Story Workbench Quickstart Guide

Version 1.2.0

Mark A. Finlayson
(markaf@mit.edu)

1 Basic Concepts

Annotation An indivisible piece of data attached to a text is called an *annotation*. Annotations, also known as *descriptions*, are the fundamental object in the workbench. The purpose of the workbench is to ease the viewing, editing, and creation of annotations on text. Each annotation has a type, called its *representation* or *layer*. The annotation's type determines what information the annotation contains, and what the annotator (be it a human annotator or an automatic annotator) must provide for the annotation to be well-formed.

Annotation Layer An annotation layer, also known as a *representation*, is the data format for type of annotation. They are called *layers* because you can think of them being "layered" on top of the natural language text you are annotating. Currently the Story Workbench has built-in support for over twenty different types of annotations, running the range from Syntactic annotations (such as token and sentence positions, part of speech tags, CFG parse trees, etc.), to Discourse annotations (referring expressions, co-reference structure), to Semantics (Wordnet word sense, semantic roles, etc.). You can also define your own tagset-based annotation layer. Layers are often configurable – for example, the Wordnet sense layer can be configured (for each file) with the version of Wordnet to use.

Annotation File Each natural language text and its associated anntations are gathered together into a single file. These files have the `sty` extension (short for "story" file), and contain XML is a fairly constrained format. For information about how to parse this file yourself, and what the different tags and content means, see the Story Workbench File Format Specification.

Annotation Factory Annotation factories, also known as *automatic annotators*, are programs that automatically provide annotations for a text. In the current version of the workbench, most layers have associated factories that can be used to generate a first pass set of annotations for that layer. For example, the token layer has an associated *tokenizer* that produces a set of token positions for a given text. Annotation factories are often configurable with different algorithms or implementations. Not all layers currently have factories available, and every layer may be configured with no factory (the *null* factory).

Build Rule A build rule defines under what conditions an annotation, or set of annotations, is considered ill-formed or has suspicious semantics. Build rules generate *build problems* and are associated with specific annotation layers. For example, there is a build rule for the token representation that specifies that tokens may not overlap. If they do, the build rule marks the file with an error that is shown in the GUI.

Corpus Annotation files are gathered together into a corpus, also known as a *story project*, or just a *project*. A corpus is a container that carries a variety of configuration information for both the representation layers and the annotation factories. Annotations files created inside a corpus are created by default with the configuration of that corpus. The configuration on a corpus also determines what build rules are run over the files in that corpus.

Workspace The workspace is a folder on your computer somewhere that contains the corpora shown in the Story Workbench. The workspace also contains the Story Workbench configuration, such as the layout of the application window. You may have multiple workspaces which you can switch between by selecting the appropriate workspace on startup, or going to *File → Switch Workspace*.

Eclipse Workbench The Story Workbench is built atop the Eclipse Application Framework. This means that all the functionality of Eclipse is included in the Story Workbench. Relevant features include: arbitrary placement and sizing of views inside the application window; ability to set your own keybindings; auto-update; and ability to install of third-party plugins. For more information see the tutorials on the Eclipse website.

2 Installing the Workbench

The Story Workbench comes packaged as a ZIP file. To install it, you merely unzip the file to where you want to the workbench installed. Inside the workbench folder is the executable.

Important: Mac Users: Do not move the executable from inside the folder; this will break the workbench. If you want a shortcut on your application bar to the workbench, create an *alias* to the executable and move that onto the application bar.

You have to have at least Sun Java 1.5 installed on your machine. If you have multiple Java JVMs installed (e.g., on a Linux or Windows machine), there are command line switches that can be used to make sure the Story Workbench uses the Sun Java installation.

When running the workbench, ignore the error in the Error Log that says "Could not find running profile instance." This is expected.

Its a good idea to close all files and close the workbench when you are done working for the day. The workbench is set to automatically check for updates on startup. Thus, when you start up again, the workbench will check for updates and you will always be sure to have an up-to-date version. If you don't want the workbench to automatically update, you can turn this off in Preferences. You can also manually check for software updates by going to *Help* → *Software Updates* → *Find and Install*, and selecting "Search for updates of the currently installed features."

3 Creating a Corpus

To create a corpus from scratch in the workspace, go to *File* → *New* → *Story Corpus*. This opens the New Story Corpus wizard.

1. On the "Create Story Corpus" wizard page, specify the name of the corpus. This will be the name of the folder on the computer that contains the corpus files.

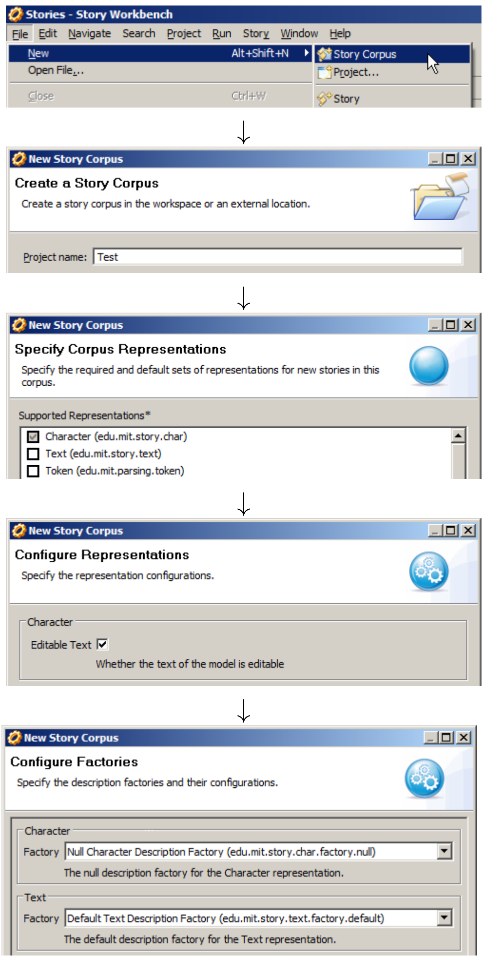
2. On the "Specify Corpus Representations" page, select the representations you want to annotation on files in the corpus. You may safely ignore the "Required representations" checkbox list.

3. On the "Configure Representations" page, you can set the parameters of those representations that are configurable. For example, if you want to protect the verbatim text from accidental modification by the annotator, uncheck the "editable text."

4. On the "Configure Factories" page, you can select the factories that will be used to automatically generate annotations. Each layer has, at the very least, a "Null Factory" option, which means that all the workbench will do is adjust annotation positions if text is inserted – it will not generate new annotations.

5. Select "Finish."

The Story Workbench comes with source control support, so if you have corpora stored in a reporisoty, you can check them out into the workspace. CVS and SVN are supported out of the box, and most other source control systems are supported by third-party Eclipse plugins.


Stories - Story Workbench
File Edit Navigate Search Project Run Story Window Help
New Alt+Shift+N
Open File...
Close Ctrl+W
Story Corpus
Project...
Story
↓
↓
↓
↓
New Story Corpus

Create a Story Corpus

Create a story corpus in the workspace or an external location.

Project name: Test

New Story Corpus

Specify Corpus Representations

Specify the required and default sets of representations for new stories in this corpus.

Supported Representations*

- [x] Character (edu.mit.story.char)
- [ ] Text (edu.mit.story.text)
- [ ] Token (edu.mit.parsing.token)

New Story Corpus

Configure Representations

Specify the representation configurations.

Character

Editable Text ✓

Whether the text of the model is editable

New Story Corpus

Configure Factories

Specify the description factories and their configurations.

Character

Factory: Null Character Description Factory (edu.mit.story.char.factory.null)

The null description factory for the Character representation.

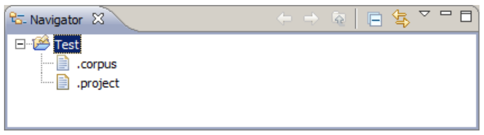
Text

Factory: Default Text Description Factory (edu.mit.story.text.factory.default)

The default description factory for the Text representation.

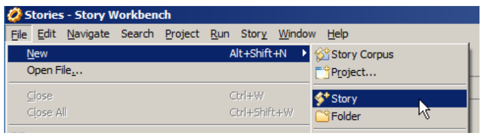
You can also create sub-folders within the corpus to organize your files. Files others than `.sty` files may also be included in the corpus.

When created, the story corpus will show up in the Navigator view. It will contain two files: `.project` and `.corpus` – do not delete either! The first is an Eclipse configuration file. The second is the Story Workbench configuration file, which specifies the representation layers, their parameters and factories.



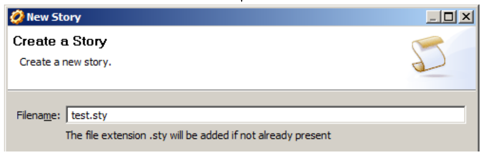
4 Creating an Annotation File

To create a story file, go to *File* → *New* → *Story*. This opens the New Story wizard.



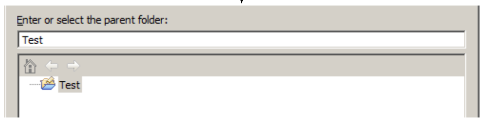
↓

1. On the “Create a Story” wizard page, specify the name of the story file. This will be the name of the file (with appended `.sty` extension) on the computer that contains the corpus files. Note: The metadata group currently doesn’t do anything.


New Story
Create a Story
Create a new story.
Filename: test.sty
The file extension .sty will be added if not already present

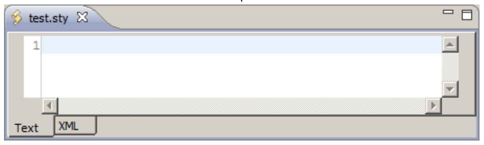
↓

2. Select the location for the story file in the part of the page titled “Enter or select the parent folder.”



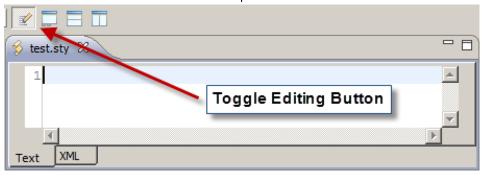
↓

3. Select “Finish.” This will create and open an empty file in the specified location in the corpus.

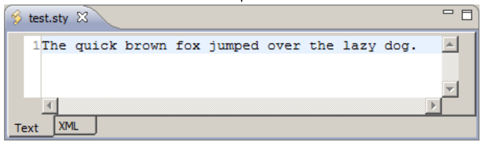


↓

4. Insert text into the story file. While it is possible to type text directly into the editor, there is a concurrency bugs that may make it difficult if you don’t type slowly enough (these will be fixed in the next major release). The better option is to type your text into another text editor, make sure your line breaks are in the right place, and then copy-and-paste your text into the Story Editor. To insert text either by typing or cut-and-paste, you will make sure editing is turned on; this may be done by using the toggle editing button in the toolbar above the editor.



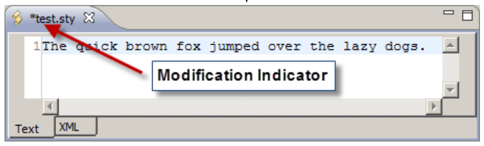
↓



The quick brown fox jumped over the lazy dog.

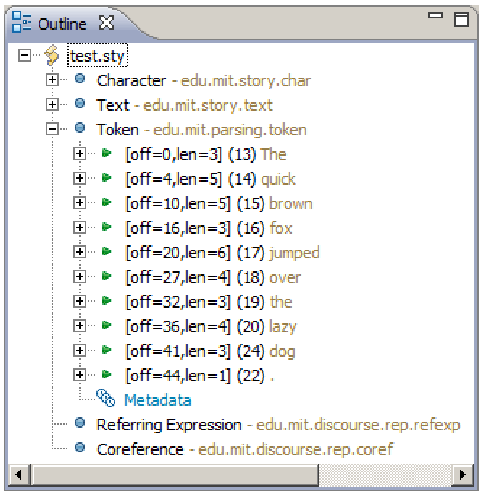
↓

5. When the text, an annotation, or an annotation layer is modified on a file, the editor will indicate that the file has been modified with a small star to the left of the filename in the editor tab. This means there are changes to the file that have not yet been saved to disk. To save them, go to *File* → *Save*.



The quick brown fox jumped over the lazy dogs.

To see an overview of what annotation layers and annotations are contained in the file, use the Outline View. If not already opened, a view may be opened by going to *Window* → *Show View* → *Other*. Each first-level entry in the tree is an annotation layer; each second-level entry is an annotation. Annotations are shown with their offset (`off`), length (`len`), and id number (the number in parentheses). To the right of that information is the actual string data for that annotation found in the story file.

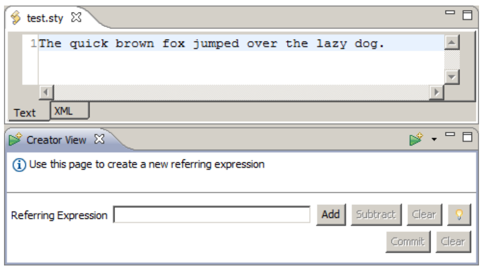


Outline

```
test.sty
  Character - edu.mit.story.char
  Text - edu.mit.story.text
  Token - edu.mit.parsing.token
    [off=0,len=3] (13) The
    [off=4,len=5] (14) quick
    [off=10,len=5] (15) brown
    [off=16,len=3] (16) fox
    [off=20,len=6] (17) jumped
    [off=27,len=4] (18) over
    [off=32,len=3] (19) the
    [off=36,len=4] (20) lazy
    [off=41,len=3] (24) dog
    [off=44,len=1] (22) .
    Metadata
  Referring Expression - edu.mit.discourse.rep.refexp
  Coreference - edu.mit.discourse.rep.coref
```

5 Creating Annotations

Most annotation layers have a general interface for creating annotations available via the Creator View. Shown to the right is the Creator View page for the Referring Expression annotation layer.

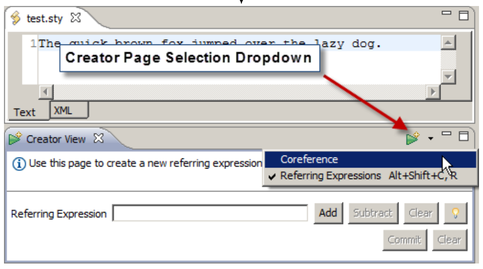


1 The quick brown fox jumped over the lazy dog.

Use this page to create a new referring expression

↓

You can switch between pages for different layers by using the page selection dropdown as shown.

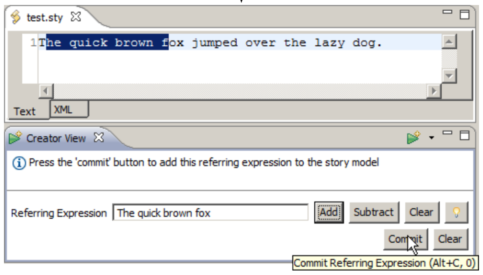


1 The quick brown fox jumped over the lazy dog.

Use this page to create a new referring expression

↓

Most creator pages follow the same semantics: you select text in the editor, and add or subtract it to fields in the creator view. The Creator View might also have combo dropdowns or free-entry fields to create information. When a valid annotation is specified in the creator view, the "Commit" button will be enabled. Press it to add the annotation to the file.

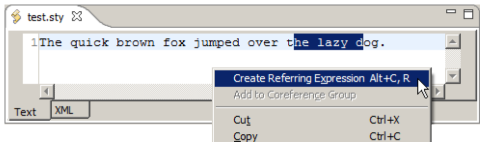


1 The quick brown fox jumped over the lazy dog.

Press the 'commit' button to add this referring expression to the story model

Referring Expression: The quick brown fox

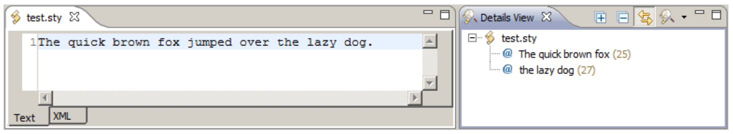
Each layer may have its own specific interface for creating annotations. For example, creating referring expressions may be done via a context menu.



1 The quick brown fox jumped over the lazy dog.

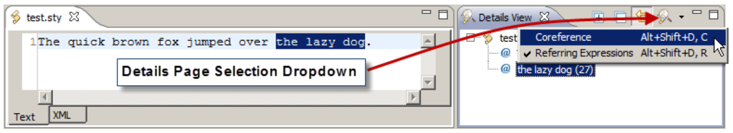
6 Viewing Annotations

Most annotation layers also provide an interface for viewing the details of annotations in that layer, available via Details View. Shown to the right is the Details View page for the Referring Expression annotation layer.



The quick brown fox jumped over the lazy dog.

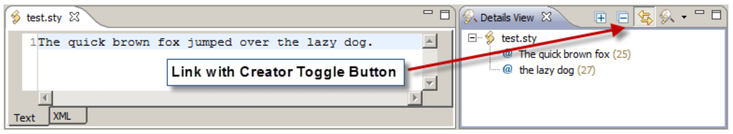
You can switch between pages for different layers by using the page selection dropdown as shown.



The quick brown fox jumped over the lazy dog.

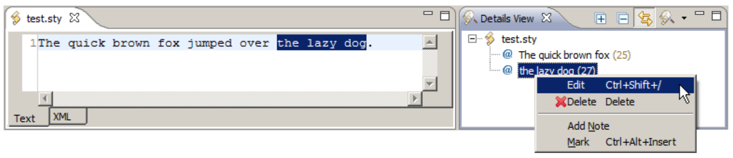
test.sty
Details Page Selection Dropdown
Text
XML
Details View
Coreference Alt+Shift+D, C
✓ Referring Expressions Alt+Shift+D, R
the lazy dog (27)

If you toggle the “Link to Creator” button, the details view will switch to the preferred details page when the creator view page is changed.



The quick brown fox jumped over the lazy dog.

The context menu on each annotation in the details view will have multiple options. Most details view pages allow you delete an annotation from the details view, or load it back into the creator view to adjust its properties. Most details view pages also allow you to add an arbitrary note to an annotation, or to add a visual mark to the annotation so you can remember it for later consideration. Some layers allow you to edit annotation features directly in the details view.



The quick brown fox jumped over the lazy dog.