**Find Me Or Not Design and Implementation**

**Requisites:**

* Eclipse Oxygen IDE/STS
* Tomcat Server 9.0/8.0
* Java 1.6/1.8
* REST API
* AWS

**Description:**

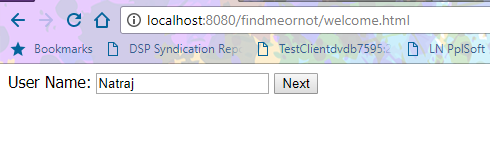
As an user, I will be entering my name in the login screen after which I will be taken to home screen. In the home screen, I will be entering an animal name and on clicking Search/Find, I will be taken to the results screen where I can see a message that the Animal I entered is found or NOT.

**Acceptance Criteria:**

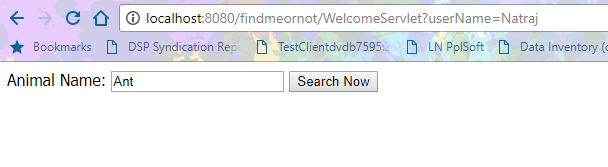
* The welcome screen should accept a name in a text box and login button in shown.
* On clicking the login button, the Search screen should be displayed.
* The Search screen should have a text box to accept ONLY alphabets which is the input (Animal Name).
* On entering empty value, error message should be displayed stating “Animal Name should not be empty”.
* On entering special characters or numerical values, error message should be displayed stating “Animal Name should not contain special characters or numbers”.
* On successful input value matching the input file value, proper message should be displayed to the user as “Hey <login name>, The animal <animal name> is Found”.
* Failure to find the input value from the input file, proper message should be displayed to the user as “Hey <login name>, The animal <animal name> is Found”.

**Demo/Output:**

**Welcome Screen:**



**Home Screen:**

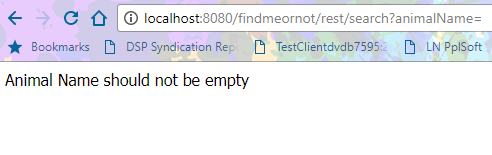


**Results Screen:**

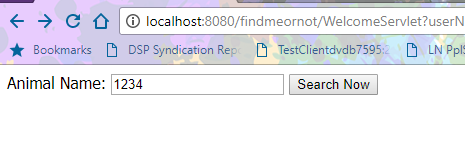


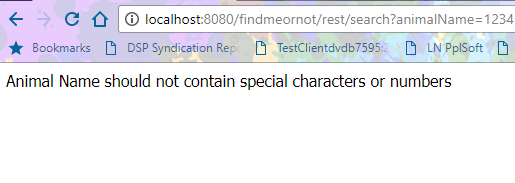
**Validations:**

When Animal name is Empty:



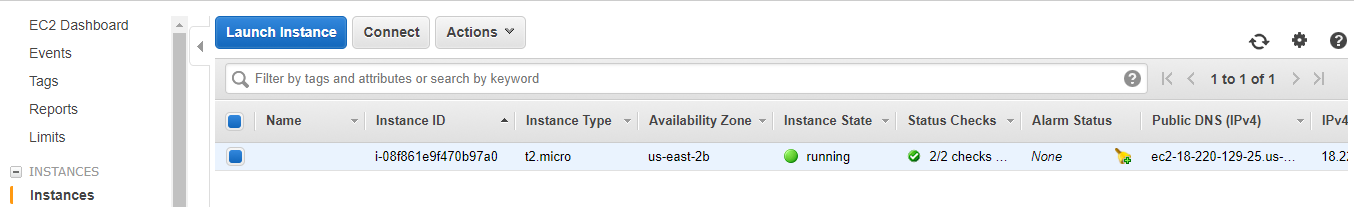
When the Animal Name has numerals:





**Implementation in AWS:**

1. Used the existing account in AWS for initializing an EC2 instance (The free tier one – Amazon AMI instance)
2. Used the instance type as t2.micro
3. Chose the default VPC and a public subnet in there.
4. Added the storage as Magnetic one since that’s the low cost one.
5. In security group, added an entry for tomcat’s 8080 port and source as anywhere since I am not sure about the user’s IP details.
6. Since this is first time, I had to create a new key-value pair of private key (.pem)



1. Since there is no Tomcat in AWS, I had to manually login into the IP address and install the Tomcat Server.
2. Once the instance is started, we need to start the tomcat server in the same host as well.
3. URL – **http:// ec2-18-220-129-25.us-east-2.compute.amazonaws.com:8080/findmeornot/welcome.html**

**Issues Faced:**

1. Referred google a lot of times when facing issues.
2. Spent more time in recalling J2EE concepts.
3. Many times I was getting 404 error when trying to hit the local server. The reason was web.xml mapping was incorrect. I used <https://dzone.com/articles/restful-web-services-java>

As reference to overcome those kind of issues.

1. Referred list of animals from <https://en.wikipedia.org/wiki/List_of_animal_names>
2. I had to really concentrate on list of jars added. While starting the server I got so much errors because of the same I have added many dependency jars. Just listing 2 errors alone.

Error:

|  |
| --- |
| java.lang.NoClassDefFoundError: org/glassfish/hk2/api/ServiceLocatorFactory  at org.glassfish.jersey.inject.hk2.AbstractHk2InjectionManager.<clinit>(AbstractHk2InjectionManager.java:75)  at org.glassfish.jersey.inject.hk2.Hk2InjectionManagerFactory$Hk2InjectionManagerStrategy$1.createInjectionManager(Hk2InjectionManagerFactory.java:79)  at org.glassfish.jersey.inject.hk2.Hk2InjectionManagerFactory.create(Hk2InjectionManagerFactory.java:97)  at org.glassfish.jersey.internal.inject.Injections.createInjectionManager(Injections.java:93)  at org.glassfish.jersey.server.ApplicationHandler.<init>(ApplicationHandler.java:282)  at org.glassfish.jersey.servlet.WebComponent.<init>(WebComponent.java:335)  at org.glassfish.jersey.servlet.ServletContainer.init(ServletContainer.java:178)  at org.glassfish.jersey.servlet.ServletContainer.init(ServletContainer.java:370)  at javax.servlet.GenericServlet.init(GenericServlet.java:158)  at org.apache.catalina.core.StandardWrapper.initServlet(StandardWrapper.java:1174)  at org.apache.catalina.core.StandardWrapper.loadServlet(StandardWrapper.java:1090)  at org.apache.catalina.core.StandardWrapper.load(StandardWrapper.java:980)  at org.apache.catalina.core.StandardContext.loadOnStartup(StandardContext.java:4829)  at org.apache.catalina.core.StandardContext.startInternal(StandardContext.java:5139)  at org.apache.catalina.util.LifecycleBase.start(LifecycleBase.java:183)  at org.apache.catalina.core.ContainerBase$StartChild.call(ContainerBase.java:1425)  at org.apache.catalina.core.ContainerBase$StartChild.call(ContainerBase.java:1415)  at java.util.concurrent.FutureTask.run(FutureTask.java:266)  at org.apache.tomcat.util.threads.InlineExecutorService.execute(InlineExecutorService.java:75)  at java.util.concurrent.AbstractExecutorService.submit(AbstractExecutorService.java:134)  at org.apache.catalina.core.ContainerBase.startInternal(ContainerBase.java:941)  at org.apache.catalina.core.StandardHost.startInternal(StandardHost.java:839)  at org.apache.catalina.util.LifecycleBase.start(LifecycleBase.java:183)  at org.apache.catalina.core.ContainerBase$StartChild.call(ContainerBase.java:1425)  at org.apache.catalina.core.ContainerBase$StartChild.call(ContainerBase.java:1415)  at java.util.concurrent.FutureTask.run(FutureTask.java:266)  at org.apache.tomcat.util.threads.InlineExecutorService.execute(InlineExecutorService.java:75)  at java.util.concurrent.AbstractExecutorService.submit(AbstractExecutorService.java:134)  at org.apache.catalina.core.ContainerBase.startInternal(ContainerBase.java:941)  at org.apache.catalina.core.StandardEngine.startInternal(StandardEngine.java:258)  at org.apache.catalina.util.LifecycleBase.start(LifecycleBase.java:183)  at org.apache.catalina.core.StandardService.startInternal(StandardService.java:422)  at org.apache.catalina.util.LifecycleBase.start(LifecycleBase.java:183)  at org.apache.catalina.core.StandardServer.startInternal(StandardServer.java:770)  at org.apache.catalina.util.LifecycleBase.start(LifecycleBase.java:183)  at org.apache.catalina.startup.Catalina.start(Catalina.java:671)  at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)  at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)  at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)  at java.lang.reflect.Method.invoke(Method.java:498)  at org.apache.catalina.startup.Bootstrap.start(Bootstrap.java:355)  at org.apache.catalina.startup.Bootstrap.main(Bootstrap.java:495)  Caused by: java.lang.ClassNotFoundException: org.glassfish.hk2.api.ServiceLocatorFactory  at org.apache.catalina.loader.WebappClassLoaderBase.loadClass(WebappClassLoaderBase.java:1275)  at org.apache.catalina.loader.WebappClassLoaderBase.loadClass(WebappClassLoaderBase.java:1104)  ... 42 more |

Jar - hk2-api-2.5.0-b42.jar

Error:

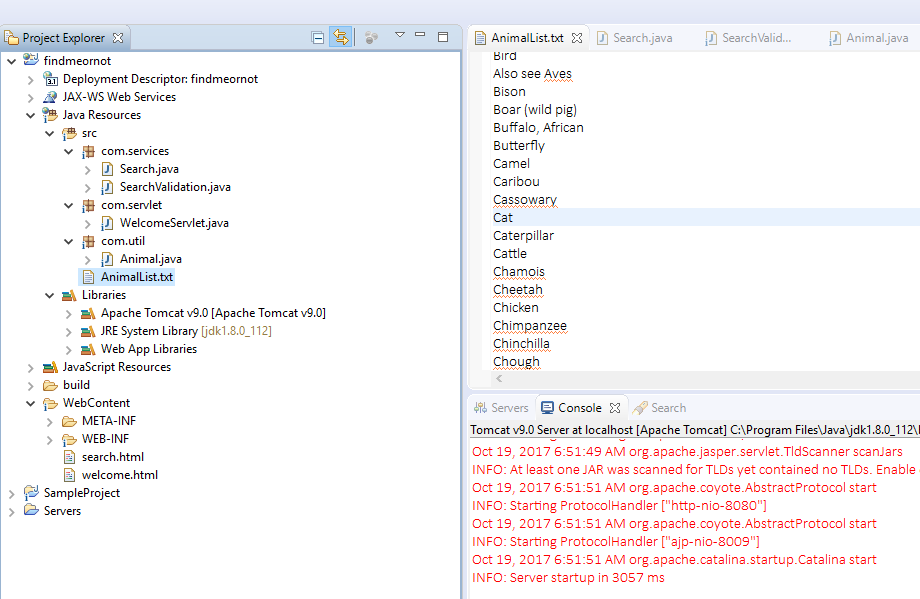
|  |
| --- |
| SEVERE: StandardWrapper.Throwable  java.lang.IllegalStateException: No generator was provided and there is no default generator registered  at org.glassfish.hk2.internal.ServiceLocatorFactoryImpl.internalCreate(ServiceLocatorFactoryImpl.java:308)  at org.glassfish.hk2.internal.ServiceLocatorFactoryImpl.create(ServiceLocatorFactoryImpl.java:268)  at org.glassfish.jersey.inject.hk2.AbstractHk2InjectionManager.createLocator(AbstractHk2InjectionManager.java:114)  at org.glassfish.jersey.inject.hk2.AbstractHk2InjectionManager.<init>(AbstractHk2InjectionManager.java:86)  at org.glassfish.jersey.inject.hk2.ImmediateHk2InjectionManager.<init>(ImmediateHk2InjectionManager.java:62)  at org.glassfish.jersey.inject.hk2.Hk2InjectionManagerFactory$Hk2InjectionManagerStrategy$1.createInjectionManager(Hk2InjectionManagerFactory.java:79)  at org.glassfish.jersey.inject.hk2.Hk2InjectionManagerFactory.create(Hk2InjectionManagerFactory.java:97)  at org.glassfish.jersey.internal.inject.Injections.createInjectionManager(Injections.java:93)  at org.glassfish.jersey.server.ApplicationHandler.<init>(ApplicationHandler.java:282)  at org.glassfish.jersey.servlet.WebComponent.<init>(WebComponent.java:335)  at org.glassfish.jersey.servlet.ServletContainer.init(ServletContainer.java:178)  at org.glassfish.jersey.servlet.ServletContainer.init(ServletContainer.java:370)  at javax.servlet.GenericServlet.init(GenericServlet.java:158)  at org.apache.catalina.core.StandardWrapper.initServlet(StandardWrapper.java:1174 |

Jar - hk2-locator-2.5.0-b42.jar

1. The simplest call I have done in the past in the GET instead of POST call. That’s one reason I used query param and not form param.
2. I thought of doing javascript validation since that’s an important one as it is on client side but I could not achieve it, so used the regular Java side Validation API to validate.
3. Here is the animal list text file I used in the project.



1. Sample screenshot of the code:



1. Spent some 3 hours in AWS to learn how to launch tomcat manually in AWS through google and also after launching I was unable to see the end result because of the following facts:
   1. The jersey jar I was using required java 1.8 but the box had 1.7. So updated to java 1.8
   2. Used catliana.sh start instead of regular sh start.sh