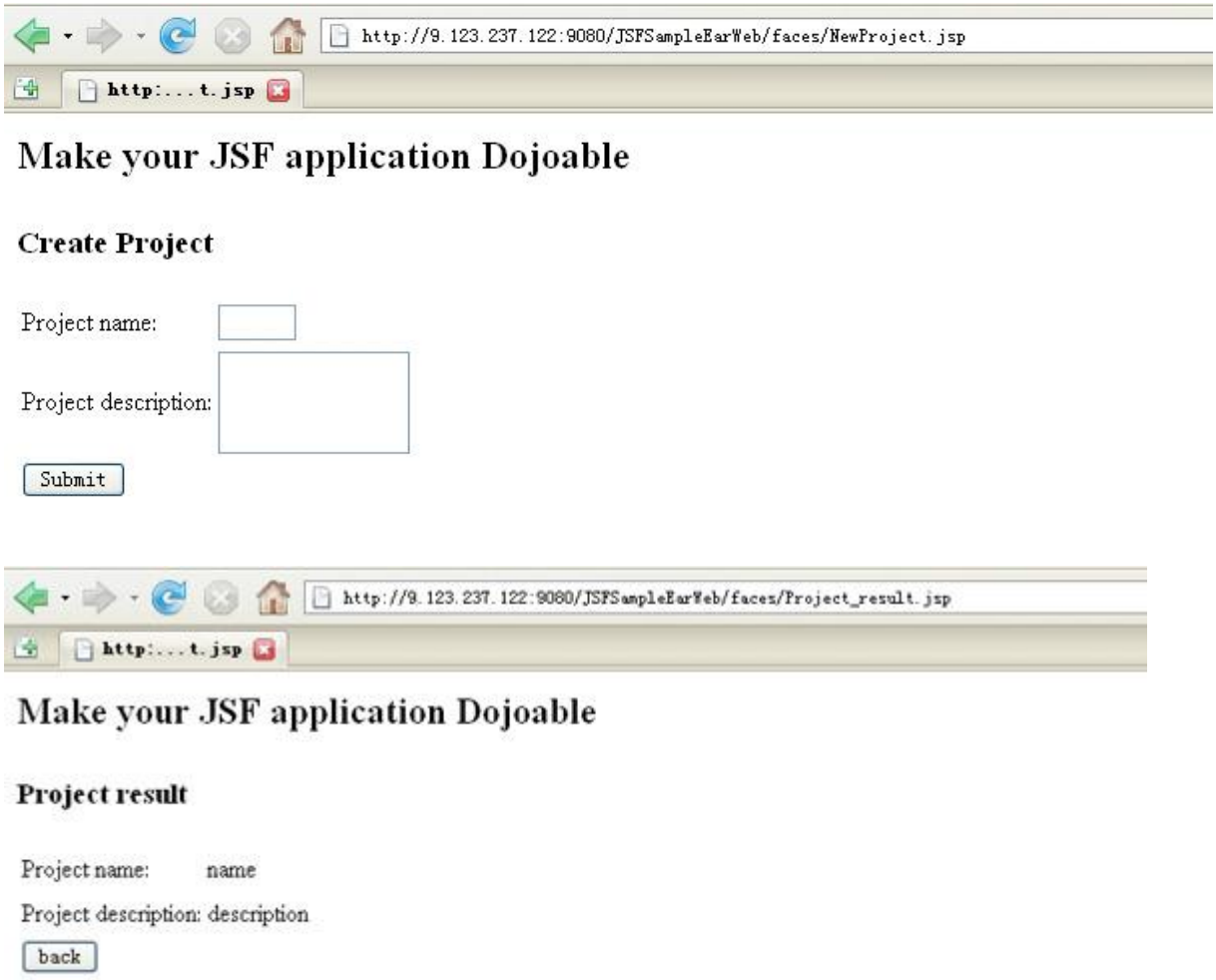
Exercice12 : Combinez JSF avec des widgets Dojo de créer une meilleure expérience utilisateur

L'application JSF est composé de deux pages: créer un projet Exercice11. Les figures 1 et 2 montrent ces deux pages. Pour faire simple et facile à comprendre, les pages présentées sont utilisées pour montrer comment combiner Dojo avec une application JSF.



Le scénario est vraiment très simple. L'utilisateur peut entrer des informations pour créer un projet, et quand ils cliquent sur le bouton d'envoi, les informations du projet est affichée. Donc, votre tâche est de convertir la zone de saisie, zone de texte, et d'autres composants JSF dans les widgets Dojo.

Etape1 Différer la liaison d'un composant JSF et un widget Dojo du côté client

Cette approche est la façon la plus simple d'appliquer le style Dojo au composant JSF. Nous utilisons seulement le JavaScript pour lier le composant JSF et le widget Dojo ensemble sur le côté client. En utilisant JavaScript, les données peuvent être transférées et seront conformes entre le composant JSF et le widget Dojo.

Ecrire la page d'accueil : Il ya deux champs de saisie: nom du projet et la description du projet. Ces deux domaines sont tous liés avec le back-end bean projectFormBean.

Commençons à convertir cette page avec un style Dojo.

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>

<%@ taglib uri="http://java.sun.com/jsf/core" prefix="f" %>
<%@ taglib uri="http://java.sun.com/jsf/html" prefix="h"%>
<%@ taglib uri="http://www.ibm.com/jsf/html_extended" prefix="hx"%>
<%@ taglib uri="http://java.sun.com/jstl/core" prefix="c" %>

<body>
<f:view>
     <h2>Make your JSF application Dojoable</h2>
     <h3>Create Project</h3>
     <h:form id="project">
     <table>
          <tbody>
               <tr>
                    <td>Project Name:</td>
                    <td><h:inputText id="projectName"
value="#{projectFormBean.projectName}" size="5"/></td>
               </tr>
               <tr>
                    <td>Project description:</td>
                    <td><h:inputTextarea id="projectDescription"
value="#{projectFormBean.projectDescription}" rows="2" cols="15"/></td>
               </tr>
               <tr>
                    <td colspan=2>
                         <h:commandButton id="button_submit" action="success"
value="Submit" type="submit"></h:commandButton>
                    </td>
               </tr>
          </tbody>
     </table>
     </h:form>
</f:view>
</body>
```

Masquer le composant JSF et ajouter le widget Dojo

Vous avez besoin de cacher les composants originaux JSF et les changer à invisible, puis ajoutez les widgets Dojo liés à cette page. Votre première étape consiste à importer la bibliothèque Dojo et déclarer que les widgets sont nécessaires.

Dans cet exemple, nous utilisons le thème tundra comme le style par défaut, si vous devez d'abord importer le tundra.css et la dojo.css. Ensuite, vous devez importer le fichier dojo.js, qui est le fichier de script js principal Dojo. Enfin, vous devez déclarer les widgets que vous souhaitez utiliser. Dans ce cas,

nous allons utiliser validationTextBox et Textarea. Dojo utilise le mécanisme à la demande. Il ne chargera les fichiers requis selon la déclaration. Par conséquent, on peut améliorer les performances de l'application et de réduire le transfert de réseau.

```
<style type="text/css">
    @import
url("${pageContext.request.contextPath}/script/dojo_lib/dijit/themes/tundra/tundra.css"
);
     @import
url("${pageContext.request.contextPath}/script/dojo_lib/dojo/resources/dojo.css");
</style>
<script type="text/javascript"
src="${pageContext.request.contextPath}/script/dojo_lib/dojo/dojo.js"
    djConfig="parseOnLoad: true, isDebug:false"></script>

<script type="text/javascript">
   dojo.require("dojo.parser");
   dojo.require("dijit.form.ValidationTextBox");
   dojo.require("dijit.form.Textarea");
</script>
```

Importer la bibliothèque Dojo et déclarer les widgets

La prochaine étape est de changer les composants JSF pour être invisible et ajouter les widgets Dojo pour l'affichage.

Nous changeons les composants originaux de JSF pour inputHidden, mais engageons toujours avec les données de projectFormBean back-end. Après cette composante cachée, nous ajoutons les widgets Dojo correspondant. Pour le champ "Nom de projet", nous utilisons un validationTextBox pour valider la boîte d'entrée sur le côté client, et pour le champ "Description du projet", nous le dojo textarea d'utilisation, qui peut être auto-développé sans une barre de défilement.

```
<tr>
<td>Project name:</td>
<td><h:inputHidden value="#{projectFormBean.projectName}"/><f:verbatim>
     <input type="text" name="projectName"
 dojoType="dijit.form.ValidationTextBox"
 regExp="[\w]+"
 required="true"
 invalidMessage="Invalid project name."/></f:verbatim>
</td>
</tr>
<tr>
<td>Project description: </td>
<td><h:inputHidden value="#{projectFormBean.projectDescription}"/><f:verbatim>
 <textarea dojoType="dijit.form.Textarea" style="width:80%">
     </textarea></f:verbatim>
</td>
</tr>
```

Changer les composants JSF en invisible et ajouter le widget Dojo

Etape2 Copiez les données de composants JSF aux widgets Dojo pendant la phase de calcul de rendu de la page

Plus tôt, nous avons remplacé les composants JSF avec des widgets Dojo dans la page, mais les données dans la page ne s'affiche pas dans les widgets Dojo. Ensuite, nous allons vous montrer comment copier les données des composants JSF pour les widgets Dojo.

Dans la phase de rendu, nous utilisons JavaScript pour extraire les données des composants JSF et y mettre les widgets Dojo liés.

```
<script type="text/javascript">
   function dojoInit(){
        dijit.registry.byClass('dijit.form.ValidationTextBox').forEach(function(pane){
          pane.setValue(pane.domNode.previousSibling.value);
     });

     dijit.registry.byClass('dijit.form.Textarea').forEach(function(pane){
          pane.setValue(pane.domNode.previousSibling.value);
     });
   }

   dojo.addOnLoad(dojoInit);
</script>
```

Copiez les données de composants JSF à des widgets Dojo au cours de la phase de rendu

Dans la fonction dojoInit vous trouverez tous les widgets Dojo invouqués par le nom de la classe. Pour chaque widget, vous trouverez son précédent nœud frère dans l'arbre Dom, qui est le composant JSF. Ensuite, vous extrayez les données de la composante JSF et réglez sur le widget Dojo demandées. De cette façon, vous pouvez synchroniser les données entre les composants JSF et les widgets Dojo. Enfin, vous utilisez la fonction dojo.addOnLoad pour s'assurer que la fonction dojoInit est appelée au cours de la phase de rendu.

Remarque: Lorsque vous utilisez le Dojo Toolkit, vous devriez éviter d'utiliser <body onload="..."> ou window.onload parce Dojo a son propre mécanisme d'initialisation qui utilise window.onload. Grâce à ce mécanisme vous va interférer avec les routines d'initialisation de Dojo. Dojo a fourni dojo.addOnLoad, qui vous permet de charger fonctions après Dojo a terminé son initialisation.

Etape3 Copiez les données de composants JSF aux widgets Dojo pendant la phase de calcul de rendu de la page

Actuellement, les widgets du dojo peuvent être affichés avec des données correctes dans la page. Mais si un utilisateur modifie les données, comment ces changements peuvent être reflétés à la correspondante d'arrière-forme haricot? Dans cette étape, nous montrons comment synchroniser les données entre les widgets Dojo et composants JSF pendant la page présenter la phase.

Tout d'abord, vous devez créer une fonction JavaScript pour copier les données à partir des widgets Dojo de composants JSF.

```
function setDojoValue(){
     dijit.registry.byClass('dijit.form.ValidationTextBox').forEach(function(pane){
          pane.domNode.previousSibling.value = pane.getValue();
     });

     dijit.registry.byClass('dijit.form.Textarea').forEach(function(pane){
          pane.domNode.previousSibling.value = pane.getValue();
     });
}
```

Copiez les données de widgets Dojo à composants JSF

Cette fonction est similaire à la fonction dojoInit montré à l'étape 2. Nous utilisons toujours la méthode dojo de recherche pour trouver tous les widgets du dojo par un nom de classe spécifié. Ensuite, nous extrayons les données de ces widgets Dojo et mis à composants JSF correspondant.

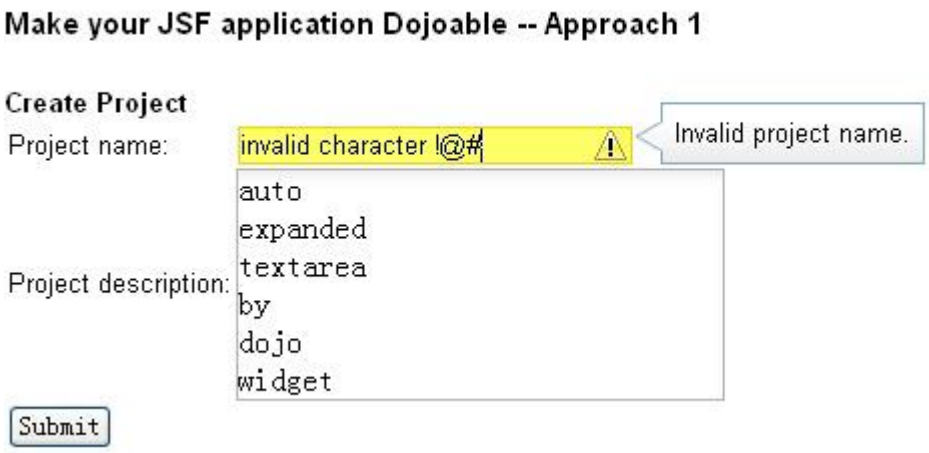
Ensuite, nous devons appeler la fonction setDojoValue lorsque la page présente, si nous changeons le bouton de soumission originale en

```
<h:commandButton id="button_submit" action="success"
    value="Submit" type="submit" onclick="setDojoValue();"></h:commandButton>
```

Bouton Submit avec le JavaScript associé

Nous ajoutons l'attribut onclick dans le JSF commandButton, donc quand un utilisateur clique sur ce bouton, la fonction JavaScript est invoqué en premier, puis la page sera soumise. De cette façon, nous pouvons copier toutes les données dans les widgets Dojo à composants JSF avant que la page est présentée, et les modifications de données seront prises en compte sous forme de back-end bean.

C'est tout ce que vous devez faire en utilisant cette approche. La figure 3 montre la nouvelle page Créer un projet.



Comme vous pouvez le voir, nous utilisons une boîte d'entrée validées qui peut soutenir la validation côté client. Nous utilisons également la zone de texte de dojo, qui peut être auto-développé sans une barre de défilement.