# AdminFaces

Version 1.0.0-RC7

# Table of Contents

Read this documentation in HTML5 here.

# Chapter 1. Introduction

AdminFaces is an open source project which brings Bootstrap and AdminLTE to your application via a PrimeFaces theme and a JSF `responsive` template.

AdminFaces ecosystem is composed by the following projects:

- Admin Theme: A Primefaces theme based on Bootstrap and Admin LTE where Primefaces components are customized to look like mentioned frameworks.
- Admin Template: A **fully responsive** Java Server Faces `admin` template (used by this Showcase) which is also based on **Bootstrap** and **Admin LTE**.
- Admin Showcase: A showcase web application, deployed on openshift, which demonstrates AdminFaces main features and components.
- Admin Designer: The same showcase application with Admin Theme and Admin Template bundled (instead of being library dependencies) in order to make it easier to customize the theme and the template.
- Admin Starter: A simple starter project to get you started with AdminFaces.
- Admin Mobile: A simple Android Studio project which uses Webview to create an hybrid web app based on Admin Showcase.

In subsequent chapters we will drive through each project in detail.

# Chapter 2. Admin Theme

Admin Theme is a PrimeFaces theme where components are styled to look like AdminLTE ones (which in turn are based on Bootstrap).

## 2.1. Prerequisites

The only pre-requisite is `PrimeFaces` and `Font Awesome`.

Since PrimeFaces `5.1.1` font awesome comes embedded, you just need to activate it in `web.xml`:

```
<context-param>
    <param-name>primefaces.FONT_AWESOME</param-name>
    <param-value>true</param-value>
</context-param>
```

For **previous versions** or if you need to upgrade FA version you may include it in your pages by using webjars:

```
<h:outputStylesheet library="webjars" name="font-
awesome/4.7.0/css/font-awesome-jsf.css" />
```

and add fontawesome webjar in your classpath:

```
<dependency>
    <groupId>org.webjars</groupId>
    <artifactId>font-awesome</artifactId>
    <version>4.7.0</version>
</dependency>
```

## 2.2. Usage

To start using the theme you need the following:

1. Add admin theme to your classpath:

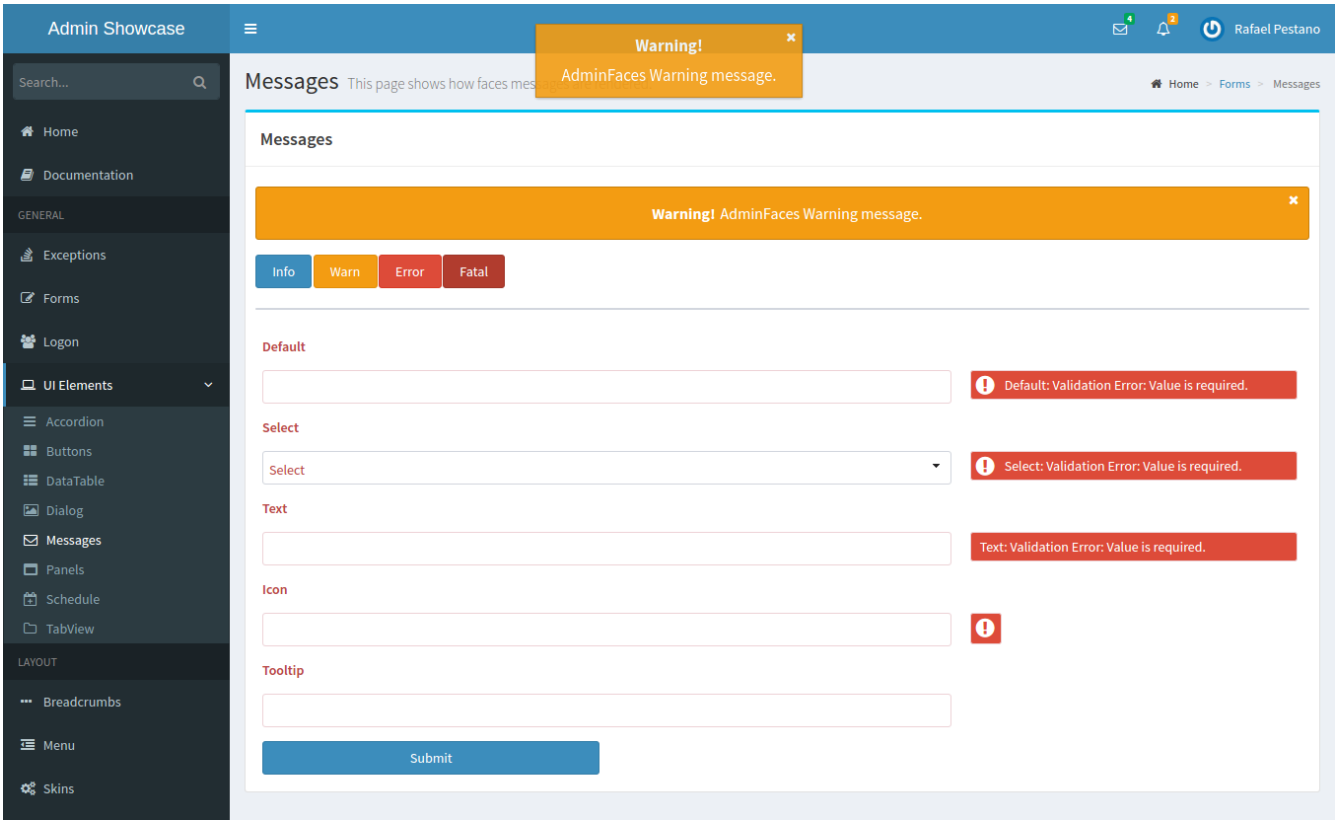
```
<dependency>
    <groupId>com.github.adminfaces</groupId>
    <artifactId>admin-theme</artifactId>
    <version>1.0.0-RC7</version>
</dependency>
```

2. Activate the theme in `web.xml`

```
<context-param>
    <param-name>primefaces.THEME</param-name>
    <param-value>admin</param-value>
</context-param>
```

If you use Admin Template the theme already comes activated.

Now PrimeFaces components are styled like Bootstrap and AdminLTE.




See showcase forms page to get an idea.

## 2.3. Architecture

The theme uses less as css pre-processor. Each PrimeFaces component has its own less file:

*src/main/resources/less*

```
├──── admin-lte
│     │
│     ├──── bootstrap ①
│     │
│     ├──── skins ②
│     │
│     └──── admin lte less files
│
├──── primefaces-admin
│     │
│     ├── components ③
│     │    │
│     │    ├──── accordeon.less
│     │    │
│     │    ├──── autocomplete.less
│     │    │
│     │    └──── etc...
│     │
│     ├── theme.less ④
│     │
│     └── variables.less
```

① Bootstrap variables and mixins are used as reference in AdminLTE and admin theme less files

② Built in skins

③ PrimeFaces components

④ Components and Admin-LTE less files are included in theme.less

After compilation it will generate the theme.css with Admin-LTE, Bootstrap and Primefaces components.

Bootstrap.css (from src/META-INF/resources) is included in theme.less but can be removed via maven classifiers

Bootstrap less is not maintained in this project only it's mixins.

## 2.4. Theme classifiers

This project uses maven classifiers to offer multiple `faces` (pum intended) of Admin Theme. Below is the description of each classifier and how to use it.

### 2.4.1. Default (no classifier)

The default theme comes `compressed`, with `Bootstrap (3.3.7)` embedded and uses `JSF resource handling` for loading images and web fonts.

*Maven usage*

```
<dependency>
    <groupId>com.github.adminfaces</groupId>
    <artifactId>admin-theme</artifactId>
    <version>1.0.0-RC7</version>
</dependency>
```

### 2.4.2. Dev classifier

The `dev` classifier will bring a theme.css **without minification**.

*Maven usage*

```
<dependency>
    <groupId>com.github.adminfaces</groupId>
    <artifactId>admin-theme</artifactId>
    <version>1.0.0-RC7</version>
    <classifier>dev</classifier>
</dependency>
```

### 2.4.3. Without Bootstrap classifier

The `without-bootstrap` classifier will bring a theme.css **without bootstrap embedded** so it's up to the developer to provide Bootstrap within the application.

*Maven usage*

```
<dependency>
    <groupId>com.github.adminfaces</groupId>
    <artifactId>admin-theme</artifactId>
    <version>1.0.0-RC7</version>
    <classifier>without-bootstrap</classifier>
</dependency>
```

### 2.4.4. Without JSF classifier

The `without-jsf` classifier will bring a theme.css **without JSF resource handling** so the theme can be used on projects (derived from PrimeFaces) without JSF like Prime React, PrimeUI or PrimeNG.

*Maven usage*

```
<dependency>
    <groupId>com.github.adminfaces</groupId>
    <artifactId>admin-theme</artifactId>
    <version>1.0.0-RC7</version>
    <classifier>without-jsf</classifier>
</dependency>
```

## 2.5. Avoiding theme caching

Whenever the theme is updated to a new version in the project users may have to clear their browser caches to get the changes of the new theme.
Sometimes a theme update even introduces conflicts and only clearing browser cache fixes them.

To solve this issues you can use a theme classifier called **no-cache**:

*pom.xml*

```
<dependency>
    <groupId>com.github.adminfaces</groupId>
    <artifactId>admin-theme</artifactId>
    <version>1.0.0-RC7</version>
    <classifier>no-cache</classifier>
</dependency>
```

This classifier **appends the theme version** to the name of theme so you need to change the theme name in web.xml:

*web.xml*

```
<context-param>
    <param-name>primefaces.THEME</param-name>
    <param-value>admin-1.0.0-RC7</param-value>
</context-param>
```

## 2.6. Development

To get your hands dirty with admin theme it is recommended to use Admin Designer in combination with Brackets or any tool that `compile less` files to css on save.

Using designer (which is backed by wildfly swarm) plus brackets will let you change the components less files and see the results instantly.

theme.less is already brackets aware so you just need to change any component less file, save it and see the results in Admin Designer.

## 2.7. Snapshots

Theme `Snapshots` are published to maven central on each commit, to use it just declare the repository below on your `pom.xml`:

```
<repositories>
    <repository>
        <snapshots/>
        <id>snapshots</id>
        <name>libs-snapshot</name>
        <url>https://oss.sonatype.org/content/repositories/snapshots</url>
    </repository>
</repositories>
```

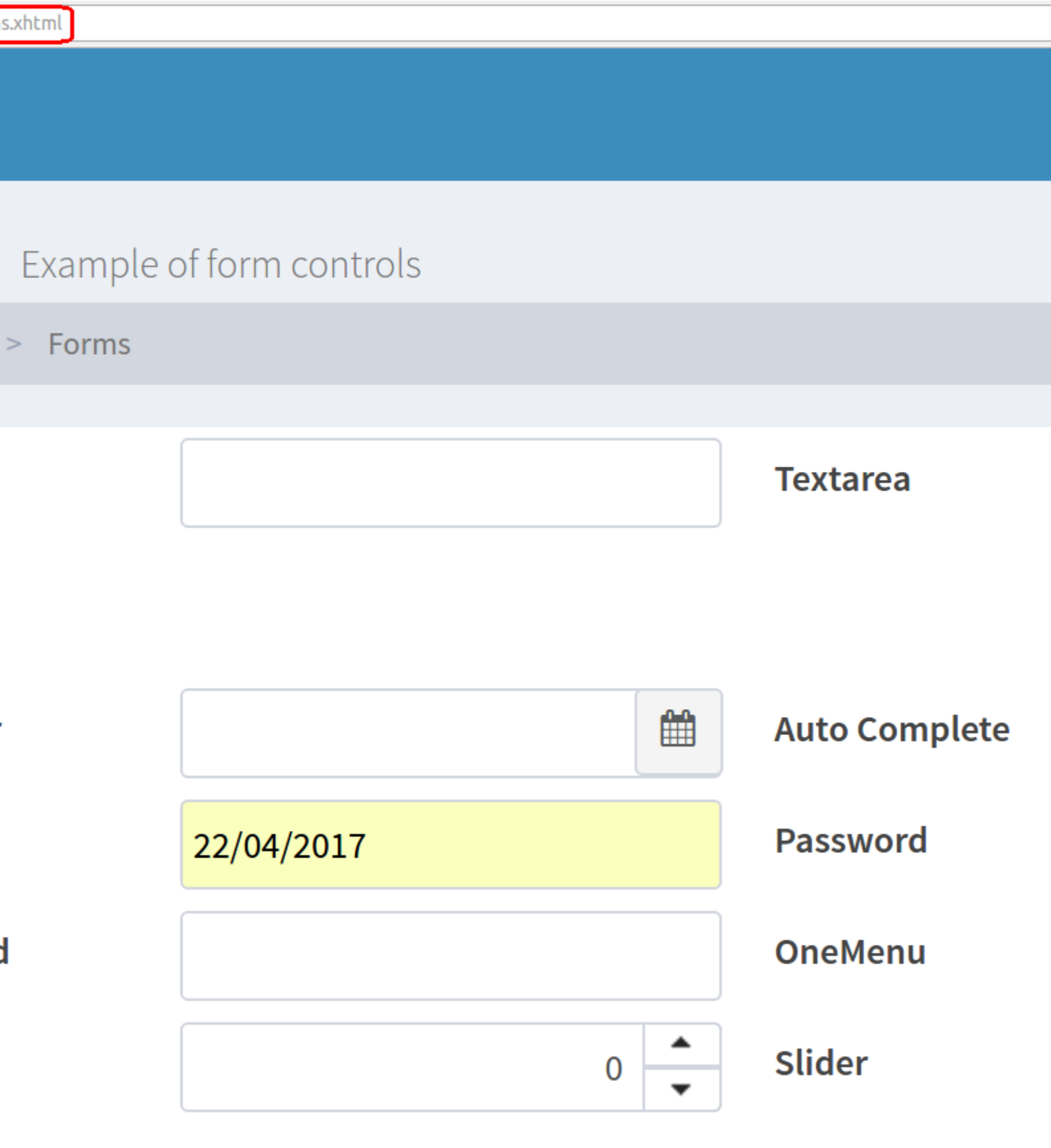
# Chapter 3. Admin Template

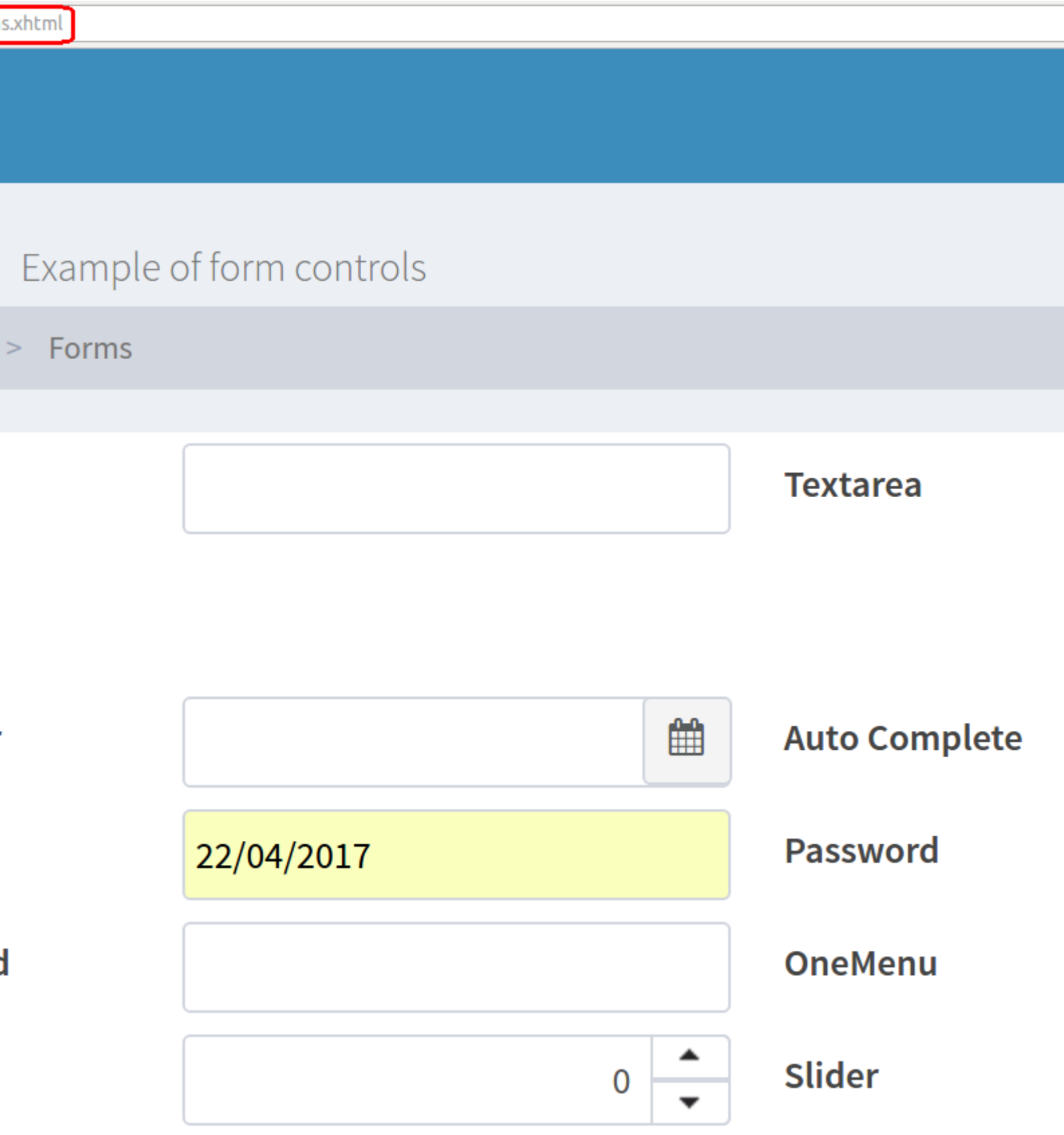
Admin Template is a **fully responsive** *Java Server Faces* `admin template` based on Bootstrap and Admin LTE.

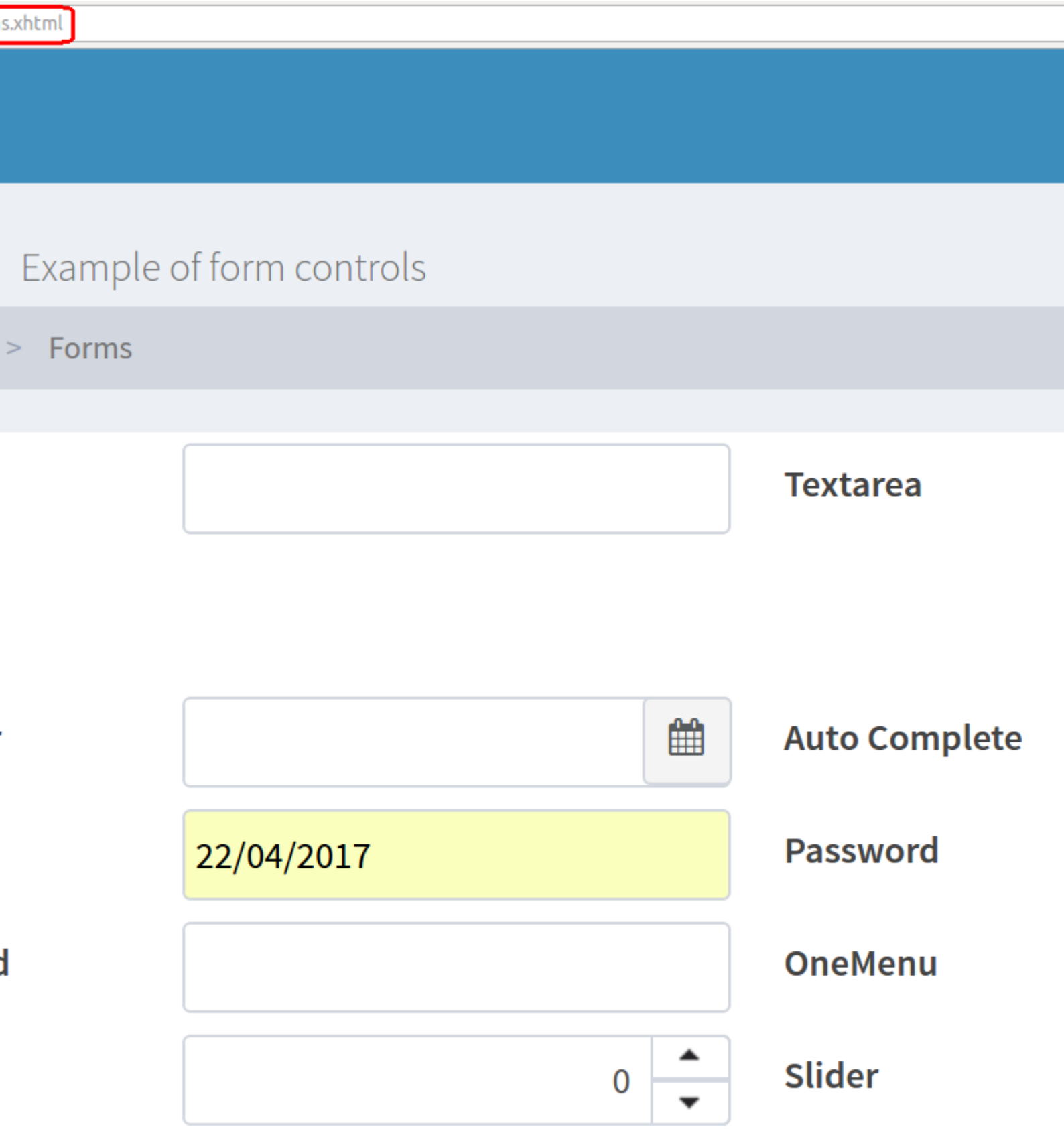
## 3.1. Features

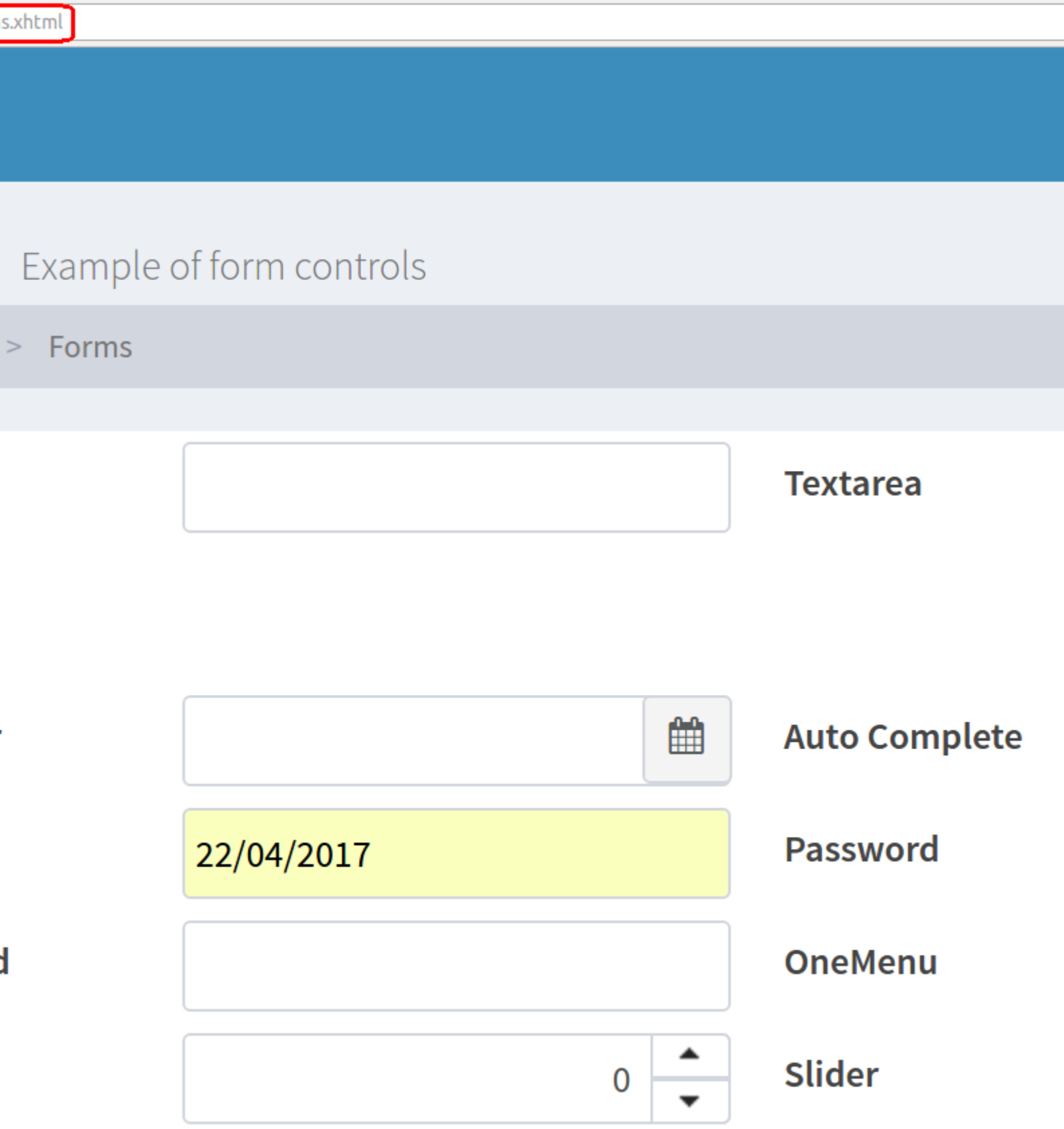
Below is a non exhaustive list of notable features brought out of the box by this template:

- Fully responsive
  - Its based on Bootstrap and AdminLTE two well tested and solid frameworks
- Mobile and PWA friendly
- Some functionalities are enabled only on mobile devices, like:
  - Material design load bar
  - Ripple effect based on materialize
  - Touch enabled menu to slide in/out
    [dd37121e 2296 11e7 855c 8f20b59dcf5f]
  - Scroll to top
- Automatically activates (highlight) menu based on current page


s.xhtml
Example of form controls
> Forms
Textarea
Auto Complete
22/04/2017
Password
OneMenu
0
Slider


s.xhtml
Example of form controls
> Forms
Textarea
Auto Complete
22/04/2017
Password
OneMenu
0
Slider


s.xhtml
Example of form controls
> Forms
Textarea
Auto Complete
22/04/2017
Password
OneMenu
0
Slider




- Custom error pages
- Configurable, see Configuration
- Breadcrumb based navigation
- Back to previous screen when logging in again after session expiration (or accessing a page via

url without being logged in)

Most of the above features can be disabled via configuration mechanism;

## 3.2. Usage

Add the following dependency to your classpath:

```
<dependency>
    <groupId>com.github.adminfaces</groupId>
    <artifactId>admin-template</artifactId>
    <version>1.0.0-RC7</version>
</dependency>
```

Admin template will bring the following transitive dependencies:

```
<dependency>
    <groupId>com.github.adminfaces</groupId>
    <artifactId>admin-template</artifactId>
    <version>1.0.0-RC7</version>
</dependency>

<dependency>
    <groupId>org.primefaces</groupId>
    <artifactId>primefaces</artifactId>
    <version>6.1</version>
</dependency>

<dependency>
    <groupId>org.omnifaces</groupId>
    <artifactId>omnifaces</artifactId>
    <version>2.1</version>
</dependency>
```

Which you can override in your pom.xml as needed.

With the template dependency in classpath now you can use `admin` facelets template into your JSF pages.

### 3.2.1. Example

Consider the following sample page:

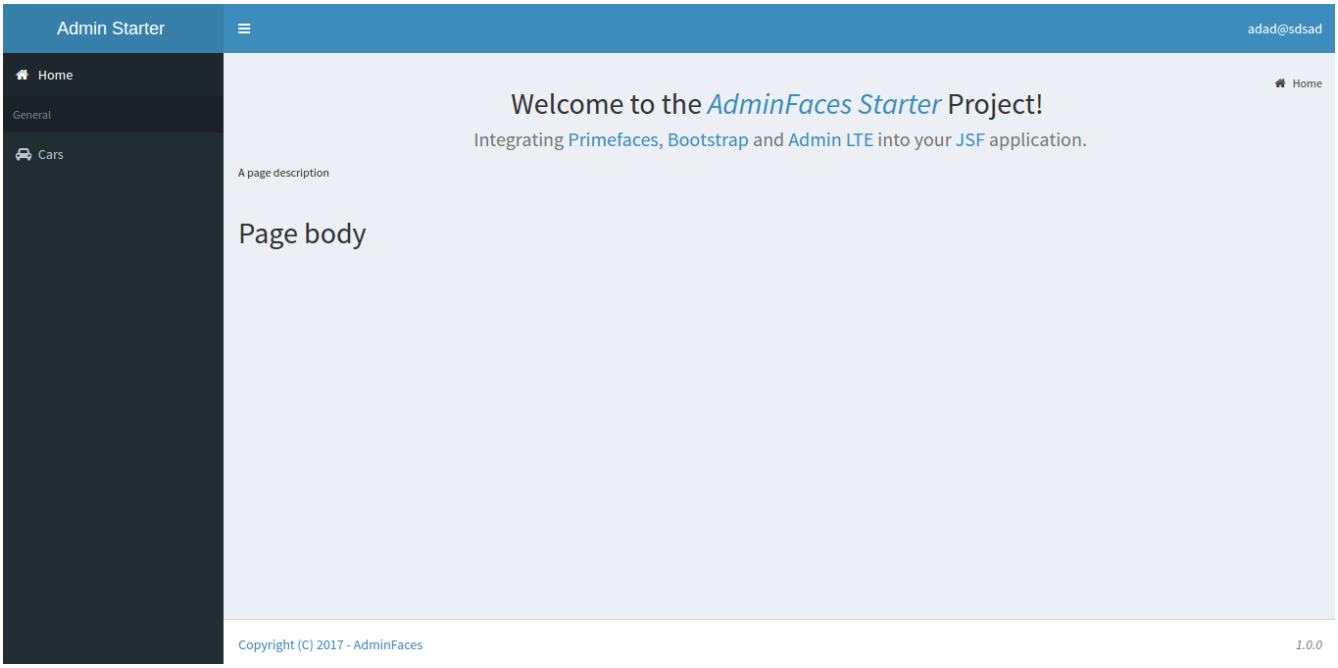
```
<?xml version="1.0" encoding="UTF-8"?>
<ui:composition xmlns="http://www.w3.org/1999/xhtml"
                xmlns:ui="http://java.sun.com/jsf/facelets"
```

```
                xmlns:p="http://primefaces.org/ui"
                template="/admin.xhtml"> ①

    <ui:define name="head">
        <title>Admin Starter</title>
    </ui:define>

    <ui:define name="logo-lg">
        Admin Starter
    </ui:define>

    <ui:define name="logo-mini">
        Admin
    </ui:define>

    <ui:define name="menu">
        <ul class="sidebar-menu">
            <li>
                <p:link href="/index.xhtml" onclick="clearBreadCrumbs()">
                    <i class="fa fa-home"></i>
                    <span>Home</span>
                </p:link>
            </li>
            <li class="header">
                General
            </li>
            <li>
                <p:link href="/car-list.xhtml">
                    <i class="fa fa-car"></i>
                    <span>Cars</span>
                </p:link>
            </li>
        </ul>
     </ui:define>

    <ui:define name="top-menu">
        <ui:include src="/includes/top-bar.xhtml"/>
    </ui:define>

      <ui:define name="title">
        <h2 class="align-center">
            Welcome to the <span class="text-aqua"> <i><a href=
"https://github.com/adminfaces/admin-starter" target="_blank"
                                                        style="text-transform: none
;text-decoration: none"> AdminFaces Starter</a></i></span> Project!
            <br/>
            <small>Integrating <p:link value="Primefaces" href="http://primefaces.org
"/>, <p:link value="Bootstrap"

href="http://getbootstrap.com/"/> and
                <p:link value="Admin LTE" href=
```

```
"https://almsaeedstudio.com/themes/AdminLTE/index2.html/"/> into your
                <p:link value="JSF" href="https://javaserverfaces.java.net/"/>
application.
            </small>
        </h2>
    </ui:define>

    <ui:define name="description">
        A page description
    </ui:define>

    <ui:define name="body">
        <h2>Page body</h2>
    </ui:define>


    <ui:define name="footer">
          <a target="_blank"
           href="https://github.com/adminfaces/">
            Copyright (C) 2017 - AdminFaces
        </a>

        <div class="pull-right hidden-xs" style="color: gray">
            <i>1.0.0</i>
        </div>
    </ui:define>


</ui:composition>
```

① /admin.xhtml is the location of the template

The above page definition renders as follows:



There are also other regions defined in admin.xhtml template, see here.

## 3.3. Configuration

Template configuration is made through `admin-config.properties` file present in `src/main/resources` folder.

Here are the default values as well as its description:

```
admin.loginPage=login.xhtml ①
admin.indexPage=index.xhtml ②
admin.dateFormat=MM/dd/yyyy HH:mm:ss ③
admin.templatePath=admin.xhtml ④
admin.breadcrumbSize=5 ⑤
admin.renderMessages=true ⑥
admin.renderAjaxStatus=true ⑦
admin.disableFilter=false ⑧
admin.renderBreadCrumb=true ⑨
admin.enableSlideMenu=true ⑩
admin.enableRipple=true ⑪
admin.rippleElements= .ripplelink,button.ui-button,.ui-selectlistbox-item,.ui-
multiselectlistbox-item,.ui-selectonemenu-label,.ui-selectcheckboxmenu,\
.ui-autocomplete-dropdown, .ui-autocomplete-item ... (the list goes on) ⑫
admin.skin=skin-blue ⑬
```

① login page location (relative to webapp). It you only be used if you configure Admin Session.

② index page location. User will be redirected to it when it access app root (contextPath/).

③ Date format used in error page (500.xhtml).

④ facelets template to be used on build in admin-template pages like 500.xhtml, 404.xhtml, viewexpired.xhtml, see Error Pages. By default it uses admin.xhtml but you can define any template (e.g one that extends admin.xhtml).

⑤ Number of breadcrumbs to queue before removing the older ones.

⑥ When false, p:messages defined in admin template will not be rendered.

⑦ When false ajaxStatus, which triggers the loading bar on every ajax request, will not be rendered.

⑧ Disables AdminFilter, responsible for redirecting user after session timeout, sending user to logon page when it is not logged in among other things.

⑨ When false, the breadCrumb component, declared in admin template, will not be rendered.

⑩ If true will make left menu touch enable (can be closed or opened via touch)

⑪ When true it will create a wave/ripple effect on elements specified by `rippleElements`.

⑫ A list of comma separated list of (jquery) selector which elements will be affected by ripple effect.

⑬ Default template skin

You don't need to declare all values in your admin-config.properties, you can specify only the ones you need in order to change.

## 3.4. Admin Session

AdminSession is a simple session scoped bean which controls whether user is logged in or not.

```
public boolean isLoggedIn(){
       return isLoggedIn; //always true by default
   }
```

By default the user **is always logged in** and you need to override it (by using bean specialization or via injection and calling `setIsLoggedIn()` method) to change its value, see Overriding AdminSession.

When isLoggedIn is `false` you got the following mechanisms activated:

1. Access to any page, besides the login, redirects user to login;
2. When session is expired user is redirected to logon and current page (before expiration) is saved so user is redirected back to where it was before session expiration.

It is up to you to decide whether the user is logged in or not.

### 3.4.1. Overriding AdminSession

There are two ways to override AdminSession default value which is specialization and injection.

#### AdminSession Specialization

A simple way to change AdminSession logged in value is by extending it:

```
import javax.enterprise.context.SessionScoped;
import javax.enterprise.inject.Specializes;
import com.github.adminfaces.template.session.AdminSession;
import org.omnifaces.util.Faces;
import java.io.Serializable;

@SessionScoped
@Specializes
public class LogonMB extends AdminSession implements Serializable {

    private String currentUser;
    private String email;
    private String password;
    private boolean remember;


    public void login() throws IOException {
        currentUser = email;
        addDetailMessage("Logged in successfully as <b>" + email + "</b>");
        Faces.getExternalContext().getFlash().setKeepMessages(true);
        Faces.redirect("index.xhtml");
    }

    @Override
    public boolean isLoggedIn() {

        return currentUser != null;
    }

    //getters&setters
}
```

### 3.4.2. AdminSession Injection

Another way is to inject it into your security authentication logic:

```
import com.github.adminfaces.template.session.AdminSession;
import org.omnifaces.util.Messages;
import org.omnifaces.util.Faces;

@SessionScoped
@Named("authorizer")
public class CustomAuthorizer implements Serializable {

    private String currentUser;

    @Inject
    AdminSession adminSession;

    public void login(String username) {
        currentUser = username;
        adminSession.setIsLoggedIn(true);
        Messages.addInfo(null,"Logged in sucessfully as <b>"+username+"</b>");
        Faces.redirect("index.xhtml");
    }

}
```

As isLoggedIn is `true by default` you need to set it to false on application startup so user is redirected to login page. This step is not needed when AdminSession Specialization.

## 3.5. Error Pages

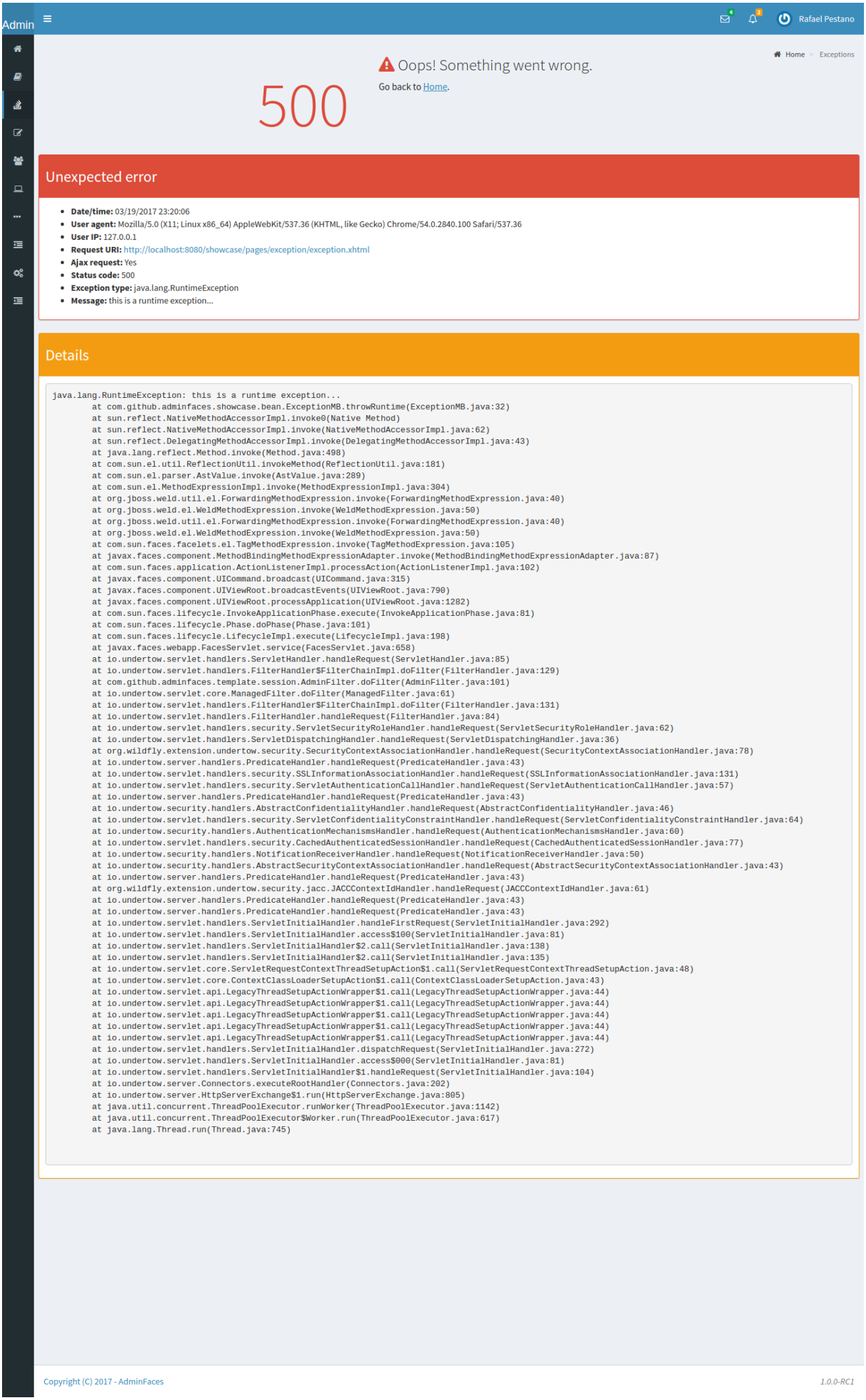
The template comes with custom error pages like `403`, `404`, `500`, `ViewExpired` and `OptimisticLock`.

*500*

User is going to be redirected to **500.xhtml** whenever a *500* response code is returned in a request.

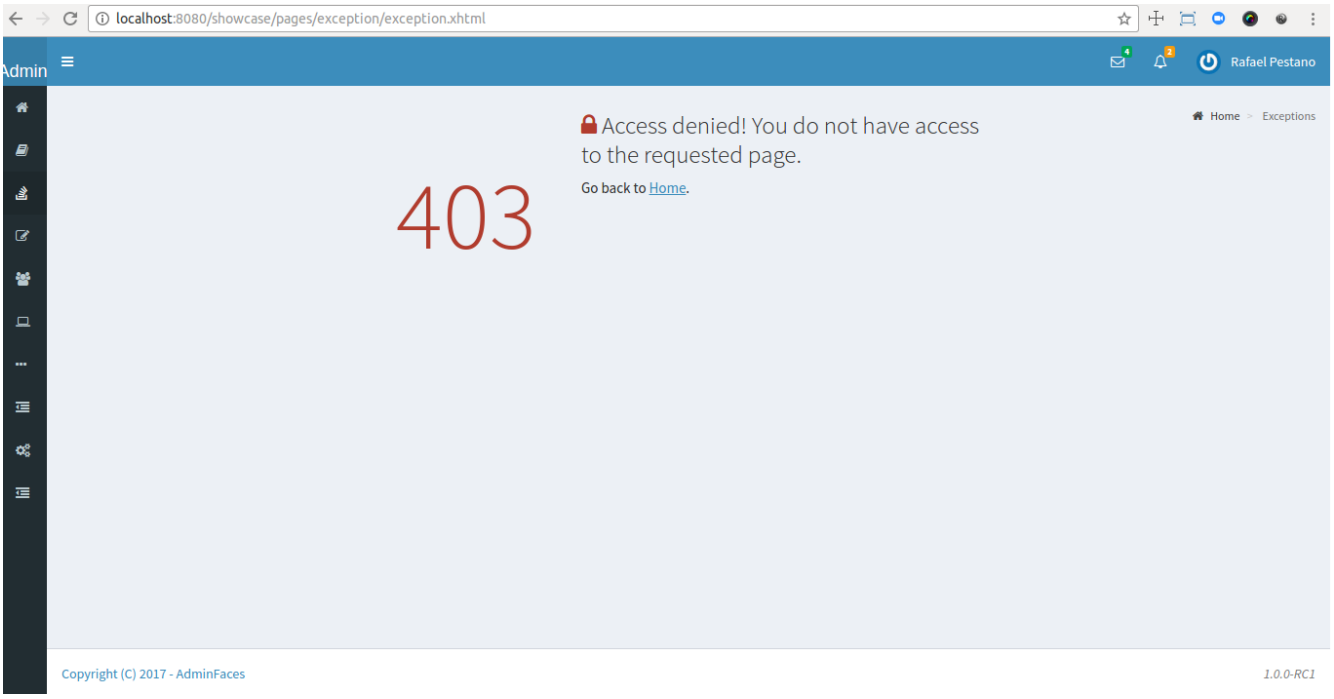
The page will also be triggered when a `Throwable` is raised (and not catch).

Here is how 500 page look like:


Admin
Rafael Pestano
Home > Exceptions
Oops! Something went wrong.
500
Go back to Home.
Unexpected error
Date/time: 03/19/2017 23:20:06
User agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/54.0.2840.100 Safari/537.36
User IP: 127.0.0.1
Request URI: http://localhost:8080/showcase/pages/exception/exception.xhtml
Ajax request: Yes
Status code: 500
Exception type: java.lang.RuntimeException
Message: this is a runtime exception...
Details
java.lang.RuntimeException: this is a runtime exception...
at com.github.adminfaces.showcase.bean.ExceptionMB.throwRuntime(ExceptionMB.java:32)
at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)
at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
at java.lang.reflect.Method.invoke(Method.java:498)
at com.sun.el.util.ReflectionUtil.invokeMethod(ReflectionUtil.java:181)
at com.sun.el.parser.AstValue.invoke(AstValue.java:289)
at com.sun.el.MethodExpressionImpl.invoke(MethodExpressionImpl.java:304)
at org.jboss.weld.util.el.ForwardingMethodExpression.invoke(ForwardingMethodExpression.java:40)
at org.jboss.weld.el.WeldMethodExpression.invoke(WeldMethodExpression.java:50)
at org.jboss.weld.util.el.ForwardingMethodExpression.invoke(ForwardingMethodExpression.java:40)
at org.jboss.weld.el.WeldMethodExpression.invoke(WeldMethodExpression.java:50)
at com.sun.faces.facelets.el.TagMethodExpression.invoke(TagMethodExpression.java:105)
at javax.faces.component.MethodBindingMethodExpressionAdapter.invoke(MethodBindingMethodExpressionAdapter.java:87)
at com.sun.faces.application.ActionListenerImpl.processAction(ActionListenerImpl.java:102)
at javax.faces.component.UICommand.broadcast(UICommand.java:315)
at javax.faces.component.UIViewRoot.broadcastEvents(UIViewRoot.java:790)
at javax.faces.component.UIViewRoot.processApplication(UIViewRoot.java:1282)
at com.sun.faces.lifecycle.InvokeApplicationPhase.execute(InvokeApplicationPhase.java:81)
at com.sun.faces.lifecycle.Phase.doPhase(Phase.java:101)
at com.sun.faces.lifecycle.LifecycleImpl.execute(LifecycleImpl.java:198)
at javax.faces.webapp.FacesServlet.service(FacesServlet.java:658)
at io.undertow.servlet.handlers.ServletHandler.handleRequest(ServletHandler.java:85)
at io.undertow.servlet.handlers.FilterHandler$FilterChainImpl.doFilter(FilterHandler.java:129)
at com.github.adminfaces.template.session.AdminFilter.doFilter(AdminFilter.java:101)
at io.undertow.servlet.core.ManagedFilter.doFilter(ManagedFilter.java:61)
at io.undertow.servlet.handlers.FilterHandler$FilterChainImpl.doFilter(FilterHandler.java:131)
at io.undertow.servlet.handlers.FilterHandler.handleRequest(FilterHandler.java:84)
at io.undertow.servlet.handlers.security.ServletSecurityRoleHandler.handleRequest(ServletSecurityRoleHandler.java:62)
at io.undertow.servlet.handlers.ServletDispatchingHandler.handleRequest(ServletDispatchingHandler.java:36)
at org.wildfly.extension.undertow.security.SecurityContextAssociationHandler.handleRequest(SecurityContextAssociationHandler.java:78)
at io.undertow.server.handlers.PredicateHandler.handleRequest(PredicateHandler.java:43)
at io.undertow.servlet.handlers.security.SSLInformationAssociationHandler.handleRequest(SSLInformationAssociationHandler.java:131)
at io.undertow.servlet.handlers.security.ServletAuthenticationCallHandler.handleRequest(ServletAuthenticationCallHandler.java:57)
at io.undertow.server.handlers.PredicateHandler.handleRequest(PredicateHandler.java:43)
at io.undertow.security.handlers.AbstractConfidentialityHandler.handleRequest(AbstractConfidentialityHandler.java:46)
at io.undertow.servlet.handlers.security.ServletConfidentialityConstraintHandler.handleRequest(ServletConfidentialityConstraintHandler.java:64)
at io.undertow.security.handlers.AuthenticationMechanismsHandler.handleRequest(AuthenticationMechanismsHandler.java:60)
at io.undertow.servlet.handlers.security.CachedAuthenticatedSessionHandler.handleRequest(CachedAuthenticatedSessionHandler.java:77)
at io.undertow.security.handlers.NotificationReceiverHandler.handleRequest(NotificationReceiverHandler.java:50)
at io.undertow.security.handlers.AbstractSecurityContextAssociationHandler.handleRequest(AbstractSecurityContextAssociationHandler.java:43)
at io.undertow.server.handlers.PredicateHandler.handleRequest(PredicateHandler.java:43)
at org.wildfly.extension.undertow.security.jacc.JACCContextIdHandler.handleRequest(JACCContextIdHandler.java:61)
at io.undertow.server.handlers.PredicateHandler.handleRequest(PredicateHandler.java:43)
at io.undertow.server.handlers.PredicateHandler.handleRequest(PredicateHandler.java:43)
at io.undertow.servlet.handlers.ServletInitialHandler.handleFirstRequest(ServletInitialHandler.java:292)
at io.undertow.servlet.handlers.ServletInitialHandler.access$100(ServletInitialHandler.java:81)
at io.undertow.servlet.handlers.ServletInitialHandler$2.call(ServletInitialHandler.java:138)
at io.undertow.servlet.handlers.ServletInitialHandler$2.call(ServletInitialHandler.java:135)
at io.undertow.servlet.core.ServletRequestContextThreadSetupAction$1.call(ServletRequestContextThreadSetupAction.java:48)
at io.undertow.servlet.core.ContextClassLoaderSetupAction$1.call(ContextClassLoaderSetupAction.java:43)
at io.undertow.servlet.api.LegacyThreadSetupActionWrapper$1.call(LegacyThreadSetupActionWrapper.java:44)
at io.undertow.servlet.api.LegacyThreadSetupActionWrapper$1.call(LegacyThreadSetupActionWrapper.java:44)
at io.undertow.servlet.api.LegacyThreadSetupActionWrapper$1.call(LegacyThreadSetupActionWrapper.java:44)
at io.undertow.servlet.api.LegacyThreadSetupActionWrapper$1.call(LegacyThreadSetupActionWrapper.java:44)
at io.undertow.servlet.api.LegacyThreadSetupActionWrapper$1.call(LegacyThreadSetupActionWrapper.java:44)
at io.undertow.servlet.handlers.ServletInitialHandler.dispatchRequest(ServletInitialHandler.java:272)
at io.undertow.servlet.handlers.ServletInitialHandler.access$000(ServletInitialHandler.java:81)
at io.undertow.servlet.handlers.ServletInitialHandler$1.handleRequest(ServletInitialHandler.java:104)
at io.undertow.server.Connectors.executeRootHandler(Connectors.java:202)
at io.undertow.server.HttpServerExchange$1.run(HttpServerExchange.java:805)
at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1142)
at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:617)
at java.lang.Thread.run(Thread.java:745)
Copyright (C) 2017 - AdminFaces
1.0.0-RC1

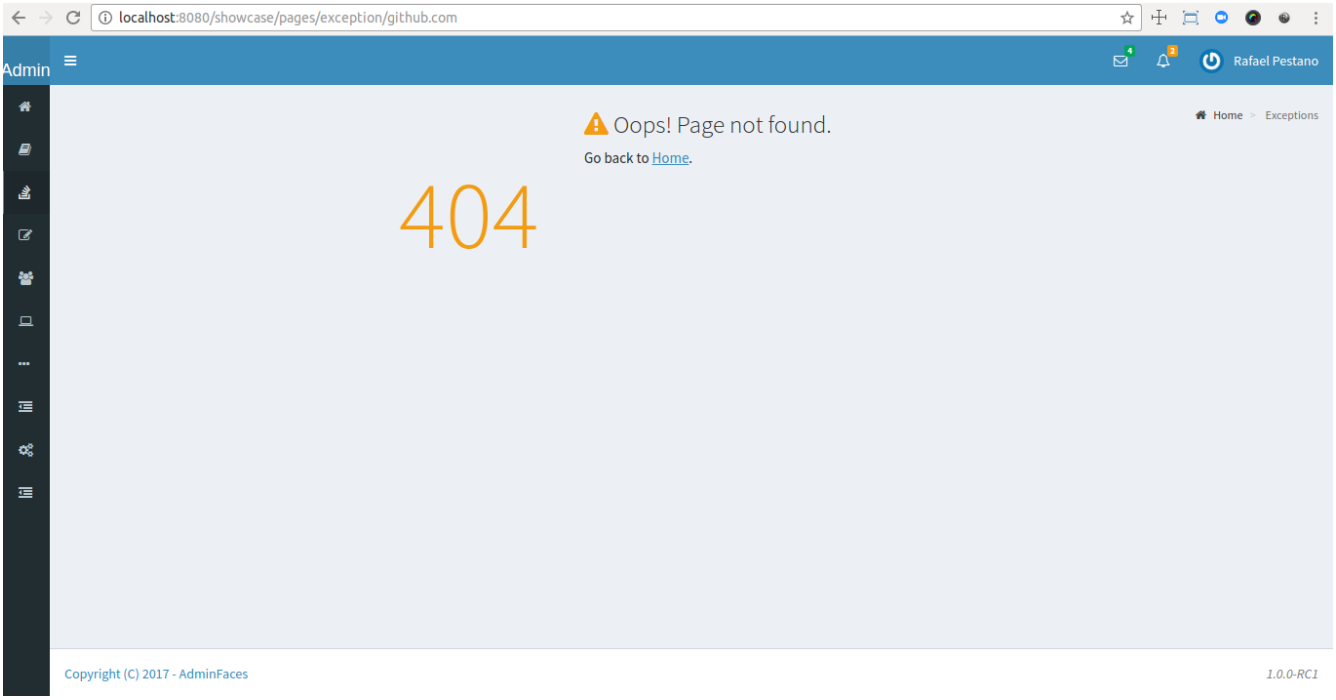
*403*

User is redirected to **403.xhtml** whenever a *403* response code is returned in a request. The page will also be triggered when a `com.github.adminfaces.template.exception.AccessDeniedException` is raised.



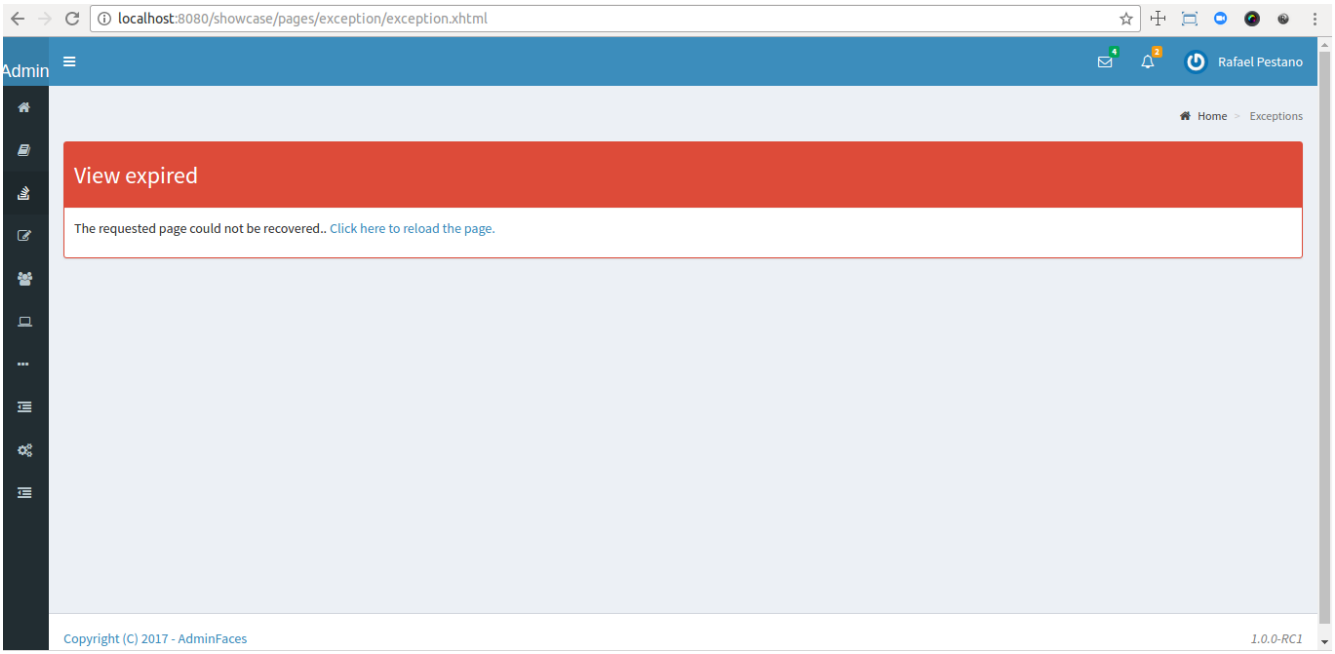

*404*

User will be redirected to **404.xhtml** whenever a 404 response code is returned from a request.



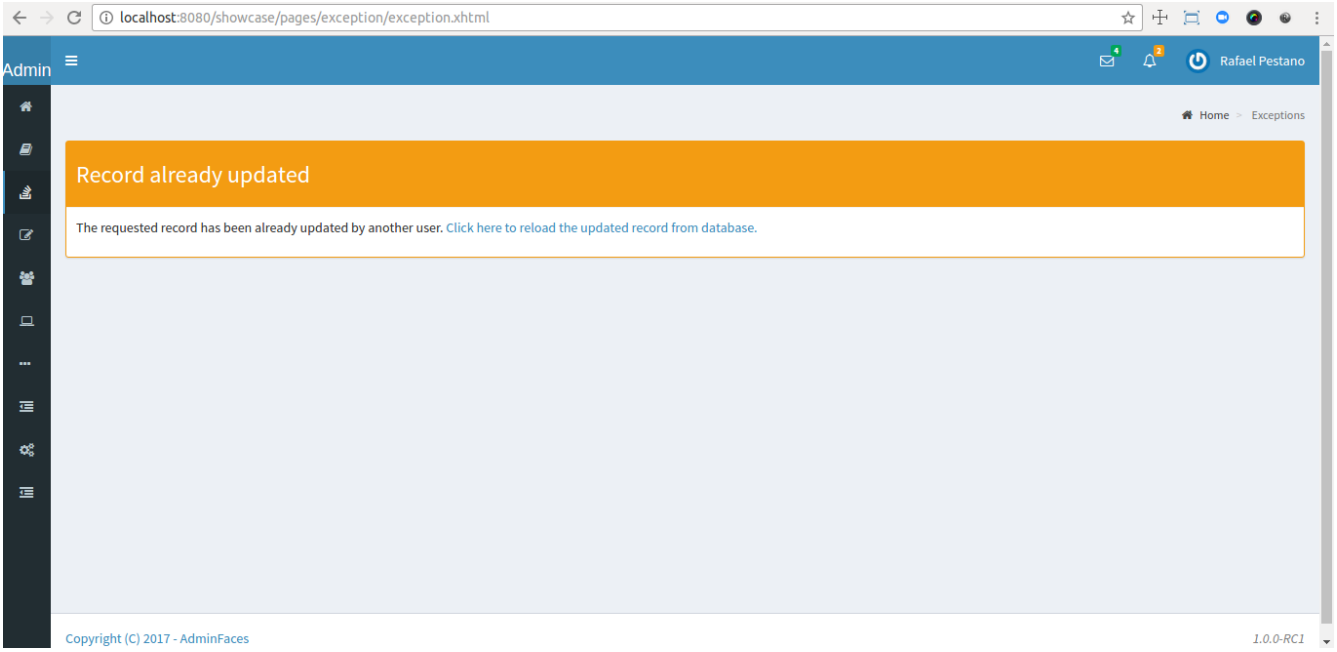

*ViewExpired*

When a JSF `javax.faces.application.ViewExpiredException` is raised user will be redirected to **expired.xhtml**.




*OptimisticLock*

When a JPA `javax.persistence.OptimisticLockException` is thrown user will be redirected to **optimistic.xhtml**.




### 3.5.1. Providing custom error pages

You can provide your own custom pages (and other status codes) by configuring them in web.xml, example:

```
<error-page>
    <error-code>404</error-code>
    <location>/404.xhtml</location>
</error-page>
<error-page>
    <error-code>500</error-code>
    <location>/500.xhtml</location>
</error-page>
<error-page>
    <exception-type>java.lang.Throwable</exception-type>
    <location>/500.xhtml</location>
</error-page>
```

### 3.5.2. Overriding error pages

You can also override error pages by placing the pages (with same name) described in Error Pages section on the root of your application (`webapp/`).

### 3.5.3. Labels and Messages in error pages

Labels and messages in error pages are provided via JSF resource bundle mechanism.

Below is `admin.properties` resource bundle which provides messages for error pages:

*src/main/resources/admin.properties*

```
label.go-back=Go back to

#403
label.403.header=403
label.403.message=Access denied! You do not have access to the requested page.

#404
label.404.header=404
label.404.message=Oops! Page not found

#500
label.500.header=500
label.500.message=Oops! Something went wrong
label.500.title=Unexpected error
label.500.detail=Details

#expired
label.expired.title=View expired
label.expired.message= The requested page could not be recovered.
label.expired.click-here= Click here to reload the page.

#optimistic
label.optimistic.title=Record already updated
label.optimistic.message= The requested record has been already updated by another
user.
label.optimistic.click-here= Click here to reload the updated record from database.
```

To override labels just provide a file named *admin.properties* in your application `resources` folder.

## 3.6. Snapshots

Template `Snapshots` are published to maven central on each commit, to use it just declare the repository below on your `pom.xml`:

```
<repositories>
    <repository>
        <snapshots/>
        <id>snapshots</id>
        <name>libs-snapshot</name>
        <url>https://oss.sonatype.org/content/repositories/snapshots</url>
    </repository>
</repositories>
```

# Chapter 4. Admin Designer

The aim of Admin Designer is to make it easier to customize Admin theme and Admin Template.

## 4.1. What is it?

This is the same Admin Showcase application with `admin template` and `admin theme` bundled inside instead of being project dependencies.

It uses Wildfly Swarm to run the `exploded` application so one can change the theme or template and see the modifications without needing to restart the application.

## 4.2. Objectives

The initial idea was to speed AdminFaces development but it turns out that it can easy `contributions` from non Java developers (like designers and frontend developers) as the project is about front end components and layout.

Also another great feature of Admin Designer is the possibility to **download the customized project** as a maven project.

The downloaded project is the Admin Starter with modified admin theme and template embedded in the project.

This is the most flexible approach but at the same time `you lose the updates on Admin Theme and template projects` because you don't depend on them anymore.

## 4.3. How it works

In application root directory:

1. First start the application by running the command:

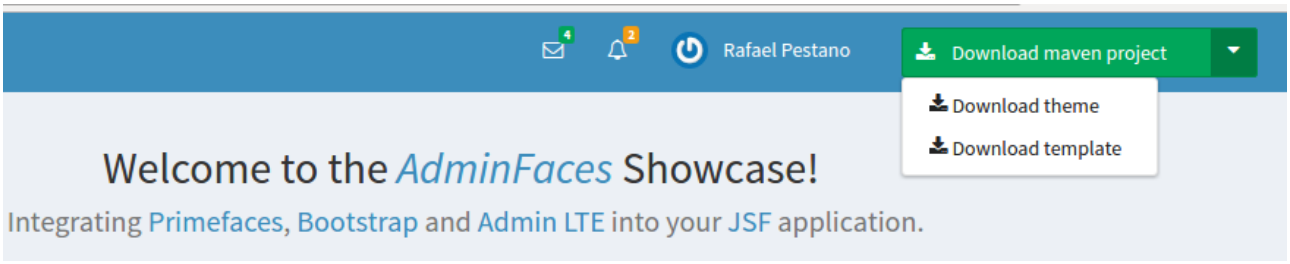
```
./mvnw wildfly-swarm:run (or mvnw.cmd wildfly-swarm:run)
```

2. Second edit any less file in directory `src/main/resources/less`.
3. Now to compile the application using:

```
./mvnw compile (or mvnw.cmd compile)
```

If you don't want to compile every time you change a less file, use the flag `-Dlesscss.watch=true`. Or use a tool like brackets with less extension installed.

4. Finally when you're done you can download the customized `theme` and `template` as a **sample maven project** or as separated jar files;



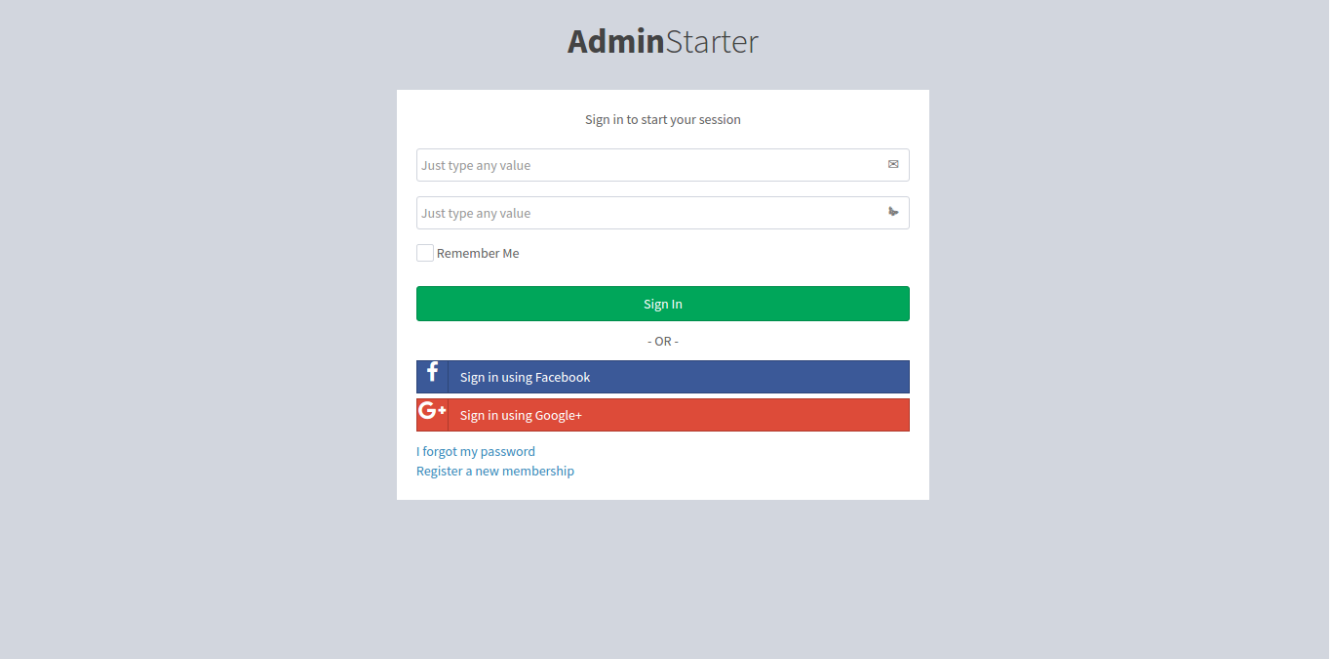

The changes made to less files should be visible in running application `http://localhost:8080/showcase`

There is no need to stop and run the application again.

You can see this workflow in the following video: https://youtu.be/X1UEpN942s0

# Chapter 5. Admin Starter

Admin Starter is a simple project to get you started with AdminFaces.




## 5.1. Running

It should run in any JavaEE 6 or greater application server.

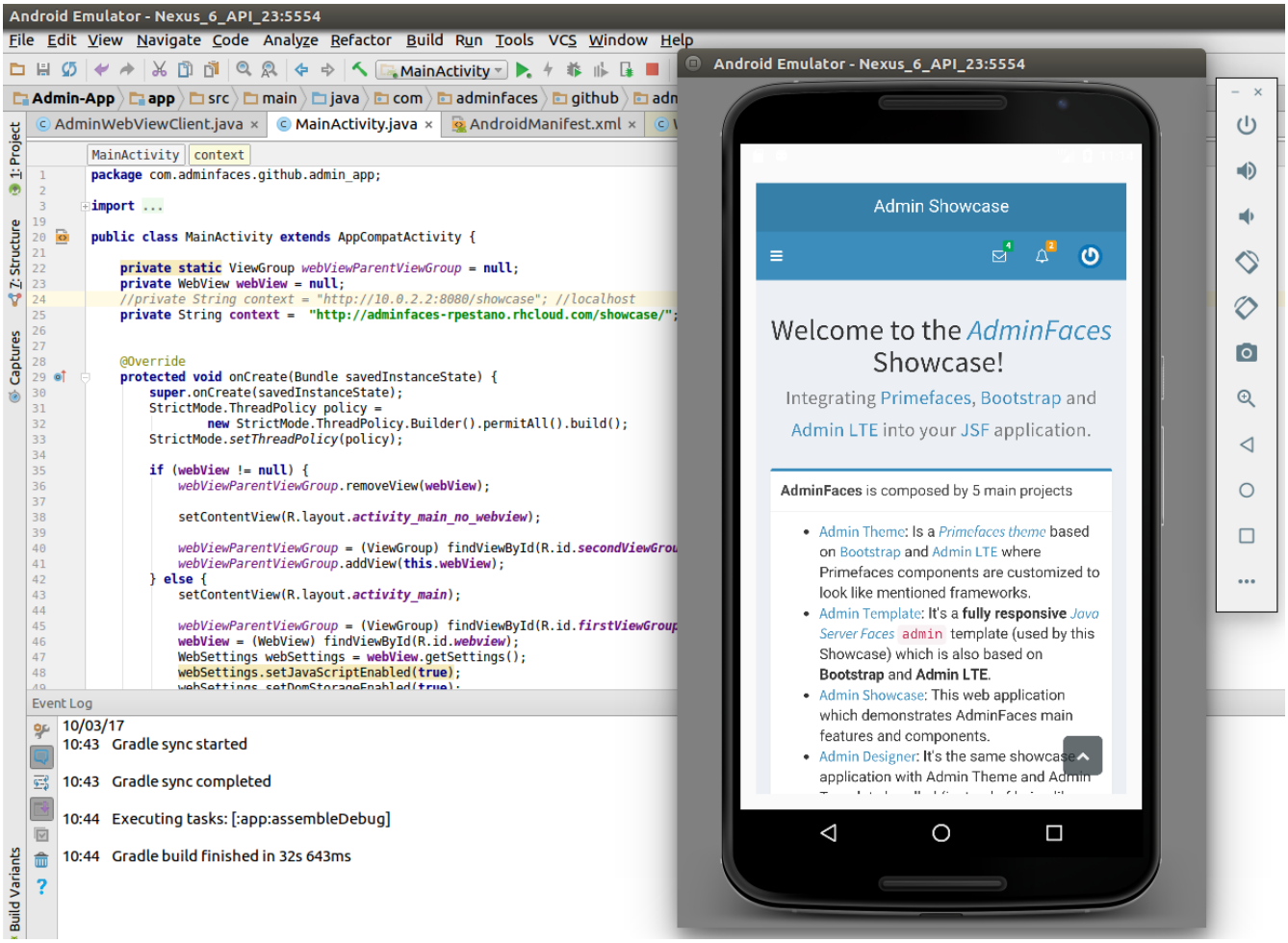
You can also run via wildfly-swarm with command `mvn wildfly-swarm:run -Pswarm`.

## 5.2. Demo

A live demo is available on Openshift here: http://starter-adminfaces.rhcloud.com/admin-starter/

# Chapter 6. Admin Mobile

Admin Mobile is a simple Android Studio project which uses Webview to create an hybrid web app based on Admin Showcase.




The app is a proof of concept to check AdminFaces user experience in mobile apps. The following behaviours will be enabled only on this kind of devices:

- Loading bar based on google material design;
- Go to top link at the bottom right corner of the page;
- Larger icons on panel, dialog and messages;
- Some components like growl, tabview and datatable are optimized for mobile devices;
- Ripple/waves effect based on materialize;
- Slideout menu
  [dd37121e 2296 11e7 855c 8f20b59dcf5f]