http://www.egovframe.go.kr/wiki/doku.php?id=egovframework:rte2:ptl:jquery_ajax

jQuery Guideline

Outline

Being the Java Script Library offering browser compatibility, jQuery provides open libraries that makes feasible Java-based Ajax, Event and UI Functions. In the WikiGuide we're going to learn the basic functions of jQuery (Ajax, Callback Function, Post Call).

Visit jQuery webfor more information.

Refer to the following for how to work Combo Box and Select Box via jQuery Ajax.

- Basic jQuery Ajax Functions

- jQuery.ajax()
- jQuery.get()
- jQuery.post()

- Advanced jQuery Ajax Functions

- Select box
- Tabs

Settings

You'll need to add jQuery Java Script before using jQuery. When you intend to provide jQuery URL, you may have jQuery Java Script referred by the Project.

- How to provide jQuery URL:

<script src="//code.jquery.com/jquery-1.11.0.min.js"></script>

- How to add the relevant jQuery script for reference:

<script type="text/javascript" src="jQueryFile Directory"></script>

You'll then need to download jQuery-Ver.js, add the file to the Project hierarchy and provide the directory information.

Basic jQuery Ajax Functions

jQuery.ajax()

In order to use jQuery Ajax Function, you'll need to refer to the Function jQuery.ajax(url[,settings]).

Settings in Ajax Function

You can configure in jQuery Ajax as follows:

Settings	Description	default	type
url	The target URL for request	N/A	url string

data	Name and value of data to be contained in request	N/A	String/Plain Object/Array
contentType	contentType upon transfer of data to server	'application/x-www-form-urlencoded; charset=UTF-8'	contentType String
dataType	Type of data to e transferred in from the server	xml, json, script, or html	xml/html/script/json/jsonp/multiple, space-separated values
statusCode	Function branch-handled by HTTP status code	N/A	Function divided by status code
beforeSend	Callback Function called prior to the request reaching the server	N/A	Function(jqXHR jqXHR, PlainObject settings)
error	Function called upon request error	N/A	Function(jqXHR jqXHR, String textStatus, String errorThrown)
success	Function called upon successful request	N/A	Function(PlainObject data, String textStatus, jqXHR jqXHR)
crossDomain	To force crossDomain request(e.g. jsonP) (cross-domain request required)	"False" for same-domain request, "True" for cross-domain request	Boolean

Example 1

Refer to the following example for how to send a parameter to Ajax request.

Here, you need to work example01.do and transfer the String "sampleData" entitled sampleInput. The following Java Script code calls either Function Success or Function Error depending on the request successful or failed.

```
$.ajax({
   url : "<c:url value='/example01.do'/>",
   data : {
      sampleInput : "sampleData"
   },
   success : function(data, textStatus, jqXHR) {
      //If Sucess, Proceed
   },
   error : function(jqXHR, textStatus, errorThrown){
      //If Error, Proceed
}
});
```

Function Callback for Ajax

Inclusive of the Function Callback for Ajax, every Ajax function available in jQuery 1.5 or nwer edition is now entitled to get the super set of the Object XMLHttpRequest, referred to as "jqXHR". You can define a Callback in the function of jqXHR.

Meanwhile, the Function Callback defined previously and varying depending on error or success is quite different from the Function Callback for Ajax in the following perspectives:

- The Functions Callback for Ajax take turn as defined by the user.
- You can have the request returned back from Ajax to call.

Function	Description

jqXHR.done(function(data, textStatus, jqXHR) {});	Function Callback called upon success
jqXHR.fail(function(jqXHR, textStatus, errorThrown) {});	Function Callback called upon error
jqXHR.always(function(data|jqXHR, textStatus, jqXHR'|errorThrown) { });	Function Callback always called with no exce ption

Example 2

Refer to the following example for how to send multiple data out to get the Function Callback returned back from the server.

Working the following example calls example02.do and sends name and location out in the form of request data. The Function Callback "done" is returned upon success.

```
$.ajax({
   url : "<c:url value='/example02.do'/>",
   data : {
      name : "gil-dong",
      location : "seoul"
   },
})
   .done(function( data ) {
      if ( console && console.log ) {
         console.log( "Sample of data:", data.slice( 0, 100 ) );
      }
   });
```

Example 3

Example 3 calls example03.do, returning either Function Callback "done" or "fail" as per the result.

You'll see the Function Callback "always" called with no exception. These Functions Callback are returned by way of Function Ajax and callable by request.

```
var jqxhr = $.ajax( "<c:url value='/example03.do'/>", )
   .done(function() {
      alert( "success" );
   })
   .fail(function() {
      alert( "error" );
   })
   .always(function() {
      alert( "complete" );
   });

jqxhr.always(function() {
   alert( "second complete" );
});
```

jQuery.get()

From jQuery 1.5 on, the Function Callback "success" is entitled to get jqXHR(the super set object of XMLHttpRequest). Contrary to cross-domain request where request for "Get" like JSONP involves use of jqXHR, the Factor XHR is recognized "undefined" in the FUnction Success.

Requesting for Ajax, Query.get() returns jqXHR back, in which case the Functions Callback (done, fail, always) are made available like $.ajax() is.
Refer to the following for how to represent the Function Get in the function of Ajax:

```
$.ajax({
    url: url,
    data: data,
    success: success,
    dataType: dataType
});
```

How to represent Function Get

Settings	Description	default	type
url	The target URL for request	N/A	url String
data	Name and value of data to be contained in request	N/A	String/Plain Object
dataType	Type of data to e transferred in from the server	xml, json, script, or html	String
success	Function called upon successful request	N/A	Function(PlainObject data, String textStatus, jqXHR jqXHR)

Example 1

URL called, Result ignored

```
$.get( "example.do" );
```

Example 2

Data out to URL, Result ignored

```
$.get( "example.do", { name: "gil-dong", location: "seoul" } );
```

Example 3

URL called, Result available in Alert

```
$.get( "test.php", function( data ) {
    alert( "Data Loaded: " + data );
});
```

Example 4

URL called, Result available in Alert

```
$.get( "example.do", { name: "gil-dong", location: "seoul" } )
    .done(function( data ) {
        alert( "Data Loaded: " + data );
});
```

jQuery.getJSON()

Function jQuery.getJSON() requests data encoded in JSON strings by way of the Method HTTP GET.

Refer to the following for how to represent the MEthod $.ajax():

```
$.ajax({
    url: url,
    data: data,
    success: success,
    dataType: 'json'
});
```

How to configure getJSON

Settings	Description	default	type
url	The target URL for request	N/A	url String
data	Name and value of data to be contained in request	N/A	String/Plain Object
dataType	Type of data to e transferred in from the server	xml, json, script, or html	String

jQuery.post()

Requesting POST for Ajax, jQuery.post() gets jqXHR back, in which case the Functions Callback (done, fail, always) are made available like ajax() and get() are.
You may work the same to configure jQuery.post as the Function Get: (jQuery.post(url [, data] [, success] [, dataType]))

Refer to the following for how to represent jQuery.post in the function of Ajax:

```
$.ajax({
    type: "POST",
    url: url,
    data: data,
    success: success,
    dataType: dataType
});
```

Example 1

URL called, Result ignored

```
$.post( "example.do" );
```

Example 2

Data out to URL, Result ignored

```
$.post( "example.do", { name: "gil-dong", location: "seoul" } );
```

Example 3

URL called, Console Log left

```
$.post( "example.do", function( data ) {
```

```
    console.log( data.name );
    console.log( data.location );
  });
```

Example 4

Data called by URL, Result available in Alert

```
  $.post( "example.do", { name: "gil-dong", location: "seoul" } )
     .done(function( data ) {
        alert( "Data Loaded: " + data );
  });
```

Advanced jQuery Ajax Functions

You'll need to configure jQuery UI for advanced jQuery Ajax functions such as Auto-complete and Panel Tabs, described as follows:

How to configure jQuery UI

In order to add jQuery UI(version 1.11.0) you'll need to add the basic jQuery script and jQuery UI Script to jsp, as follows:

```
...
<link rel="stylesheet"
            href="http://code.jquery.com/ui/1.11.0/themes/smoothness/jquery-ui.css" />
<script src="//code.jquery.com/jquery-1.11.0.min.js"></script>
<script src="http://code.jquery.com/ui/1.11.0/jquery-ui.js"></script>
...
```

When you add jQuery UI script yourself to refer to, you'll need to add jQuery-ui.js and jQuery-ui.css to the Project hierarchy and provide the directory information.

Auto complete

jQuery offers you autoComplete() that demonstrates the projected strings on the given input.

How to configure autoComplete

Category	Settings	Description	Type
Options	source	autoComplete list (required) made available at the bottom	Array, String, function
Options	minLength	Mininum length of strings to work autoComplete	Integer
Options	disabled	State disabled	Boolean
Events	change(event, ui)	Function Event occurring upon change of value	autocompletechange
Events	focus(event, ui)	Function Event occurring upon focus of value	autocompletefocus
Events	select(event, ui)	Function Event occurring when value is selected	autocompleteselect

Visit jQuery autocomplete api for more information.

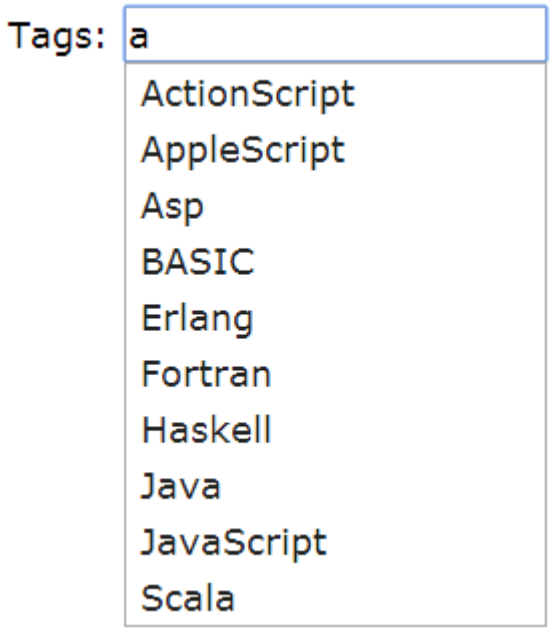
autoComplete Examples

Refer to the following for how to implement autoComplete:

```
<html lang="en">
<head>
   <meta charset="utf-8">
   <title>jQuery UI Autocomplete - Default functionality</title>
   <link rel="stylesheet" href="//code.jquery.com/ui/1.11.0/themes/smoothness/jquery-ui.css">
   <script src="//code.jquery.com/jquery-1.10.2.js"></script>
   <script src="//code.jquery.com/ui/1.11.0/jquery-ui.js"></script>
   <link rel="stylesheet" href="/resources/demos/style.css">
   <script>
   $(function() {
      var availableTags = [ "ActionScript", "AppleScript", "Asp", "BASIC",
                                          "C", "C++", "Clojure", "COBOL", "ColdFusion", "Erlang",
                                          "Fortran", "Groovy", "Haskell", "Java", "JavaScript", "Lisp",
                                          "Perl", "PHP", "Python", "Ruby", "Scala", "Scheme" ];
      $( "#tags" ).autocomplete({
         source: availableTags
      });
   });
   </script>
</head>
<body>

<div class="ui-widget">
   <label for="tags">Tags: </label>
   <input id="tags">
</div>
</body>
</html>
```

See the following for the result of the foregoing:



With minLength having the dfault value of 1, you'll see the Strings of source appears in the form of autoComplete list when you make an input.

Advanced example for autoComplete and Ajax

Ajax also works to get the list for broadcasting to source of autoComplete.

In order to get source via Ajax, the result from server called should be either:

- String Array; or
- Object Array (w/ id, label, and value).

1. Getting String Array

You'll call the source by Ajax via example.do while getting the value alerted.

<java script>

```
  ...
<script type="text/javaScript">
$(function() {
    $('#autoValue').autocomplete(
        {
            source : function(request, response) {
                $.ajax({
                    url : "<c:url value='/example.do'/>",
                    data : { input : request.term },
                    success : function(data) {
                        response( data.locations );
                    }
                });
            },
            minLength : 1,
            select : function(event, ui) {
                alert( ui.item ? "Selected : " + ui.item.label
                        : "Nothing select, input was " + this.value);
            }
        });
});
</script>
//... Omitted
<input type="text" id="autoValue" />
//... Omitted
```

Keep in mind request.term sent out refers to the user's input. This signifies that when the user inputs “je”, request.term would contain the string "input = je", in which case Controller request.getParameter(“input”) or @RequestParam(“input”) String input to get the value out.

Meanwhile, you might want to check the following example when the Bean for MappingJacksonJsonView is registered in jsonView to extract the date out of Controller and send the result out to client. (See <Ajax support java code>, the corresponding part in WikiGuide for coding for Ajax communication)

```
  @RequestMapping(value="/autoList.do")
public String autoList(HttpServletRequest request, ModelMap model) {

            String input = request.getParameter("input");
            List<String> resultList = new ArrayList<String>();
            //...omitted...
            //Result contained in       resultListvia service class
            model.addAttribute("locations", resultList );

            return "jsonView";
}
```

Refer to the following example for the Query called (mybatis example):

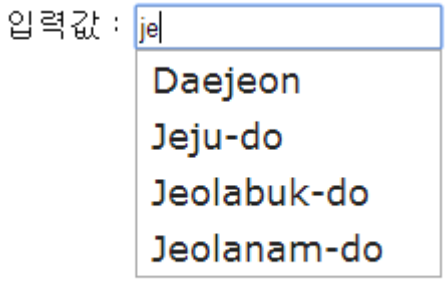
```
<select id="selectLocationList" parameterType="string" resultType="string">
SELECT
        LOCATION_NM
FROM LOCATION
        WHERE upper(LOCATION_NM) LIKE '%' || upper(#{input}) || '%'
</select>
```

If the result is out:

{"locations":["Daejeon","Jeju-do","Jeolabuk-do","Jeolanam-do"]}

then you may get the value out to data.locations in the Function Callback "success" for configuration of source for autoComplete.

Refer to the following for the result:



2. Getting Object Array

When attempting to get object array, you'll retrieve data by the Query input to get the Object Array from the server.

When you hand the value of List<Object> entitled "location" from Controller off to ModelMap and get the json data:

{"locations":[{"locationId":"0006","locationNm":"Daejeon","localNb":"042"},{"locationId":"0010","locationNm":"Jeju-do","localNb":"064"},{"locationId":"0011","locationNm":"Jeolabuk-do","localNb":"063"},{"locationId":"0012","locationNm":"Jeolanam-do","localNb":"061"}]}

You'd have the Object appear the value from the source of autoComplete containing label, id and value, in which case you must convert the request returned into the form of Object to see the corresponding string in the Function Callback "success". The rest are all the same as other jQueries.

```
success : function(data) {
        response($.map(data.locations, function(item) {
                return{
                        id: item.locationId,
                        label: item.locationNm,
                        //value: item.localNb
                }));
        }
```

Check that, among other values, "label" appears on autoComplete List. If you have configured for "value", you'll see autoComplete List contains "value" instead of "label" as an input.

Select Box

Controlling Selectbox(combobox)

jQuery does not provide you with Select Box UI Function.

In this part of WikiGuide, we are going to learn how jQuery controls select box and how to implement the list of selectbox by way of Ajax.

For the following descriptions, we assume the selectbox is as follows:

```
<select id="combobox">
    <option value="">===locations===</option>
    <option value="01">Seoul</option>
    <option value="02">Busan</option>
    <option value="03">Jeju-do</option>
    <option value="04">Incheon</option>
</select>
```

Loading value from selectbox

You can load the value from selectbox as follows:

```
var selectedVal = $("#combobox option:selected").val();
```

Loading texts from selectbox

You can load the text selected in selectbox (e.g. Seoul) as follows:

```
var selectedText= $("#combobox option:selected").text();
```

Loading index selected in selectbox

You can load index in selectbox as follows:

```
var selectedIndex = $("#combobox option").index($("#combobox option:selected"));
```

Adding the list at the end of selectbox

You can add the list at the end of selectbox as follows:

```
$("#combobox").append("<option value="05">Daejeon</option>");
```

You can add the list at the beginning of selectbox using .prepend().

Replacing values in selectbox

You can replace values from selectbox as follows:

```
$("#combobox").html(
    "<option value="">===locations===</option>
    <option value='01'>Jeju-do</option>
    <option value='02">Seoul</option>
    <option value='03'>Incheon</option>
    <option value='04'>Daejeon</option>")
```

Replacing values in selectbox

You can replace values from selectbox as follows:

```
$("#combobox").html(
```

```
"<option value="">===locations===</option>
<option value='01'>Jeju-do</option>
<option value='02">Seoul</option>
<option value='03'>Incheon</option>
<option value='04'>Daejeon</option>")
```

Deleting values from selectbox

You can delete the value selected in the list of selectbox as follows:

```
$("#combobox option:selected").remove();
```

Function Callback when value is selected in selectbox

Function Callback is called when a value is selected in selectbox.

```
$("#combobox).change(function() {
    //Implementing Function
});
```

Implementing selectbox

By working the selectbox control function and Function Ajax, you can generate selectbox as follows:

```
<Implementing Combobox in JSP>

<script type="text/javaScript">
        $(function() {
                $.ajax({
                        url : "<c:url value='/simpleCombo.do'/>",
                        success : function(data) {
                                loadCombo($("#combobox"), data.locations);
                            $("#combobox").val("");
                        }
                });

                $("#combobox").change(function() {
                        alert("Selected : " + $("#combobox option:selected").val());
                });
        });

        function loadCombo(target, data) {
                var dataArr = [];
                var inx = 0;
                target.empty();

                $(data).each( function() {
                        dataArr[inx++] = "<option value=" + this.locationId + ">" + this.locationNm +
"</option> ";
                });

                target.append(dataArr);
        }
</script>

//... Omitted
<select id="combobox">
```

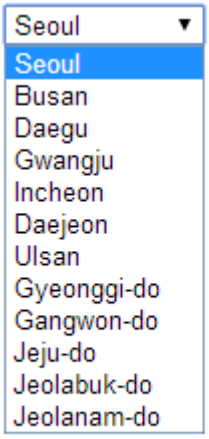
```
        <option>===locations===</option>
</select>
```

<Result Returned from Server>

{"locations":[{"locationId":"0001","locationNm":"Seoul"},{"locationId":"0002","locationNm":"Busan"},{"locationId":"0003","locationNm":"Daegu"},{"locationId":"0004","locationNm":"Gwangju"},{"locationId":"0005","locationNm":"Incheon"},{"locationId":"0006","locationNm":"Daejeon"},{"locationId":"0007","locationNm":"Ulsan"},{"locationId":"0008","locationNm":"Gyeonggi-do"},{"locationId":"0009","locationNm":"Gangwon-do"},{"locationId":"0010","locationNm":"Jeju-do"},{"locationId":"0011","locationNm":"Jeolabuk-do"},{"locationId":"0012","locationNm":"Jeolanam-do"}]}

When the Function Ajax is executed to load the data via simpleCombo.do and the json described above, you can implement the function comprising combobox to refer to the list available in combobox.

See the following for the result:



Tabs

Implementing Fundamental Tabs

The Function tabs () implements tabs in Query UI. The configurations are as follows:

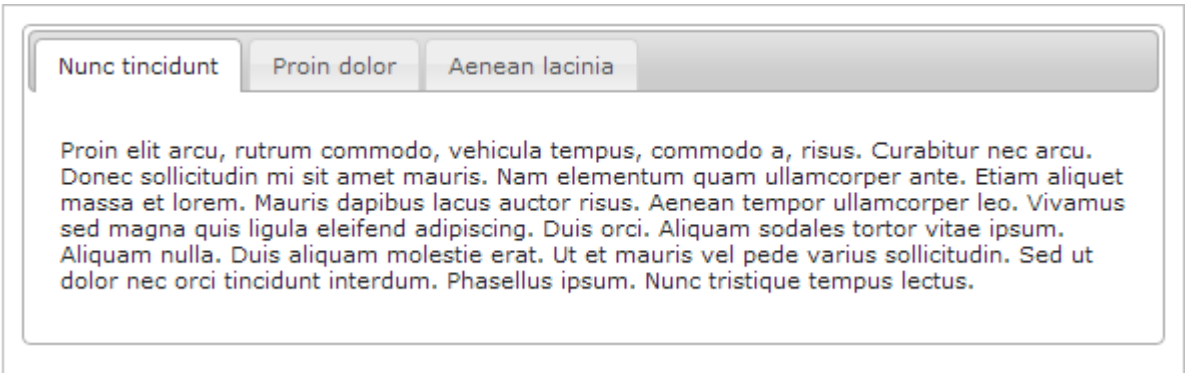
Settings	Description	Category
active	Chooses the panel for implementation	options
event	Events in which tabs are implemented	options
hide	Animation when the panel gets hidden	options
show	Animation when the panel is shown	options
beforeLoad	Event functions implemented prior to the remote tab loaded	events
create		events

Refer to the following example for how to implement tabs:

```
<!doctype html>
<html lang="en">
<head>
```

```
    <meta charset="utf-8">
    <title>jQuery UI Tabs - Default functionality</title>
    <link rel="stylesheet" href="//code.jquery.com/ui/1.11.0/themes/smoothness/jquery-ui.css">
    <script src="//code.jquery.com/jquery-1.10.2.js"></script>
    <script src="//code.jquery.com/ui/1.11.0/jquery-ui.js"></script>
    <link rel="stylesheet" href="/resources/demos/style.css">
    <script>
    $(function() {
      $( "#tabs" ).tabs();
    });
    </script>
</head>
<body>

<div id="tabs">
    <ul>
      <li><a href="#tabs-1">Nunc tincidunt</a></li>
      <li><a href="#tabs-2">Proin dolor</a></li>
      <li><a href="#tabs-3">Aenean lacinia</a></li>
    </ul>
    <div id="tabs-1">
      <p>Proin elit arcu, rutrum commodo, vehicula tempus, commodo a, risus. Curabitur nec arcu. Donec sollicitudin mi sit amet mauris. Nam elementum quam ullamcorper ante. Etiam aliquet massa et lorem. Mauris dapibus lacus auctor risus. Aenean tempor ullamcorper leo. Vivamus sed magna quis ligula eleifend adipiscing. Duis orci. Aliquam sodales tortor vitae ipsum. Aliquam nulla. Duis aliquam molestie erat. Ut et mauris vel pede varius sollicitudin. Sed ut dolor nec orci tincidunt interdum. Phasellus ipsum. Nunc tristique tempus lectus.</p>
    </div>
    <div id="tabs-2">
      <p>Morbi tincidunt, dui sit amet facilisis feugiat, odio metus gravida ante, ut pharetra massa metus id nunc. Duis scelerisque molestie turpis. Sed fringilla, massa eget luctus malesuada, metus eros molestie lectus, ut tempus eros massa ut dolor. Aenean aliquet fringilla sem. Suspendisse sed ligula in ligula suscipit aliquam. Praesent in eros vestibulum mi adipiscing adipiscing. Morbi facilisis. Curabitur ornare consequat nunc. Aenean vel metus. Ut posuere viverra nulla. Aliquam erat volutpat. Pellentesque convallis. Maecenas feugiat, tellus pellentesque pretium posuere, felis lorem euismod felis, eu ornare leo nisi vel felis. Mauris consectetur tortor et purus.</p>
    </div>
    <div id="tabs-3">
      <p>Mauris eleifend est et turpis. Duis id erat. Suspendisse potenti. Aliquam vulputate, pede vel vehicula accumsan, mi neque rutrum erat, eu congue orci lorem eget lorem. Vestibulum non ante. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos. Fusce sodales. Quisque eu urna vel enim commodo pellentesque. Praesent eu risus hendrerit ligula tempus pretium. Curabitur lorem enim, pretium nec, feugiat nec, luctus a, lacus.</p>
      <p>Duis cursus. Maecenas ligula eros, blandit nec, pharetra at, semper at, magna. Nullam ac lacus. Nulla facilisi. Praesent viverra justo vitae neque. Praesent blandit adipiscing velit. Suspendisse potenti. Donec mattis, pede vel pharetra blandit, magna ligula faucibus eros, id euismod lacus dolor eget odio. Nam scelerisque. Donec non libero sed nulla mattis commodo. Ut sagittis. Donec nisi lectus, feugiat porttitor, tempor ac, tempor vitae, pede. Aenean vehicula velit eu tellus interdum rutrum. Maecenas commodo. Pellentesque nec elit. Fusce in lacus. Vivamus a libero vitae lectus hendrerit hendrerit.</p>
    </div>
</div>
</body>
</html>
```



- referenced jquery ui site

How to implement Tabs via Ajax

All you need to implement Tabs via Ajax is to designate a URL to call Ajax.

```
<script src="http://code.jquery.com/jquery-1.10.2.js"></script>
<script src="http://code.jquery.com/ui/1.11.0/jquery-ui.js"></script>
<script type="text/javaScript">
$(function() {
      $( "#tabs" ).tabs({
         beforeLoad: function( event, ui ) {
            ui.jqXHR.error(function() {
               ui.panel.html(
                  "Couldn't load this tab. We'll try to fix this as soon as possible. " +
                  "If this wouldn't be a demo." );
            });
         }
      });
   });
</script>
</head>
<body>
    <div id="tabs">
        <ul>
            <li><a href="${pageContext.request.contextPath }/simpleCombo.do">Tab 1</a></li>
             <li><a href="${pageContext.request.contextPath }/tabTwoForm.do">Tab 2</a></li>
             <li><a href="${pageContext.request.contextPath }/tabThreeForm.do">Tab 3</a></li>
        </ul>
    </div>
</body>
```

When the first tab is designated, you'll see simpleCombo.do is called so that the panel displays what is called from simpleCombo.do.

References

- jQuery Web Site
- jQuery UI Web Site