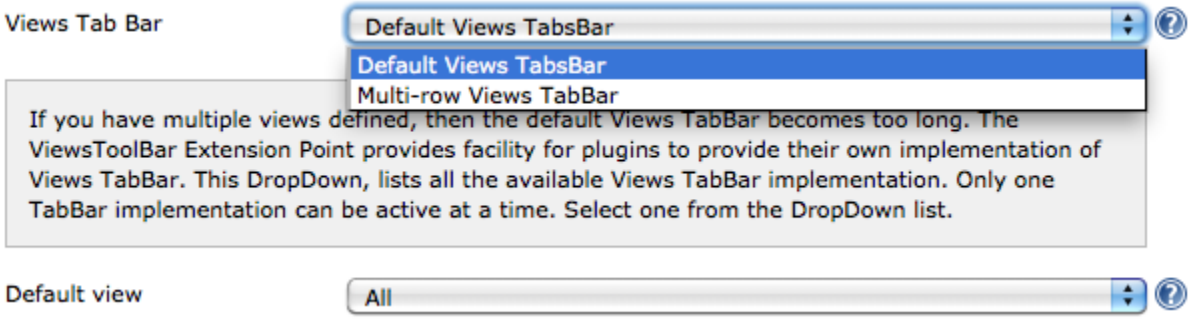
Extension Point for Project Views Navigation

If you have many views, the TabBar that displays the view names becomes very large. The solution would be to use alternate UI for the the Project View List such as

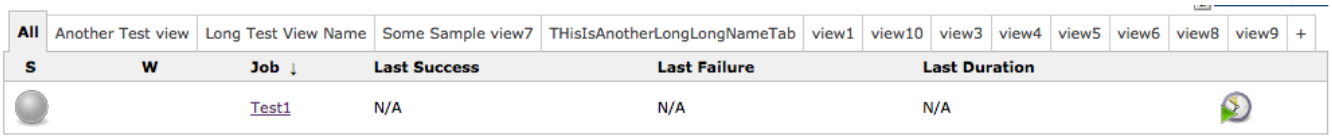
- Multi-line TabBar
- Scrollling TabBar
- TabBar with dropdown List
- ..

The best solution would be to provide some kind of Extension mechanism so that plugin developer can provide their own Views Tab Bar UI.

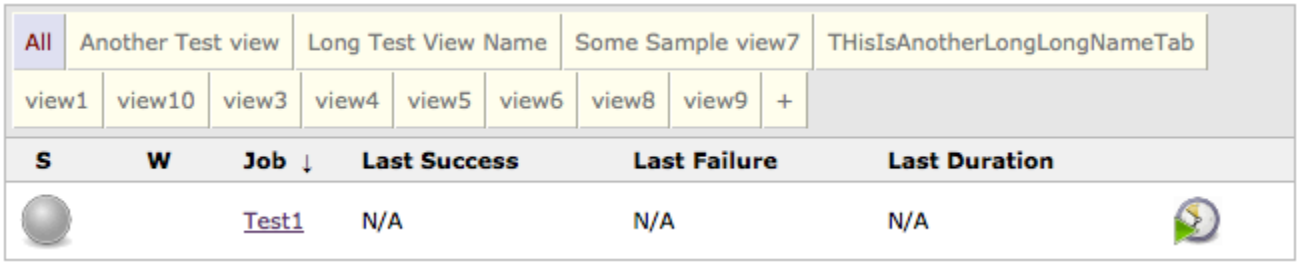
Such an Extension Point (ViewsTabBar) is introduced now (in a branch as of 1.378) and plugin developers can provide their own UI by extending the Extension Point. The alternate UI provided by plugin developers would appear in a DropDown list in the main Hudson Configuration page as



On selecting the Default Views TabBar, the Dashboard would show the tabs with the default UI.



I have implemented a simple plugin that enables displaying the tabs in a muti-row fashion. The alternate UI looks like



Extending `ViewsTabBar` Extension Point

To provide alternate UI for TabBar extend the ***ViewsTabBar*** in your plugin as

```
public class MultilineViewsTabBar extends ViewsTabBar {
   @DataBoundConstructor
   public MultilineViewsTabBar() {
   }

   @Extension
   public static class DescriptorImpl extends ViewsTabBarDescriptor {

       @Override
       public String getDisplayName() {
           return "Multi-row Views TabBar";
       }
   }
}
```

The value returned by getDisplayName() would be displayed in the DropDown List.

The UI itself is provided via a page named "viewTabs.jelly". To this page three attributes are passed in

```
<%@ attribute name="it" required="true" type="<Your Plugin Class>" %>
<%@ attribute name="views" required="true" type="java.util.Collection<hudson.model.View>" %>
<%@ attribute name="currentView" required="true" type="hudson.model.View" %>
```

viewTabs.jelly may look something like

viewTabs.jelly

```
<j:jelly xmlns:j="jelly:core" xmlns:st="jelly:stapler" xmlns:d="jelly:define" xmlns:lc="/lib/layout/tabbar" >
  <!-- view tab bar -->
    <lc:tabBar>
        <j:forEach var="v" items="${views}">
            <lc:tab name="${v.viewName}" active="${v==currentView}" href="${rootURL}/${v.url}" />
        </j:forEach>
        <j:if test="${currentView.hasPermission(currentView.CREATE)}">
            <lc:tab name="+" href="${rootURL}/${currentView.owner.url}newView" active="false"
             title="${%New View}" />
        </j:if>
    </lc:tabBar>
</j:jelly>
```

For Multi-Row Tabbar I wrote the TabBar.jelly and Tab.jelly as

TabBar.jelly

```
<j:jelly xmlns:j="jelly:core" xmlns:d="jelly:define" xmlns:st="jelly:stapler">

   <style type="text/css">

           /* CSS Tabs */
           #tabBarContainer {
               background: #E6E6E6;
               margin: 10px auto 0;
               padding: 3px;
               border: 1px solid #BBBBBB;
           }

           /* to stretch the container div to contain floated list */
           #tabBarContainer:after {
               content: ".";
               display: block;
               line-height: 1px;
               font-size: 1px;
               clear: both;
           }
```

```
            ul#tabList {
                list-style: none;
                padding: 0;
                margin: 0 auto;
                width: 100%;
            }

            ul#tabList li {
                display: block;
                float: left;
                /*width: 10%;*/
                margin: 0;
                padding: 0;
                text-align: center
            }

            ul#tabList li a {
                display: block;
                width: 100%;
                padding: 6px;
                border-width: 1px;
                border-color: #ffe #aaab9c #ccc #fff;
                border-style: solid;
                color: #777;
                text-decoration: none;
                background: #FEFEEE;
            }

            #tabBarContainer>ul#tabList li a { width: auto; }

            ul#tabList li#active a {
                background: #E0E0F0;
                color: #800000;
            }

            ul#tabList li a:hover, ul#tabList li#active a:hover {
                color: #800000;
                background: transparent;
                border-color: #aaab9c #fff #fff #ccc;
            }

    </style>

    <div id="tabBarContainer">
        <ul id="tabList">
            <j:set var="tab" value="${tabs}" />
            <d:invokeBody />
        </ul>
    </div>

</j:jelly>
```

Tab.jelly

```
<!--
<%@ attribute name="name" required="true" type="java.lang.String" %>
<%@ attribute name="href" required="true" type="java.lang.String" %>
<%@ attribute name="active" required="true" type="java.lang.Boolean" %>
<%@ attribute name="title" required="false" type="java.lang.String" %>
-->

<j:jelly xmlns:j="jelly:core">

    <j:choose>
        <j:when test="${active}">
            <li id="active">
                <a id="current" href="${href}" title="${attrs.title}">${name}</a>
            </li>
        </j:when>
        <j:otherwise>
            <li>
                <a href="${href}" title="${attrs.title}">${name}</a>
            </li>
        </j:otherwise>
    </j:choose>

</j:jelly
```