
Ada Server Faces Programmer's Guide

Stephane Carrez

2023-08-14

Contents

1	Introduction	8
2	Installation	10
2.1	Using Alire	10
2.2	Without Alire	10
2.2.1	Before Building	10
2.2.2	Configuration	11
2.2.3	Build	11
2.2.4	Installation	12
2.3	Using	12
3	Tutorial	13
3.1	View definition	13
3.2	Writing the Cylinder Ada Bean	17
3.3	Register the Cylinder Ada Bean	18
3.4	Command buttons and method expression	18
3.5	Method Bean Declaration	19
3.6	Implement the action	20
3.7	Define the action binding	20
3.8	Register and expose the action bindings	20
3.9	What happens now?	21
3.10	Application Initialization	21
3.11	Servlets	22
3.12	Application and Web Container	23
3.13	Global Objects	23
3.14	Starting the server	24
3.15	What happens to a request?	24
4	Request Processing Lifecycle	26
5	Converters	27
5.1	Date converter	27
5.2	Number converter	27
5.3	Size converter	27
6	Validators	28
6.1	Length validator	28

6.2	Regex validator	28
6.3	Range validator	28
6.4	Components	29
7	Facelet Components	30
7.1	ui:composition	30
7.1.1	Attributes	30
7.1.2	Example	30
7.2	ui:decorate	31
7.2.1	Attributes	31
7.2.2	Example	31
7.3	ui:define	31
7.3.1	Attributes	32
7.3.2	Example	32
7.4	ui:include	32
7.4.1	Attributes	32
7.4.2	Example	33
7.5	ui:insert	33
7.5.1	Attributes	33
7.5.2	Example	33
7.6	ui:param	34
7.6.1	Attributes	34
7.6.2	Example	34
8	JSTL Components	35
8.1	c:choose	35
8.1.1	Attributes	35
8.1.2	Example	35
8.2	c:if	35
8.2.1	Attributes	36
8.2.2	Example	36
8.3	c:otherwise	36
8.3.1	Attributes	36
8.3.2	Example	36
8.4	c:set	37
8.4.1	Attributes	37
8.4.2	Example	37

8.5	c:when	37
8.5.1	Attributes	37
8.5.2	Example	37
9	Core Components	39
9.1	f:attribute	39
9.1.1	Attributes	39
9.1.2	Example	40
9.2	f:convertDateTime	40
9.2.1	Attributes	40
9.2.2	Example	41
9.3	f:converter	42
9.3.1	Attributes	43
9.3.2	Example	43
9.4	f:facet	43
9.4.1	Attributes	43
9.4.2	Example	44
9.5	f:metadata	44
9.5.1	Attributes	44
9.5.2	Example	44
9.6	f:param	45
9.6.1	Attributes	45
9.6.2	Example	45
9.7	f:selectItem	45
9.7.1	Attributes	46
9.7.2	Example	46
9.8	f:selectItems	46
9.8.1	Attributes	47
9.8.2	Example	47
9.9	f:validateLength	47
9.9.1	Attributes	48
9.9.2	Example	48
9.10	f:validateLongRange	49
9.10.1	Attributes	49
9.10.2	Example	49
9.11	f:validator	50
9.11.1	Attributes	50
9.11.2	Example	50

9.12	f:view	51
9.12.1	Attributes	51
9.12.2	Example	52
9.13	f:viewAction	52
9.13.1	Attributes	52
9.13.2	Example	53
9.14	f:viewParam	53
9.14.1	Attributes	53
9.14.2	Example	54
10	HTML Components	55
10.1	h:body	55
10.1.1	Attributes	55
10.1.2	Example	56
10.2	h:commandButton	56
10.2.1	Attributes	56
10.2.2	Example	59
10.3	h:form	59
10.3.1	Attributes	60
10.3.2	Example	61
10.4	h:head	62
10.4.1	Attributes	62
10.4.2	Example	62
10.5	h:inputFile	63
10.5.1	Attributes	63
10.5.2	Example	63
10.6	h:inputHidden	64
10.6.1	Attributes	64
10.6.2	Example	65
10.7	h:inputSecret	66
10.7.1	Attributes	66
10.7.2	Example	69
10.8	h:inputText	70
10.8.1	Attributes	70
10.8.2	Example	72
10.9	h:inputTextarea	73
10.9.1	Attributes	73
10.9.2	Example	76

10.10h:list	77
10.10.1 Attributes	77
10.10.2 Example	78
10.11h:outputFormat	78
10.11.1 Attributes	78
10.11.2 Example	80
10.12h:outputLabel	81
10.12.1 Attributes	81
10.12.2 Example	83
10.13h:outputLink	83
10.13.1 Attributes	84
10.13.2 Example	85
10.14h:outputText	86
10.14.1 Attributes	86
10.14.2 Example	87
10.15h:panelGroup	87
10.15.1 Attributes	87
10.15.2 Example	88
10.16h:selectBooleanCheckbox	88
10.16.1 Attributes	88
10.16.2 Example	90
10.17h:selectOneMenu	91
10.17.1 Attributes	91
10.17.2 Example	93
11 Util Components	95
11.1 util:escape	95
11.1.1 Attributes	95
11.1.2 Example	95
11.2 util:file	96
11.2.1 Attributes	96
11.2.2 Example	96
11.3 util:flush	97
11.3.1 Attributes	97
11.3.2 Example	97
11.4 util:script	97
11.4.1 Attributes	98
11.4.2 Example	98

11.5	util:set	99
11.5.1	Attributes	99
11.5.2	Example	99
12	Widget Components	100
12.1	w:accordion	100
12.1.1	Attributes	100
12.1.2	Example	100
12.2	w:autocomplete	101
12.2.1	Attributes	101
12.2.2	Example	102
12.3	w:chosen	102
12.3.1	Attributes	103
12.3.2	Example	103
12.4	w:gravatar	104
12.4.1	Attributes	104
12.4.2	Example	104
12.5	w:inputDate	105
12.5.1	Attributes	105
12.5.2	Example	106
12.6	w:inputText	106
12.6.1	Attributes	107
12.6.2	Example	107
12.7	w:like	108
12.7.1	Facebook	108
12.7.2	Twitter	108
12.7.3	Attributes	108
12.7.4	Example	109
12.8	w:panel	109
12.8.1	Attributes	109
12.8.2	Example	110
12.9	w:tab	110
12.9.1	Attributes	110
12.9.2	Example	111
12.10	w:tabView	112
12.10.1	Attributes	112
12.10.2	Example	112

13 Tips	114
13.1 Open dialog box	114
13.2 ASF Actions	114
13.2.1 Updating content	114
13.2.2 Hide or show	115
13.2.3 Updating CSS class	115
13.2.4 Redirect	116
13.2.5 Get content	116
13.2.6 clear action	116

1 Introduction

Ada Server Faces (ASF) is a user interface framework for Ada web applications. It uses the same design patterns as the Java Server Faces (JSF) that is standardized through the JSR 252, JSR 314 and JSR 344.

Ada Server Faces uses a model which is very close to Java Server Faces. JSF and ASF use a component-based model for the design and implementation of a web application. Like traditional MVC models, the presentation layer is separated from the control and model parts. Unlike the MVC model, JSF and ASF are not request-based meaning there is not a specific controller associated with the request. Instead, each component that is part of the page (view) participate in the control and each component brings a piece of the model.

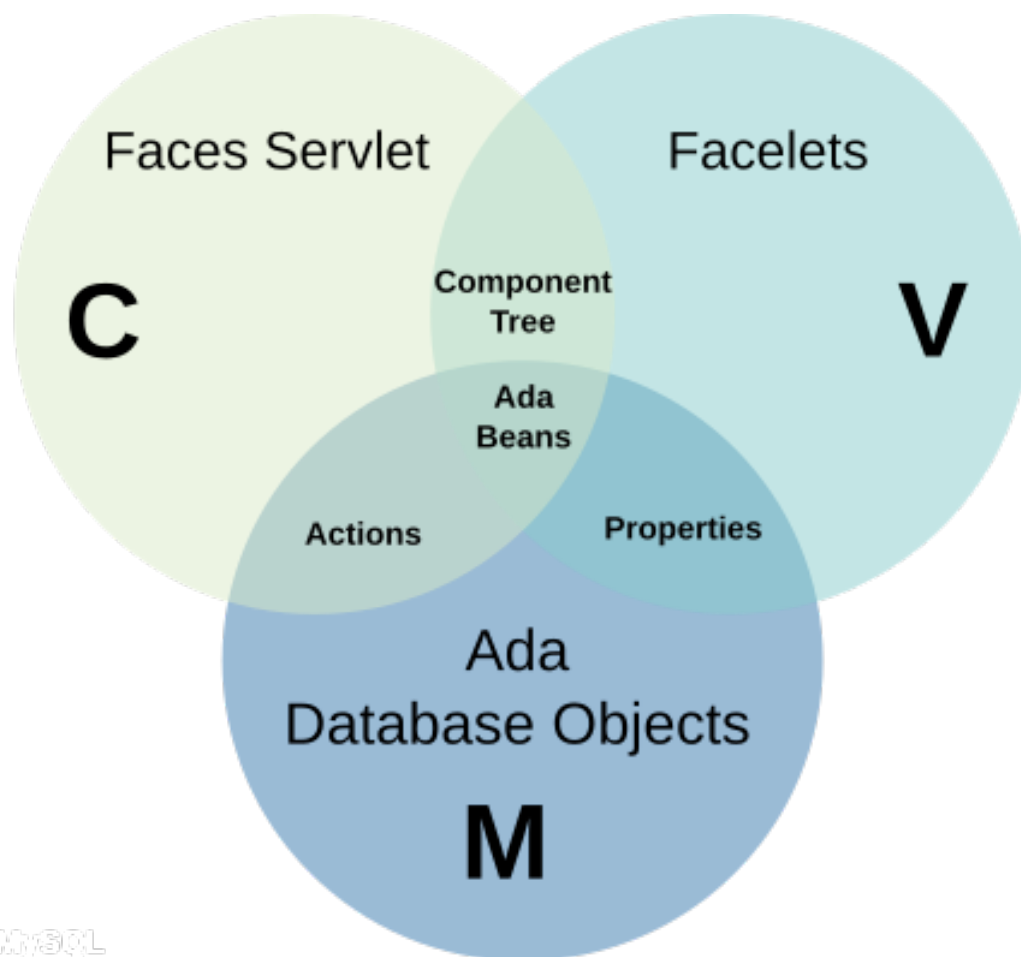


Figure 1: Model View Controller

The Faces Servlet is the controller that handles incoming requests. It builds a component tree from the

facelet view description files. The component tree uses Ada beans which are instances of Ada objects that provide a getter and a setter specific method. The Ada bean object has access to the application data optionally in some database. While the component tree is traversed and used, properties are retrieved from the model to provide values in the view (under the control of the component tree). For some user requests, the controller can invoke actions and trigger some operations in the model for example to load or save some data from the database.

The ASF framework solves a number of problems when designing and implementing a web application:

- It provides reusable UI components to construct User interfaces,
- It defines a framework to validate the data provided by the user,
- It makes a link between the presentation layer and the Ada implementation,
- It defines a page-to-page navigation which describes the user model interactions,
- It allows the creation and building of reusable and custom UI components.

The framework is part of Ada Web Application framework but it can also be used as a standalone framework.

Writing an Ada Server Faces application consists in several steps:

- a set of XHTML file defines the presentation layer by describing the HTML elements of the web page,
- CSS files are written to provide a presentation design to the rendered HTML content,
- Ada beans are written to implement a link between the facelet presentation layer and the data model,
- actions are written in Ada implementation to take into account user interaction and perform work asked by the user,
- XML configuration files are written to configure the Ada Server Faces navigation rules, declare the name of Ada beans.

This document describes how to build the library and how you can use the framework to write interactive web applications.

2 Installation

This chapter explains how to build and install the framework.

2.1 Using Alire

The Ada Server Faces Library is available as several Alire crates to simplify the installation and setup your project. Run the following commands to setup your project to use the library:

```
1 alr index --update-all
2 alr with serverfaces
3 alr with serverfaces_unit
```

If you want to use the Ada Server Faces library in a web server, you must choose a servlet web container that will handle the requests. Two web server implementations are provided:

- AWS
- EWS

and you should run one of the following `alr` command depending on your choice:

```
1 alr with servletada_aws
2 alr with servletada_ews
```

2.2 Without Alire

2.2.1 Before Building

Before building the framework, you will need:

- Ada Utility Library,
- Ada Expression Language Library,
- Ada Security Library,
- Ada Servlet Library,
- XML/Ada
- AWS or EWS

First get, build and install the above components and then get, build and install the Ada Server Faces.

2.2.2 Configuration

The library uses the `configure` script to detect the build environment, check whether XML/Ada, AWS support are available and configure everything before building. If some component is missing, the `configure` script will report an error or it will disable the feature. The `configure` script provides several standard options and you may use:

- `--prefix=DIR` to control the installation directory,
- `--enable-shared` to enable the build of shared libraries,
- `--disable-static` to disable the build of static libraries,
- `--enable-distrib` to build for a distribution and strip symbols,
- `--disable-distrib` to build with debugging support,
- `--enable-coverage` to build with code coverage support (`-fprofile-arcs -ftest-coverage`),
- `--with-ada-util=PATH` to control the installation path of Ada Utility Library,
- `--with-ada-el=PATH` to control the installation path of Ada Expression Language Library,
- `--with-ada-security=PATH` to control the installation path of Ada Security Library,
- `--with-ada-servlet=PATH` to control the installation path of Ada Servlet Library,
- `--help` to get a detailed list of supported options.

In most cases you will configure with the following command:

```
1 ./configure
```

Building to get a shared library can sometimes be a real challenge. With GNAT 2021, you can configure as follows:

```
1 ./configure --enable-shared
```

2.2.3 Build

After configuration is successful, you can build the library by running:

```
1 make
```

After building, it is good practice to run the unit tests before installing the library. The unit tests are built and executed using:

```
1 make test
```

And unit tests are executed by running the `bin/asf_harness` test program.

2.2.4 Installation

The installation is done by running the `install` target:

```
1 make install
```

If you want to install on a specific place, you can change the `prefix` and indicate the installation direction as follows:

```
1 make install prefix=/opt
```

2.3 Using

To use the library in an Ada project, add the following line at the beginning of your GNAT project file:

```
1 with "asf";
```

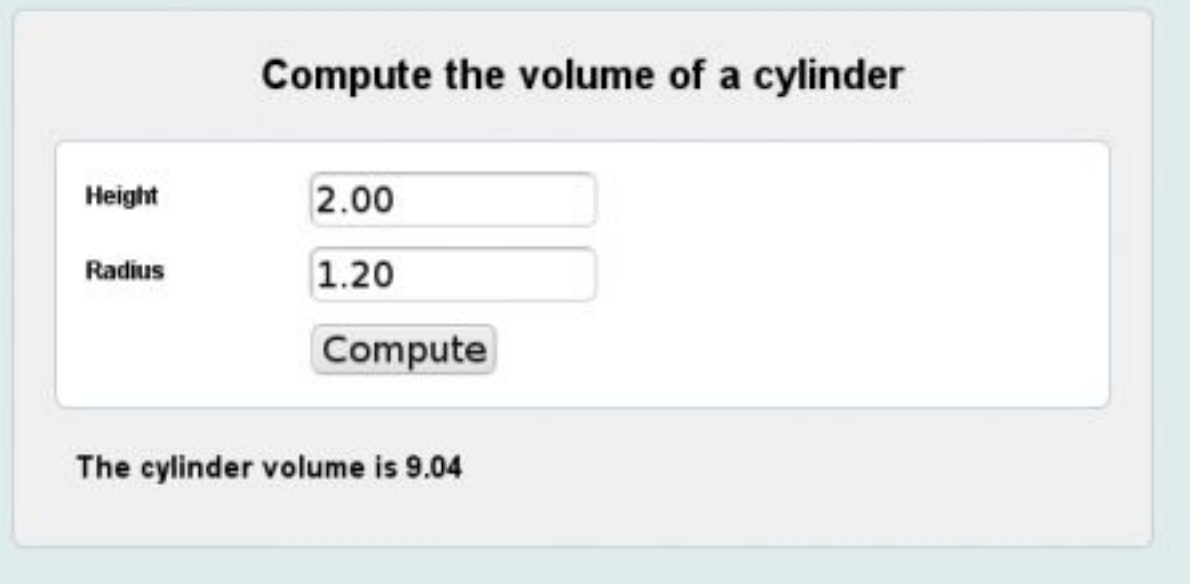
and if you write unit tests for your server faces components, you can benefit from the unit testing support by using the following GNAT project:

```
1 with "asf_unit";
```

3 Tutorial

Ada Server Faces uses a model which is very close to Java Server Faces. JSF and ASF use a component-based model for the design and implementation of a web application. Like traditional MVC models, the presentation layer is separated from the control and model parts. Unlike the MVC model, JSF and ASF are not request-based meaning there is not a specific controller associated with the request. Instead, each component that is part of the page (view) participate in the control and each component brings a piece of the model.

This tutorial shows how to write and use the Ada Server Faces framework to write a simple cylinder form computation. The simple form has two input fields to allow entering the cylinder dimensions (the unit does not matter). It provides a button to submit the form and obtain the result.



The screenshot shows a web form with a light blue border. The title "Compute the volume of a cylinder" is centered at the top in bold black text. Below the title, there are two input fields. The first is labeled "Height" and contains the text "2.00". The second is labeled "Radius" and contains the text "1.20". Below these fields is a button labeled "Compute". At the bottom of the form, the text "The cylinder volume is 9.04" is displayed.

Figure 2: Volume form

3.1 View definition

The presentation part is implemented by a facelet file and a CSS file. The facelet file is an XML file which contains XHTML elements as well as facelets and JSF/ASF components. The facelets and ASF components are specified in their own XML namespace. The ASF components form a tree of components (**UIComponent**) which is then used for displaying and processing form submissions.

At the root of the XML file is an `f:view` component which represents the root of the component tree. The typical page layout looks as follows. Note the `#{contextPath}` notation in the link

reference. This is an EL expression that will be evaluated when the view is displayed (rendered in JSF terminology).

```
1 <f:view contentType="text/html"
2     xmlns:ui="http://java.sun.com/jsf/facelets"
3     xmlns:f="http://java.sun.com/jsf/core"
4     xmlns:c="http://java.sun.com/jstl/core"
5     xmlns:u="http://code.google.com/p/ada-asf/util"
6     xmlns:h="http://java.sun.com/jsf/html">
7   <html xmlns="http://www.w3.org/1999/xhtml">
8     <head>
9       <link media="screen" type="text/css" rel="stylesheet"
10         href="#{contextPath}/themes/main.css"/>
11       <title>Volume Cylinder</title>
12     </head>
13     <body>
14       <div>
15         <h1>Compute the volume of a cylinder</h1>
16         ...
17       </div>
18     </body>
19   </html>
20 </f:view>
```

The form, input fields and submit buttons have to be specified using a JSF/ASF component. The JSF/ASF component will make the link between the presentation (view) and the controller (beans). The `h:form` is the JSF/ASF component that represents our form. Note that there is no need to specify any form `action` attribute: the form action will be managed by JSF/ASF.

The input fields are identified by the `h:input` components. The input field is linked to the bean through the `value` EL expression. This expression specifies the bean name and attribute. When rendering the view, JSF/ASF will fetch the value from the named bean. On form submission, JSF/ASF will populate the bean with the submitted value.

The `h:input` component can contain a `f:converter` element which indicates a conversion operation to call when displaying or before populating the bean value.

```
1 <h:form id='compute'>
2   <dl>
3     <dt>Height</dt>
4     <dd>
5       <h:inputText id='height' size='10' value='#{compute.height}'>
6         <f:converter converterId="float" />
7       </h:inputText>
8     </dd>
9     <dt>Radius</dt>
10    <dd>
11      <h:inputText id='radius' size='10' value='#{compute.radius}'>
12        <f:converter converterId="float"/>
13      </h:inputText>
14    </dd>
15  </dl>
16 </h:form>
```

```
13         </h:inputText>
14     </dd>
15     <dt></dt>
16     <dd>
17         <h:commandButton id='run' value='Compute'
18             action="#{compute.run}"/>
19     </dd>
20 </dl>
21 </h:form>
```

The `#{compute.height}` is an EL expression that refers to the `height` property of the Ada bean identified as `compute`. The value is fetched from the Ada bean when the view is displayed and it is populated when the form is submitted.

At the form end, the `h:commandButton` represents the submit button and the controller action to invoke on form submission. The method to invoke is defined with an EL method expression in the `action` attribute. Before invoking the method, JSF/ASF will verify the submitted values, convert them according to associated converters, populate the beans with the values.

When the facelet file is instantiated, a component tree is created to describe the view and handle the request. The component tree instantiation is specific to each request and the above facelet file will be represented by the following simplified component tree:

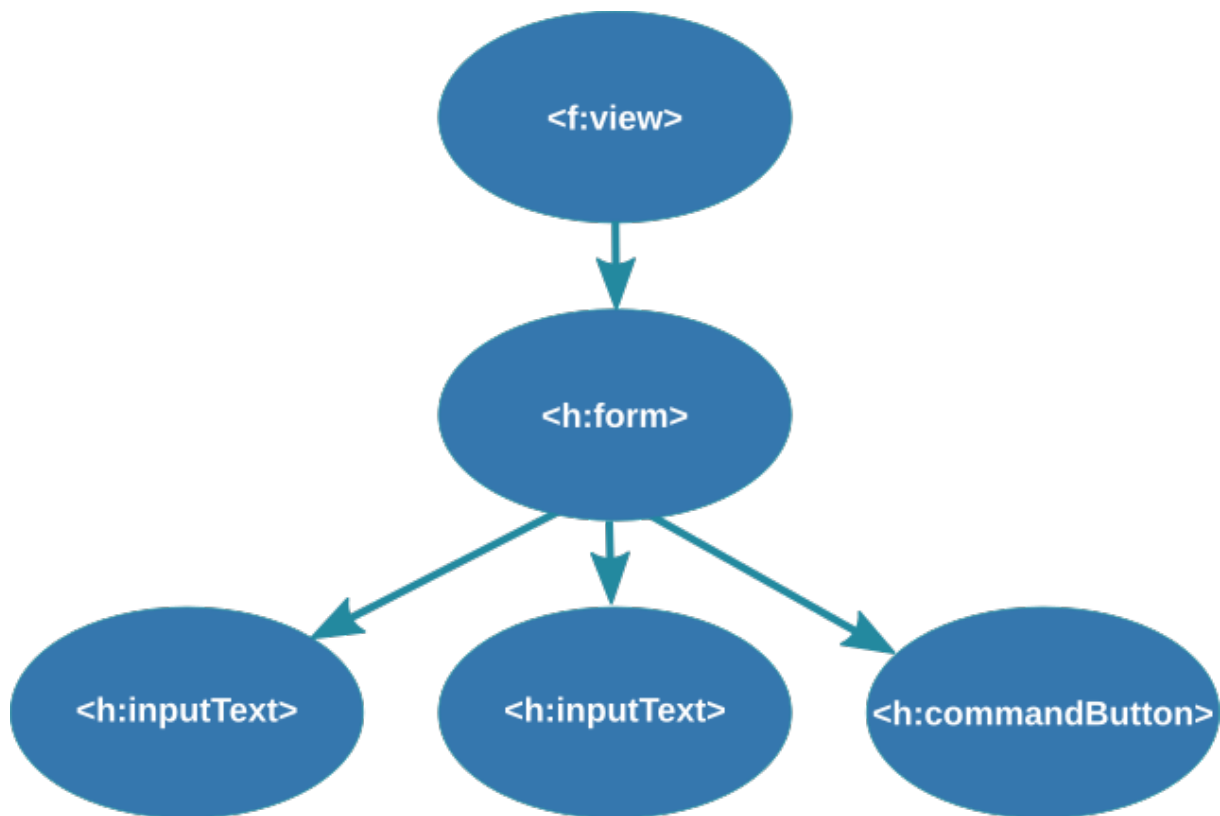


Figure 3: Component tree

The root of the component tree is the `f:view` element which is represented by a `UIView` component. The root component contains the `h:form` element which is represented by a `UIForm` component. That form contains the two `h:inputText` elements which are represented by a `UIInputText` component and they are followed by the `h:commandButton` element represented by a `UICommandButton` component.

The page style is provided by a specific CSS file. The `dl/dt/dd` list is rendered as a table using the following CSS definitions. By changing the CSS file, a new presentation can be provided to users.

```
1 dl {
2   float: left;
3   width: 500px;
4   border: 1px solid #bcd;
5   background-color: #ffffff;
6   padding: 10px;
7   -moz-border-radius: 6px;
8   -webkit-border-radius: 6px 6px;
9 }
10 dt {
11   clear: left;
```

```
12  float: left;
13  font-weight: bold;
14  width: 20%;
15  height: 20px;
16  line-height: 24px;
17  padding: 5px;
18 }
19 dd {
20  float: left;
21  padding: 5px;
22 }
```

3.2 Writing the Cylinder Ada Bean

The Ada bean is an instance of an Ada tagged record that must implement a getter and a setter operation. These operations are invoked through an EL expression. Basically the getter is called when the view is rendered and the setter is called when the form is submitted and validated. The [Bean](#) interface defines the two operations that must be implemented by the Ada type:

```
1  with Util.Beatns.Basic;
2  with Util.Beatns.Objects;
3  ...
4  type Compute_Bean is new Util.Beatns.Basic.Bean with record
5      Height : My_Float := -1.0;
6      Radius : My_Float := -1.0;
7  end record;
8
9  -- Get the value identified by the name.
10 overriding
11 function Get_Value (From : Compute_Bean;
12                     Name : String) return Util.Beatns.Objects.Object;
13
14 -- Set the value identified by the name.
15 overriding
16 procedure Set_Value (From : in out Compute_Bean;
17                     Name : in String;
18                     Value : in Util.Beatns.Objects.Object);
```

The getter and setter will identify the property to get or set through a name. The value is represented by an [Object](#) type that can hold several data types (boolean, integer, floats, strings, dates, ...). The getter looks for the name and returns the corresponding value in an [Object](#) record. Several [To_Object](#) functions help in creating the result value.

```
1  function Get_Value (From : Compute_Bean;
2                      Name : String) return Util.Beatns.Objects.Object is
3  begin
4      if Name = "radius" and From.Radius >= 0.0 then
```

```
5      return Util.Beans.Objects.To_Object (Float (From.Radius));
6
7      elsif Name = "height" and From.Height >= 0.0 then
8          return Util.Beans.Objects.To_Object (Float (From.Height));
9
10     else
11         return Util.Beans.Objects.Null_Object;
12     end if;
13 end Get_Value;
```

The setter is similar.

```
1 procedure Set_Value (From : in out Compute_Bean;
2                     Name  : in String;
3                     Value : in Util.Beans.Objects.Object) is
4 begin
5     if Name = "radius" then
6         From.Radius := My_Float (Util.Beans.Objects.To_Float (Value));
7     elsif Name = "height" then
8         From.Height := My_Float (Util.Beans.Objects.To_Float (Value));
9     end if;
10 end Set_Value;
```

3.3 Register the Cylinder Ada Bean

The next step is to register the cylinder bean and associate it with the `compute` name. There are several ways to do that but for the purpose of this example, there will be a global instance of the bean. That instance must be `aliased` so that we can use the `Access` attributes.

```
1 Bean : aliased Compute_Bean;
```

The Ada bean is registered on the application object by using the `Set_Global` procedure. This creates a global binding between a name and an `Object` record. In our case, the object will hold a reference to the Ada bean.

```
1 App : aliased ASF.Applications.Main.Application;
2 ...
3 App.Set_Global ("compute", Util.Beans.Objects.To_Object (Bean'
4                 Unchecked_Access));
```

3.4 Command buttons and method expression

This submit button can be associated with an action that will be executed when the button is pressed. The EL expression is the mechanism by which we create a binding between the XHTML presentation page and the component implemented in Java or Ada. A method expression is a simple EL expression

that represents a bean and a method to invoke on that bean. This method expression represent our action.

A typical use is on the **h:commandButton** component where we can specify an action to invoke when the button is pressed. This is written as:

```
1 <h:commandButton id='run' value='Compute'
2   action="#{compute.run}"/>
```

The method expression `#{compute.run}` indicates to execute the method `run` of the bean identified by `compute`.

3.5 Method Bean Declaration

Java implements method expressions by using reflection. It is able to look at the methods implemented by an object and then invoke one of these method with some parameters. Since we cannot do this in Ada, some developer help is necessary.

For this an Ada bean that implements an action must implement the `Method_Bean` interface. If we take the `Compute_Bean` type, we just have to extend that interface and implement the `Get_Method_Bindings` function. This function will indicate the methods which are available for an EL expression and somehow how they can be called.

```
1 with Util.Beans.Methods;
2 ...
3 type Compute_Bean is new Util.Beans.Basic.Bean
4   and Util.Beans.Methods.Method_Bean with record
5     Height : My_Float := -1.0;
6     Radius : My_Float := -1.0;
7     Volume: My_Float := -1.0;
8   end record;
9   -- This bean provides some methods that can be used in a
10  Method_Expression
11  overriding
12  function Get_Method_Bindings (From : in Compute_Bean)
13    return Util.Beans.Methods.Method_Binding_Array_Access;
```

Our Ada type can now define a method that can be invoked through a method expression. The action bean always receives the bean object as an **in out** first parameter and it must return the action outcome as an `Unbounded_String` also as **in out**.

```
1 procedure Run (From : in out Compute_Bean;
2               Outcome : in out Unbounded_String);
```

3.6 Implement the action

The implementation of our action is quite simple. The `Radius` and `Height` parameters submitted in the form have been set on the bean before the action is called. We can use them to compute the cylinder volume.

```
1 procedure Run (From      : in out Compute_Bean;  
2                Outcome : in out Unbounded_String) is  
3   V : My_Float;  
4 begin  
5   V := (From.Radius * From.Radius);  
6   V := V * From.Height;  
7   From.Volume := V * 3.141;  
8   Outcome := To_Unbounded_String ("compute");  
9 end Run;
```

3.7 Define the action binding

To be able to call the `Run` procedure from an EL method expression, we have to create a binding object. This binding object will hold the method name as well as a small procedure stub that will somehow tie the method expression to the procedure. This step is easily done by instantiating the `ASF.Events.Actions.Action_Method.Bind` package.

```
1 with ASF.Events.Actions;  
2 ...  
3 package Run_Binding is  
4   new ASF.Events.Actions.Action_Method.Bind  
5     (Bean  => Compute_Bean,  
6      Method => Run,  
7      Name   => "run");
```

3.8 Register and expose the action bindings

The last step is to implement the `Get_Method_Bindings` function. Basically it has to return an array of method bindings which indicate the methods provided by the Ada bean.

```
1 Binding_Array : aliased constant Util.Beans.Methods.  
   Method_Binding_Array  
2 := (Run_Binding.Proxy'Unchecked_Access, Run_Binding.Proxy'  
   Unchecked_Access);  
3  
4 overriding  
5 function Get_Method_Bindings (From : in Compute_Bean)  
6   return Util.Beans.Methods.Method_Binding_Array_Access is
```

```
7 begin
8   return Binding_Array'Unchecked_Access;
9 end Get_Method_Bindings;
```

3.9 What happens now?

When the user presses the **Compute** button, the browser will submit the form and the ASF framework will do the following:

- It will check the validity of input parameters,
- It will save the input parameters on the `compute` bean by using the `Set_Value` procedure,
- It will execute the method expression `#{compute.run}`:
 - It calls the `Get_Method_Bindings` function to get a list of valid method,
 - Having found the right binding, it calls the binding procedure
 - The binding procedure invokes the `Run` procedure on the object.

3.10 Application Initialization

An Ada Server Faces Application is represented by the `Application` type which holds all the information to process and dispatch requests. First, let's declare a variable that represents our application.

"Note: for the purpose of this article, we will assume that every variable is declared at some package level scope. If those variables are declared in another scope, the `Access` attribute should be replaced by `Unchecked_Access`."

```
1 with ASF.Applications.Main;
2 ...
3 App : aliased ASF.Applications.Main.Application;
```

To initialize the application, we will also need some configuration properties and a factory object. The configuration properties are used to configure the various components used by ASF. The factory allows to customize some behavior of Ada Server Faces. For now, we will use the default factory.

```
1 with ASF.Applications;
2 ...
3 C      : ASF.Applications.Config;
4 Factory : ASF.Applications.Main.Application_Factory;
```

The initialization requires to define some configuration properties. The `VIEW_EXT` property indicates the URI extension that are recognized by ASF to associate an XHTML file (the `compute.html` corresponds to the XHTML file `compute.xhtml`). The `VIEW_DIR` property defines the root directory where the XHTML files are stored.

```
1 C.Set (ASF.Applications.VIEW_EXT, ".html");
2 C.Set (ASF.Applications.VIEW_DIR, "samples/web");
3 C.Set ("web.dir", "samples/web");
4 App.Initialize (C, Factory);
```

3.11 Servlets

Ada Server Faces uses the Ada Servlet framework to receive and dispatch web requests. It provides a `Faces_Servlet` servlet which can be plugged in the servlet container. This servlet is the entry point for ASF to process incoming requests. We will also need a `File_Servlet` to process the static files. Note that these servlets are implemented using tagged records and you can easily override the entry points (`Do_Get` or `Do_Post`) to implement specific behaviors.

```
1 with ASF.Servlets.Faces;
2 with ASF.Servlets.Files;
3 ...
4 Faces : aliased ASF.Servlets.Faces.Faces_Servlet;
5 Files : aliased ASF.Servlets.Files.File_Servlet;
```

The servlet instances are registered in the application.

```
1 App.Add_Servlet (Name => "faces", Server => Faces'Access);
2 App.Add_Servlet (Name => "files", Server => Files'Access);
```

Once registered, we have to define a mapping that tells which URI path is mapped to the servlet.

```
1 App.Add_Mapping (Name => "faces", Pattern => "*.html");
2 App.Add_Mapping (Name => "files", Pattern => "*.css");
```

For the purpose of debugging, ASF provides a servlet filter that can be plugged in the request processing flow. The `Dump_Filter` will produce a dump of the request with the headers and parameters.

```
1 with ASF.Filters.Dump;
2 ...
3 Dump : aliased ASF.Filters.Dump.Dump_Filter;
```

The filter instance is registered as follows:

```
1 App.Add_Filter (Name => "dump", Filter => Dump'Access);
```

And a mapping is defined to tell which URL will trigger the filter.

```
1 App.Add_Filter_Mapping (Name => "dump", Pattern => "*.html");
```

3.12 Application and Web Container

The application object that we created is similar to a Java Web Application packaged in a WAR file. It represents the application and it must be deployed in a Web Container. With Ada Server Faces this is almost the same, the application needs a Web container. By default, ASF provides a web container based on the excellent Ada Web Server implementation (other web containers could be provided in the future based on other web servers).

```
1 with ASF.Server.Web;  
2 ...  
3 WS : ASF.Server.Web.AWS_Container;
```

To register the application, we indicate the URI context path to which the application is associated. Several applications can be registered, each of them having a unique URI context path.

```
1 CONTEXT_PATH : constant String := "/volume";  
2 ...  
3 WS.Register_Application (CONTEXT_PATH, App'Access);
```

3.13 Global Objects

An application can provide some global objects which will be available during the request processing through the EL expression. First, we will expose the application context path which allows to write links in the XHTML page that match the URI used for registering the application in the web container.

```
1 App.Set_Global ("contextPath", CONTEXT_PATH);
```

Below is an example of use of this `contextPath` variable:

```
1 <link media="screen" type="text/css" rel="stylesheet"  
2 href="#{contextPath}/themes/main.css"/>
```

Now, we will register the bean that we created for our application! This was explained in the Ada beans previous article.

```
1 with Volume;  
2 ...  
3 Bean : aliased Volume.Compute_Bean;  
4 ...  
5 App.Set_Global ("compute", Util.Beans.Objects.To_Object (Bean'Access  
6 ));
```

“Note: For the purpose of this example, the `Compute_Bean` is registered as a global object. This means that it will be shared by every request. A future article will explain how to get a session or a request bean as in Java Server Faces.”

3.14 Starting the server

Once the application is registered, we can start our server. Note that since Ada Web Server starts several threads that listen to requests, the `Start` procedure does not block and returns as soon as the server is started. The delay is necessary to let the server wait for requests during some time.

```
1 WS.Start;  
2 delay 1000.0;
```

3.15 What happens to a request?

Let's say the server receives a HTTP GET request on `/volume/compute.html`. Here is what happens:

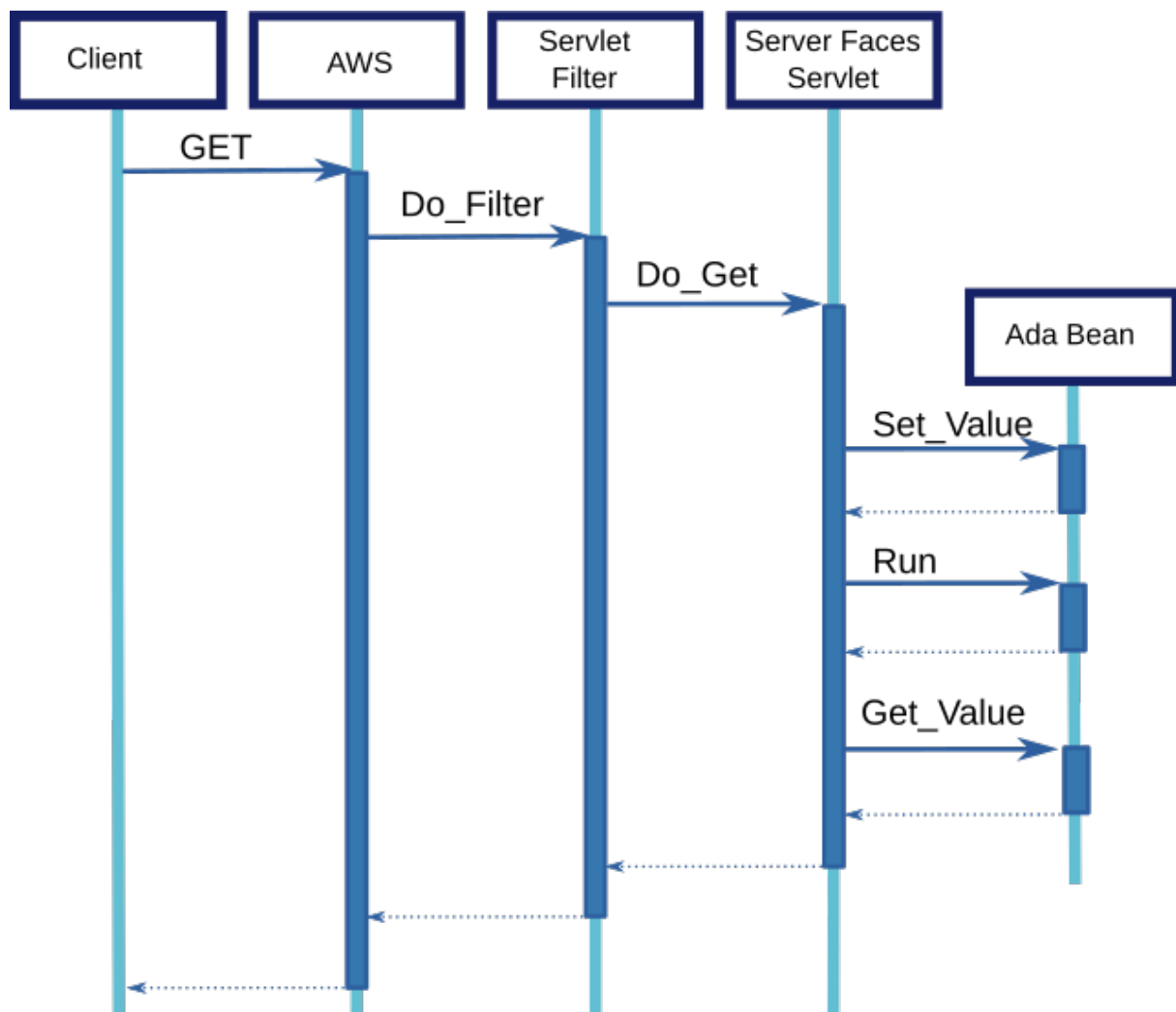


Figure 4: Volume ASF Flow

- Ada Web Server receives the HTTP request,
- It identifies the application that matches `/volume` (our context path) and gives the control to it,
- The application identifies the servlet that processes the remaining URI, which is `compute.html`,
- It gives the control to the `Dump_Filter` filter and then to the `Faces_Servlet` servlet,
- The faces servlet identifies the XHTML facelet file and reads the `compute.xhtml` file,
- ASF builds the component tree that describes the page and invokes the render response phase,
- While rendering, the EL expressions such as `#{compute.radius}` are evaluated and the value is obtained on our `Bean` global instance,
- The HTML content is produced as part of the rendering process and returned by AWS.

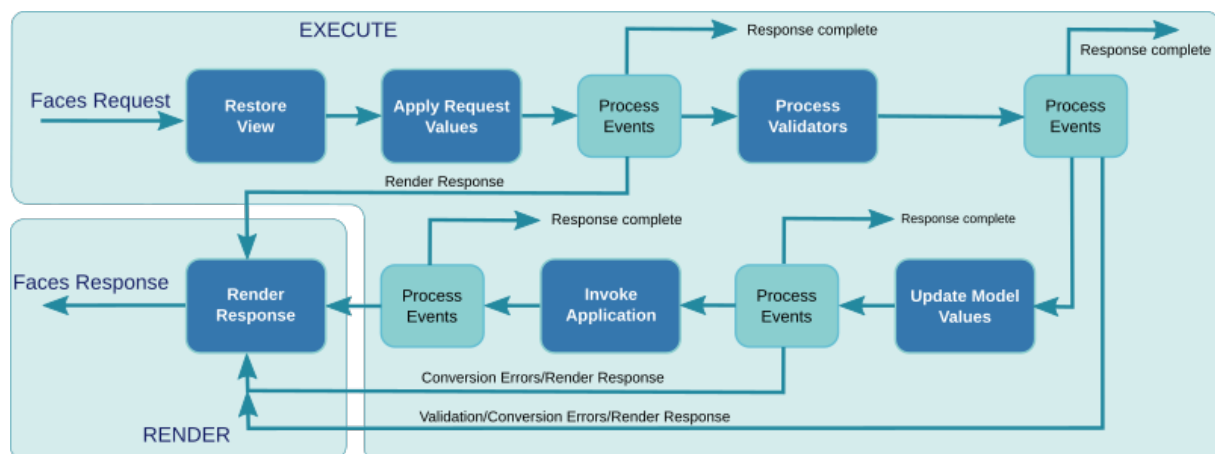
4 Request Processing Lifecycle

The request lifecycle is decomposed in two groups: an execute group handles the incoming request and a render group handles the production of the response. The execution group is decomposed in several phases with some of them being executed. After each phase, some decision is made to proceed, to render the response or finish the request when it is completed. The request lifecycle handles both initial requests (an HTTP GET) and postbacks (an HTTP POST).

The **Restore View** phase is always executed to handle the request and it is responsible for building the components in the view. The XHTML file associated with the view is read or obtained from the facelet cache and the components described by the XHTML tags are created to form the component tree.

The **Apply Request Values** phase is then handled to obtain the request parameters.

The **Process Validators** phase executes the input validators on the component tree to validate the request parameters. If a parameter is invalid, some message can be posted and associated with the component that triggered it.



The **Update Model Values** phase invokes the **Set_Value** procedure on every Ada bean for which an input parameter was submitted and was valid. The Ada bean may raise an exception and an error will be associated with the associated component.

The **Invoke Application** phase executes the Ada bean actions that have been triggered by the **f:viewAction** for an initial request or by the postback actions. The Ada bean method is invoked so that it gets the control of the request and it returns an outcome that serves for the request navigation.

The **Render Response** phase is the final phase that walks the component tree and renders the HTML response.

5 Converters

The `ASF.Converters` package defines an interface used by the conversion model to translate an object into a string when formatting the response and translate a string into an object during the apply request or validation phases (JSF postback).

The `Converter` interface defines two functions for the conversion of a object to a string and (`To_String`) and convert back a string to an object (`To_Object`). See JSR 314 - JavaServer Faces Specification 3.3.2 Converter (`To_String` is the JSF `getAsString` method and `To_Object` is the JSF `getAsObject` method)

5.1 Date converter

The `ASF.Converters.Dates` defines the date converter to format a date object into a localized representation. It is automatically created when the `f:convertDateTime` tag is used in the facetlet file, for example as follows:

```
1 <h:outputText value='{messages.today}'>
2   <f:convertDateTime dateStyle="short"/>
3 </h:outputText>
```

5.2 Number converter

The `ASF.Converters.Numbers` provides a floating point number converter. It can be used to print floating point numbers in various formats.

5.3 Size converter

The `ASF.Converters.Sizes` defines a converter to display a file size in bytes, kilo bytes, mega bytes or giga bytes.

6 Validators

The validators participate in the validation of submitted values during the request validation phase. They are responsible for checking whether the input parameter complies with the validation rules associated with the input field. The validator is expected to raise an exception and an error message is then associated with the faulty input field.

The validator is described in the XHTML file by using one of the following JSF core components:

Component	Validation type
f:validateLength	The input field must have a minimum/maximum length
f:validateLongRange	The input field must be a number in the given range
f:validator	An Ada registered or custom validator is used
f:validateRegex	The input field must match the regular expression

A validator instance must implement the `ASF.Validators.Validator` interface. It only needs to implement the `Validate` procedure which gets the UI component, the faces request context and the submitted value. It must verify the value according to the validator's rule and the UI component. When the value is incorrect, it must set an error message in the UI component so that some user friendly message is reported. In case of error, it must also raise the `Invalid_Value` exception.

6.1 Length validator

The `ASF.Validators.Texts.Length_Validator` implements the validator for the `<f:validateLength>` XHTML validator.

6.2 Regex validator

The `ASF.Validators.Texts.Regex_Validator` implements the validator for the `<f:validateRegex>` XHTML validator.

6.3 Range validator

The `ASF.Validators.Numbers` defines various number oriented validators.

6.4 Components

The [ASF.Components](#) describes the components that form the tree view. Each component has attributes and children. Children represent sub-components and attributes control the rendering and behavior of the component.

The component tree is created from the [ASF.Views](#) tag nodes for each request. Unlike tag nodes, the component tree is not shared.

7 Facelet Components

The facelets is the default view declaration language that uses XML and XHTML. It is a composition and templating framework that allows to create the component tree.

The facelet components are defined in the following namespace:

```
1 xmlns:ui="http://java.sun.com/jsf/facelets"
```

The facelet components are implemented by the `ASF.Views.Nodes.Facelets` package which defines the pre-defined tags for composing a view. Nodes of this package are instantiated when the facelet XML tag is found when reading the XHTML view description.

7.1 ui:composition

Defines a composition that optionally uses a template, as outlined in the description of the ui tag library. Multiple compositions can use the same template, thus encapsulating and reusing layout. JSF disregards everything outside of the composition, which lets developers embed compositions in well-formed XHTML pages that can be viewed in an XHTML viewer, such as Dreamweaver or a browser, without including extraneous elements such as head and body.

7.1.1 Attributes

Name	Required	Type	Description
template	false	String	A URI that points to a template, also known as a layout, that inserts pieces of the page defined in the composition.

7.1.2 Example

```
1 <ui:composition xmlns:ui="http://java.sun.com/jsf/facelets">
2
3     <h2>Title #{empty name ? '?' : name}</h2>
4     <p>
5         <ui:insert name="description">
6             There is no description
7         </ui:insert>
8     </p>
9     <h2>List</h2>
10    <ul style='padding-left: 20px;'>
```

```

11         <ui:insert name="list">
12             <li>
13                 There is no list.
14             </li>
15         </ui:insert>
16     </ul>
17 </ui:composition>

```

7.2 ui:decorate

The decorate tag is identical to the composition tag, except that `ui:decorate`, unlike `ui:composition`, does not disregard all content outside of the tag. The decorate is useful when you want to decorate some content in a page, for example, you might want to decorate a list of items.

7.2.1 Attributes

Name	Required	Type	Description
template	true	String	A URI that points to a template, also known as a layout, that inserts pieces of the page defined in the decorator.

7.2.2 Example

```

1 <ui:decorate xmlns:ui="http://java.sun.com/jsf/facelets"
2     template="/facelet/composition.xhtml">
3     <ui:param name="name" value="decorate"/>
4     <ui:define name="description">
5         The decorate tag allows to use a template and insert data
6         at various places.
7     </ui:define>
8     <ui:define name="list">
9         <li>The decorate tag uses a template</li>
10        <li>It includes optional ui:param elements</li>
11        <li>It fills the template by using the ui:define element.</li>
12    </ui:define>
13 </ui:decorate>

```

7.3 ui:define

The define tag defines content that is inserted into a page by a template. The define tag can be used inside `ui:composition`, `ui:component`, `ui:decorate`, and `ui:fragment` tags.

7.3.1 Attributes

Name	Required	Type	Description
name	true	String	Assigns a name to the content inside a define tag. That name is used by corresponding ui:insert tags in a template that insert the named content into a page.

7.3.2 Example

```
1 <ui:decorate xmlns:ui="http://java.sun.com/jsf/facelets"
2     template="/facelet/composition.xhtml">
3     <ui:param name="name" value="decorate"/>
4     <ui:define name="description">
5         The decorate tag allows to use a template and insert data
6         at various places.
7     </ui:define>
8     <ui:define name="list">
9         <li>The decorate tag uses a template</li>
10        <li>It includes optional ui:param elements</li>
11        <li>It fills the template by using the ui:define element.</li>
12    </ui:define>
13 </ui:decorate>
```

7.4 ui:include

Use this tag—which is very similar to JSP's `jsp:include` to encapsulate and reuse content among multiple XHTML pages. There are three things this tag can include: plain XHTML, and XHTML pages that have either a composition tag or a component tag.

You supply a filename, through `ui:include`'s `src` attribute for JSF to include. That filename is relative to the XHTML file that was rendered as a result of the last request. So, for example, if JSF loaded the view `login.xhtml`, and that file included `pageDecorations/header.xhtml`, and `pageDecorations/header.xhtml` included `companyLogo.xhtml`, then `companyLogo.xhtml` will not be found if it's in the `pageDecorations` directory, because `companyLogo.xhtml` has to be in the same directory as `login.xhtml`.

7.4.1 Attributes

Name	Required	Type	Description
src	true	String	The filename of an XHTML page to include. The filename is relative to the XHTML page that was originally loaded.

7.4.2 Example

```
1 <div xmlns:ui="http://java.sun.com/jsf/facelets">
2   <ui:include src="composition.xhtml">
3     <ui:param name="name" value="include"/>
4   </ui:include>
5   <ui:include src="../jstl/if.xhtml"/>
6 </div>
```

7.5 ui:insert

Inserts content into a template. That content is defined with the `ui:define` tag in either a `ui:composition`, `ui:component`, `ui:decorate`, or `ui:fragment`.

7.5.1 Attributes

Name	Required	Type	Description
name	true	String	The fragment name to insert.

7.5.2 Example

```
1 <ui:composition xmlns:ui="http://java.sun.com/jsf/facelets">
2
3   <h2>Title #{empty name ? '?' : name}</h2>
4   <p>
5     <ui:insert name="description">
6       There is no description
7     </ui:insert>
8   </p>
9   <h2>List</h2>
10  <ul style='padding-left: 20px;'>
11    <ui:insert name="list">
12      <li>
```

```
13         There is no list.
14     </li>
15 </ui:insert>
16 </ul>
17 </ui:composition>
```

7.6 ui:param

Use this tag to pass parameters to an included file (using `ui:include`), or a template (linked to either a composition or decorator). Embed `ui:param` tags in either `ui:include`, `ui:composition`, or `ui:decorate` to pass the parameters.

7.6.1 Attributes

Name	Required	Type	Description
name	true	String	The name of the parameter.
value	true	String	The value of the parameter. Notice that this attribute's value can be an EL expression, which means that you can pass objects to either an included file or a template.

7.6.2 Example

```
1 <ui:decorate xmlns:ui="http://java.sun.com/jsf/facelets"
2     template="/facelet/composition.xhtml">
3     <ui:param name="name" value="decorate"/>
4     <ui:define name="description">
5         The decorate tag allows to use a template and insert data
6         at various places.
7     </ui:define>
8     <ui:define name="list">
9         <li>The decorate tag uses a template</li>
10        <li>It includes optional ui:param elements</li>
11        <li>It fills the template by using the ui:define element.</li>
12    </ui:define>
13 </ui:decorate>
```

8 JSTL Components

The JSTL components are defined in the following namespace:

```
1 xmlns:c="http://java.sun.com/jstl/core"
```

The facelet components are implemented by the `ASF.Views.Nodes.Facelets` package which defines the pre-defined tags for composing a view. Nodes of this package are instantiated when the facelet XML tag is found when reading the XHTML view description.

8.1 c:choose

This tag associates a parameter name-value pair with the nearest parent `UIComponent`. A `UIComponent` is created to represent this name-value pair, and stored as a child of the parent component; what effect this has depends upon the renderer of that parent component.

Unless otherwise specified, all attributes accept static values or EL expressions.

8.1.1 Attributes

No attributes are defined.

8.1.2 Example

```
1 <c:choose>
2   <c:when test="#{not empty compute.radius}">
3     Radius is not empty: #{compute.radius}.
4   </c:when>
5   <c:when test="#{empty compute.radius}">
6     Radius is empty
7   </c:when>
8 </c:choose>
```

8.2 c:if

Simple conditional tag, which evaluates its body if the supplied condition is true and optionally exposes a Boolean scripting variable representing the evaluation of this condition.

8.2.1 Attributes

Name	Type	Description
test	Boolean	The test condition that determines whether or not the body content should be processed.
var	String	Name of the exported scoped variable for the resulting value of the test condition. The type of the scoped variable is Boolean.

8.2.2 Example

```
1 <c:if test="#{not empty compute.radius}">
2   Radius is not empty: #{compute.radius}.
3 </c:if>
4 <c:if test="#{empty compute.radius}">
5   Radius is empty
6 </c:if>
```

8.3 c:otherwise

Subtag of `choose` that follows `when` tags and runs only if all of the prior conditions evaluated to 'false'.

8.3.1 Attributes

This tag has no attribute.

8.3.2 Example

```
1 <c:choose>
2   <c:when test="#{not empty compute.radius}">
3     Radius is not empty: #{compute.radius}.
4   </c:when>
5   <c:otherwise>
6     Radius is empty
7   </c:otherwise>
8 </c:choose>
```

8.4 c:set

Sets the result of an expression evaluation based on the value of the attributes.

8.4.1 Attributes

Name	Type	Description
var	String	Name of the variable.
value	ValueExpression	Expression to be evaluated.

8.4.2 Example

```
1 <c:set var="name" value="23"/>
2 #{name}
3 <c:set var="name" value="#{23 + 1}"/>
4 #{name}
```

8.5 c:when

Subtag of `choose` that includes its body if its condition evaluates to 'true'.

8.5.1 Attributes

Name	Type	Description
test	Boolean	The test condition that determines whether or not the body content should be processed.

8.5.2 Example

```
1 <c:choose>
2   <c:when test="#{not empty compute.radius}">
3     Radius is not empty: #{compute.radius}.
4   </c:when>
5   <c:when test="#{empty compute.radius}">
```

```
6     Radius is empty
7   </c:when>
8 </c:choose>
```

The ASF.Views.Nodes.Core package defines some pre-defined core tag nodes which are mapped in the following namespaces:

```
1 xmlns:c="http://java.sun.com/jstl/core"
2 xmlns:ui="http://java.sun.com/jsf/facelets"
3 xmlns:fn="http://java.sun.com/jsp/jstl/functions"
```

9 Core Components

The facelets is the default view declaration language that uses XML and XHTML. It is a composition and templating framework that allows to create the component tree.

The core components are defined in the following namespace:

```
1 xmlns:f="http://java.sun.com/jsf/core"
```

The core components are implemented by the `ASF.Components.Core` package which defines the `UIComponent` that describes the various elements provided by the core components. These components are instantiated when the view is created from the facelet tree that was read from the XHTML view description.

9.1 f:attribute

This tag associates an attribute with the nearest parent `UIComponent`. When the value is not an EL expression, this tag has the same effect as calling `Component.Set_Attribute (Name, Value)`. When the attribute name specified matches a standard property of the component, that property is set. However it is also valid to assign attributes to components using any arbitrary name; the component itself won't make any use of these but other objects such as custom renderers, validators or action listeners can later retrieve the attribute from the component by name.

When the value is an EL expression, this tag has the same effect as calling `Component.Set_Attribute (Tag, Value)`. A call to method `Component.Get_Attribute (Name)` will then cause that expression to be evaluated and the result of the expression is returned, not the original EL expression string.

See the `ASF.Components.Base` package for more details.

Unless otherwise specified, all attributes accept static values or EL expressions.

9.1.1 Attributes

Name	Required	Type	Description
name	false	String	The name of the attribute.
value	false	String	The attribute's value.

9.1.2 Example

```
1 <div xmlns:f="http://java.sun.com/jsf/core"
2     xmlns:h="http://java.sun.com/jsf/html">
3     <h:panelGroup layout="block">
4         <h:outputText value="Hello world!">
5             <!-- The 'styleClass' attribute is applied
6                 on the h:outputText -->
7             <f:attribute name="styleClass" value="left"/>
8         </h:outputText>
9         <p>
10             <!-- The 'styleClass' attribute is applied
11                 on the h:panelGroup -->
12             <f:attribute name="styleClass"
13                         value="ui-widget asf-container ui-corner-all"
14                         />
15         </p>
16     </h:panelGroup>
17 </div>
```

9.2 f:convertDateTime

This tag registers an instance of a `Date_Time_Converter`, and associates it with the nearest parent `UIComponent`.

9.2.1 Attributes

Name	Required	Type	Description
dateStyle	false	String	Predefined formatting style which determines how the date component of a date string is to be formatted and parsed. Applied only if type is “date” or “both”. Valid values are “default”, “short”, “medium”, “long”, and “full”. Default value is “default”.
locale	false	String	Locale whose predefined styles for dates and times are used during formatting or parsing. If not specified, the Locale returned by <code>FacesContext.Get_View_Root().Get_Locale()</code> will be used. Value must be either a VB expression that evaluates to a valid String locale.

Name	Required	Type	Description
pattern	false	String	Custom formatting pattern which determines how the date/time string should be formatted and parsed.
timeStyle	false	String	Predefined formatting style which determines how the time component of a date string is to be formatted and parsed. Applied only if type is “time” or “both”. Valid values are “default”, “short”, “medium”, “long”, and “full”. Default value is “default”.
timeZone	false	String	Time zone in which to interpret any time information in the date String. Value must be either a VB expression that evaluates to a valid string that is a timezone ID.
type	false	String	Specifies what contents the string value will be formatted to include, or parsed expecting. Valid values are “date”, “time”, and “both”. Default value is “date”.

9.2.2 Example

```

1  <div xmlns:f="http://java.sun.com/jsf/core"
2    xmlns:h="http://java.sun.com/jsf/html">
3    <dl>
4      <dt>No converter</dt>
5      <dd>
6        <!-- pi is a float, use a default converter -->
7        #{messages.today}
8      </dd>
9      <dt>With converter</dt>
10     <dd>
11       <h:outputText value='#{messages.today}'>
12         <!-- use the 'float' converter defined by the
13           application -->
14         <f:convertDateTime dateStyle="short"/>
15       </h:outputText>
16     </dd>
17   </dl>
18   <h4>dateStyle</h4>
19   <dl class='list'>

```

```
19      <dt>short</dt>
20      <dd>
21          <h:outputText value='{messages.today}'>
22              <f:convertDateTime dateStyle="short"/>
23          </h:outputText>
24      </dd>
25      <dt>medium</dt>
26      <dd>
27          <h:outputText value='{messages.today}'>
28              <f:convertDateTime dateStyle="medium"/>
29          </h:outputText>
30      </dd>
31      <dt>long</dt>
32      <dd>
33          <h:outputText value='{messages.today}'>
34              <f:convertDateTime dateStyle="long"/>
35          </h:outputText>
36      </dd>
37      <dt>full</dt>
38      <dd>
39          <h:outputText value='{messages.today}'>
40              <f:convertDateTime dateStyle="full"/>
41          </h:outputText>
42      </dd>
43  </dl>
44  <h4>timeStyle</h4>
45  <dl class='list'>
46      <dt>short</dt>
47      <dd>
48          <h:outputText value='{messages.today}'>
49              <f:convertDateTime timeStyle="short"/>
50          </h:outputText>
51      </dd>
52      <dt>medium</dt>
53      <dd>
54          <h:outputText value='{messages.today}'>
55              <f:convertDateTime timeStyle="long"/>
56          </h:outputText>
57      </dd>
58  </dl>
59 </div>
```

9.3 f:converter

This tag creates an instance of the specified Converter, and associates it with the nearest parent UIComponent.

Register a named Converter instance on the UIComponent associated with the closest parent UIComponent custom action.

9.3.1 Attributes

Name	Required	Type	Description
converterId	true	String	The converter's registered identifier.

9.3.2 Example

```
1 <div xmlns:f="http://java.sun.com/jsf/core"
2     xmlns:h="http://java.sun.com/jsf/html">
3     <dl>
4         <dt>No converter</dt>
5         <dd>
6             <!-- pi is a float, use a default converter -->
7             #{compute.pi}
8         </dd>
9         <dt>With converter</dt>
10        <dd>
11            <h:outputText value=#{compute.pi}>
12                <!-- use the 'float' converter defined by the
13                     application -->
14                <f:converter converterId="float"/>
15            </h:outputText>
16        </dd>
17    </dl>
</div>
```

9.4 f:facet

This tag allows to register the named facet to the closest parent UIComponent.

Facets are used by some components to render and control specific parts of the component: for example a table header or footer, the column header, the widget panel titles. Example of components that use facets: `h:dataTable`, `h:panelGrid`, `w:panel`.

Warning: if a facet is used within a component that does not recognize the name, the facet content will be ignored.

9.4.1 Attributes

Name	Required	Type	Description
name	true	String	The facet name.

9.4.2 Example

```
1 <div xmlns:f="http://java.sun.com/jsf/core"
2     xmlns:h="http://java.sun.com/jsf/html">
3     <f:facet name="title">
4         title
5     </f:facet>
6 </div>
```

9.5 f:metadata

Declares the metadata facet for the view.

9.5.1 Attributes

None.

9.5.2 Example

```
1 <f:view xmlns:f="http://java.sun.com/jsf/core">
2     <f:metadata>
3         <f:viewParam id='height' value='#{compute.height}'>
4             <f:converter converterId="float" />
5         </f:viewParam>
6         <f:viewParam id='radius' value='#{compute.radius}'>
7             <f:converter converterId="float" />
8         </f:viewParam>
9         <f:viewAction action="#{compute.run}" />
10    </f:metadata>
11    <dl>
12        <dt>Height</dt>
13        <dd>
14            #{compute.height}
15        </dd>
16        <dt>Radius</dt>
17        <dd>
18            #{compute.radius}
19        </dd>
```

```
20         <dt>Volume</dt>
21         <dd>
22             #{compute.volume}
23         </dd>
24     </dl>
25 </f:view>
```

9.6 f:param

This tag associates a parameter name-value pair with the nearest parent `UIComponent`. A `UIComponent` is created to represent this name-value pair, and stored as a child of the parent component; what effect this has depends upon the renderer of that parent component.

Unless otherwise specified, all attributes accept static values or EL expressions.

9.6.1 Attributes

Name	Required	Type	Description
name	false	String	The name under which the value is stored.
value	false	String	The value of this component.
id	false	String	Get a string which uniquely identifies this <code>UIComponent</code> within the nearest ancestor naming component.

9.6.2 Example

```
1 <div xmlns:f="http://java.sun.com/jsf/core"
2     xmlns:h="http://java.sun.com/jsf/html">
3     <h:outputFormat value="The value of PI is {0} and 2 x PI is {1}.">
4         <f:param value="#{compute.pi}"/>
5         <f:param value="#{compute.pi + compute.pi}"/>
6     </h:outputFormat>
7 </div>
```

9.7 f:selectItem

This tag associates a single `SelectItem` with the nearest parent `UIComponent`. The item represents a single option for a component such as an `h:selectBooleanCheckbox` or `h:selectOneMenu`.

See also component `f:selectItems`.

Unless otherwise specified, all attributes accept static values or EL expressions.

`UISelectItem` should be nested inside a `UISelectMany` or `UISelectOne` component, and results in the addition of a `SelectItem` instance to the list of available options for the parent component.

9.7.1 Attributes

Name	Required	Type	Description
<code>itemDisabled</code>	false	Boolean	Determine whether this item can be chosen by the user. When true, this item cannot be chosen by the user. If this method is ever called, then any EL-binding for the disabled property will be ignored.
<code>escape</code>	false	Boolean	The escape setting for the label of this selection item.
<code>itemDescription</code>	false	String	The item description.
<code>itemLabel</code>	false	String	The string which will be presented to the user for this option.
<code>itemValue</code>	false	String	The value for this item.
<code>value</code>	false	ValueExpression	The initial value of this component.

9.7.2 Example

```
1 <div xmlns:f="http://java.sun.com/jsf/core"
2     xmlns:h="http://java.sun.com/jsf/html">
3     <h:selectOneMenu id='height' size='10' value='{compute.height}'
4         styleClass="ui-state-default ui-corner-all">
5         <f:selectItem itemLabel="1 inch" itemValue="25.4"/>
6         <f:selectItem itemLabel="1 feet" itemValue="304.8"/>
7         <f:selectItem itemLabel="1 yard" itemValue="914.4"/>
8         <f:converter converterId="float" />
9     </h:selectOneMenu>
10 </div>
```

9.8 f:selectItems

This tag associates a list of `SelectItem` with the nearest parent `UIComponent`. The list of items is retrieved via a value-binding. See also component `f:selectItem`.

Unless otherwise specified, all attributes accept static values or EL expressions.

`UISelectItem` should be nested inside a `UISelectMany` or `UISelectOne` component, and results in the addition of one or more `SelectItem` instance to the list of available options for the parent component.

9.8.1 Attributes

Name	Required	Type	Description
<code>itemDisabled</code>	false	Boolean	Determine whether this item can be chosen by the user. When true, this item cannot be chosen by the user. If this method is ever called, then any EL-binding for the disabled property will be ignored.
<code>escape</code>	false	Boolean	The escape setting for the label of this selection item.
<code>itemDescription</code>	false	String	The item description.
<code>itemLabel</code>	false	String	The string which will be presented to the user for this option.
<code>itemValue</code>	false	String	The value for this item.
<code>value</code>	false	ValueExpression	The initial value of this component.

9.8.2 Example

```
1 <div xmlns:f="http://java.sun.com/jsf/core"
2     xmlns:h="http://java.sun.com/jsf/html">
3     <h:selectOneMenu id='height' size='10' value='#{compute.height}'
4         styleClass="ui-state-default ui-corner-all">
5         <f:selectItems value="#{countries}"/>
6     </h:selectOneMenu>
7 </div>
```

9.9 `f:validateLength`

Creates a validator and associates it with the nearest parent `UIComponent`. When invoked, the validator ensures that values are valid strings with a length that lies within the minimum and maximum values specified. Commonly associated with a `h:inputText` entity. Unless otherwise specified, all attributes accept static values or EL expressions.

9.9.1 Attributes

Name	Required	Type	Description
maximum	false	Natural	The largest value that should be considered valid.
minimum	false	Natural	The smallest value that should be considered valid.

9.9.2 Example

```

1 <div xmlns:f="http://java.sun.com/jsf/core"
2   xmlns:h="http://java.sun.com/jsf/html">
3   <h:form id='text-form'>
4     <dl class='ui-widget container_12'>
5       <dt><label for='height'>Height</label><h:message for='
6         height' /></dt>
7       <dd>
8         <h:inputText id='height' size='10' value='#{compute.
9           height}'
10           styleClass="ui-corner-all">
11           <f:converter converterId="float" />
12           <f:validateLength minimum="2" maximum="3"/>
13         </h:inputText>
14       </dd>
15       <dt><label for='radius'>Radius</label><h:message for='
16         radius' /></dt>
17       <dd>
18         <h:inputText id='radius' size='10' value='#{compute.
19           radius}'
20           styleClass="ui-corner-all">
21           <f:converter converterId="float"/>
22           <f:validateLength minimum="1" maximum="4"/>
23         </h:inputText>
24       </dd>
25     </dl>
26     <ul class='container_12 buttons'>
27       <li>
28         <h:commandButton id='run' value='Compute' action="#{
29           compute.run}"
30           styleClass="ui-button ui-state-default
31             ui-corner-all"/>
32       </li>
33     </ul>
34   </h:form>
35 </div>

```

9.10 f:validateLongRange

Creates a validator and associates it with the nearest parent `UIComponent`. When invoked, the validator ensures that values are valid longs that lie within the minimum and maximum values specified. Commonly associated with a `h:inputText` entity. Unless otherwise specified, all attributes accept static values or EL expressions.

9.10.1 Attributes

Name	Required	Type	Description
maximum	false	Long	The largest value that should be considered valid.
minimum	false	Long	The smallest value that should be considered valid.

9.10.2 Example

```

1  <div xmlns:f="http://java.sun.com/jsf/core"
2    xmlns:h="http://java.sun.com/jsf/html">
3    <h:form id='text-form'>
4      <dl class='ui-widget container_12'>
5        <dt><label for='height'>Height</label><h:message for='
6          height' /></dt>
7        <dd>
8          <h:inputText id='height' size='10' value='#{compute.
9            height}'
10             styleClass="ui-corner-all">
11            <f:converter converterId="float" />
12            <f:validateLongRange minimum="1" maximum="3"/>
13          </h:inputText>
14        </dd>
15        <dt><label for='radius'>Radius</label><h:message for='
16          radius' /></dt>
17        <dd>
18          <h:inputText id='radius' size='10' value='#{compute.
19            radius}'
20             styleClass="ui-corner-all">
21            <f:converter converterId="float" />
22            <f:validateLongRange minimum="1" maximum="5"/>
23          </h:inputText>
24        </dd>
25      </dl>
26      <ul class='container_12 buttons'>
27        <li>

```

```

24         <h:commandButton id='run' value='Compute' action="#{
           compute.run}"
25                               styleClass="ui-button ui-state-default
                                   ui-corner-all"/>
26     </li>
27 </ul>
28 </h:form>
29 </div>

```

9.11 f:validator

Creates a validator and associates it with the nearest parent UIComponent.

During the validation phase (or the apply-request-values phase for immediate components), if the associated component has any submitted value and the conversion of that value to the required type has succeeded then the specified validator type is invoked to test the validity of the converted value.

Commonly associated with an `h:inputText` entity, but may be applied to any input component. Some validators may allow the component to use attributes to define component-specific validation constraints; see the `f:attribute` tag. See also the “validator” attribute of all input components, which allows a component to specify an arbitrary validation `method` (rather than a registered validation type, as this tag does).

Unless otherwise specified, all attributes accept static values or EL expressions.

9.11.1 Attributes

Name	Required	Type	Description
validatorId	false	String	The registered ID of the desired Validator.

9.11.2 Example

```

1 <div xmlns:f="http://java.sun.com/jsf/core"
2   xmlns:h="http://java.sun.com/jsf/html">
3   <h:form id='text-form'>
4       <dl class='ui-widget container_12'>
5           <dt><label for='height'>Height</label><h:message for='
              height' /></dt>
6           <dd>
7               <h:inputText id='height' size='10' value='#{compute.
                  height}'

```

```

8           styleClass="ui-corner-all">
9           <f:converter converterId="float" />
10          <f:validator validatorId="validateDimension"/>
11        </h:inputText>
12      </dd>
13      <dt><label for='radius'>Radius</label><h:message for='
14        radius' /></dt>
15      <dd>
16        <h:inputText id='radius' size='10' value='#{compute.
17          radius}'
18          styleClass="ui-corner-all">
19          <f:converter converterId="float"/>
20          <f:validator validatorId="validateDimension"/>
21        </h:inputText>
22      </dd>
23    </dl>
24    <ul class='container_12 buttons'>
25      <li>
26        <h:commandButton id='run' value='Compute' action="#{
27          compute.run}"
28          styleClass="ui-button ui-state-default
29          ui-corner-all"/>

```

9.12 f:view

Creates a JSF View, which is a container that holds all of the components that are part of the view. The `UIView` represents the root of the component tree.

Unless otherwise specified, all attributes accept static values or EL expressions.

9.12.1 Attributes

Name	Required	Type	Description
locale	false	String	The locale of this view. Default: the default locale from the configuration file.
contentType	false	String	The content type to be placed in the response header. The default content type is <code>text/html</code> .

9.12.2 Example

```

1 <f:view contentType="application/json"
2     xmlns:f="http://java.sun.com/jsf/core"
3     xmlns:util="http://code.google.com/p/ada-asf/util"
4     xmlns:h="http://java.sun.com/jsf/html">
5 [
6 { "action": "update", "id": "#result", "data": "<util:escape>
7   <h:panelGroup rendered="#{not empty compute.volume}">
8     <h2>The cylinder volume is
9     <h:outputText value="#{compute.volume}">
10       <f:converter converterId="float"/>
11     </h:outputText>
12   </h2>
13 </h:panelGroup>
14 </util:escape>" }
15 ]
16 </f:view>

```

9.13 f:viewAction

The viewAction element is used in a metadata facet. It allows to execute an Ada bean action method when a request is processed. The Ada bean method is executed before rendering the page.

9.13.1 Attributes

Name	Required	Type	Description
itemDisabled	false	Boolean	Determine whether this item can be chosen by the user. When true, this item cannot be chosen by the user. If this method is ever called, then any EL-binding for the disabled property will be ignored.
escape	false	Boolean	The escape setting for the label of this selection item.
itemDescription	false	String	The item description.
itemLabel	false	String	The string which will be presented to the user for this option.
itemValue	false	String	The value for this item.
value	false	ValueExpression	The initial value of this component.

9.13.2 Example

```
1 <f:view xmlns:f="http://java.sun.com/jsf/core">
2   <f:metadata>
3     <f:viewParam id='height' value='#{compute.height}'>
4       <f:converter converterId="float" />
5     </f:viewParam>
6     <f:viewParam id='radius' value='#{compute.radius}'>
7       <f:converter converterId="float" />
8     </f:viewParam>
9     <f:viewAction action="#{compute.run}"/>
10  </f:metadata>
11  <dl>
12    <dt>Height</dt>
13    <dd>
14      #{compute.height}
15    </dd>
16    <dt>Radius</dt>
17    <dd>
18      #{compute.radius}
19    </dd>
20    <dt>Volume</dt>
21    <dd>
22      #{compute.volume}
23    </dd>
24  </dl>
25 </f:view>
```

9.14 f:viewParam

The viewParam element is used in a metadata facet. It allows to initialize a bean attribute from a request parameter.

9.14.1 Attributes

Name	Required	Type	Description
name	false	String	The name of the request parameter from which the value for this component is retrieved on an initial request or to override the stored value on a postback.

Name	Required	Type	Description
from	false	String	A value or expression that is evaluated to initialize the parameter value. This is an extension compared to the standard implementation.

9.14.2 Example

```
1 <f:view xmlns:f="http://java.sun.com/jsf/core">
2   <f:metadata>
3     <f:viewParam name='height' value='#{compute.height}'>
4       <f:converter converterId="float" />
5     </f:viewParam>
6     <f:viewParam id='radius' value='#{compute.radius}'>
7       <f:converter converterId="float" />
8     </f:viewParam>
9     <f:viewAction action="#{compute.run}" />
10  </f:metadata>
11  <dl>
12    <dt>Height</dt>
13    <dd>
14      #{compute.height}
15    </dd>
16    <dt>Radius</dt>
17    <dd>
18      #{compute.radius}
19    </dd>
20    <dt>Volume</dt>
21    <dd>
22      #{compute.volume}
23    </dd>
24  </dl>
25 </f:view>
```

10 HTML Components

The `html` components provide the HTML components.

```
1 xmlns:h="http://java.sun.com/jsf/html"
```

10.1 h:body

Render an html `body` element.

10.1.1 Attributes

Name	Required	Type	Description
dir	false	String	Direction indication for text that does not inherit directionality. Valid values are “LTR” (left-to-right) and “RTL” (right-to-left).
lang	false	String	Code describing the language used in the generated markup for this component.
onchange	false	String	Javascript code executed when the value of the element changes.
onclick	false	String	Javascript code executed when a pointer button is clicked over this element.
ondblclick	false	String	Javascript code executed when a pointer button is double clicked over this element.
onkeydown	false	String	Javascript code executed when a key is pressed down over this element.
onkeypress	false	String	Javascript code executed when a key is pressed or released over this element.
onkeyup	false	String	Javascript code executed when a key is released over this element.
onmousedown	false	String	Javascript code executed when a pointer button is pressed down over this element.

Name	Required	Type	Description
onmousemove	false	String	Javascript code executed when a pointer button is moved within this element.
onmouseout	false	String	Javascript code executed when a pointer button is moved away from this element.
onmouseover	false	String	Javascript code executed when a pointer button is moved onto this element.
onmouseup	false	String	Javascript code executed when a pointer button is released over this element.
onload	false	String	Javascript code executed when the user agent finishes loading a window or all frames within a frameset.
onunload	false	String	Javascript code executed when the user agent removes a document from a window or frame.

10.1.2 Example

```
1 <f:view xmlns:f="http://java.sun.com/jsf/core"
2     xmlns:h="http://java.sun.com/jsf/html"
3     contentType="text/html; charset=UTF-8">
4     <h:head dir="ltr" lang="en">
5         <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
6         <link media="screen" type="text/css" rel="stylesheet" href="#{contextPath}/css/samples.css"/>
7     </h:head>
8     <h:body dir="ltr" lang="en">
9         <p>Hello world!</p>
10    </h:body>
11 </f:view>
```

10.2 h:commandButton

Renders an HTML `input` element.

10.2.1 Attributes

Name	Required	Type	Description
value	true	ValueExpression	The current value of this component.
id	false	String	The component identifier for this component. This value must be unique within the closest parent component that is a naming container.
rendered	false	Boolean	Flag indicating whether or not this component should be rendered (during Render Response Phase), or processed on any subsequent form submit. The default value for this property is true.
action	false	String	MethodExpression representing the application action to invoke when this component is activated by the user. The expression must evaluate to a public method that takes no parameters, and returns an Object (the toString()) of which is called to derive the logical outcome) which is passed to the NavigationHandler for this application.
id	false	String	The component identifier for this component. This value must be unique within the closest parent component that is a naming container.
rendered	false	Boolean	Flag indicating whether or not this component should be rendered (during Render Response Phase), or processed on any subsequent form submit. The default value for this property is true.
dir	false	String	Direction indication for text that does not inherit directionality. Valid values are “LTR” (left-to-right) and “RTL” (right-to-left).
lang	false	String	Code describing the language used in the generated markup for this component.
accesskey	false	String	Access key that, when pressed, transfers focus to this element.
tabindex	false	Natural	Position of this element in the tabbing order for the current document. This value must be an integer between 0 and 32767.

Name	Required	Type	Description
alt	false	String	Alternate textual description of the element rendered by this component.
image	false	String	Absolute or relative URL of the image to be displayed for this button. If specified, this “input” element will be of type “image”. Otherwise, it will be of the type specified by the “type” property with a label specified by the “value” property. Note that if the value of this attribute starts with “/”, the rendered value for this attribute will be prefixed with the context-root for this application.
disabled	false	Boolean	Flag indicating that this element must never receive focus or be included in a subsequent submit. A value of false causes no attribute to be rendered, while a value of true causes the attribute to be rendered as disabled=“disabled”.
label	false	String	A localized user presentable name for this component.
onblur	false	String	Javascript code executed when this element loses focus.
onchange	false	String	Javascript code executed when this element loses focus and its value has been modified since gaining focus.
onfocus	false	String	Javascript code executed when this element receives focus.
onselect	false	String	Javascript code executed when text within this element is selected by the user.
readonly	false	String	Flag indicating that this component will prohibit changes by the user. The element may receive focus unless it has also been disabled. A value of false causes no attribute to be rendered, while a value of true causes the attribute to be rendered as readonly=“readonly”.

Name	Required	Type	Description
type	false	String	Type of button to create. Valid values are “submit”, “button”, and “reset”. If not specified, or not a valid value, the default value is “submit”.

10.2.2 Example

```

1 <div xmlns:f="http://java.sun.com/jsf/core"
2   xmlns:h="http://java.sun.com/jsf/html">
3   <h:form id='text-form'>
4     <dl class='ui-widget container_12'>
5       <dt><label for='height'>Height</label><h:message for='
6         height' /></dt>
7       <dd>
8         <h:inputText id='height' size='10' value='#{compute.
9           height}'
10          styleClass="ui-corner-all">
11           <f:converter converterId="float" />
12         </h:inputText>
13       </dd>
14       <dt><label for='radius'>Radius</label><h:message for='
15         radius' /></dt>
16       <dd>
17         <h:inputText id='radius' size='10' value='#{compute.
18           radius}'
19          styleClass="ui-corner-all">
20           <f:converter converterId="float" />
21         </h:inputText>
22       </dd>
23     </dl>
24     <ul class='container_12 buttons'>
25       <li>
26         <h:commandButton id='run' value='Compute' action="#{
27           compute.run}"
28          styleClass="ui-button ui-state-default
29            ui-corner-all"/>
26       </li>
27     </ul>
28   </h:form>
29 </div>

```

10.3 h:form

Renders the HTML `form` element.

10.3.1 Attributes

Name	Required	Type	Description
accept	false	String	List of content types that a server processing this form will handle correctly.
acceptcharset	false	String	List of character encodings accepted by the server for this form.
enctype	false	String	Content type used to submit the form to the server. If not specified, the default value is <code>application/x-www-form-urlencoded</code>
validate	false	Natural	The validity duration of the CSRF token created for the form submit protection.
expireMessage	false	String	A ValueExpression enabled attribute that, if present, will be used as the text of the expiration message when the form CSRF token has expired.
dir	false	String	Direction indication for text that does not inherit directionality. Valid values are “LTR” (left-to-right) and “RTL” (right-to-left).
lang	false	String	Code describing the language used in the generated markup for this component.
onchange	false	String	Javascript code executed when the value of the element changes.
onclick	false	String	Javascript code executed when a pointer button is clicked over this element.
ondblclick	false	String	Javascript code executed when a pointer button is double clicked over this element.
onkeydown	false	String	Javascript code executed when a key is pressed down over this element.
onkeypress	false	String	Javascript code executed when a key is pressed or released over this element.
onkeyup	false	String	Javascript code executed when a key is released over this element.

Name	Required	Type	Description
onmousedown	false	String	Javascript code executed when a pointer button is pressed down over this element.
onmousemove	false	String	Javascript code executed when a pointer button is moved within this element.
onmouseout	false	String	Javascript code executed when a pointer button is moved away from this element.
onmouseover	false	String	Javascript code executed when a pointer button is moved onto this element.
onmouseup	false	String	Javascript code executed when a pointer button is released over this element.
onreset	false	String	Javascript code executed when this form is reset.
onsubmit	false	String	Javascript code executed when this form is submitted.

10.3.2 Example

```

1  <div xmlns:f="http://java.sun.com/jsf/core"
2     xmlns:h="http://java.sun.com/jsf/html">
3     <h:form id='text-form'>
4         <dl class='ui-widget container_12'>
5             <dt><label for='height'>Height</label>
6                 <h:message for='height' /></dt>
7             <dd>
8                 <h:inputText id='height' size='10'
9                     value='#{compute.height}'
10                    styleClass="ui-corner-all">
11                     <f:converter converterId="float" />
12                 </h:inputText>
13             </dd>
14             <dt><label for='radius'>Radius</label>
15                 <h:message for='radius' /></dt>
16             <dd>
17                 <h:inputText id='radius' size='10'
18                     value='#{compute.radius}'
19                     styleClass="ui-corner-all">
20                     <f:converter converterId="float" />
21                 </h:inputText>
22             </dd>
23         </dl>

```

```

24         <ul class='container_12 buttons'>
25             <li>
26                 <h:commandButton id='run' value='Compute'
27                                 action="#{compute.run}"
28                                 styleClass="ui-button ui-state-default
29                                     ui-corner-all"/>
29             </li>
30         </ul>
31     </h:form>
32 </div>

```

10.4 h:head

Renders an HTML `head` element.

10.4.1 Attributes

Name	Required	Type	Description
dir	false	String	Direction indication for text that does not inherit directionality. Valid values are “LTR” (left-to-right) and “RTL” (right-to-left).
lang	false	String	Code describing the language used in the generated markup for this component.

10.4.2 Example

```

1 <f:view xmlns:f="http://java.sun.com/jsf/core"
2     xmlns:h="http://java.sun.com/jsf/html"
3     contentType="text/html; charset=UTF-8">
4     <h:head dir="ltr" lang="en">
5         <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
6         <link media="screen" type="text/css" rel="stylesheet" href="#{
7             contextPath}/css/samples.css"/>
8     </h:head>
9     <h:body dir="ltr" lang="en">
10         <p>Hello world!</p>
11     </h:body>
12 </f:view>

```

10.5 h:inputFile

Renders an HTML `input` element of type `file`.

10.5.1 Attributes

Name	Required	Type	Description
id	false	String	The component identifier for this component. This value must be unique within the closest parent component that is a naming container.
rendered	false	Boolean	Flag indicating whether or not this component should be rendered (during Render Response Phase), or processed on any subsequent form submit. The default value for this property is true.
converterMessage	false	String	A ValueExpression enabled attribute that, if present, will be used as the text of the converter message, replacing any message that comes from the converter.
required	false	Boolean	Flag indicating that the user is required to provide a submitted value for this input component.
requiredMessage	false	String	A ValueExpression enabled attribute that, if present, will be used as the text of the validation message for the “required” facility, if the “required” facility is used.
validatorMessage	false	String	A ValueExpression enabled attribute that, if present, will be used as the text of the validator message, replacing any message that comes from the validator.
value	false	ValueExpression	The current value of this component.

10.5.2 Example

```
1 <div xmlns:f="http://java.sun.com/jsf/core"
2   xmlns:h="http://java.sun.com/jsf/html">
```



```

3      <div class="ui-widget ui-widget-header">
4          Compute the volume of a cylinder
5      </div>
6      <h:form id='text-form' enctype='multipart/form-data'>
7          <dl>
8              <dt><label for='file'>File to upload</label></dt>
9              <dd>
10                 <h:inputFile id='file' size='50' value='#{image.image}'
11                     styleClass="ui-corner-all">
12                 </h:inputFile>
13                 <h:message for='file' />
14             </dd>
15         </dl>
16         <ul class='buttons'>
17             <li>
18                 <h:commandButton id='run' value='Upload' action="#{
19                     image.post}"
20                     styleClass="ui-button ui-state-default
21                         ui-corner-all"/>
22             </li>
23         </ul>
24     </h:form>
25 </div>

```

10.6 h:inputHidden

Renders an HTML `input` element of type `hidden`.

10.6.1 Attributes

Name	Required	Type	Description
id	false	String	The component identifier for this component. This value must be unique within the closest parent component that is a naming container.
rendered	false	Boolean	Flag indicating whether or not this component should be rendered (during Render Response Phase), or processed on any subsequent form submit. The default value for this property is true.

Name	Required	Type	Description
converterMessage	false	String	A ValueExpression enabled attribute that, if present, will be used as the text of the converter message, replacing any message that comes from the converter.
required	false	Boolean	Flag indicating that the user is required to provide a submitted value for this input component.
requiredMessage	false	String	A ValueExpression enabled attribute that, if present, will be used as the text of the validation message for the “required” facility, if the “required” facility is used.
validatorMessage	false	String	A ValueExpression enabled attribute that, if present, will be used as the text of the validator message, replacing any message that comes from the validator.
value	false	ValueExpression	The current value of this component.

10.6.2 Example

```

1  <div xmlns:f="http://java.sun.com/jsf/core"
2      xmlns:h="http://java.sun.com/jsf/html">
3      <h:form id='text-form'>
4          <h:inputHidden id='radius' value='#{compute.radius}'>
5              <f:converter converterId="float"/>
6          </h:inputHidden>
7          <dl class='ui-widget container_12'>
8              <dt><label for='height'>Height</label><h:message for='
9                  height' /></dt>
10             <dd>
11                 <h:inputText id='height' size='10' value='#{compute.
12                     height}'
13                     styleClass="ui-corner-all">
14                     <f:converter converterId="float" />
15                 </h:inputText>
16             </dd>
17         </dl>
18         <ul class='container_12 buttons'>
19             <li>
20                 <h:commandButton id='run' value='Compute' action="#">
21                     compute.run}</h:commandButton>
22             </li>
23         </ul>
24     </h:form>
25 </div>

```

```
19                                     styleClass="ui-button ui-state-default
20                                     ui-corner-all"/>
21     </li>
22 </ul>
23 </h:form>
24 </div>
```

10.7 h:inputSecret

Renders an HTML `input` element of type `password`.

10.7.1 Attributes

Name	Required	Type	Description
id	false	String	The component identifier for this component. This value must be unique within the closest parent component that is a naming container.
rendered	false	Boolean	Flag indicating whether or not this component should be rendered (during Render Response Phase), or processed on any subsequent form submit. The default value for this property is true.
converterMessage	false	String	A ValueExpression enabled attribute that, if present, will be used as the text of the converter message, replacing any message that comes from the converter.
style	false	String	CSS style(s) to be applied when this component is rendered.
styleClass	false	String	Space-separated list of CSS style class(es) to be applied when this element is rendered. This value must be passed through as the “class” attribute on generated markup.
title	false	String	Advisory title information about markup elements generated for this component.

Name	Required	Type	Description
dir	false	String	Direction indication for text that does not inherit directionality. Valid values are “LTR” (left-to-right) and “RTL” (right-to-left).
lang	false	String	Code describing the language used in the generated markup for this component.
onchange	false	String	Javascript code executed when the value of the element changes.
onclick	false	String	Javascript code executed when a pointer button is clicked over this element.
ondblclick	false	String	Javascript code executed when a pointer button is double clicked over this element.
onkeydown	false	String	Javascript code executed when a key is pressed down over this element.
onkeypress	false	String	Javascript code executed when a key is pressed or released over this element.
onkeyup	false	String	Javascript code executed when a key is released over this element.
onmousedown	false	String	Javascript code executed when a pointer button is pressed down over this element.
onmousemove	false	String	Javascript code executed when a pointer button is moved within this element.
onmouseout	false	String	Javascript code executed when a pointer button is moved away from this element.
onmouseover	false	String	Javascript code executed when a pointer button is moved onto this element.
onmouseup	false	String	Javascript code executed when a pointer button is released over this element.
required	false	Boolean	Flag indicating that the user is required to provide a submitted value for this input component.

Name	Required	Type	Description
requiredMessage	false	String	A ValueExpression enabled attribute that, if present, will be used as the text of the validation message for the “required” facility, if the “required” facility is used.
validatorMessage	false	String	A ValueExpression enabled attribute that, if present, will be used as the text of the validator message, replacing any message that comes from the validator.
value	false	ValueExpression	The current value of this component.
accesskey	false	String	Access key that, when pressed, transfers focus to this element.
tabindex	false	Natural	Position of this element in the tabbing order for the current document. This value must be an integer between 0 and 32767.
maxlength	false	Natural	The maximum number of characters that may be entered in this field.
size	false	Natural	The number of characters used to determine the width of this field.
redisplay	false	Boolean	Flag indicating that any existing value in this field should be rendered when the form is created. Because this is a potential security risk, password values are not displayed by default.
readonly	false	Boolean	Flag indicating that this component will prohibit changes by the user. The element may receive focus unless it has also been disabled. A value of false causes no attribute to be rendered, while a value of true causes the attribute to be rendered as readonly=“readonly”.
disabled	false	Boolean	Flag indicating that this element must never receive focus or be included in a subsequent submit. A value of false causes no attribute to be rendered, while a value of true causes the attribute to be rendered as disabled=“disabled”.

Name	Required	Type	Description
label	false	String	A localized user presentable name for this component.

10.7.2 Example

```

1  <div xmlns:f="http://java.sun.com/jsf/core"
2      xmlns:h="http://java.sun.com/jsf/html">
3      <div class="ui-widget ui-widget-header">
4          Compute the volume of a cylinder
5      </div>
6      <h:form id='text-form'>
7          <dl>
8              <dt><label for='height'>Secret height</label></dt>
9              <dd>
10                 <h:inputSecret id='height' size='10' value='#{compute.
11                     height}'
12                     styleClass="ui-corner-all">
13                     <f:converter converterId="float" />
14                 </h:inputSecret>
15                 <h:message for='height' />
16             </dd>
17             <dt><label for='radius'>Secret radius</label></dt>
18             <dd>
19                 <h:inputSecret id='radius' size='10' value='#{compute.
20                     radius}'
21                     styleClass="ui-corner-all">
22                     <f:converter converterId="float" />
23                 </h:inputSecret>
24                 <h:message for='radius' />
25             </dd>
26         </dl>
27         <ul class='buttons'>
28             <li>
29                 <h:commandButton id='run' value='Compute' action="#{
30                     compute.run}"
31                     styleClass="ui-button ui-state-default
32                     ui-corner-all"/>
33             </li>
34         </ul>
35     </h:form>
36 </div>

```

10.8 h:inputText

Renders an HTML `input` element of type `text`.

10.8.1 Attributes

Name	Required	Type	Description
id	false	String	The component identifier for this component. This value must be unique within the closest parent component that is a naming container.
rendered	false	Boolean	Flag indicating whether or not this component should be rendered (during Render Response Phase), or processed on any subsequent form submit. The default value for this property is true.
converterMessage	false	String	A ValueExpression enabled attribute that, if present, will be used as the text of the converter message, replacing any message that comes from the converter.
style	false	String	CSS style(s) to be applied when this component is rendered.
styleClass	false	String	Space-separated list of CSS style class(es) to be applied when this element is rendered. This value must be passed through as the “class” attribute on generated markup.
title	false	String	Advisory title information about markup elements generated for this component.
dir	false	String	Direction indication for text that does not inherit directionality. Valid values are “LTR” (left-to-right) and “RTL” (right-to-left).
lang	false	String	Code describing the language used in the generated markup for this component.
onchange	false	String	Javascript code executed when the value of the element changes.

Name	Required	Type	Description
onclick	false	String	Javascript code executed when a pointer button is clicked over this element.
ondblclick	false	String	Javascript code executed when a pointer button is double clicked over this element.
onkeydown	false	String	Javascript code executed when a key is pressed down over this element.
onkeypress	false	String	Javascript code executed when a key is pressed or released over this element.
onkeyup	false	String	Javascript code executed when a key is released over this element.
onmousedown	false	String	Javascript code executed when a pointer button is pressed down over this element.
onmousemove	false	String	Javascript code executed when a pointer button is moved within this element.
onmouseout	false	String	Javascript code executed when a pointer button is moved away from this element.
onmouseover	false	String	Javascript code executed when a pointer button is moved onto this element.
onmouseup	false	String	Javascript code executed when a pointer button is released over this element.
required	false	Boolean	Flag indicating that the user is required to provide a submitted value for this input component.
requiredMessage	false	String	A ValueExpression enabled attribute that, if present, will be used as the text of the validation message for the “required” facility, if the “required” facility is used.
validatorMessage	false	String	A ValueExpression enabled attribute that, if present, will be used as the text of the validator message, replacing any message that comes from the validator.
value	false	ValueExpression	The current value of this component.

Name	Required	Type	Description
accesskey	false	String	Access key that, when pressed, transfers focus to this element.
tabindex	false	Natural	Position of this element in the tabbing order for the current document. This value must be an integer between 0 and 32767.
maxlength	false	Natural	The maximum number of characters that may be entered in this field.
size	false	Natural	The number of characters used to determine the width of this field.
redisplay	false	Boolean	Flag indicating that any existing value in this field should be rendered when the form is created. Because this is a potential security risk, password values are not displayed by default.
readonly	false	Boolean	Flag indicating that this component will prohibit changes by the user. The element may receive focus unless it has also been disabled. A value of false causes no attribute to be rendered, while a value of true causes the attribute to be rendered as <code>readonly="readonly"</code> .
disabled	false	Boolean	Flag indicating that this element must never receive focus or be included in a subsequent submit. A value of false causes no attribute to be rendered, while a value of true causes the attribute to be rendered as <code>disabled="disabled"</code> .
label	false	String	A localized user presentable name for this component.

10.8.2 Example

```

1 <div xmlns:f="http://java.sun.com/jsf/core"
2   xmlns:h="http://java.sun.com/jsf/html">
3   <div class="ui-widget ui-widget-header">
4       Compute the volume of a cylinder
5   </div>

```

```

6      <h:form id='text-form'>
7          <dl>
8              <dt><label for='height'>Height</label></dt>
9              <dd>
10                 <h:inputText id='height' size='10' value='#{compute.
11                     height}'
12                     styleClass="ui-corner-all">
13                     <f:converter converterId="float" />
14                 </h:inputText>
15                 <h:message for='height' />
16             </dd>
17             <dt><label for='radius'>Radius</label></dt>
18             <dd>
19                 <h:inputText id='radius' size='10' value='#{compute.
20                     radius}'
21                     styleClass="ui-corner-all">
22                     <f:converter converterId="float" />
23                 </h:inputText>
24                 <h:message for='radius' />
25             </dd>
26         </dl>
27         <ul class='buttons'>
28             <li>
29                 <h:commandButton id='run' value='Compute' action="#{
30                     compute.run}"
31                     styleClass="ui-button ui-state-default
32                     ui-corner-all"/>
33             </li>
34         </ul>
35     </h:form>
36 </div>

```

10.9 h:inputTextarea

Renders an HTML `textarea` element.

10.9.1 Attributes

Name	Required	Type	Description
id	false	String	The component identifier for this component. This value must be unique within the closest parent component that is a naming container.

Name	Required	Type	Description
rendered	false	Boolean	Flag indicating whether or not this component should be rendered (during Render Response Phase), or processed on any subsequent form submit. The default value for this property is true.
converterMessage	false	String	A ValueExpression enabled attribute that, if present, will be used as the text of the converter message, replacing any message that comes from the converter.
style	false	String	CSS style(s) to be applied when this component is rendered.
styleClass	false	String	Space-separated list of CSS style class(es) to be applied when this element is rendered. This value must be passed through as the “class” attribute on generated markup.
title	false	String	Advisory title information about markup elements generated for this component.
dir	false	String	Direction indication for text that does not inherit directionality. Valid values are “LTR” (left-to-right) and “RTL” (right-to-left).
lang	false	String	Code describing the language used in the generated markup for this component.
onchange	false	String	Javascript code executed when the value of the element changes.
onclick	false	String	Javascript code executed when a pointer button is clicked over this element.
ondblclick	false	String	Javascript code executed when a pointer button is double clicked over this element.
onkeydown	false	String	Javascript code executed when a key is pressed down over this element.
onkeypress	false	String	Javascript code executed when a key is pressed or released over this element.

Name	Required	Type	Description
onkeyup	false	String	Javascript code executed when a key is released over this element.
onmousedown	false	String	Javascript code executed when a pointer button is pressed down over this element.
onmousemove	false	String	Javascript code executed when a pointer button is moved within this element.
onmouseout	false	String	Javascript code executed when a pointer button is moved away from this element.
onmouseover	false	String	Javascript code executed when a pointer button is moved onto this element.
onmouseup	false	String	Javascript code executed when a pointer button is released over this element.
required	false	Boolean	Flag indicating that the user is required to provide a submitted value for this input component.
requiredMessage	false	String	A ValueExpression enabled attribute that, if present, will be used as the text of the validation message for the “required” facility, if the “required” facility is used.
validatorMessage	false	String	A ValueExpression enabled attribute that, if present, will be used as the text of the validator message, replacing any message that comes from the validator.
value	false	ValueExpression	The current value of this component.
accesskey	false	String	Access key that, when pressed, transfers focus to this element.
tabindex	false	Natural	Position of this element in the tabbing order for the current document. This value must be an integer between 0 and 32767.
rows	false	Natural	The number of rows to be displayed.
cols	false	Natural	The number of columns to be displayed.

Name	Required	Type	Description
readonly	false	Boolean	Flag indicating that this component will prohibit changes by the user. The element may receive focus unless it has also been disabled. A value of false causes no attribute to be rendered, while a value of true causes the attribute to be rendered as <code>readonly="readonly"</code> .
label	false	String	A localized user presentable name for this component.

10.9.2 Example

```

1 <div xmlns:f="http://java.sun.com/jsf/core"
2   xmlns:h="http://java.sun.com/jsf/html">
3
4   <div class="ui-widget ui-widget-header">
5       Write a message in a textarea
6   </div>
7   <h:form id='textarea-form'>
8       <dl>
9           <dt><label for='email'>Email</label> <h:message for='email'
10              /></dt>
11           <dd>
12               <h:inputText id='email' size='80' value='#{message.
13                  email}'
14                  styleClass="ui-corner-all">
15               </h:inputText>
16           </dd>
17           <dt><label for='message'>Message</label> <h:message for='
18              message' /></dt>
19           <dd>
20               <h:inputTextarea id='message' rows='20' cols='30' value
21                  = '#{message.text}' />
22           </dd>
23       </dl>
24       <ul class='buttons'>
25           <li>
26               <h:commandButton id='send' value='Send' action="#{
27                  message.post}"
28                  styleClass="ui-button ui-state-default
29                      ui-corner-all"/>
30           </li>
31       </ul>
32   </h:form>

```

27

</div>

10.10 h:list

Renders a list of items.

When the list layout is `orderedList` an `ol/li` list is generated, when the layout is `unorderedList` an `ul/li` list is generated. The default (`simple`) renders the list as is.

10.10.1 Attributes

Name	Required	Type	Description
var	true	String	Name of the variable which holds the current row value.
value	true	ValueExpression	The value expression representing the list to iterate on.
id	false	String	The component identifier for this component. This value must be unique within the closest parent component that is a naming container.
rendered	false	Boolean	Flag indicating whether or not this component should be rendered (during Render Response Phase), or processed on any subsequent form submit. The default value for this property is true.
style	false	String	CSS style(s) to be applied when this component is rendered.
styleClass	false	String	Space-separated list of CSS style class(es) to be applied when this element is rendered. This value must be passed through as the “class” attribute on generated markup.
title	false	String	Advisory title information about markup elements generated for this component.
layout	false	String	The layout of the list: <code>simple</code> , <code>unorderedList</code> , <code>orderedList</code> .
itemStyleClass	false	String	The CSS class attribute to apply to the <code>li</code> or <code>div</code> items.

10.10.2 Example

```
1 <div xmlns:h="http://java.sun.com/jsf/html">
2   <h:list value="#{messages}" var="msg">
3     <div class="message">
4       <div class="email">
5         #{msg.email}
6       </div>
7       <div class="text">
8         #{msg.text}
9       </div>
10    </div>
11  </h:list>
12  <!-- Demo Hint: use the 'Forum' demo to populate the above list if
13    it is empty -->
14 </div>
```

10.11 h:outputFormat

Render parameterized text.

Obtain the style, styleClass, dir, and lang attributees from this component. If any are present, render a `span` element. Output the styleClass attribute (if present) as the value of the class attribute. Output the style attribute as the value of the style attribute. Output the dir and lang attributes as pass through attributes. Accrue a list of the values of all child `UIParameter` components of this component. If there are one or more accumulated parameter values, convert the list of parameter values to an Object array, call `MessageFormat.format()`, passing the value of this component as the first argument, and the array of parameter values as the second argument, and render the result. Otherwise, render the value of this component unmodified.

10.11.1 Attributes

Name	Required	Type	Description
id	false	String	The component identifier for this component. This value must be unique within the closest parent component that is a naming container.

Name	Required	Type	Description
rendered	false	Boolean	Flag indicating whether or not this component should be rendered (during Render Response Phase), or processed on any subsequent form submit. The default value for this property is true.
style	false	String	CSS style(s) to be applied when this component is rendered.
styleClass	false	String	Space-separated list of CSS style class(es) to be applied when this element is rendered. This value must be passed through as the “class” attribute on generated markup.
title	false	String	Advisory title information about markup elements generated for this component.
dir	false	String	Direction indication for text that does not inherit directionality. Valid values are “LTR” (left-to-right) and “RTL” (right-to-left).
lang	false	String	Code describing the language used in the generated markup for this component.
accesskey	false	String	Access key that, when pressed, transfers focus to this element.
tabindex	false	Natural	Position of this element in the tabbing order for the current document. This value must be an integer between 0 and 32767.
onchange	false	String	Javascript code executed when the value of the element changes.
onclick	false	String	Javascript code executed when a pointer button is clicked over this element.
ondblclick	false	String	Javascript code executed when a pointer button is double clicked over this element.
onkeydown	false	String	Javascript code executed when a key is pressed down over this element.
onkeypress	false	String	Javascript code executed when a key is pressed or released over this element.

Name	Required	Type	Description
onkeyup	false	String	Javascript code executed when a key is released over this element.
onmousedown	false	String	Javascript code executed when a pointer button is pressed down over this element.
onmousemove	false	String	Javascript code executed when a pointer button is moved within this element.
onmouseout	false	String	Javascript code executed when a pointer button is moved away from this element.
onmouseover	false	String	Javascript code executed when a pointer button is moved onto this element.
onmouseup	false	String	Javascript code executed when a pointer button is released over this element.
for	false	String	Client identifier of the component for which this element is a label.
value	false	String	The current value of this component.
escape	false	Boolean	Flag indicating that characters that are sensitive in HTML and XML markup must be escaped. This flag is set to "true" by default.

10.11.2 Example

```

1 <div xmlns:f="http://java.sun.com/jsf/core"
2   xmlns:h="http://java.sun.com/jsf/html">
3   <!-- Write a text string escaping special characters -->
4   <p><h:outputFormat value="The application name is: {0}">
5     <f:param value="#{sampleName}"/>
6   </h:outputFormat>
7   </p>
8   <!-- Write a text with a style. Generate a <span> element. -->
9   <p><h:outputFormat value="The context path is: #{contextPath}"
10     style="font-weight: bold;">
11     <f:param value="#{contextPath}"/>
12   </h:outputFormat>
13   </p>
14 </div>

```

10.12 h:outputLabel

Renders an HTML **label** element. Render the current value of the component as label text if it is specified. If a **for** attribute is specified, find the component specified by the value of the **for** attribute, and render its client id as the value of the **for** attribute. If “styleClass” attribute is specified, render its value as the value of the “class” attribute.

10.12.1 Attributes

Name	Required	Type	Description
id	false	String	The component identifier for this component. This value must be unique within the closest parent component that is a naming container.
rendered	false	Boolean	Flag indicating whether or not this component should be rendered (during Render Response Phase), or processed on any subsequent form submit. The default value for this property is true.
style	false	String	CSS style(s) to be applied when this component is rendered.
styleClass	false	String	Space-separated list of CSS style class(es) to be applied when this element is rendered. This value must be passed through as the “class” attribute on generated markup.
title	false	String	Advisory title information about markup elements generated for this component.
dir	false	String	Direction indication for text that does not inherit directionality. Valid values are “LTR” (left-to-right) and “RTL” (right-to-left).
lang	false	String	Code describing the language used in the generated markup for this component.
accesskey	false	String	Access key that, when pressed, transfers focus to this element.

Name	Required	Type	Description
tabindex	false	Natural	Position of this element in the tabbing order for the current document. This value must be an integer between 0 and 32767.
onchange	false	String	Javascript code executed when the value of the element changes.
onclick	false	String	Javascript code executed when a pointer button is clicked over this element.
ondblclick	false	String	Javascript code executed when a pointer button is double clicked over this element.
onkeydown	false	String	Javascript code executed when a key is pressed down over this element.
onkeypress	false	String	Javascript code executed when a key is pressed or released over this element.
onkeyup	false	String	Javascript code executed when a key is released over this element.
onmousedown	false	String	Javascript code executed when a pointer button is pressed down over this element.
onmousemove	false	String	Javascript code executed when a pointer button is moved within this element.
onmouseout	false	String	Javascript code executed when a pointer button is moved away from this element.
onmouseover	false	String	Javascript code executed when a pointer button is moved onto this element.
onmouseup	false	String	Javascript code executed when a pointer button is released over this element.
for	false	String	Client identifier of the component for which this element is a label.
value	false	String	The current value of this component.
escape	false	Boolean	Flag indicating that characters that are sensitive in HTML and XML markup must be escaped. This flag is set to “true” by default.

10.12.2 Example

```
1 <dl xmlns:f="http://java.sun.com/jsf/core"
2   xmlns:h="http://java.sun.com/jsf/html">
3   <dt><h:outputLabel for='height' value="Height"/><h:message for='
4     height'/></dt>
5   <dd>
6     <h:selectOneMenu id='height' size='10' value='#{compute.height}'
7       styleClass="ui-state-default ui-corner-all">
8       <f:selectItem itemLabel="1 inch" itemValue="25.4"/>
9       <f:selectItem itemLabel="1 feet" itemValue="304.8"/>
10      <f:selectItem itemLabel="1 yard" itemValue="914.4"/>
11      <f:converter converterId="float" />
12    </h:selectOneMenu>
13  </dd>
14  <dt><h:outputLabel for='radius' value="Radius"/><h:message for='
15    radius'/></dt>
16  <dd>
17    <h:selectOneMenu id='radius' size='10' value='#{compute.radius}'
18      styleClass="ui-state-default ui-corner-all">
19      <f:selectItem itemLabel="1 inch" itemValue="25.4"/>
20      <f:selectItem itemLabel="1 feet" itemValue="304.8"/>
21      <f:selectItem itemLabel="1 yard" itemValue="914.4"/>
22      <f:converter converterId="float" />
23    </h:selectOneMenu>
24  </dd>
25 </dl>
```

10.13 h:outputLink

Render an HTML “a” anchor element. The value of the component is rendered as the value of the “href” attribute. Any child UIParameter components are appended to the String to be output as the value of the “href” attribute as query parameters before rendering. The entire “href” string must be passed through a call to the `encodeResourceURL()` method of the `ExternalContext`. The name of the UIParameter goes on the left hand side, and the value of the UIParameter on the right hand side. The name and the value must be URLEncoded. Each UIParameter instance is separated by an ampersand, as dictated in the URL spec. If the “styleClass” attribute is specified, render its value as the value of the “class” attribute. If the “id” attribute is specified, follow the same steps as mentioned in the “General Notes on Encoding” regarding the “id” attribute for UIInput components. If the “disabled” attribute is specified, do not render the HTML “a” anchor element or the “href” element. Instead, render a “span” element. If the “styleClass” attribute is specified, render its value as the value of the “class” attribute on the “span”.

10.13.1 Attributes

Name	Required	Type	Description
id	false	String	The component identifier for this component. This value must be unique within the closest parent component that is a naming container.
rendered	false	Boolean	Flag indicating whether or not this component should be rendered (during Render Response Phase), or processed on any subsequent form submit. The default value for this property is true.
style	false	String	CSS style(s) to be applied when this component is rendered.
styleClass	false	String	Space-separated list of CSS style class(es) to be applied when this element is rendered. This value must be passed through as the “class” attribute on generated markup.
title	false	String	Advisory title information about markup elements generated for this component.
dir	false	String	Direction indication for text that does not inherit directionality. Valid values are “LTR” (left-to-right) and “RTL” (right-to-left).
lang	false	String	Code describing the language used in the generated markup for this component.
accesskey	false	String	Access key that, when pressed, transfers focus to this element.
tabindex	false	Natural	Position of this element in the tabbing order for the current document. This value must be an integer between 0 and 32767.
onchange	false	String	Javascript code executed when the value of the element changes.
onclick	false	String	Javascript code executed when a pointer button is clicked over this element.

Name	Required	Type	Description
ondblclick	false	String	Javascript code executed when a pointer button is double clicked over this element.
onkeydown	false	String	Javascript code executed when a key is pressed down over this element.
onkeypress	false	String	Javascript code executed when a key is pressed or released over this element.
onkeyup	false	String	Javascript code executed when a key is released over this element.
onmousedown	false	String	Javascript code executed when a pointer button is pressed down over this element.
onmousemove	false	String	Javascript code executed when a pointer button is moved within this element.
onmouseout	false	String	Javascript code executed when a pointer button is moved away from this element.
onmouseover	false	String	Javascript code executed when a pointer button is moved onto this element.
onmouseup	false	String	Javascript code executed when a pointer button is released over this element.
value	false	String	The current value of this component.

10.13.2 Example

```
1 <div xmlns:f="http://java.sun.com/jsf/core"
2   xmlns:h="http://java.sun.com/jsf/html">
3   <!-- Link is enabled -->
4   <p><h:outputLink value="https://github.com/stcarrez/ada-asf">Ada
      Server Faces</h:outputLink></p>
5   <!-- Write a text with a style. Generate a <span> element. -->
6   <p><h:outputLink value="https://github.com/stcarrez/ada-el"
      disabled="true"
7       style="font-weight: bold;">Ada EL</h:outputLink></p>
8 </div>
```

10.14 h:outputText

Renders the value of the associated UIOutput component. If this element has an ID or CSS style properties, the text is wrapped in a span element.

10.14.1 Attributes

Name	Required	Type	Description
id	false	String	The component identifier for this component. This value must be unique within the closest parent component that is a naming container.
rendered	false	Boolean	Flag indicating whether or not this component should be rendered (during Render Response Phase), or processed on any subsequent form submit. The default value for this property is true.
style	false	String	CSS style(s) to be applied when this component is rendered.
styleClass	false	String	Space-separated list of CSS style class(es) to be applied when this element is rendered. This value must be passed through as the “class” attribute on generated markup.
title	false	String	Advisory title information about markup elements generated for this component.
dir	false	String	Direction indication for text that does not inherit directionality. Valid values are “LTR” (left-to-right) and “RTL” (right-to-left).
lang	false	String	Code describing the language used in the generated markup for this component.
value	false	String	The current value of this component.
escape	false	Boolean	Flag indicating that characters that are sensitive in HTML and XML markup must be escaped. This flag is set to “true” by default.

10.14.2 Example

```
1 <div xmlns:f="http://java.sun.com/jsf/core"
2     xmlns:h="http://java.sun.com/jsf/html">
3     <!-- Write a text string escaping special characters -->
4     <p><h:outputText value="The application name is: #{sampleName}"/></p>
5     <!-- Write a text with a style. Generate a <span> element. -->
6     <p><h:outputText value="The context path is: #{contextPath}"
7         style="font-weight: bold;"/></p>
8     <!-- Write a text without escaping -->
9     <p><h:outputText escape="false" value="&lt;i&gt;This string is not
10         escaped.&lt;/i&gt;"/></p>
11 </div>
```

10.15 h:panelGroup

This element is used to group other components where the specification requires one child element. If any of the HTML or CSS attributes are set, its content is rendered within a span or div element.

10.15.1 Attributes

Name	Required	Type	Description
id	false	String	The component identifier for this component. This value must be unique within the closest parent component that is a naming container.
rendered	false	Boolean	Flag indicating whether or not this component should be rendered (during Render Response Phase), or processed on any subsequent form submit. The default value for this property is true.
style	false	String	CSS style(s) to be applied when this component is rendered.
styleClass	false	String	Space-separated list of CSS style class(es) to be applied when this element is rendered. This value must be passed through as the “class” attribute on generated markup.

Name	Required	Type	Description
title	false	String	Advisory title information about markup elements generated for this component.
layout	false	String	The type of layout markup to use when rendering this group. If the value is “block” the renderer must produce an HTML “div” element. Otherwise HTML “span” element must be produced.

10.15.2 Example

```
1 <div xmlns:f="http://java.sun.com/jsf/core"
2   xmlns:h="http://java.sun.com/jsf/html">
3   <h:panelGroup layout="block">
4     A div element
5   </h:panelGroup>
6   <h:panelGroup rendered="#{sampleName eq 'demo'}">
7     A text block which is not present (rendered = false)
8   </h:panelGroup>
9   <h:panelGroup rendered="#{sampleName ne 'demo'}">
10    A text block which is present (rendered = true)
11  </h:panelGroup>
12  <h:panelGroup rendered="#{sampleName ne 'demo'}" style="padding-top
13    : 20px; color: red;">
14    A text block which is present (rendered = true)
15  </h:panelGroup>
16 </div>
```

10.16 h:selectBooleanCheckbox

10.16.1 Attributes

Name	Required	Type	Description
id	false	String	The component identifier for this component. This value must be unique within the closest parent component that is a naming container.

Name	Required	Type	Description
rendered	false	Boolean	Flag indicating whether or not this component should be rendered (during Render Response Phase), or processed on any subsequent form submit. The default value for this property is true.
style	false	String	CSS style(s) to be applied when this component is rendered.
styleClass	false	String	Space-separated list of CSS style class(es) to be applied when this element is rendered. This value must be passed through as the “class” attribute on generated markup.
title	false	String	Advisory title information about markup elements generated for this component.
dir	false	String	Direction indication for text that does not inherit directionality. Valid values are “LTR” (left-to-right) and “RTL” (right-to-left).
lang	false	String	Code describing the language used in the generated markup for this component.
onchange	false	String	Javascript code executed when the value of the element changes.
onclick	false	String	Javascript code executed when a pointer button is clicked over this element.
ondblclick	false	String	Javascript code executed when a pointer button is double clicked over this element.
onkeydown	false	String	Javascript code executed when a key is pressed down over this element.
onkeypress	false	String	Javascript code executed when a key is pressed or released over this element.
onkeyup	false	String	Javascript code executed when a key is released over this element.
onmousedown	false	String	Javascript code executed when a pointer button is pressed down over this element.

Name	Required	Type	Description
onmousemove	false	String	Javascript code executed when a pointer button is moved within this element.
onmouseout	false	String	Javascript code executed when a pointer button is moved away from this element.
onmouseover	false	String	Javascript code executed when a pointer button is moved onto this element.
onmouseup	false	String	Javascript code executed when a pointer button is released over this element.
required	false	Boolean	Flag indicating that the user is required to provide a submitted value for this input component.
requiredMessage	false	String	A ValueExpression enabled attribute that, if present, will be used as the text of the validation message for the “required” facility, if the “required” facility is used.
validatorMessage	false	String	A ValueExpression enabled attribute that, if present, will be used as the text of the validator message, replacing any message that comes from the validator.
value	false	ValueExpression	The current value of this component.
layout	false	String	The type of layout markup to use when rendering this group. If the value is “block” the renderer must produce an HTML “div” element. Otherwise HTML “span” element must be produced.

10.16.2 Example

```

1 <div xmlns:f="http://java.sun.com/jsf/core"
2   xmlns:h="http://java.sun.com/jsf/html">
3   <div class="ui-widget ui-widget-header">
4     Compute the volume of a cylinder
5   </div>
6   <h:form id='text-form'>
7     <dl>

```

```
8      <dt><label for='flag'>Is active</label> <h:message for='
9      flag' /></dt>
10     <dd>
11         <h:selectBooleanCheckbox id="flag" value="{empty
12         compute.radius}" />
13     </dd>
14     <dt><label for='radius'>Radius</label></dt>
15     <dd>
16         <h:inputText id='radius' size='10' value='{compute.
17         radius}'
18         styleClass="ui-corner-all">
19             <f:converter converterId="float" />
20         </h:inputText>
21         <h:message for='radius' />
22     </dd>
23 </dl>
24 <ul class='buttons'>
25     <li>
26         <h:commandButton id='run' value='Compute' action="{
27         compute.run}"
28         styleClass="ui-button ui-state-default
29         ui-corner-all" />
30     </li>
31 </ul>
32 </h:form>
33 </div>
```

10.17 h:selectOneMenu

Allow the user to choose one option from a set of options. Renders a drop-down menu (aka “combo-box”) containing a set of choices, of which only one can be chosen at a time. The available choices are defined via child `f:selectItem` or `f:selectItems` elements.

The value attribute of this component is read to determine which of the available options is initially selected; its value should match the “value” property of one of the child `SelectItem` objects.

On submit of the enclosing form, the value attribute’s bound property is updated to contain the “value” property from the chosen `SelectItem`.

10.17.1 Attributes

Name	Required	Type	Description
id	false	String	The component identifier for this component. This value must be unique within the closest parent component that is a naming container.
rendered	false	Boolean	Flag indicating whether or not this component should be rendered (during Render Response Phase), or processed on any subsequent form submit. The default value for this property is true.
style	false	String	CSS style(s) to be applied when this component is rendered.
styleClass	false	String	Space-separated list of CSS style class(es) to be applied when this element is rendered. This value must be passed through as the “class” attribute on generated markup.
title	false	String	Advisory title information about markup elements generated for this component.
dir	false	String	Direction indication for text that does not inherit directionality. Valid values are “LTR” (left-to-right) and “RTL” (right-to-left).
lang	false	String	Code describing the language used in the generated markup for this component.
onchange	false	String	Javascript code executed when the value of the element changes.
onclick	false	String	Javascript code executed when a pointer button is clicked over this element.
ondblclick	false	String	Javascript code executed when a pointer button is double clicked over this element.
onkeydown	false	String	Javascript code executed when a key is pressed down over this element.
onkeypress	false	String	Javascript code executed when a key is pressed or released over this element.
onkeyup	false	String	Javascript code executed when a key is released over this element.

Name	Required	Type	Description
onmousedown	false	String	Javascript code executed when a pointer button is pressed down over this element.
onmousemove	false	String	Javascript code executed when a pointer button is moved within this element.
onmouseout	false	String	Javascript code executed when a pointer button is moved away from this element.
onmouseover	false	String	Javascript code executed when a pointer button is moved onto this element.
onmouseup	false	String	Javascript code executed when a pointer button is released over this element.
required	false	Boolean	Flag indicating that the user is required to provide a submitted value for this input component.
requiredMessage	false	String	A ValueExpression enabled attribute that, if present, will be used as the text of the validation message for the “required” facility, if the “required” facility is used.
validatorMessage	false	String	A ValueExpression enabled attribute that, if present, will be used as the text of the validator message, replacing any message that comes from the validator.
value	false	ValueExpression	The current value of this component.
layout	false	String	The type of layout markup to use when rendering this group. If the value is “block” the renderer must produce an HTML “div” element. Otherwise HTML “span” element must be produced.

10.17.2 Example

```

1 <div xmlns:f="http://java.sun.com/jsf/core"
2   xmlns:h="http://java.sun.com/jsf/html">
3   <div class="ui-widget ui-widget-header">
4     Compute the volume of a cylinder
5   </div>

```

```
6      <h:form id='select-form'>
7          <dl>
8              <dt><label for='height'>Height</label> <h:message for='
9                  height' /></dt>
10             <dd>
11                 <h:selectOneMenu id='height' size='10' value='#{compute
12                     .height}'
13                     styleClass="ui-state-default ui-corner
14                         -all">
15                     <f:selectItem itemLabel="1 inch" itemValue="25.4"/>
16                     <f:selectItem itemLabel="1 feet" itemValue="304.8"
17                         />
18                     <f:selectItem itemLabel="1 yard" itemValue="914.4"
19                         />
20                     <f:converter converterId="float" />
21                 </h:selectOneMenu>
22             </dd>
23             <dt><label for='radius'>Radius</label> <h:message for='
24                 radius' /></dt>
25             <dd>
26                 <h:selectOneMenu id='radius' size='10' value='#{compute
27                     .radius}'
28                     styleClass="ui-state-default ui-corner
29                         -all">
30                     <f:selectItem itemLabel="1 inch" itemValue="25.4"/>
31                     <f:selectItem itemLabel="1 feet" itemValue="304.8"
32                         />
33                     <f:selectItem itemLabel="1 yard" itemValue="914.4"
34                         />
35                     <f:converter converterId="float" />
36                 </h:selectOneMenu>
37             </dd>
38         </dl>
39         <ul class='buttons'>
40             <li>
41                 <h:commandButton id='run' value='Compute' action="#{
42                     compute.run}"
43                     styleClass="ui-button ui-state-default
44                         ui-corner-all"/>
45             </li>
46         </ul>
47     </h:form>
48 </div>
```

11 Util Components

The `util` components are specific to Ada Server Faces and they are provided to help in writing server faces pages. The component is defined in the following namespace:

```
1 xmlns:util="http://code.google.com/p/ada-asf/util"
```

11.1 util:escape

Render the inner component children and escape the resulting HTML text using Javascript or XML escape rules. Using the `util:escape` component is useful when rendering a Javascript extract, a Json response or some XML content.

11.1.1 Attributes

Name	Required	Type	Description
mode	false	String	When set to <code>xml</code> , use the XML escape rules to escape the content. Otherwise, use Javascript escape rules.
rendered	false	Boolean	Flag indicating whether or not this component should be rendered (during Render Response Phase), or processed on any subsequent form submit. The default value for this property is true.

11.1.2 Example

```
1 <div xmlns:util="http://code.google.com/p/ada-asf/util">
2   <util:script>
3     var code = "<util:escape>This is a javascript message.
4     <b>It can be inserted in an HTML element.</b>
5     <p>
6       It can contain HTML code as well as special characters
7       such as quotes (' or ")
8     </p>
9   </util:escape>";
10
11   $('#code-raw').html(code);
12 </util:script>
13
14 <div id='code-raw' />
```



```
15
16 </div>
```

11.2 util:file

This component allows to include an external file in the render response phase.

11.2.1 Attributes

Name	Required	Type	Description
src	true	String	The relative path for the file to be included.
rendered	false	Boolean	Flag indicating whether or not this component should be rendered (during Render Response Phase), or processed on any subsequent form submit. The default value for this property is true.

11.2.2 Example

```
1 <div xmlns:util="http://code.google.com/p/ada-asf/util"
2   xmlns:h="http://java.sun.com/jsf/html">
3   <h:form>
4       <input type="submit" name="file" value="Display 'file.xhtml'"
5           onclick="return ASF.Submit(this);"/>
6       <input type="submit" name="escape" value="Display 'escape.xhtml"
7           onclick="return ASF.Submit(this);"/>
8       <div style='overflow: hidden; width: 100%;'>
9           <code>
10              <util:file src="/util/file.xhtml" escape="true"
11                  rendered="#{param['file'] eq '1'}/>
12              <util:file src="/util/escape.xhtml" escape="true"
13                  rendered="#{param['file'] ne '1'}/>
14          </code>
15      </div>
16  </h:form>
17 </div>
```

11.3 util:flush

This component is used in the render response phase only. It flushes the javascript code that has been queued either by some component or by the `util:script` tag. This allows to flush the javascript at well known places. When `response` is specified and true, it also flushes the response stream.

11.3.1 Attributes

Name	Required	Type	Description
response	false	Boolean	Flag indicating whether the response stream must be flushed.

11.3.2 Example

```
1 <div xmlns:util="http://code.google.com/p/ada-asf/util">
2   <!-- A first javascript piece to fade out the block
3       and update its content -->
4   <util:script>
5       $('#code-raw').fadeOut('slow', function() {
6           var code = "Fade in code";
7           $('#code-raw').html(code).fadeIn();
8       });
9   </util:script>
10  <div id='code-raw' />
11  <div id='code-raw-2' />
12  <util:script>
13      $('#code-raw').fadeOut('slow', function() {
14          var code = "<util:escape>Code</util:escape>";
15
16          $('#code-raw-2').html(code).fadeIn();
17      });
18  </util:script>
19
20  <!-- Javascript enclosed by <util:script> generated here -->
21  <util:flush/>
22 </div>
```

11.4 util:script

In the render response phase, queue some Javascript code in the response stream or queue a Javascript file inclusion. The Javascript code is automatically flushed before sending the response stream. It can

be flushed explicitly by using the `util:flush` component.

11.4.1 Attributes

Name	Required	Type	Description
async	false	Boolean	Flag indicating whether the external Javascript file is loaded asynchronously.
src	false	String	When not empty, render the inclusion of an external Javascript file. The Javascript file location is defined by the src attribute. The Javascript file inclusions are rendered before all javascript code by the <code>util:flush</code> component.
rendered	false	Boolean	Flag indicating whether or not this component should be rendered (during Render Response Phase), or processed on any subsequent form submit. The default value for this property is true.

11.4.2 Example

```
1 <div xmlns:util="http://code.google.com/p/ada-asf/util">
2   <!-- A first javascript piece to fade out the block
3       and update its content -->
4   <util:script>
5       $('#code-raw').fadeOut('slow', function() {
6           var code = "<util:escape>Code appears</util:escape>";
7
8           $('#code-raw').html(code).fadeIn();
9       });
10  </util:script>
11
12  <div id='code-raw' />
13
14  <util:flush/>
15 </div>
```

11.5 util:set

Sets a value on a managed bean attribute. The `var` attribute is a Value expression that describe a managed bean attribute to set. The `value` attribute corresponds to the value that will be assigned to the value expression. This allows to invoke the `Set_Value` method of the managed bean to assign a named value.

11.5.1 Attributes

Name	Type	Description
var	ValueExpression	Value expression to set.
value	any	Value to assign.

11.5.2 Example

```
1 <html xmlns:h="http://java.sun.com/jsf/html"
2     xmlns:c="http://java.sun.com/jstl/core"
3     xmlns:util="http://code.google.com/p/ada-asf/util">
4   <body>
5     <util:set var="#{form.email}" value="Potter@gmail.com" />
6     <util:set var="#{form.name}" value="Harry" />
7     Email: #{form.email}
8     Name: #{form.name}
9   </body>
10 </html>
```

12 Widget Components

The `widget` components are specific to Ada Server Faces and they provide high level components to help in designing and providing a web interface.

```
1 xmlns:w="http://code.google.com/p/ada-asf/widget"
```

12.1 w:accordion

The `w:accordion` component provides a vertical tab component.

12.1.1 Attributes

Name	Required	Type	Description
collapsible	false	Boolean	This flag indicates whether the accordion must close all the sections at once.
rendered	false	Boolean	Flag indicating whether or not this component should be rendered (during Render Response Phase), or processed on any subsequent form submit. The default value for this property is true.

12.1.2 Example

```
1 <div xmlns:w="http://code.google.com/p/ada-asf/widget"
2   style='overflow: auto; padding: 10px'>
3   <w:accordion id="people" collapsible="true">
4     <w:tab title="Dennis">
5       Dennis MacAlistair Ritchie was an American computer
6         scientist who "helped shape the digital era."
7       He created the C programming language and, with long-time
8         colleague Ken Thompson,
9       the Unix operating system. Ritchie and Thompson received
10      the Turing Award from the ACM in 1983,
11      the Hamming Medal from the IEEE in 1990 and the National
12      Medal of Technology from President Clinton
13      in 1999. Ritchie was the head of Lucent Technologies System
14      Software Research Department when he
15      retired in 2007. He was the 'R' in K&R C and commonly
16      known by his username dmr.
```

```
11      </w:tab>
12      <w:tab title="Alan">
13          Alan Mathison Turing, was an English mathematician,
14              logician, cryptanalyst, and computer scientist.
15          He was highly influential in the development of computer
16              science, giving a formalisation of the
17              concepts of "algorithm" and "computation" with the Turing
18              machine, which can be considered a model
19              of a general purpose computer. Turing is widely considered
20              to be the father of computer science
21              and artificial intelligence.
22
23          http://en.wikipedia.org/wiki/Alan_Turing
24      </w:tab>
25      <w:tab title="Ada">
26          Augusta Ada King, Countess of Lovelace (10 December 1815 -
27              27 November 1852),
28          born Augusta Ada Byron and now commonly known as Ada
29              Lovelace, was an English mathematician
30              and writer chiefly known for her work on Charles Babbage's
31              early mechanical general-purpose computer,
32              the Analytical Engine. Her notes on the engine include what
33              is recognised as the first algorithm
34              intended to be processed by a machine. Because of this, she
35              is often described as the world's first
36              computer programmer.
37      </w:tab>
38  </w:accordion>
39 </div>
```

12.2 w:autocomplete

The `w:autocomplete` component combines the `h:inputText` and `h:message` components and provides autocomplete functionality on the input field. It renders the title and the input form field. The error message associated with the input field is rendered if there is one. The title, input field and message are combined within an HTML `dl`, `dt` and `dd` elements.

When the user enters some text, the form is submitted for autocomplete. The `w:autocomplete` component handles the form submission and uses the `autocompleteList` attribute to find out possible completions. It then returns that list that is then displayed by the client.

12.2.1 Attributes

Name	Required	Type	Description
title	false	String	The title to display for the input field.
autocompleteList	false	String	The list of values for the autocompletion.
rendered	false	Boolean	Flag indicating whether or not this component should be rendered (during Render Response Phase), or processed on any subsequent form submit. The default value for this property is true.

12.2.2 Example

```

1 <div xmlns:f="http://java.sun.com/jsf/core"
2   xmlns:w="http://code.google.com/p/ada-asf/widget"
3   xmlns:h="http://java.sun.com/jsf/html">
4   <div class="ui-widget ui-widget-header">
5       Compute the volume of a cylinder
6   </div>
7   <h:form id='text-form'>
8       <w:autocomplete title="Country" id='country' size='30'
9           autocompleteList="#{countries}"
10              value='#{messages.email}'
11              styleClass="ui-corner-all">
12   </w:autocomplete>
13   <ul class='buttons'>
14       <li>
15           <h:commandButton id='run' value='Compute' action="#{
16               compute.run}"
17               styleClass="ui-button ui-state-default
18               ui-corner-all"/>
19       </li>
20   </ul>
21 </h:form>
22 </div>

```

12.3 w:chosen

The `w:chosen` component is a `h:selectOne` component that uses jQuery Chosen support. It renders the `select` component with its options and activates the jQuery Chosen support on it.

The `w:chosen` component uses the `options` facet to allow to provide specific options to the jQuery Chosen library. The `events` facet can be used to invoke jQuery specific operations on the Chosen selector and bind some events.

12.3.1 Attributes

Name	Required	Type	Description
id	true	String	The id of the element (this is mandatory for the correct jQuery Chosen support).
rendered	false	Boolean	Flag indicating whether or not this component should be rendered (during Render Response Phase), or processed on any subsequent form submit. The default value for this property is true.

12.3.2 Example

```

1 <div xmlns:f="http://java.sun.com/jsf/core"
2   xmlns:w="http://code.google.com/p/ada-asf/widget"
3   style='padding: 10px'>
4   <div style="padding: 10px">
5     <w:chosen value '#{compute.height}' id="chosen-example-1">
6       <f:facet name="options">width: "95%", disable_search: true
7         </f:facet>
8       <f:facet name="events">.bind("change", function(event,
9         params) {
10           alert("Selected country: " + $(this).val());
11         } )
12       </f:facet>
13       <f:selectItems value="#{countries}" />
14     </w:chosen>
15   </div>
16   <div style="padding: 10px">
17     <w:chosen value '#{compute.height}' id="chosen-example-2">
18       <f:facet name="options">width: "95%"</f:facet>
19       <f:facet name="events">.bind("change", function(event,
20         params) {
21           alert("Selected country: " + $(this).val());
22         } )
23       </f:facet>
24       <f:selectItems value="#{countries}" />
25     </w:chosen>
26   </div>
27 </div>

```


12.4 w:gravatar

This component renders an image whose link is the gravatar's link of a person's email address.

12.4.1 Attributes

Name	Required	Type	Description
email	true	String	The email address to create the gravatar image link.
secure	false	Boolean	When True, the secure link to the gravatar image is created (https). The default creates an http link only.
size	false	Integer	The size of the gravatar image (from 1 to 2048 pixels max).
default	false	String	Default.
alt	false	String	The image alt attribute. By default, the email address is used for the alt attribute.
rendered	false	Boolean	Flag indicating whether or not this component should be rendered (during Render Response Phase), or processed on any subsequent form submit. The default value for this property is true.

12.4.2 Example

```

1 <div xmlns:f="http://java.sun.com/jsf/core"
2   xmlns:w="http://code.google.com/p/ada-asf/widget"
3   xmlns:h="http://java.sun.com/jsf/html" style='overflow: auto;*>
4   <div class="ui-widget ui-widget-header">
5     Display a person's gravatar
6   </div>
7   <h:form id='text-form'>
8     <w:inputText title="Type the person email address" id='email'
9       size='40'
10      value='#{message.email}'
11      styleClass="ui-corner-all">
12   </w:inputText>
13   <div class='gravatar grid_6'>
14     <span>default</span>
15     <w:gravatar email='#{message.email}' size='120'/*>

```

```

15         </div>
16         <div class='gravatar grid_6'>
17             <span>wavatar</span>
18             <w:gravatar email='{message.email}' default='wavatar' size=
               ='120' />
19         </div>
20         <div class='gravatar grid_6'>
21             <span>retry</span>
22             <w:gravatar email='{message.email}' default='retro' size='
               120' />
23         </div>
24         <div class='gravatar grid_6'>
25             <span>monsterid</span>
26             <w:gravatar email='{message.email}' default='monsterid'
               size='120' />
27         </div>
28         <ul class='buttons'>
29             <li>
30                 <h:commandButton id='run' value='Display gravatar'
31                                     styleClass="ui-button ui-state-default
32                                         ui-corner-all"/>
32             </li>
33         </ul>
34     </h:form>
35 </div>

```

12.5 w:inputDate

The `w:inputDate` component combines the `h:inputText` and `h:message` components and a date picker to select a date. It renders the title and the input form field with the Javascript support to activate the date picker. The error message associated with the input field is rendered if there is one. The title, input field and message are combined within an HTML `dl`, `dt` and `dd` elements.

The date picker is based on the jQuery date picker.

12.5.1 Attributes

Name	Required	Type	Description
title	false	String	The title to display for the input field.

Name	Required	Type	Description
rendered	false	Boolean	Flag indicating whether or not this component should be rendered (during Render Response Phase), or processed on any subsequent form submit. The default value for this property is true.

12.5.2 Example

```

1 <div xmlns:f="http://java.sun.com/jsf/core"
2   xmlns:w="http://code.google.com/p/ada-asf/widget"
3   xmlns:h="http://java.sun.com/jsf/html">
4   <div class="ui-widget ui-widget-header">
5       Select a date
6   </div>
7   <h:form id='text-form'>
8       <w:inputDate title="Message date" id='date' size='20'
9           value='#{message.text}'
10          dateFormat="yy-dd-mm"
11          styleClass="ui-corner-all">
12   </w:inputDate>
13   <w:inputText title="Email" id='email' size='10'
14       value='#{message.email}'
15       styleClass="ui-corner-all">
16   </w:inputText>
17   <ul class='buttons'>
18       <li>
19           <h:commandButton id='run' value='Post' action="#{
20               message.post}"
21               styleClass="ui-button ui-state-default
22                   ui-corner-all"/>
23       </li>
24   </ul>
25 </h:form>
26 </div>

```

12.6 w:inputText

The `w:inputText` component combines the `h:inputText` and `h:message` components. It renders the title and the input form field. The error message associated with the input field is rendered if there is one. The title, input field and message are combined within an HTML `dl`, `dt` and `dd` elements.

12.6.1 Attributes

Name	Required	Type	Description
title	false	String	The title to display for the input field.
id	false	String	The component identifier for this component. This value must be unique within the closest parent component that is a naming container.
rendered	false	Boolean	Flag indicating whether or not this component should be rendered (during Render Response Phase), or processed on any subsequent form submit. The default value for this property is true.
converterMessage	false	String	A ValueExpression enabled attribute that, if present, will be used as the text of the converter message, replacing any message that comes from the converter.

12.6.2 Example

```

1 <div xmlns:f="http://java.sun.com/jsf/core"
2   xmlns:w="http://code.google.com/p/ada-asf/widget"
3   xmlns:h="http://java.sun.com/jsf/html">
4   <div class="ui-widget ui-widget-header">
5       Compute the volume of a cylinder
6   </div>
7   <h:form id='text-form'>
8       <w:inputText title="Height" id='height' size='10'
9           value '#{compute.height}'
10          styleClass="ui-corner-all">
11           <f:converter converterId="float" />
12       </w:inputText>
13       <w:inputText title="Radius" id='radius' size='10'
14           value '#{compute.radius}'
15           styleClass="ui-corner-all">
16           <f:converter converterId="float" />
17       </w:inputText>
18       <ul class='buttons'>
19           <li>
20               <h:commandButton id='run' value='Compute' action="#{
21                   compute.run}"
22                   styleClass="ui-button ui-state-default
23                   ui-corner-all"/>

```

```
22         </li>
23     </ul>
24 </h:form>
25 </div>
```

12.7 w:like

This component renders a like button for Facebook or Twitter (more like implementations can be added programatically in Ada). The like button code is rendered within a `div` element whose style and class can be customized.

The `kind` attribute defines what like button must be generated.

12.7.1 Facebook

The Facebook like button is generated with the `facebook` kind attribute value.

When using the Facebook button, the component will pass several attributes to the Facebook button: data-layout, data-show-faces, data-width, data-action, data-font, data-colorscheme, data-ref, data-kid_directed_site, data-send.

The Facebook like button requires that you register your application and get a facebook client ID. The like component will use the configuration property `facebook.client_id` to retrieve this client ID.

12.7.2 Twitter

The Tweet Button is generated with the `twitter` kind attribute value. The following attributes are passed to the Tweet button: data-via, data-count, data-size.

12.7.3 Attributes

Name	Required	Type	Description
kind	true	String	The type of like button to generate: “facebook”, “twitter”.
href	false	String	The optional URL to pass to the like button. The default is to use the current page URL.

Name	Required	Type	Description
styleClass	false	String	The CSS class to be applied in the div element that contains the like button.
style	false	String	The CSS style to be applied in the div element that contains the like button.
rendered	false	Boolean	Flag indicating whether or not this component should be rendered (during Render Response Phase), or processed on any subsequent form submit. The default value for this property is true.

12.7.4 Example

```
1 <div xmlns:w="http://code.google.com/p/ada-asf/widget" style='overflow:
  auto;'>
2   <div class="ui-widget ui-widget-header">
3     Display a social like button
4   </div>
5   <w:like type="facebook" href="http://www.google.com"/>
6   <w:like type="twitter" data-count="vertical"/>
7 </div>
```

12.8 w:panel

The `w:panel` component provides a collapsible panel with a header, a content and an optional footer.

12.8.1 Attributes

Name	Required	Type	Description
header	true	String	The header title to display at the top of the panel.
footer	false	String	The optional title to display at the bottom of the panel.
closable	false	Boolean	When true, the panel can be closed by clicking on the close icon action in the header.

Name	Required	Type	Description
toggleable	false	Boolean	When true, the panel can be collapsed by clicking on the expand/collapse icon action in the header.
rendered	false	Boolean	Flag indicating whether or not this component should be rendered (during Render Response Phase), or processed on any subsequent form submit. The default value for this property is true.

12.8.2 Example

```
1 <div xmlns:f="http://java.sun.com/jsf/core"
2     xmlns:w="http://code.google.com/p/ada-asf/widget"
3     xmlns:h="http://java.sun.com/jsf/html" style='overflow: auto;
4     padding: 10px'>
5     <w:panel header='Ada Lovelace' closable="true" toggleable="true">
6         Augusta Ada King, Countess of Lovelace (10 December 1815 – 27
7             November 1852),
8         born Augusta Ada Byron and now commonly known as Ada Lovelace,
9         was an English mathematician
10        and writer chiefly known for her work on Charles Babbage's
11        early mechanical general-purpose computer,
12        the Analytical Engine. Her notes on the engine include what is
13        recognised as the first algorithm
14        intended to be processed by a machine. Because of this, she is
15        often described as the world's first
16        computer programmer.
17    </w:panel>
18    <f:facet name="footer">
19        <a href="http://en.wikipedia.org/wiki/Ada_Lovelace">
20            More on wikipedia
21        </a>
22    </f:facet>
23 </div>
```

12.9 w:tab

The `w:tab` component defines a tab content to be displayed within a tab selection.

12.9.1 Attributes

Name	Required	Type	Description
title	false	String	The tab title.
rendered	false	Boolean	Flag indicating whether or not this component should be rendered (during Render Response Phase), or processed on any subsequent form submit. The default value for this property is true.

12.9.2 Example

```

1 <div xmlns:f="http://java.sun.com/jsf/core"
2   xmlns:w="http://code.google.com/p/ada-asf/widget"
3   style='overflow: auto; padding: 10px'>
4   <w:tabView header='Ada Lovelace' closable="true" toggleable="true"
5     effect="blind" collapsible="true">
6     <w:tab title="Dennis">
7         Dennis MacAlistair Ritchie was an American computer
8         scientist who "helped shape the digital era."
9         He created the C programming language and, with long-time
10        colleague Ken Thompson,
11        the Unix operating system. Ritchie and Thompson received
12        the Turing Award from the ACM in 1983,
13        the Hamming Medal from the IEEE in 1990 and the National
14        Medal of Technology from President Clinton
15        in 1999. Ritchie was the head of Lucent Technologies System
16        Software Research Department when he
17        retired in 2007. He was the 'R' in K&R C and commonly
18        known by his username dmr.
19    </w:tab>
20    <w:tab title="Alan">
21        Alan Mathison Turing, was an English mathematician,
22        logician, cryptanalyst, and computer scientist.
23        He was highly influential in the development of computer
24        science, giving a formalisation of the
25        concepts of "algorithm" and "computation" with the Turing
26        machine, which can be considered a model
27        of a general purpose computer. Turing is widely considered
28        to be the father of computer science
29        and artificial intelligence.
30
31        http://en.wikipedia.org/wiki/Alan_Turing
32    </w:tab>
33    <w:tab title="Ada">
34        Augusta Ada King, Countess of Lovelace (10 December 1815 -
35        27 November 1852),

```



```

24      born Augusta Ada Byron and now commonly known as Ada
25      Lovelace, was an English mathematician
26      and writer chiefly known for her work on Charles Babbage's
27      early mechanical general-purpose computer,
28      the Analytical Engine. Her notes on the engine include what
29      is recognised as the first algorithm
30      intended to be processed by a machine. Because of this, she
31      is often described as the world's first
32      computer programmer.
33    </w:tab>
34  </w:tabView>
35</div>

```

12.10 w:tabView

The `w:tabView` component defines a tab selection. It uses the jQuery UI tabs. Each tab must be represented by a `w:tab` component which indicates the tab title and content.

12.10.1 Attributes

Name	Required	Type	Description
collapsible	false	Boolean	When true, the tabs are collapsible.
effect	false	String	The effect to use when switching tabs.
duration	false	Integer	The effect duration.
rendered	false	Boolean	Flag indicating whether or not this component should be rendered (during Render Response Phase), or processed on any subsequent form submit. The default value for this property is true.

12.10.2 Example

```

1 <div xmlns:f="http://java.sun.com/jsf/core"
2     xmlns:w="http://code.google.com/p/ada-asf/widget"
3     style='overflow: auto; padding: 10px'>
4   <w:tabView header='Ada Lovelace' closable="true" toggleable="true"
5     effect="blind" collapsible="true">
6     <w:tab title="Dennis">
7       Dennis MacAlister Ritchie was an American computer
8       scientist who "helped shape the digital era."

```

```
7      He created the C programming language and, with long-time
8      colleague Ken Thompson,
9      the Unix operating system. Ritchie and Thompson received
10     the Turing Award from the ACM in 1983,
11     the Hamming Medal from the IEEE in 1990 and the National
12     Medal of Technology from President Clinton
13     in 1999. Ritchie was the head of Lucent Technologies System
14     Software Research Department when he
15     retired in 2007. He was the 'R' in K&R C and commonly
16     known by his username dmr.
17 </w:tab>
18 <w:tab title="Alan">
19     Alan Mathison Turing, was an English mathematician,
20     logician, cryptanalyst, and computer scientist.
21     He was highly influential in the development of computer
22     science, giving a formalisation of the
23     concepts of "algorithm" and "computation" with the Turing
24     machine, which can be considered a model
25     of a general purpose computer. Turing is widely considered
26     to be the father of computer science
27     and artificial intelligence.
28     http://en.wikipedia.org/wiki/Alan\_Turing
29 </w:tab>
30 <w:tab title="Ada">
31     Augusta Ada King, Countess of Lovelace (10 December 1815 -
32     27 November 1852),
33     born Augusta Ada Byron and now commonly known as Ada
34     Lovelace, was an English mathematician
35     and writer chiefly known for her work on Charles Babbage's
36     early mechanical general-purpose computer,
37     the Analytical Engine. Her notes on the engine include what
38     is recognised as the first algorithm
39     intended to be processed by a machine. Because of this, she
40     is often described as the world's first
41     computer programmer.
42 </w:tab>
43 </w:tabView>
44 </div>
```

13 Tips

13.1 Open dialog box

The `ASF.OpenDialog` Javascript operation can be used to open a dialog box by fetching its content from the server. First, create the dialog box file in a separate XHTML file. The file should start with a `<f:view>` component with a valid `contentType` attribute.

```
1 <f:view contentType="text/html; charset=UTF-8"
2     xmlns:f="http://java.sun.com/jsf/core"
3     xmlns:w="http://code.google.com/p/ada-asf/widget"
4     xmlns:h="http://java.sun.com/jsf/html">
5   <div>
6     <h:form id='subscribe-send'>
7     ...
8       <ul class='awa-buttons'>
9         <li>
10           <h:commandButton id='send-mail'
11             title="..."
12             value='...'
13             styleClass="ui-button ui-state-default ui-corner-all"
14             action="#{action}"/>
15         </li>
16       </ul>
17     </h:form>
18   </div>
19 </f:view>
```

The `ASF.OpenDialog` is simply called to trigger the opening of the dialog. The second parameter should be the name of the dialog box JavaScript variable. The last parameter is the URL to fetch to get the dialog box content.

```
1 <a class='awa-button' href="#"
2   onclick="return ASF.OpenDialog(this, 'openDialog', '#{contextPath
3   }/');">
4   Open Dialog
5 </a>
```

13.2 ASF Actions

13.2.1 Updating content

Several actions are available to update the content of some DOM components:

- The update action replaces a complete DOM content,

- The prepend action adds some content before a DOM node,
- The append action adds some content after a DOM node.

The DOM component is identified by a jQuery identification string and passed in the `id` attribute. The content to replace, prepend or append is specified in the `data` attribute which can contain HTML tags.

```
1 {  
2   "action": "update",  
3   "id": "<name>",  
4   "data": "<content>"  
5 }
```

13.2.2 Hide or show

Several actions are available to hide, show and provide visual effects when displaying or hiding some component.

- show to make a DOM component visible,
- hide to make a DOM component invisible,
- fadeIn to show a component after a fade-in visual effect,
- fadeOut to hide a component after a fade-out visual effect,
- slideUp to slide a component up,
- slideDown to slide a component down.

The `id` attribute is used to defined the DOM component onto which the action is made.

```
1 {  
2   "action": "fadeIn",  
3   "id": "<name>"  
4 }
```

13.2.3 Updating CSS class

- addClass to add a CSS class to some DOM components,
- removeClass to remove a CSS class to some DOM components.

```
1 {  
2   "action": "addClass",  
3   "id": "<name>",  
4   "data": "<class-name>"  
5 }
```

13.2.4 Redirect

The redirect action can be used to redirect the browser to a new page. The redirection page is defined by the url attribute.

```
1 {  
2   "action": "redirect",  
3   "url": "<redirection-url>"  
4 }
```

13.2.5 Get content

The redirect action can be used to redirect the browser to a new page. The redirection page is defined by the url attribute.

```
1 {  
2   "action": "get",  
3   "id": "<name>",  
4   "url": "<get-url>"  
5 }
```

13.2.6 clear action

TBW