**Integrating Stripes Jquery And Ajax To Create A Web Application**

### Introduction:

This document describes how to create a simple web application using Stripes, different types of AJAX calls and jQuery. It also describes how an Ajax call can be synchronized with loading an image.

### Software Requirements

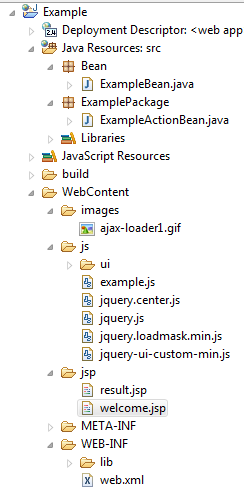
* Eclipse Ganymede IDE version 3.4.2
* Stripes Framework 1.5.6

### Jar Files Required:

1. Stripes-1.5.6-bundle.jar
2. Commons-logging.jar
3. Flexjson.jar
4. Jstl.jar
5. Stripes.jar (These jars can be downloaded.)

### Folder Structure:

The snapshot below depicts the folder structure used for the web application example:



* Example is a Dynamic web Project having ‘src’ and ‘WebContent’.
* Under ‘src’ two packages are defined :

1. Bean: It has the data model beans.
2. ExamplePackage: It has all the actionBeans.

* Under WebContent following folders are defined:

1. Images: It has all the images used.
2. Js: It has all the ‘javascript source’ files.
3. Jsp: It has all the ‘JSPs’.

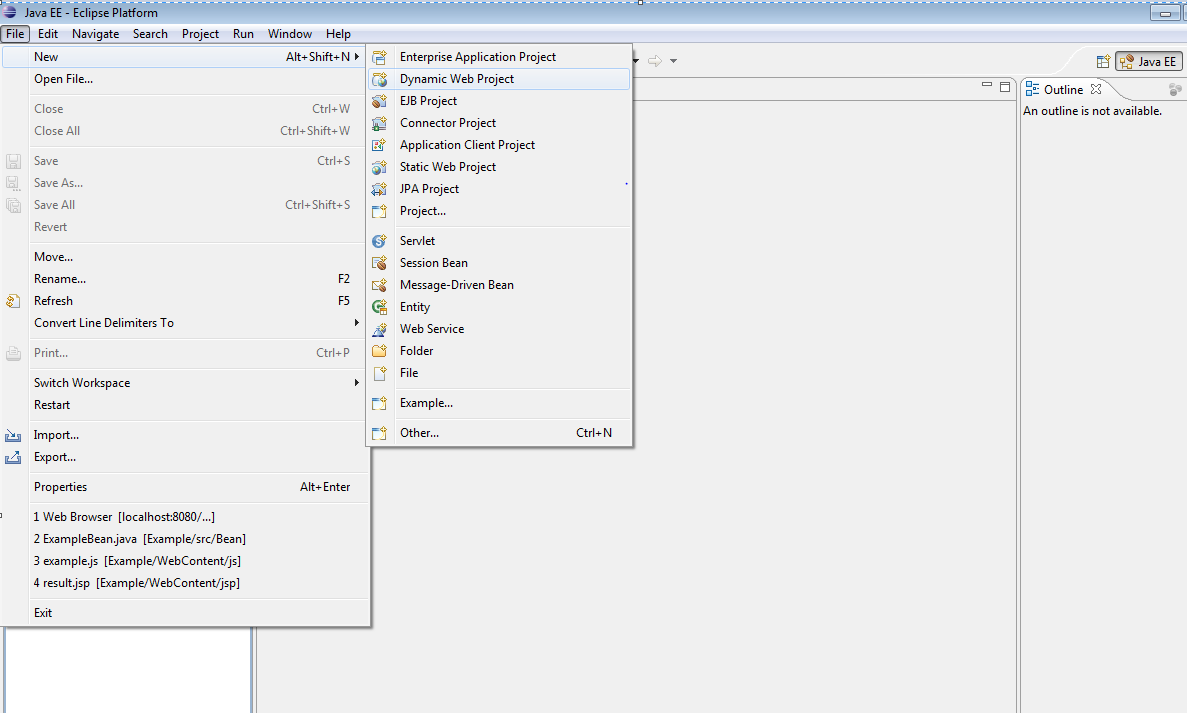
and web.xml is also defined.

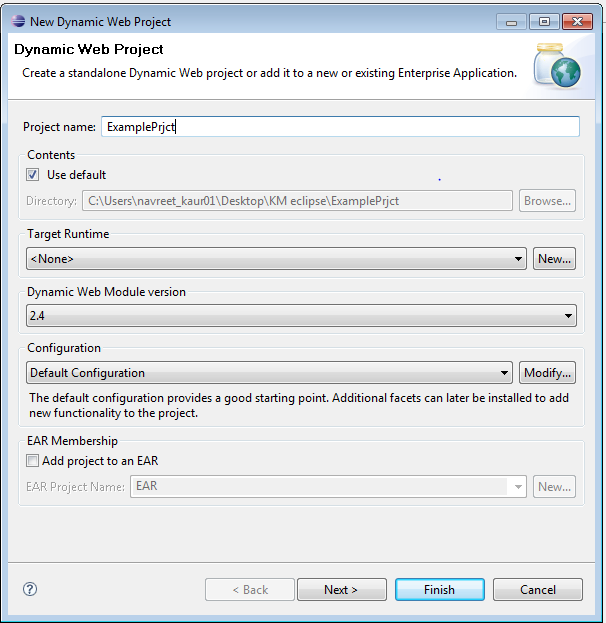
The logical steps followed to create the web application are defined in the next section.

### Steps to Implement:

**Step 1:**

1. Open Eclipse Ganymede. Start a new dynamic web project.





1. Add the JAR files included in the zipped folder (stripes-1.5.6-bundle.jar, commons-logging-1.04.jar, flexjson.jar, jstl.jar, stripes.jar) to the lib folder of WebContent/Web-INF.

**Step 2:** Update the web.xml file:

* Configuration of Stripes Filter with a parameter and the Stripes Dispatcher Servlet is required.
* ActionBeans are recognized by stripes at deployment time by scanning web application's classpath.

Any standard stripes configuration can be used. (Refer <http://www.stripesframework.org/display/stripes/Quick+Start+Guide>)

|  |
| --- |
| <filter>  ----------------------------------------  ----------------------------------------  <init-param>  <param-name>ActionResolver.Packages</param-name>  <param-value>ExamplePackage</param-value>  </init-param>  </filter> |

‘ExamplePackage’ is the package name where the ActionBean is located.

**Step 3:** Create a package named Bean and add class ExampleBean.java to it. This class will hold variables used in presentation layer.

* Add the following variables to the class:

1. private String country;
2. private String place;
3. private String meansOfTravel;
4. private String availableOptions;
5. private List<String> countries;
6. private List<String> places;
7. private List<String> meansOfTravelList;
8. private List<String> availableOptionList;

* Make getters and setters for the same.

**Step 4:** Create a jsp folder under web-content and add the following two JSP files to it.

1. This JSP page is the home page named ‘welcome.jsp’.

|  |
| --- |
| <html>  <head>  <script type="text/javascript" src="js/jquery.js"></script>  <script type="text/javascript" src="js/jquery-ui-custom-min.js"></script>  <script src="js/ui/jquery.ui.core.js"></script>  <script src="js/ui/jquery.ui.widget.js"></script>  <script src="js/ui/jquery.ui.button.js"></script>  <script type="text/javascript" src="js/jquery.loadmask.min.js"></script>  <script type="text/javascript" src="js/example.js"></script>  <script src="js/ui/jquery.ui.mouse.js"></script>  <title>Welcome To TravelWithUs</title>  <style type="text/css">  .heading {  padding-left: 12px;  text-align: center;  height: 190px;  color: grey;  width: 96%;  }  .loader1 {  position: fixed;  top: 50%;  left: 50%;  margin-left: -50px;  margin-top: -50px;  text-align: center;  z-index: 9999;  overflow: auto;  width: 150px;  height: 102px;  }  </style>  </head>  <body bgcolor="#E0EEE0">  <stripes:form action="/Example.action">  <div id="loader" class="loader1" style="display: none;"><img  id="img-loader" src="images/ajax-loader1.gif" alt="Loading" /><stripes:label  name="Plaese Wait" /></div>  <h1 id="heading" class="heading">Welcome To TravelWithUs</h1>  <table align="center" bgcolor="#CFCFCF" cellspacing="1" cellpadding="1"  style='margin: 0px 0; table-layout: fixed; width: 600px; height: 300px'>  <tr>  <td bgcolor="#C6AEC7">&nbsp;&nbsp;&nbsp;&nbsp;Country :</td>  <td bgcolor="#add8e6"><stripes:select name="exampleBean.country"  id="country" style="width:90%;">  <stripes:option value="">--Select--</stripes:option>  <stripes:options-collection  collection="${actionBean.exampleBean.countries}" />  </stripes:select></td>  </tr>  <tr>  <td bgcolor="#C6AEC7">&nbsp;&nbsp;&nbsp;&nbsp;Places :</td>  <td bgcolor="#add8e6"><stripes:select name="exampleBean.place"  id="place" style="width:90%;">  <stripes:option value="">--Select--</stripes:option>  <stripes:options-collection  collection="${actionBean.exampleBean.places}" />  </stripes:select></td>  </tr>  <tr>  <td bgcolor="#C6AEC7">&nbsp;&nbsp;&nbsp;&nbsp;Means Of Travel :  </td>  <td bgcolor="#add8e6"><stripes:select  name="exampleBean.meansOfTravel" id="meansOfTravel"  style="width:90%;">  <stripes:option value="">--Select--</stripes:option>  <stripes:options-collection  collection="${actionBean.exampleBean.meansOfTravelList}" />  </stripes:select></td>  </tr>  <tr>  <td bgcolor="#C6AEC7">&nbsp;&nbsp;&nbsp;&nbsp;Timings available  :</td>  <td bgcolor="#add8e6"><stripes:select  name="exampleBean.availableOptions" id="availableOptions"  style="width:90%;">  <stripes:option value="">--Select--</stripes:option>  <stripes:options-collection  collection="${actionBean.exampleBean.availableOptionList}" />  </stripes:select></td>  </tr>  <tr>  <td colspan="2" align="center"><stripes:submit name="showResult"  id="showResult" value="Submit" /></td>  </tr>  </table>  </stripes:form>  </body>  </html> |

Div with id as "loader" is used to define the loader image used for the Ajax call. Synchronization is done in JS file.

<stripes:form> is used for mapping the action class to the JSP.(in this case ExampleActionBean.java)

<stripes:select> is used for displaying dropdown and its ‘name’ property is used to map the value selected from the drop-down to the variable with the same name in the action class.

<stripes:options-collection> is used for displaying the options for the corresponding drop-down.

<stripes:submit> is used for submitting the page.

1. After submitting welcome page, the following page is displayed. (result.jsp)

|  |
| --- |
| <%@ page contentType="text/html;charset=UTF-8" language="java"%>  <%@ taglib prefix="stripes"  uri="http://stripes.sourceforge.net/stripes.tld"%>  <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>  <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">  <html>  <head>  <meta http-equiv="Content-Type" content="text/html; charset=utf-8">  <script type="text/javascript" src="js/jquery.js"></script>  <script type="text/javascript" src="js/jquery-ui-custom-min.js"></script>  <script src="js/ui/jquery.ui.core.js"></script>  <script src="js/ui/jquery.ui.widget.js"></script>  <script src="js/ui/jquery.ui.button.js"></script>  <title>Booking Successful!!</title>  </head>  <body bgcolor="#E0EEE0">  <h1 align="center" id="heading" class="heading">Booking  Successful!!</h1>  <stripes:form action="/Example.action">  <h2 align="center" style="color: #C6AEC7">Your Booking is  confirmed for <c:out value="${actionBean.exampleBean.place}"></c:out>,  <c:out value="${actionBean.exampleBean.country}"></c:out> by <c:out  value="${actionBean.exampleBean.meansOfTravel}"></c:out> at <c:out  value="${actionBean.exampleBean.availableOptions}"></c:out>!!!</h2>  </stripes:form>  </body>  </html> |

**Step 5:** Create a folder js under WebContent and add the following JS file to it.

|  |
| --- |
| $(document).ready(function(){  // to display wait logo while Ajax query is sent  $("#loader").bind("ajaxSend", function() {  $(this).show();  $("#parent").mask();  }).bind("ajaxStop", function() {  $(this).hide();  $("#parent").unmask();  }).bind("ajaxError", function() {  $(this).hide();  });    // to populate places when country is selected  $("select#country").change(function(){  $.getJSON("Example.action?getPlaceAjax=",{'exampleBean.country': $(this).val(), Ajax: 'true'}, function(places){  var options ='<option value="">--Select--</option>';    // to populate options in the place drop down  for (var i = 0; i < places.length; i++) {  options += '<option title = "' + places[i] +'"value="' + places[i]+ '">' + places[i] + '</option>';  }    $("select#place").html(options);  }).success(function() { })  .error(function(jqXHR, textStatus, errorThrown) {  alert("error " + textStatus);  alert("incoming Text " + jqXHR.responseText);  }).complete(function() {  });  });    // to populate means-of-travel list when place is selected  $("select#place").change(function(){  $.post("Example.action?getMeansOfTravelPost",{'exampleBean.place': $(this).val()},function() {    }) .complete(function() {  $.getJSON("Example.action?getMeansOfTravelAjax=",{Ajax: 'true'}, function(meansOfTravelList){  var options ='<option value="">--Select--</option>';    // to populate options in the means-of travel drop down  for (var i = 0; i < meansOfTravelList.length; i++) {  options += '<option title = "' + meansOfTravelList[i] +'"value="' + meansOfTravelList[i]+ '">' + meansOfTravelList[i] + '</option>';  }    $("select#meansOfTravel").html(options);  });  }).error(function(jqXHR, textStatus, errorThrown) {  alert("error " + textStatus);  alert("incoming Text " + jqXHR.responseText);  });  });    // to populate time slot list when means-of-travel is selected  $("select#meansOfTravel").change(function(){  $.getJSON("Example.action?getAvailableOptionsAjax=",{'exampleBean.meansOfTravel': $(this).val(), Ajax: 'true'}, function(availableOptionList){  var options ='<option value="">--Select--</option>';    // to populate options in the time slot drop down  for (var i = 0; i < availableOptionList.length; i++) {  options += '<option title = "' + availableOptionList[i] +'"value="' + availableOptionList[i]+ '">' + availableOptionList[i] + '</option>';  }    $("select#availableOptions").html(options);  }).success(function() { })  .error(function(jqXHR, textStatus, errorThrown) {  alert("error " + textStatus);  alert("incoming Text " + jqXHR.responseText);  }).complete(function() {  });  });    }); |

In the above JS file demonstrates two ways which can be used to get data from back-end:

1. getJSON directly (country selected is passed as a variable and place list is received by calling the function in the action bean). This approach works fine if String passed has less characters (<100) but if the String passed is large, the desired results are not obtained and the Ajax call behaves unpredictably.
2. getJSON and post (place selected is passed through a post call and then the action bean function is called using getJSON to get the meansOfTravelList). This approach is better as large Strings and lists can be sent using post call.

**Step 6:** Create a package named ExamplePackage and add the below class to it (ExampleActionBean.java).

|  |
| --- |
| package ExamplePackage;  import java.io.StringReader;  import java.util.ArrayList;  import java.util.List;  import Bean.ExampleBean;  import net.sourceforge.stripes.action.ActionBean;  import net.sourceforge.stripes.action.ActionBeanContext;  import net.sourceforge.stripes.action.DefaultHandler;  import net.sourceforge.stripes.action.ForwardResolution;  import net.sourceforge.stripes.action.HandlesEvent;  import net.sourceforge.stripes.action.Resolution;  import net.sourceforge.stripes.action.SessionScope;  import net.sourceforge.stripes.action.StreamingResolution;  import net.sourceforge.stripes.action.UrlBinding;  @UrlBinding("/Example.action")  @SessionScope  public class ExampleActionBean implements ActionBean {  public ExampleBean exampleBean;  public ActionBeanContext context;  public ExampleBean getExampleBean() {  return exampleBean;  }  public void setExampleBean(ExampleBean exampleBean) {  this.exampleBean = exampleBean;  }  @Override  public ActionBeanContext getContext() {  return context;  }  @Override  public void setContext(ActionBeanContext context) {  this.context = context;  }  @DefaultHandler  public Resolution selectionCountry() {  // creating instance of the bean  exampleBean = new ExampleBean();  //populating country list  List<String> countries = new ArrayList<String>();  countries.add("India");  countries.add("Pakistan");  countries.add("Bangladesh");  countries.add("Sri Lanka");  exampleBean.setCountries(countries);  return new ForwardResolution("/jsp/welcome.jsp");  }  @HandlesEvent("showResult")  public Resolution showResult() {  // forwarding to result page  return new ForwardResolution("/jsp/result.jsp");  }  public Resolution getPlaceAjax() {  //populating place list based on country selected  String countrySelected = exampleBean.getCountry();  List<String> places = new ArrayList<String>();  if (countrySelected.equalsIgnoreCase("India")) {  places.add("Mumbai");  places.add("Delhi");  places.add("Bangalore");  places.add("kolkata");  places.add("Chennai");  }  if (countrySelected.equalsIgnoreCase("Pakistan")) {  places.add("Karachi");  places.add("Lahore");  places.add("Peshawar");  places.add("Islamabad");  places.add("Rawalpindi");  }  if (countrySelected.equalsIgnoreCase("Bangladesh")) {  places.add("Dhaka");  places.add("Comilla");  places.add("Chitagong");  places.add("Rangpur");  places.add("Jessore");  }  if (countrySelected.equalsIgnoreCase("Sri Lanka")) {  places.add("Columbo");  places.add("Kotte");  places.add("Dambulla");  places.add("Anuradhapura");  places.add("Negombo");  }  exampleBean.setPlaces(places);  String jsonResult = new flexjson.JSONSerializer().serialize(exampleBean  .getPlaces());  return new StreamingResolution("application/javascript", jsonResult);  }  public Resolution getMeansOfTravelAjax() {  // populating means of travel list based on country selected  List<String> meansOfTravelList = new ArrayList<String>();  if (!exampleBean.getCountry().equalsIgnoreCase("Sri Lanka")) {  meansOfTravelList.add("Buses");  meansOfTravelList.add("Trains");  }  meansOfTravelList.add("Flights");  if (exampleBean.getCountry().equalsIgnoreCase("Sri Lanka")) {  meansOfTravelList.add("Ships");  }  exampleBean.setMeansOfTravelList(meansOfTravelList);  String jsonResult = new flexjson.JSONSerializer().serialize(exampleBean  .getMeansOfTravelList());  return new StreamingResolution("application/javascript", jsonResult);  }  public Resolution getMeansOfTravelPost() {  StringBuilder message = new StringBuilder();  message = message.append(" ");  return new StreamingResolution("text/html", new StringReader(message  .toString()));  }  public Resolution getAvailableOptionsAjax() {  // populating time slot list  List<String> availableOptions = new ArrayList<String>();  availableOptions.add("09:00 am");  availableOptions.add("12:00 pm");  availableOptions.add("03:00 pm");  availableOptions.add("06:00 pm");  availableOptions.add("09:00 pm");  availableOptions.add("12:00 am");  exampleBean.setAvailableOptionList(availableOptions);  String jsonResult = new flexjson.JSONSerializer().serialize(exampleBean  .getAvailableOptionList());  return new StreamingResolution("application/javascript", jsonResult);  }  } |

### Files to be added for jQuery:

JS files to be added to js folder:

1. jquery-ui-custom-min.js
2. jquery.center.js
3. jquery.js
4. jquery.loadmask.min.js
5. jquery.ui.core.js
6. jquery.ui.widget
7. jquery.ui.button (These files can be downloaded)

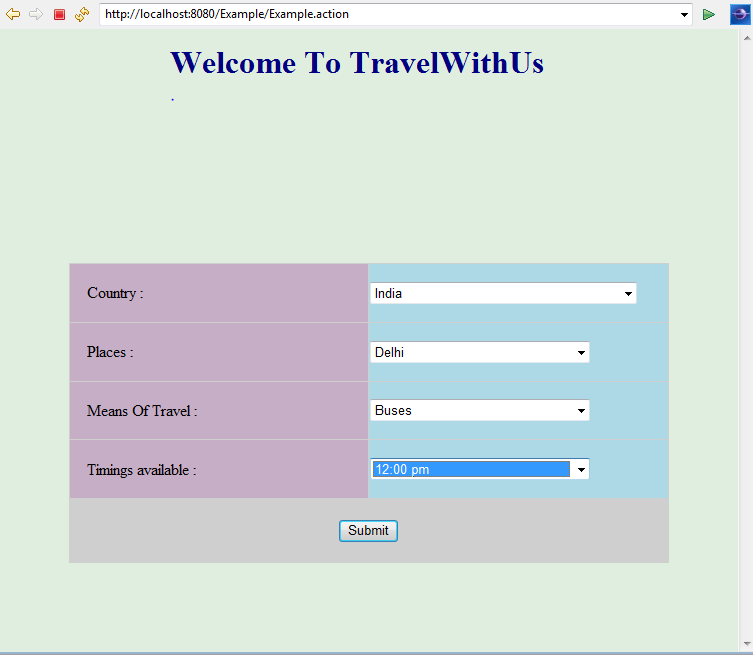
Image to be added:



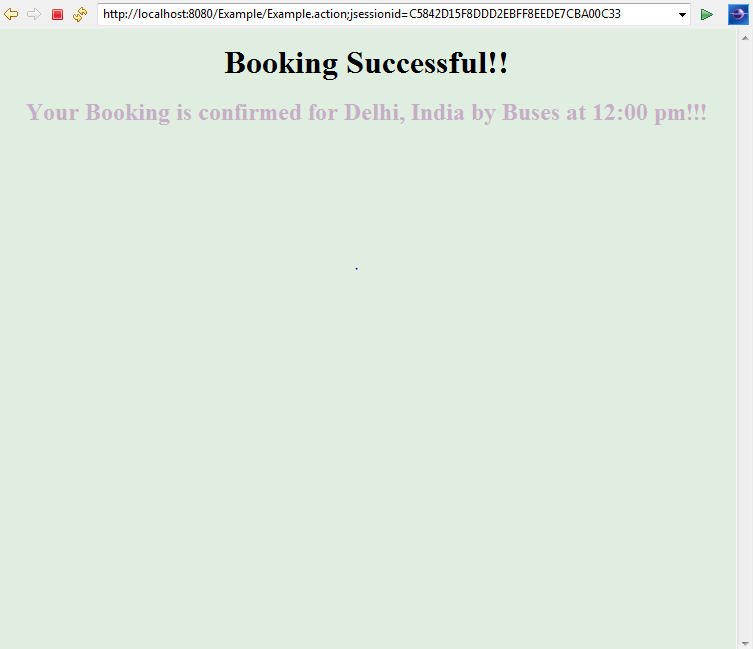
(Create a folder images under WebContent and add the above image to it. Add the files present in first folder to JS folder)

### Screen-Shots:

1. Home page for TravelWithUs:



1. Confirmation page:



### Conclusion:

Thus, a web application can be created using Stripes Framework tag libraries, AJAX and jQuery.

### Reference(s)

1. “Stripes and Java Web Development Is Fun Again” by Frederic Daoud. (2009)