Name: Dergousoff, Kristen
NSID: kkd115
Project: WinMerge
Change Number: 6
Date: 04/04/2012
Group Number: Danger Twins

User Specified Backup File Extensions

1. **Change Request*:***
   The Options -> Backup Files dialog allows the user to "Append .bak -extension". Create a way for the user to specify extensions other than ".bak".

2. **Concept Location:**

Table 1. The list of all the classes visited during concept location.

#	File name	Tool used	Located?	Comments
1	OptionsDef.h	Find in File Tool Searched for "bak"	No **CHANGED**	This class has no backup logic but it seems to contain a list of options to be used in various UI's in the project, so we will most likely have to add our extension option into this list as well.
2	PropBackups.h	Same as above	No **CHANGED**	This class looks to model the Backup File options page so this would seem to be highly related to the concept location, but it is just a header file so it does not contain any logic. We will still need to add a new field to this class for the option page to handle our new extensions.
3	PropsBackup.cpp	Same as above	Yes **CHANGED**	This class contains all of the options and handles all of the Backup File commands. It also contains the methods that actually perform the backups, so this would seem to be our Concept Location!

3. **Impact Analysis:**

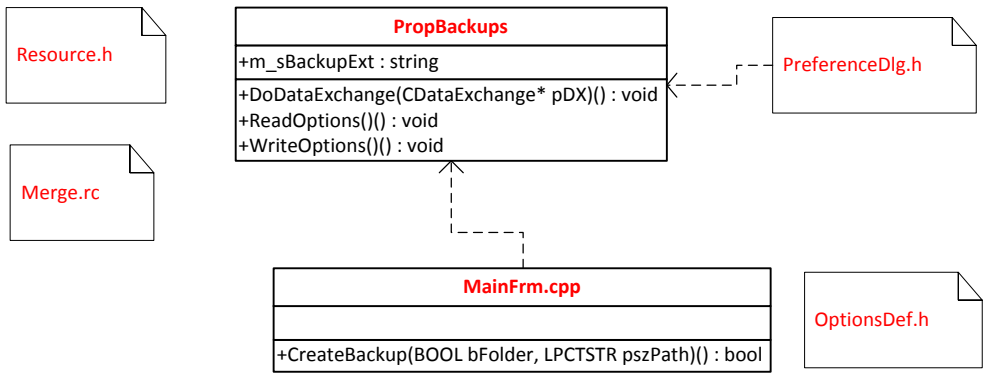
Table 2. The list of all the classes visited during impact analysis.

#	Class name	Tool used	Impacted?	Comments

1	PropBackups.cpp	Find in Files Searched for “bak”	Yes	Concept Location
2	Resource.h	Same as above	Yes	This class will need to be changed in order to add our new UI element to allow users to enter an arbitrary extension.
3	Merge.rc	Same as above	Yes	This class will also need to be changed in order to add our new UI element to the current UI.
4	MainFrm.cpp	Same as above	Yes	This class is what handles the actual processing of the form actions. This will need to be modified to use the extension submitted by the user if one was submitted.
5	PreferenceDlg.h	Find in Files Searched for “PropBackups”	No	This class contains an instance of the PropBackups class but does not use any internal methods of the class since it is just a header file.

4. Learning process:

In the following UML diagram, if a class contained a .cpp and a corresponding .h file with the same name, they were treated as a single object. (Ex. PropBackups.h and PropBackups.cpp were combined into PropBackups which outlines the main methods and attributes of both files)



5. Description of the implementation:

As has already been learnt from previous change requests, one of the first steps we need to take when adding a new UI element is to create a unique id for that element inside resource.h.

<resource.h>

```
#define IDC_COMPARE_QUICKC_LIMIT        1350
#define IDC_BACKUP_OPT_EXT               1351
```

The next step we need to take is to add our new element to the methods that handle the UI input. This will need to be done in the PropBackups.cpp class, but first we need to add a new field into the .h that can be used by the .cpp.

<PropBackups.h>

```
CString m_sGlobalFolder;
BOOL m_bAppendBak;
CString m_sBackupExt;
```

There are three main changes that need to be made in PropBackups.cpp. They are all in different places and so I will be separating the different areas of code with …

<PropBackups.cpp>

This first code snippet will bind the information from the submitted form to our new field that was added to our .h

```
void PropBackups::DoDataExchange(CDataExchange* pDX)
{
	CDialog::DoDataExchange(pDX);
	DDX_Check(pDX, IDC_BACKUP_FOLDERCMP, m_bCreateForFolderCmp);
	DDX_Check(pDX, IDC_BACKUP_FILECMP, m_bCreateForFileCmp);
	DDX_Text(pDX, IDC_BACKUP_FOLDER, m_sGlobalFolder);
	DDX_Check(pDX, IDC_BACKUP_APPEND_BAK, m_bAppendBak);
	DDX_Text(pDX, IDC_BACKUP_OPT_EXT, m_sBackupExt);
	DDX_Check(pDX, IDC_BACKUP_APPEND_TIME, m_bAppendTime);
	DDX_Radio(pDX, IDC_BACKUP_ORIGFOLD, m_nBackupFolder);
}
```

This class seems to grab the default values for each of the UI elements and sends them to the UI.

```
void PropBackups::ReadOptions()
{
	m_bCreateForFolderCmp = GetOptionsMgr()->GetBool(OPT_BACKUP_FOLDERCMP);
	m_bCreateForFileCmp = GetOptionsMgr()->GetBool(OPT_BACKUP_FILECMP);
	m_nBackupFolder = GetOptionsMgr()->GetInt(OPT_BACKUP_LOCATION);
	m_sGlobalFolder = GetOptionsMgr()-
>GetString(OPT_BACKUP_GLOBALFOLDER).c_str();
	m_bAppendBak = GetOptionsMgr()->GetBool(OPT_BACKUP_ADD_BAK);
```

```
        m_bAppendTime = GetOptionsMgr()->GetBool(OPT_BACKUP_ADD_TIME);
        m_sBackupExt = GetOptionsMgr()->GetString(OPT_BACKUP_OPT_EXT).c_str();
}
```

This next code snippet is the method that actually handles retrieving the user submitted value from the text field of our form, and stores it into our new field.

```
void PropBackups::WriteOptions()
{
        m_sGlobalFolder.TrimLeft();
        m_sGlobalFolder.TrimRight();
        if (m_sGlobalFolder.GetLength() > 3&&
                m_sGlobalFolder[m_sGlobalFolder.GetLength() - 1] != '\\')
        {
                m_sGlobalFolder += "\\";
        }

        GetOptionsMgr()->SaveOption(OPT_BACKUP_FOLDERCMP, m_bCreateForFolderCmp ==
TRUE);
        GetOptionsMgr()->SaveOption(OPT_BACKUP_FILECMP, m_bCreateForFileCmp ==
TRUE);
        GetOptionsMgr()->SaveOption(OPT_BACKUP_LOCATION, m_nBackupFolder);
        GetOptionsMgr()->SaveOption(OPT_BACKUP_GLOBALFOLDER, m_sGlobalFolder);
        GetOptionsMgr()->SaveOption(OPT_BACKUP_ADD_BAK, m_bAppendBak == TRUE);
        GetOptionsMgr()->SaveOption(OPT_BACKUP_ADD_TIME, m_bAppendTime == TRUE);
        GetOptionsMgr()->SaveOption(OPT_BACKUP_OPT_EXT, m_sBackupExt);
}
```

Once we have finished modifying PropBackups to handle the user input, we have to actually add our new element to the Backup File UI. This is done by editing the Merge.rc file as follows.

<Merge.rc>

```
BEGIN
    …
    CONTROL         "&Append .bak -extension",IDC_BACKUP_APPEND_BAK,"Button",
                    BS_AUTOCHECKBOX | WS_TABSTOP,17,99,212,10
        EDITTEXT            IDC_BACKUP_OPT_EXT,29,70,145,14,ES_AUTOHSCROLL |
WS_GROUP
    CONTROL         "A&ppend timestamp",IDC_BACKUP_APPEND_TIME,"Button",
                    BS_AUTOCHECKBOX | WS_TABSTOP,17,109,212,10
END
```

Now all that is left to do is to process our extension addition inside the class that actually processes and backs up the file.

<MainFrm.cpp>

```
BOOL success = FALSE;
if (GetOptionsMgr()->GetString(OPT_BACKUP_OPT_EXT).size() > 0)
{
        // If a custom backup extension was provided, use that instead of .bak
        // Don't add dot if there is no existing extension
        if (ext.size() > 0)
```

```
                ext += _T(".");
        ext += GetOptionsMgr()->GetString(OPT_BACKUP_OPT_EXT);
}
else if (GetOptionsMgr()->GetBool(OPT_BACKUP_ADD_BAK))
{
        // Don't add dot if there is no existing extension
        if (ext.size() > 0)
                ext += _T(".");
        ext += BACKUP_FILE_EXT;
}
```

We also found the OptionsDef.h file in our search for the Concept Location and it seemed to be a file that listed all of the form options. It included the original .bak option so we figured we would need to add our new extension option as well.

<OptionsDef.h>

```
const TCHAR OPT_BACKUP_ADD_BAK[] = _T("Backup/NameAddBak");
const TCHAR OPT_BACKUP_OPT_EXT[] = _T("Backup/NameAddExt");
```

6. **Sources:** No external sources used. Just knowledge gained from previous change requests.