + document.lastModified);</script>
        </xsl:template>
    </xsl:stylesheet>
  </forrest:template>
</forrest:contract>
```

### 4.4. Activating the contract

To see whether the new contract works we need to add it to our view. The contract usage contains the contract-tag `<forrest:contract name="siteinfo-last-published-howto"/>` Please see Getting started with forrest:view DSL for more details.

**FIXME (thorsten):**

Let us now look into advanced features of views. I will write a how-to for advanced contracts soon. :)

## 5. Further Reading

Congratulations you are now able to work with view contracts. From here we recommend to read the following How-To's:

- Getting started with forrest:view DSL

## 6. Feedback

Please provide feedback about this document via the mailing lists.