## Solution for Datastore

In phase one we weren’t required to use the datastore so we simply imported the .xml file into the *elementTree*  data structured and spewed it back out with the export button. Now that we were required to use GAE datastore we had to figure out a model diagram for us to store the data imported and later be able to retrieve and export from the GAE models. We decided that we were going to let organization, person and crisis have their own class and then have them be parents of other subclasses that are required in the schema. This would also allow for the each Organization/Person/Crisis to have a many to many relation with each other. The crisis class used location, impact, extension and info as a subclass which then each had their own smaller subclasses. The organization class only used Info and Ref as a subclass. Organization’s Ref subclass was very similar to Crisis’ extension subclass in that it had similar subclasses with the exact same attributes. Even though we could have joined them in one class and have Organization/Peron/Crisis have separated associations we decided to have each “Parent” class have its own separate hierarchy. We did this in order to make the importing of the xml data to these models much easier to iterate and save to the datastore. The person class used only Info and Ref as a subclass with Info using Brithdate as a subclass and Ref being the same as OrganizationRef. Once our models were set we then used *ElementTree* to go through the imported xml and create and save our class class instances into the datastore. ….

To export from the datastore was a bit trickier…

## Solution for Import/Export

With the help of *elementTree* we were able to import our .xml file into a treedata structure and then simply print it back out with the .out.write() function giving it the imported xml file back in a string format. At first we had trouble on how it was being displayed in the browser but found an easy fix by setting the .content\_type equal to “text/xml” from the hint on the project page. Since we weren’t supposed to use the dataStore in the first phase we were able to accomplish this fairly easily. By saving the tree in a global variable and using this tree across our import/export functions we were able to output the xml that we imported as long as we did not close the browser. If you were to import the xml and then close the browser, the export function would not have worked in a new browser since there was no global variable initiated in that instance. In the second phase we were to use the dataStore and save the imported xml into GAE models that were then to be saved to the dataStore. We saved our models similar to how we setup our xml file by keeping the same hierarchy in order to make the importing to the models much smoother and easier to read. Now, whenever we import an xml file it saved and we can close the browser and export