# *Web Programming III (420-C30-HR)*

# *Assignment 4 – Bug Tracking System*

Date assigned: Thursday, November 17, 2016

Date due: **Wednesday, December 7, 2016 (last day of classes)**

**Learning Objectives**

Upon successful completion of this assignment, the student will be able to:

* Create a web application using an All-in-One form for PHP

Overview

You are working in support for Santa’s List Inspection and Maintenance (SLIM). You are going to create a system to track the boys and girls on Santa’s naughty and nice lists. This system will have a server component running PHP which is used by the elves to create new and update existing list items. This will be done using an all-in-one page on CSDEV (the O drive). And YES, it MUST run on the O drive.

Santa and Mrs. Claus want to be able to view the list information by selecting the information from a list. They use a windows desktop browser or mobile phone (responsive design required) to access the lists. Their “dashboard” displays a list of the current good boys and girls and allows them to select one to display details. Santa can NOT make changes to the list (he doesn’t know how to type).

Santa is accessing a node.js web server for all his web needs. His “dashboard” must update automatically every 30 seconds to display the current good girls and boys.

The Application

1. There are obviously many multiple parts to this application involving PHP, node and AJAX (and XML or JSON too). Break things down to make it easier…

**PHP**

1. Child creating/editing. Create an all-in-one PHP form to create, update and save children to the naughty and nice list (list described later). This file should be called manageList.php and must run on the O drive in the folder O:\c30A04PHP\manageList.php. Boys and girls are stored in a SINGLE XML or JSON file (your choice) called SantaList.xml (or .json) in the folder O:\c30A04PHP\List. The user must be able to add a new person or select an existing person and update his or her information.
2. Return list. Create a PHP file called getListInfo.php in the same folder as above which will return a list of all the members of the list based on the requested value (naughty, nice, unknown, limbo). This means that getListInfo.php has one parameter passed either through a get or post action. The parameter is called “which” and has the value naughty, nice, unknown or limbo. This list consists of an XML or JSON response object. The fields returned are: person id, first name, last name, city, date updated. This program has no screen interface and simply returns an HTTP response object (see getAlbums.php from assignment 1).
3. Return details. Create a PHP file called getDetailInfo.php which returns all the information about a single (requested) person from the list. This php program has one parameter passed through either a get or post which is the id of the person to return.

**Node**

1. Node web server. Create a node web server using port 7546 which will serve files from the folder. This must be called santaWeb.js and be located in the folder c30A04Node. The node web server will render pages from the public folder located below the current folder (c30A04Node\public). The default file will be index.html but any html, htm, jpg, png, gif, xml, pdf file in the folder or on a path from that folder can be returned.
2. Node request. When node receives a request to get a list of the children it will send a request (use the request module) to csdev.cegep-heritage.qc.ca/student/username/c30A04PHP/ getListInfo.php with the single parameter of naughty, nice, unknown or limbo. This request will return either an XML or JSON file which the node server will then format as an HTML page (using a template as desired) and return to the calling agent (Santa).
3. Node request. When node receives a request to get a specific child it will send a request (use the request module) to csdev.cegep-heritage.qc.ca/student/username/c30A04PHP/ getDetailInfo.php. This will return the details about a single person either as XML or JSON. The node server will then format the information as an HTML page (using a template as desired) and return to the calling agent.

NOTES FOR NODE:

* You must provide a package.json with all its dependencies so that all the appropriate modules will be downloaded using an npm -install

**AJAX**

1. Santa will be using an html page through the node server to access the information. There will be a file called myList.html in the public folder of on the node web server. When this file is displayed, it will run a JavaScript program which will call the node web server with an AJAX get (use jQuery if you wish (I certainly would)). The get to the node web server will make the web server call getListInfo.php on CSDEV to return the list. By default the nice list is returned, but Santa must be able to choose (from the returned page) which list to view (naughty, nice, unknown or limbo). The list should update every 30 seconds with the current information.
2. Santa will also have the option from the list page to select one person to view the details of. How he selects the person is up to you (a hyperlink on the person’s name would be the most effective, but it could be a checkbox, a select link, or even typing in the person’s id (YUCK!)). It is up to you to decide how it is done. When Santa chooses a person a request is made to the node web server that will make the node web server call the php program on csdev with getDetailInfo.php. The detailed information returned to the node server from the php server would then be displayed on Santa’s device.

NOTES FOR HTML/AJAX:

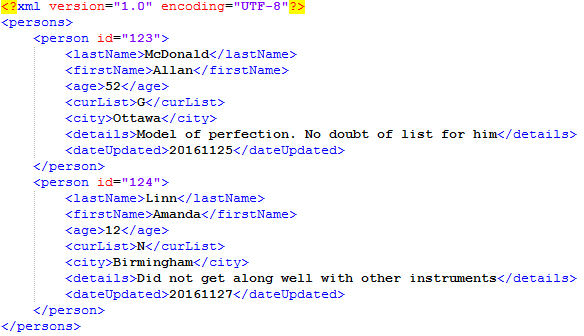
* Santa can use his phone or desktop depending on if he is in his office or the toyshop. This means that this page must be responsive at the 768px breakpoint. Feel free to use bootstrap or another library to do this.
* Colours would be nice for Santa…maybe different lists in different colours, etc.
* Santa’s information can all be done in one page; that is, the details of the chosen person and the list can be displayed at the same time. This should save you some logic/work.
* Remember that CSS and JavaScript must be kept in external files.

**Person Record**

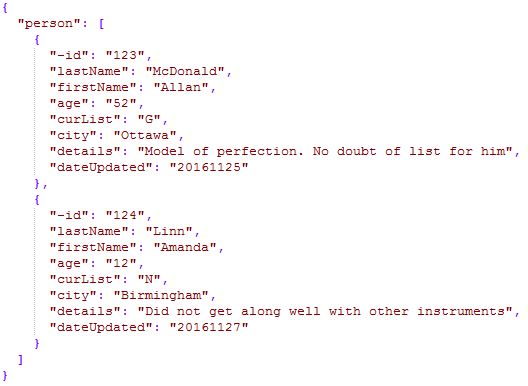
There needs to be a persons file. It can be either XML or JSON; as in persons.xml or persons.json. The following information must be collected for a person:

|  |  |  |
| --- | --- | --- |
| Item | Name | Description |
| Person Identifier | personID | A unique identifier for a person. This is an integer and generated by the system. The user never enters the ID and, indeed, should never see it |
| Last Name | lastName | The person’s last name |
| First Name | firstName | The person’s first name |
| Age | age | The person’s age |
| Current List | curList | A single character representing the list the person is on. The possible values are:   * G – Good * N – Naughty * L – Limbo, undecided which list * U – Unknown |
| City | city | The person’s city |
| Details | details | A string detailing reasons for the list the person is on. |
| Date Updated | dateUpdated | The last date the person was updated |

Here is a sample file in XML:



And the same data in JSON (notice there is not persons root entry):



I will provide a typical workflow soon to avoid confusion.

**To submit**

When you have completed the assignment, zip the files as YourUserName\_C30\_A03 and save the zip file to the page for the course.