

Getting the Request and Response Objects

Uses for request object

- Explicit session manipulation
E.g., changing inactive interval or invalidating session
- Explicit cookie manipulation (e.g., long-lived cookies)
- Reading request headers (e.g., User-Agent)
- Looking up requesting host name

Uses for response object

- Setting status codes
- Setting response headers
- Setting long-lived cookies

We Use Static methods to Get Request/Response Objects

```
ExternalContext context =  
    FacesContext.getCurrentInstance().getExternalContext();  
HttpServletRequest request =  
    (HttpServletRequest)context.getRequest();  
HttpServletResponse response =  
    (HttpServletResponse)context.getResponse();
```

Practice

1. Collect a search string and a search engine name, show the results of a search with that search engine.
2. Input form
 1. Use textfield for arbitrary search string.
 2. Use drop down menu to list only search engines that app supports.

CDI bean

3. Construct a URL by concatenating a base URL (e.g.,

http://www.google.com/search?q=) with the URLEncoded search string

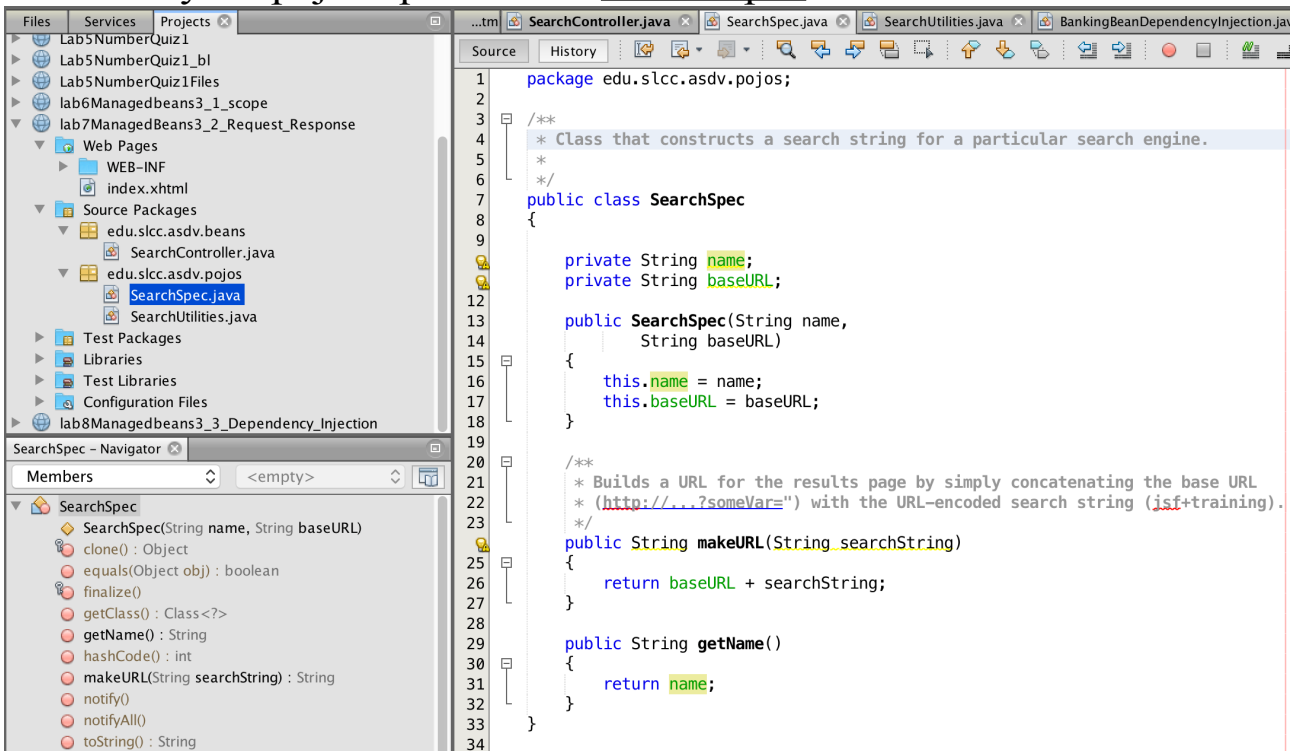
4. Do response.sendRedirect

Must use static methods to get the HttpServletResponse

Return normal strings for error pages

1. Create a new Web App, named lab7ManagedBeans3_2_Request_Response

2. Under your pojos place class SearchSpec



3. Under pojos place class SearchUtilities

```

1 package edu.sbcc.asdv.pojos;
2 import ...
3 /** Utility with static method to build a URL for any of the most popular search ...6 lines */
4 public class SearchUtilities
5 {
6     private SearchUtilities(){} // Uninstantiatable class
7
8     private static SearchSpec[] commonSpecs =
9     {
10         new SearchSpec("Google", "https://www.google.com/#q="),
11         new SearchSpec("Yahoo", "https://search.yahoo.com/search?p="),
12         new SearchSpec("Bing", "http://www.bing.com/search?q="),
13         new SearchSpec("DuckDuckGo", "https://duckduckgo.com/?q=")
14     };
15     private static List<String> searchEngineNames;
16     static{
17         searchEngineNames = new ArrayList();
18         for (SearchSpec spec : commonSpecs)
19             searchEngineNames.add(spec.getName());
20     }
21     public static SearchSpec[] commonSearchSpecs(){return commonSpecs;}
22     public static List<String> searchEngineNames(){return searchEngineNames;}
23
24     /** Given a search engine name and a search string, builds a URL for the ...5 lines */
25     public static String makeURL(String searchEngineName,
26         String searchString)
27     {
28         String searchURL = null;
29         for (SearchSpec spec : commonSpecs)
30         {
31             if (spec.getName().equalsIgnoreCase(searchEngineName))
32             {
33                 searchURL = spec.makeURL(searchString);
34                 break;
35             }
36         }
37         return (searchURL);
38     }
39 }

```

3.

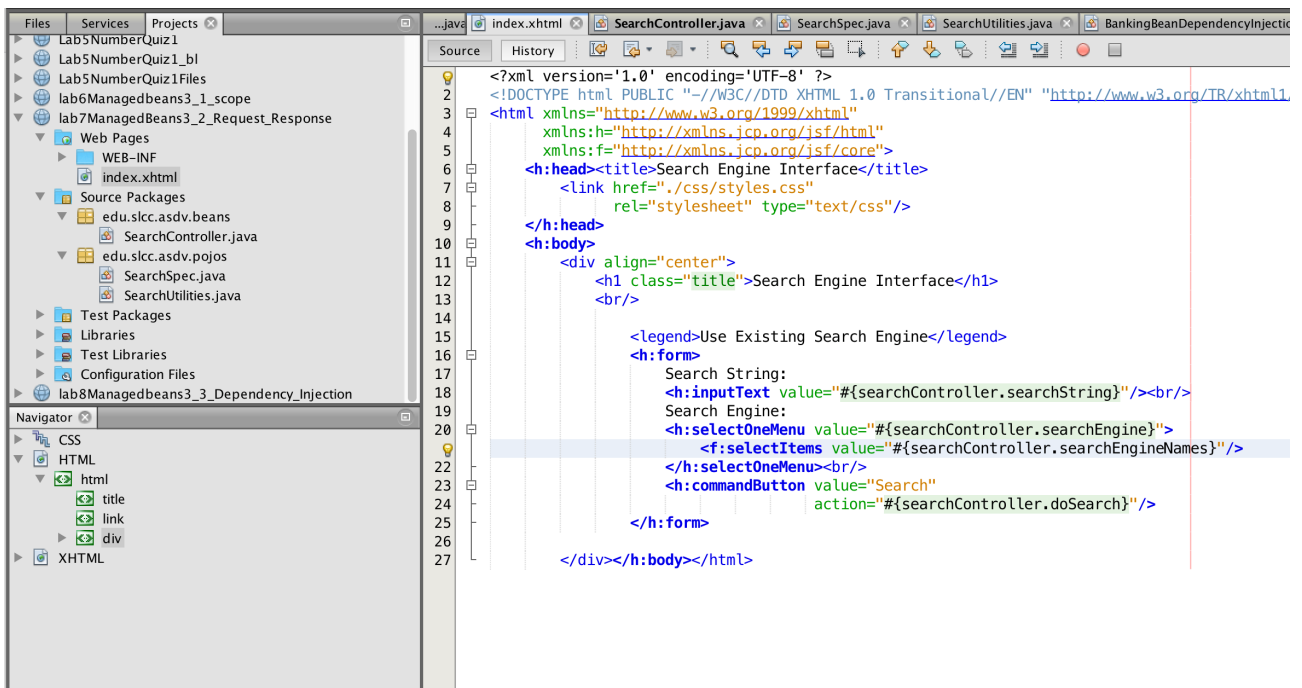
4. Under beans place the Managed bean SearchController

```

1 package edu.sbcc.asdv.beans;
2 import ...9 lines
3
4 /** Takes a search string and a search engine name, sending the query to that ...6 lines */
5 @Named(value = "searchController")
6 @RequestScoped
7 public class SearchController
8 {
9     private String searchString = "";
10    private String searchEngine;
11    public SearchController(){return searchString;}
12    public void setSearchString(String searchString){
13        this.searchString = searchString.trim();
14    }
15    public String getSearchEngine(){return searchEngine;}
16    public void setSearchEngine(String searchEngine){ this.searchEngine = searchEngine;}
17    public List<String> getSearchEngineNames(){return SearchUtilities.searchEngineNames();}
18    /** The URLEncoder changes spaces to "+" signs and other non-alphanumeric ...11 lines */
19    public String doSearch() throws IOException
20    {
21        if (searchString.isEmpty())
22            return "no-search-string";
23        searchString = URLEncoder.encode(searchString, "utf-8");
24        String searchURL
25            = SearchUtilities.makeURL(searchEngine, searchString);
26        if (searchURL != null)
27        {
28            ExternalContext context
29                = FacesContext.getCurrentInstance().getExternalContext();
30            HttpServletResponse response
31                = (HttpServletResponse) context.getResponse();
32            response.sendRedirect(searchURL);
33            return null;
34        }
35        else return ("unknown-search-engine");
36    }
37 }

```

5. Your index.html that has the form



```
<?xml version='1.0' encoding='UTF-8' ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://xmlns.jcp.org/jsf/html"
      xmlns:f="http://xmlns.jcp.org/jsf/core">
<h:head><title>Search Engine Interface</title>
<link href="/css/styles.css"
      rel="stylesheet" type="text/css"/>
</h:head>
<h:body>
<div align="center">
<h1 class="title">Search Engine Interface</h1>
<br/>
<legend>Use Existing Search Engine</legend>
<h:form>
  Search String:
  <h:inputText value="#{searchController.searchString}"/><br/>
  Search Engine:
  <h:selectOneMenu value="#{searchController.searchEngine}">
    <f:selectItems value="#{searchController.searchEngineNames}"/>
  </h:selectOneMenu><br/>
  <h:commandButton value="Search"
    action="#{searchController.doSearch}"/>
</h:form>
</div></h:body></html>
```

6. Clean and build, Run.