PowerShell

To do these tasks you need to finish the following trainings from MVA.

1. Getting Started with Microsoft PowerShell
2. Advanced Tools & Scripting with PowerShell 3.0 Jump Start
3. PowerShell work with XML and Working with XML
4. PowerShell playing with JSON and save in JSON with PowerShell

1. From one XML file Create 10 XML files and update value of the line which defined in parameters:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="/">
  <html>
  <body>
  <h2>My CD Collection</h2>
  <table border="1">
    <tr bgcolor="#9acd32">
      <th>Title</th>
      <th>Artist</th>
    </tr>
    <tr>
      <td>.</td>
      <td>.</td>
    </tr>
  </table>
  </body>
  </html>
</xsl:template>

</xsl:stylesheet>
```

2. Take this JSON file and update value of the **'SortAs'** to **'OMPL'** and then save file.

```
{
 "glossary": {
  "title": "example glossary",
   "GlossDiv": {
    "title": "S",
     "GlossList": {
      "GlossEntry": {
        "ID": "SGML",
          "SortAs": "SGML",
          "GlossTerm": "Standard Generalized Markup Language",
          "Acronym": "SGML",
          "Abbrev": "ISO 8879:1986",
          "GlossDef": {
           "para": "A meta-markup language, used to create markup languages
such as DocBook.",
```

```
          "GlossSeeAlso": ["GML", "XML"]
           },
           "GlossSee": "markup"
       }
      }
     }
   }
}
```

3. Use **JSON** file to iterate in the loop and print out to get some keys and values to the all variables.

```
[
{
"RgName": "DOTcom-dev-rg",
"Name": "dev",
"AppName": "devse-cd"
},
{
"RgName": "DOTcom-dev-rg",
"Name": "dev-cm",
"AppName": "devse-cm"
},
{
"RgName": "DOTcom-uat-rg",
"Name": "uat",
"AppName": "uatse-cd"
},
{
"RgName": "DOTcom-uat-rg",
"Name": "uat-cm",
"AppName": "uatse-cm"
}
]
```

4. Check local file system if file will be there then print out something and "**elif**" file will not exists just, download this from some URL.

5. Check that variable is undefined then, set new value and print out, "**elif**" print out value (Check for both Null and Empty variable).

6. Check parameter "**boolian**" type and if it is ***true*** execute to get information about the system disks.

7. Create one script which will call another with named parameters from main script.

8. Create new **PSM** which will be called from main script with named parameters.

9. Create array and fill with values.

10. Create object and fill with values.