New design, UX and extensibility digest for login page et. al.

This blog post gives an introduction to the new design for the login and signup forms and Jenkins is (re)starting pages introduced in Jenkins 2.128. The first part of the blog post is an introduction to the new design and UX for Jenkins users. The later part is talking about extensibility in a more technical manner, aimed at plugin developers. Tweet

Overview

The recent changes to some core pages provide new design and UX and further dropping all external dependencies to prevent any possible malicious javascript introduced by third party libraries. To be clear, this never was an issue with previous releases of Jenkins, but having read this article, this author believes that the article has good points and leading by example may raise awareness of data protection.

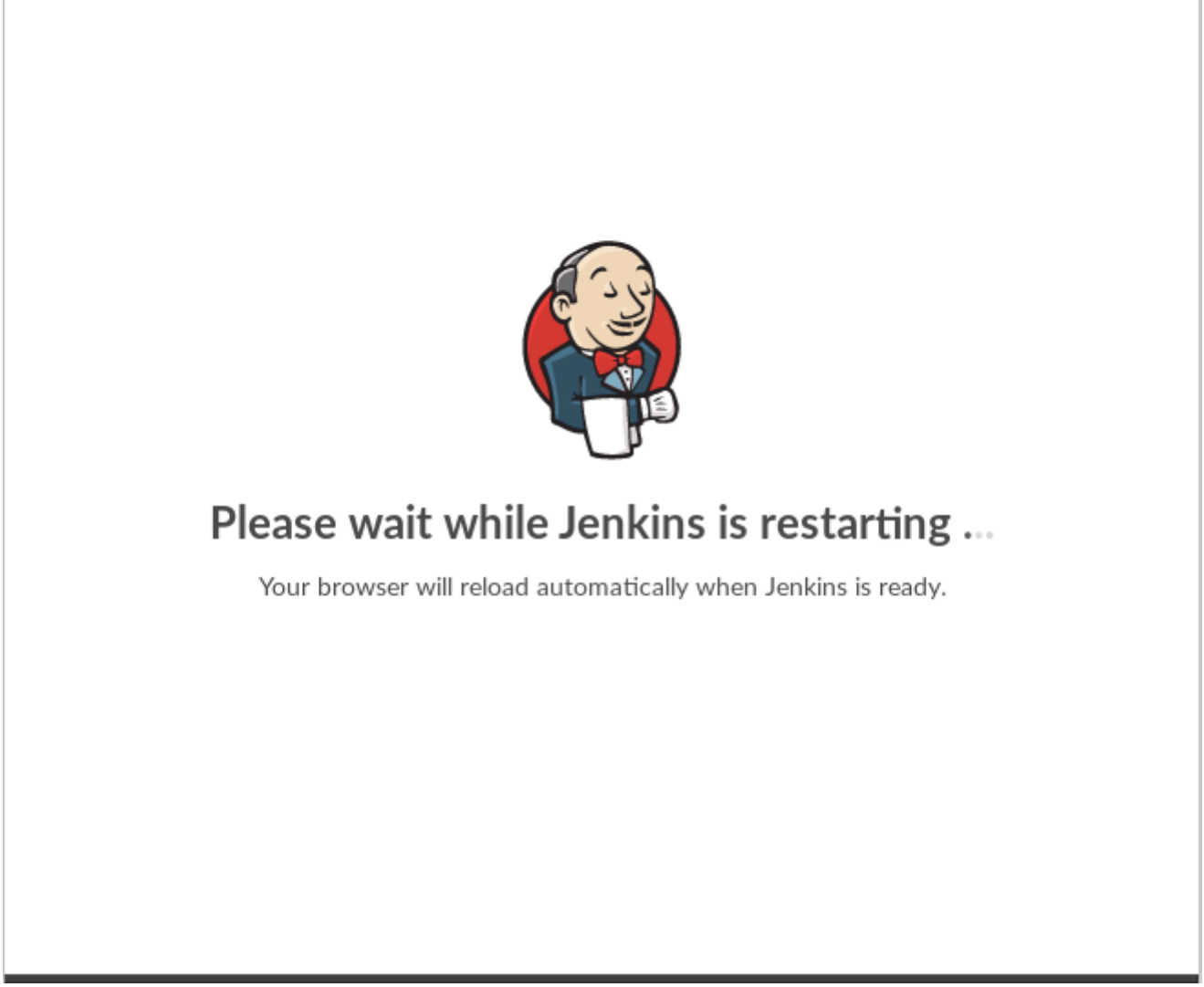
This meant to drop the usage of the jelly layout lib (aka `xmlns:l="/lib/layout"`) and as well the page decorators it supported. However there is a new `SimplePageDecorator` extension point (discussed below) which can be used to modify the look and feel for the login and sign up page.

The following pages have given a new design:

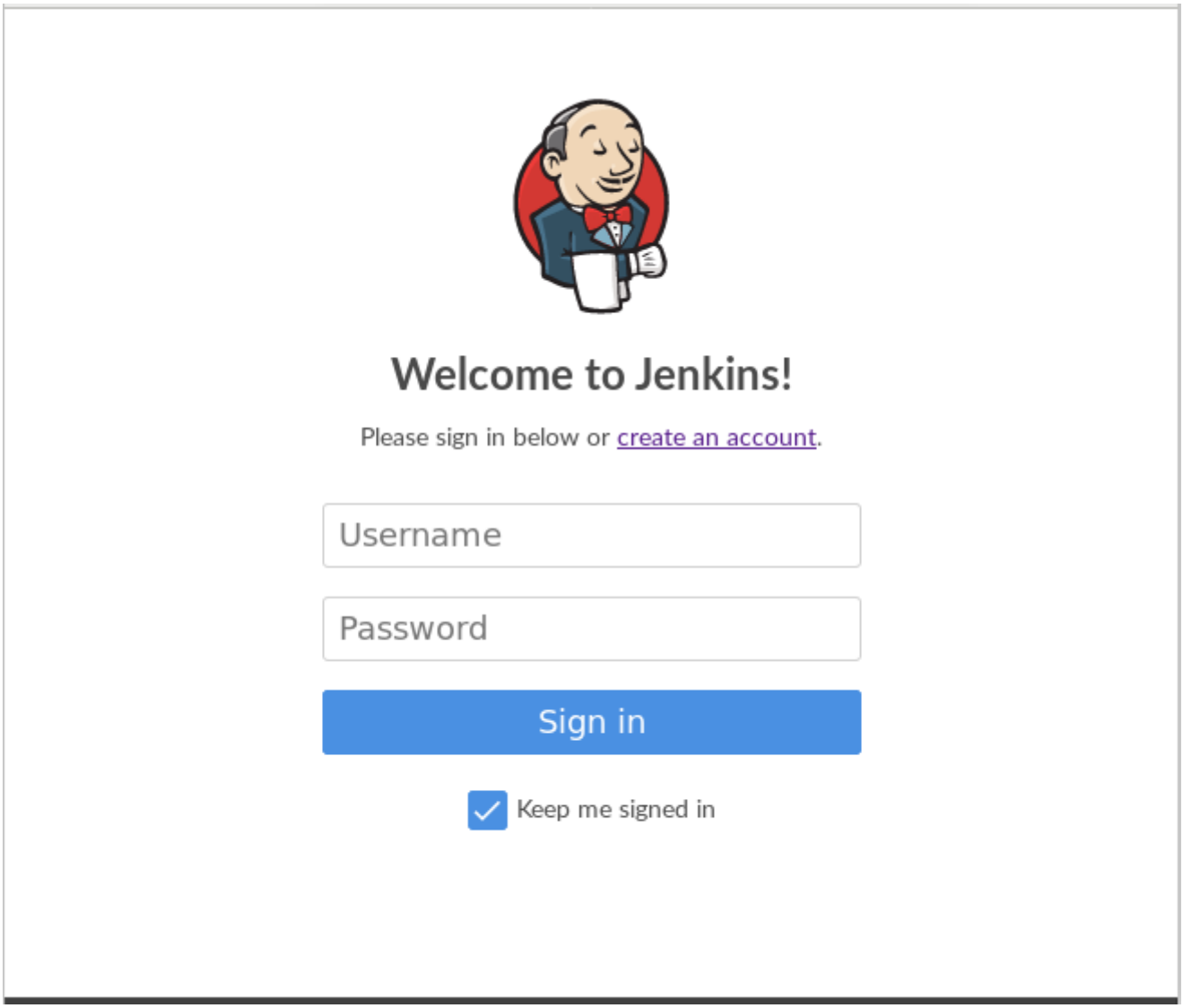
- Jenkins is (re)starting pages

Please wait while Jenkins is getting ready to work ...

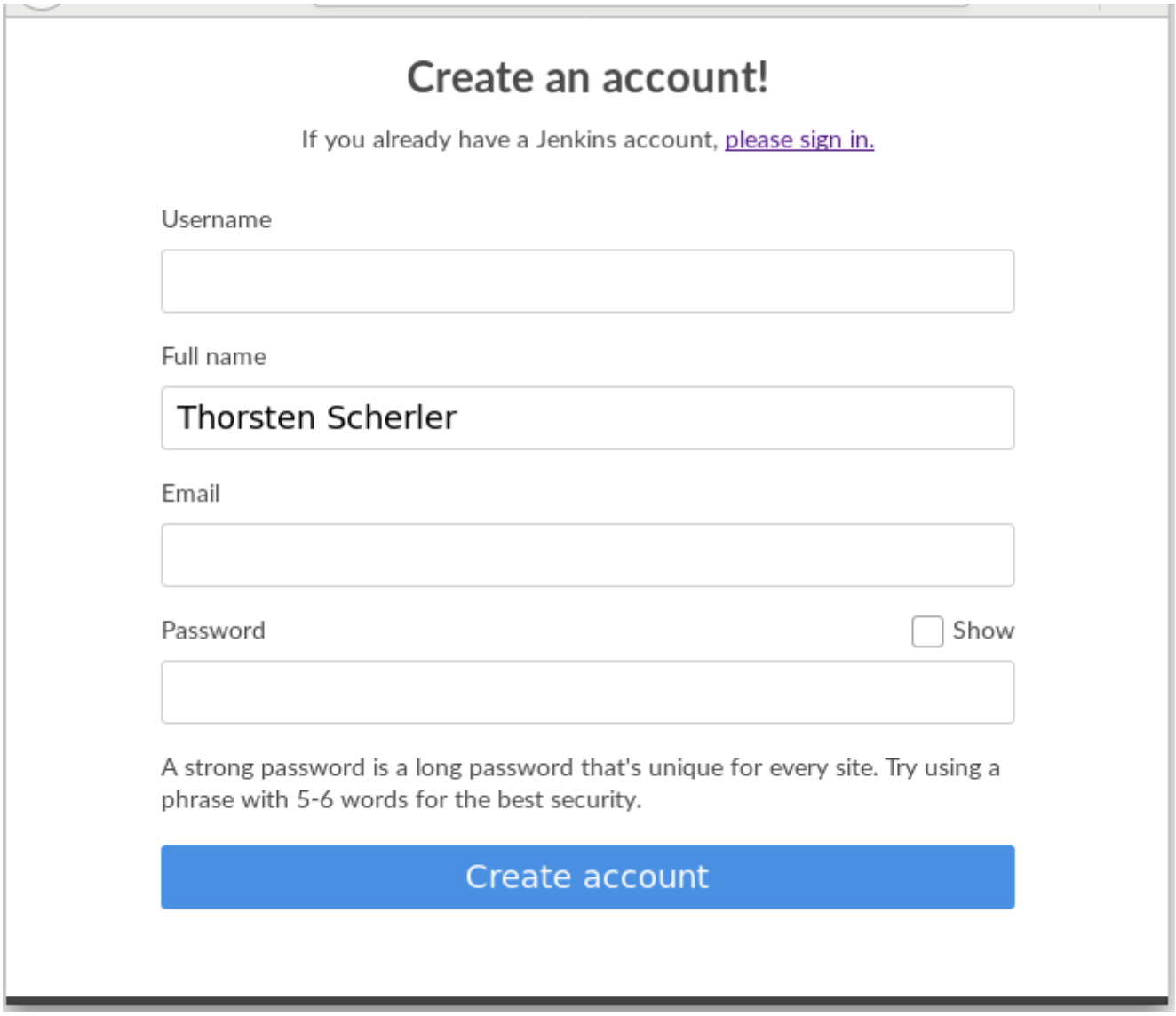
Your browser will reload automatically when Jenkins is ready.


Please wait while Jenkins is restarting ...
Your browser will reload automatically when Jenkins is ready.

- Login


Welcome to Jenkins!
Please sign in below or create an account.
Username
Password
Sign in
Keep me signed in

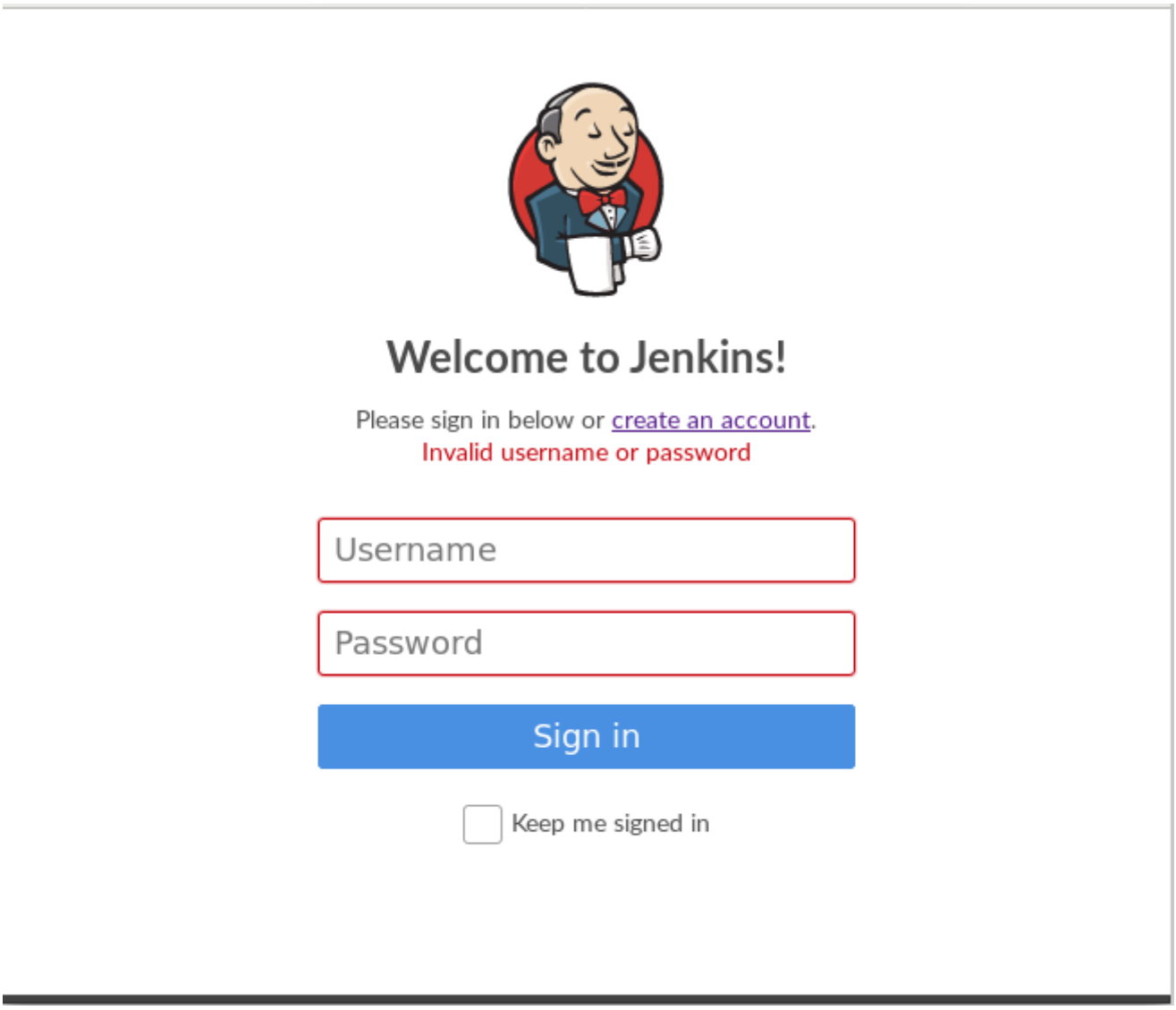
- Sign up



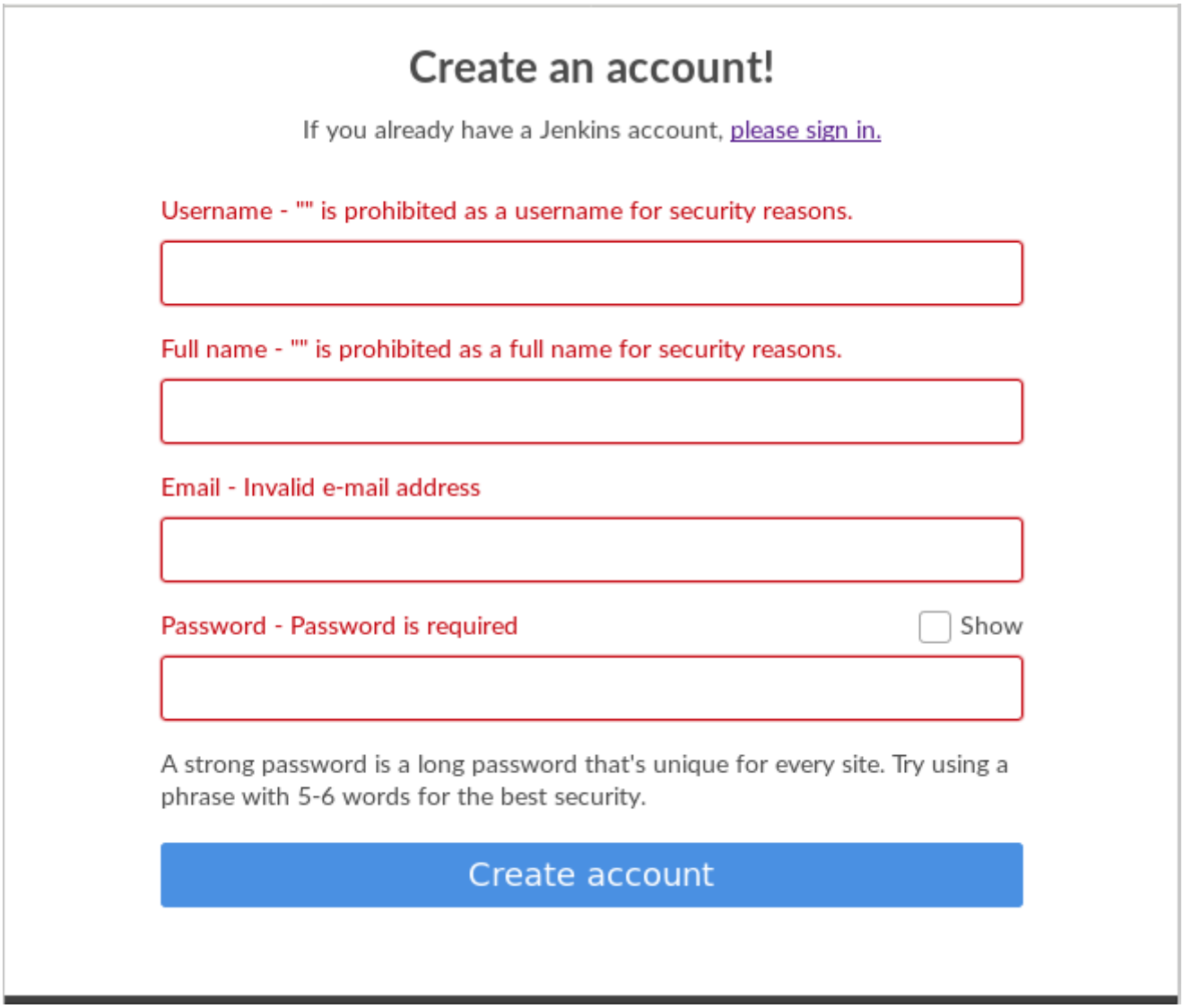
UX enhancement

Form validation has changed to give inline feedback about data validation errors in the same form.

- Login

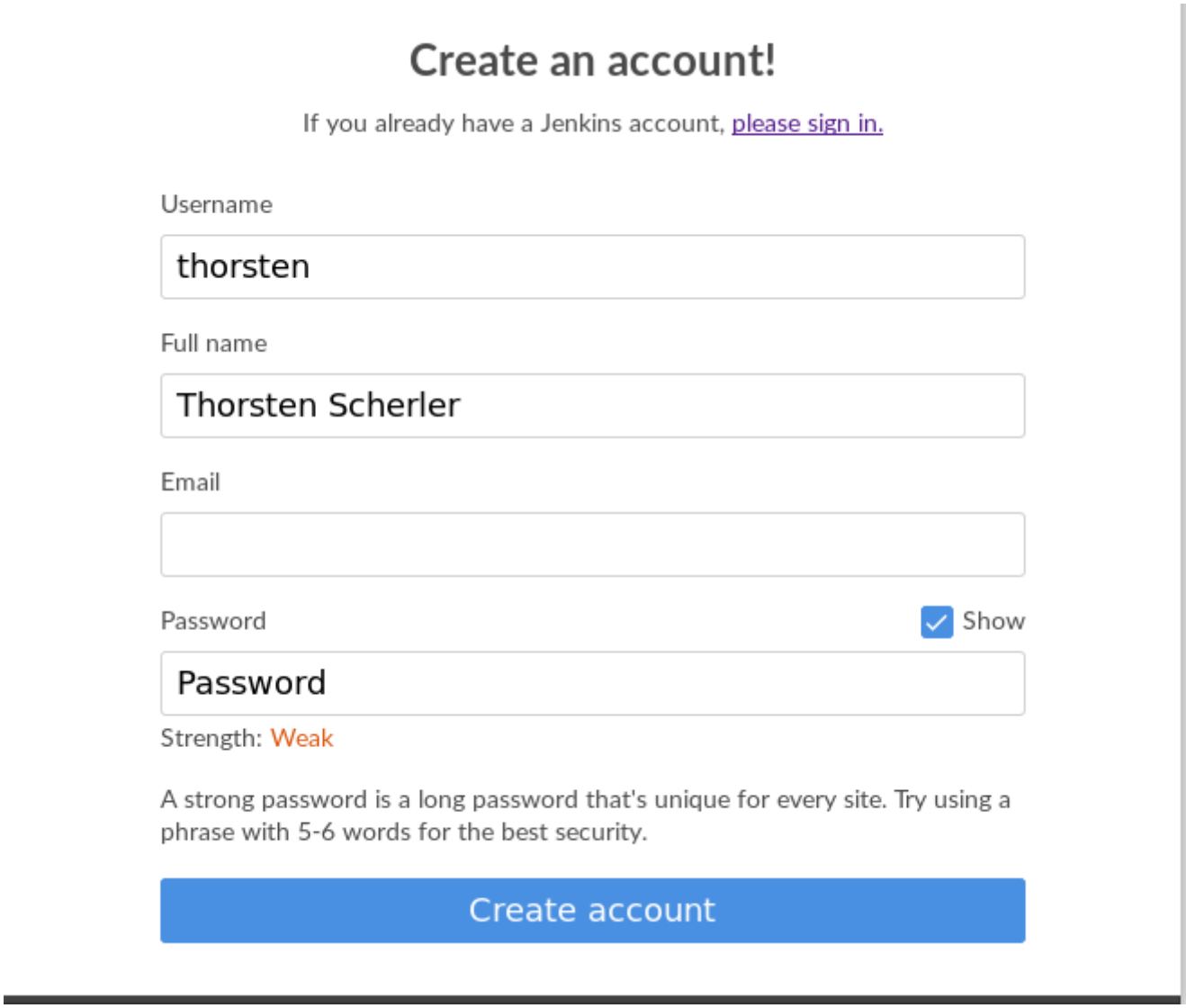

Welcome to Jenkins!
Please sign in below or create an account.
Invalid username or password
Username
Password
Sign in
Keep me signed in

- Sign up



The above image shows that the validation is now done on all input fields instead of before breaking on the first error found, which should lead to fewer retry cycles.

Instead of forcing the user to repeat the password, the new UX introduces the possibility to display the password in clear text. Further a basic password strength meter indicates password strength to the user while she enters the password.



Customizing the UI

> The re-/starting screens do not support the concept of decorators very well, hence the decision to not support them for these pages.

The SimplePageDecorator is the key component for customization and uses three different files to allow overriding the look and feel of the login and signup pages.

- `simple-head.jelly`
- `simple-header.jelly`
- `simple-footer.jelly`

All of the above `SimplePageDecorator` Jelly files are supported in the **login** page. The following snippet is a minimal excerpt of the login page, showing how it makes use of `SimplePageDecorator`.

```
<?jelly escape-by-default='true'?>
<j:jelly xmlns:j="jelly:core" xmlns:st="jelly:stapler" >
  <j:new var="h" className="hudson.Functions"/>
  <html>
    <head>
      <!-- css styling, will fallback to default implementation -->
```

```
      <st:include it="${h.simpleDecorator}" page="simple-head.jelly" optional="true"/>
    </head>
    <body>
      <div class="simple-page" role="main">
        <st:include it="${h.simpleDecorator}" page="simple-header.jelly" optional="true"/>
      </div>
      <div class="footer">
        <st:include it="${h.simpleDecorator}" page="simple-footer.jelly" optional="true"/>
      </div>
    </body>
  </html>
</j:jelly>
```

The **sign-up** page only supports the simple-head.jelly:

```
<?jelly escape-by-default='true'?>
<j:jelly xmlns:j="jelly:core" xmlns:st="jelly:stapler" >
  <j:new var="h" className="hudson.Functions"/>
  <html>
    <head>
      <!-- css styling, will fallback to default implementation -->
      <st:include it="${h.simpleDecorator}" page="simple-head.jelly" optional="true"/>
    </head>
  </html>
</j:jelly>
```

SimplePageDecorator - custom implementations

> Have a look at Login Theme Plugin, which allows you to configure your own custom content to be injected into the new login/sign-up page.

To allow easy customisation the decorator only implements one instance by the principal "first-come-first-serve". If jenkins finds an extension of the `SimplePageDecorator` it will use the Jelly files provided by that plugin. Otherwise Jenkins will fall back to the default implementation.

```
@Extension
public class MySimplePageDecorator extends SimplePageDecorator {
   public String getProductName() {
     return "MyJenkins";
   }
}
```

> The above will take override over the default because the default implementation has a very low ordinal (`@Extension(ordinal=-9999)`) If you have competing plugins implementing `SimplePageDecorator`, the implementation with the highest ordinal will be used.

As a simple example, to customize the logo we display in the login page, create a `simple-head.jelly` with the following content:

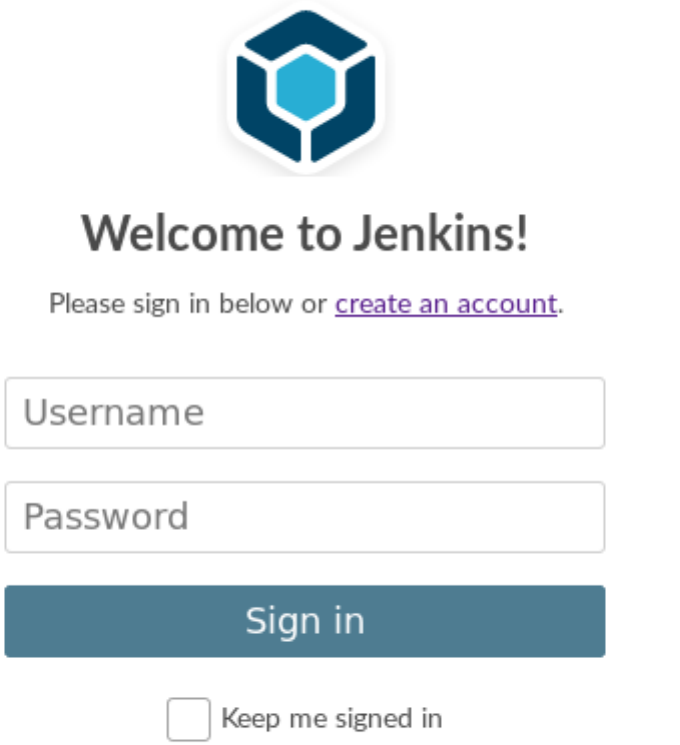
```
<?jelly escape-by-default='true'?>
<j:jelly xmlns:j="jelly:core">
```

```
  <link rel="stylesheet" href="${resURL}/css/simple-page.css" type="text/css" />
  <link rel="stylesheet" href="${resURL}/css/simple-page.theme.css" type="text/css" />
  <style>
    .simple-page .logo {
        background-image: url('${resURL}/plugin/YOUR_PLUGIN/icons/my.svg');
        background-repeat: no-repeat;
        background-position: 50% 0;
        height: 130px;
    }
  </style>
  <link rel="stylesheet" href="${resURL}/css/simple-page-forms.css" type="text/css" />
</j:jelly>
```

To customize the login page further, create a `simple-header.jelly` like this:

```
<?jelly escape-by-default='true'?>
<j:jelly xmlns:j="jelly:core">
  <div id="loginIntro">
    <div class="logo"> </div>
    <h1 id="productName">Welcome to ${it.productName}!</h1>
  </div>
</j:jelly>
```

For example, I used this technique to create a prototype of a login page for a CloudBees product I am working on:



Conclusion

We hope you like the recent changes to some core pages and as well the new design and UX. We further hope you feel enabled to customize the look and feel to adopt your needs with the `SimplePageDecorator`.

About the Author

Thorsten Scherler

Thorsten Scherler has a broad background in open source consulting, training and solutions since 2002. He's an engineer at CloudBees, working full time on the Blue Ocean front end. He left Germany in 2003 after finishing his university degrees and has been living in sunny Seville ever since.

- GitHub
- Twitter