Overview

The purpose of this project is to provide a streamlined experience to keep SalesForce and version control in synch through the following goals:

- Keep single location of code for SalesForce and Version Control
- Always keep Static Resources up to date and deployed alongside standard depoloyments
- Deploy to SFDC without modifying Version Control and commit when ready.
- Deploy to SFDC in 1/10 the time than eclipse
- Monitor changes in SalesForce against Version Control

As an aside:

The goal with this, and with all projects is to make things better.

If you notice something, have a suggestion or want to talk something over, please contact the team or proth@salesforce.com

Dependencies

- Apache Ant

(Optional)
* Git (a code repository tool)
* DiffMerge (a windows/mac code merge tool)

To get Setup

To use the local functionality goodness (like Package Lists / Smart Deploys / Templates / or configure for Automated Deployments), please see
README_Setup.md

-- However, if you are only interested in Automated Documentation, those steps are recommended but not necessary.

To setup Automated Documentation, please see
README_AutodocSetup.md

For more information about Automated Documentation please see
README_Autodoc.md

High level targets

NOTE: ANT IS CASE SENSITIVE
A list of all targets is below

```
ant test           - Tests your setup and credentials
ant help           - Minimal help
ant -p             - lists all targets
ant packageList    - Works with package lists
ant new            - Creates files based on templates
ant retrieveDeploy - High level target refreshing/deploying/storing
ant modifyPackage  - High level target working with packages
ant destroy        - High level target for destroying metadata
ant list           - High level target to list out metadata
ant git            - High level target using git on the source code.
ant auto           - AutoDeployment related Tasks
ant settings       - works with settings like test deploys.
ant doc            - Generates Documentation for all apex classes retrieved.
```

Everything you can do within the app can be reached under those commands.

Sample uses

(TODO) Update section with feedback

So, what else can you do with the SFDC Ant Project?

- You can add / subtract packages (or even add to them by a pattern - such as all classes that start with THD_)
- You can review changes not in version control and manage them in the team.
- You can do automated documentation
- You can run automated deployments
- You can refresh the entire org to see what changes are not in version control.
- You can refresh / deploy sets of files called 'packageLists'.
- You can work with Apex/VisualForce/JavaScript/CSS/SASS and bundle them as a packageList. Then combine/lint/uglify/zip/package/deploy all of them at once. (And

redeploy ever after pressing enter three times)

...

File Lists

Package Lists are all stored in the **packageList** folder and contain the list of metadata files that should be used together.

Examples could be a simple user story or assets used together for specific functionality.

For example **packageList/home.txt** could contain all the homepage components, controllers, visualforce pages and static resources used for the home page.

```
force/src/staticresources/TL_Assets.resource
force/src/pages/TL_Home.page
force/src/classes/TL_Home_CTRL.cls
```

All those assets can be refreshed from SalesForce by using the following command:

```
ant refreshFromList
```

This lets you refresh those assets in your version control to verify they are up to date. This also extracts any static resources to the resources directory so they are all version controlled as-well.

Simply make a change to those files in your files in **force/src** directory and then run

```
ant deployFromList
```

To compress all those resources back to the static resource files and deploy them all back up to the server.

Package

A package is stored at `force/src/package.xml` and should list everything that your project should contain.

To ensure that your version control is in synch with your SalesForce.com instance, simply use `ant refresh` to determine what changes are in SalesForce not in version control (note: uncommitted changes might be lost)

Environments / Credentials

An environment is configuration/credentials for an org to allow easy definition of where to retrieve/deploy.

By specifying 'ant createEnvironmentSettings' in each environment, the project automatically defaults the credentials to match the repository.

By running 'ant settings | retrieve', you specify the override where to retrieve from.

By running 'ant settings | deploy', you specify the override where to deploy to.

Using a blank value for 'ant settings' environment, retrieve or deploy, removes the override and allows the default to be used.

At any time, you can run 'ant status' to tell the current credentials used.

For more information, please read the Environment Hierarchy section below.

Environment Hierarchy

The `force.environment` specifies which environment you are currently on and the default environment to retrieve from and deploy to.

Properties are checked in the following order

1. **build.environment** will be used if specified
2. otherwise **build.properties** is checked
3. otherwise **resources/environment.settings** is checked

The `force.retrieve.environment` specifies which environment to use to retrieve. `force.deploy.environment` specifies which environment to use for deployments.

Both `force.retrieve.environment` and `force.deploy.environment` override the

`build.environment`

For example, the environment to use for deployment would be checked in the following order

1. `force.retrieve.environment` in build.environment
2. `force.retrieve.environment` in build.properties
3. `force.retrieve.environment` in resources/environment.settings
4. `build.environment` in build.environment
5. `build.environment` in build.properties
6. `build.environment` in resources/environment.settings

All Targets

```
addAllPackageTypes            Adds available types to the package

addDocMetadata                Adds in all metadata used for documentation using
wildcards
                                 (this allows retrieving ALL members, even new
ones)

addDocMetadataByName          Adds in all metadata used for documentation by
member names
                                 (this allows for removing items from the list
if desired)

addFileToPackageList          creates a packageList from a file

addPackageMember              adds a member from the package xml file

addPackageType                Retrieves a list of the current members of a type
for adding

addResourcesToDeploy          Adds multiple resources to the deployment

addStandardObjects            Adds the standard objects to the list of custom
objects

applyPackageListToPackage     Merges the package list into the current package

checkRetrieved

clean                         High level cleaning targets

cleanDeploy                   Cleans/Removes the deployment directory
```

```
cleanOutput                   cleans the documentation directory

compressResources             Uses the property lists to compress groups of
static   resources in a set

createRevisionList            creates a package list based on a range of
revisions

deploy                        Sends the deployment out

deployFromList                Creates a deployment from a deployment list

destroyFromPackageList        Creates a destructive change deployment from a
package list

doc                           SimpleTask - Creates documentation reports

docAll                        Runs all documentation reports

docCode                       Runs ApexDoc on the current set of apex classes

docObjects                    Run Report on objects

docProfiles                   Run Report on Profiles

docWorkbooks                  Create object workbooks

docWorkflows                  Run Report on Workflows

extractResources              Uses the property lists to compress groups of
static   resources in a set

git                           Runs a git command on the salesforce metadata
(used for option 1)

help                          Shows help information

list                          Lists different types of information

listFileChanges               lists the changes to a specific file for a
revision range

listMetadata                  Lists metadata types

listMetadataFiles             Lists all the items for a particular metadata type

listMetadataTypes             Lists all the items for a particular metadata type
```

```
listPackageList               Provides the files in a specific packageList

listPackageToChange           Shows the contents of the current package to
modify

loadDeploy                    Shelves the deploy folder so it can be used at a
later time

makePackageListAll            Creates a package list for all files currently
held in the src directory. *see addToPackageList

matrix                        Creates a Profile Matrix

mergeCurrentCode              Merges current code with the latest from head

newPackage                    creates a new package file

packageList                   high level methods for dealing with package lists

pull                          pulls the latest code from version control for the
force directory.

refresh                       Refreshes all code currently in version control
with that in SalesForce.com

refreshFromList               Refreshes files from a deployment list.

removeExternalPackageMembers  Removes all external metadata/members from the
package - such as app exchange

removePackageList             Deletes a package list

removePackageMember           removes a member from the package xml file

removePackageType             removes a type from the package xml file

retrieveDeploy                High level methods refreshing/deploying/storing

settings                      high level methods for dealing with package lists

setupDeployCredentials        Defines the credentials used to retrieve metadata

setupRetrieveCredentials      Defines the credentials used to retrieve metadata

shelveDeploy                  Shelves the deploy folder so it can be used at a
later time
```

```
new                     Creates metadata/files based on templates

testCredentials              Tests specific credentials

testDeploy                   Changes whether only test deployments occur

updateIndices                Updates the output indices for the different
reports
```