Contents

For problems or questions, contact: protege-help@smi.stanford.edu

For the latest implementation, see: http://protege.stanford.edu

What is Protégé-2000?

Protégé-2000 is an integrated software tool used by system developers and domain experts to develop _knowledge-based systems._ Applications developed with Protégé-2000 are used in problem-solving and decision-making in a particular _domain_.

While our earlier Protégé/Win, like a classical database system, separately defined _classes_ of information (schema) and stored _instances_ of these classes, Protégé-2000 makes it easier to work simultaneously with both classes and instances. Thus, a singular instance can be used on the level of a class definition, and a class can be stored as an instance. Similarly, _slots,_ which earlier were employed only within classes, are now elevated to the same level as classes. With this new knowledge model we also facilitate conformance to the _Open Knowledge Base Connectivity (OKBC)_ protocol for accessing knowledge bases stored in knowledge representation systems. Finally, _applications_ on top of these components are also executed within the integrated Protégé-2000 environment.

The Protégé-2000 tool accesses all of these parts through a uniform GUI (graphical user interface) whose top-level consists of overlapping tabs for compact presentation of the parts and for convenient co-editing between them. This "tabbed" top-level design permits an integration of (1) the modeling of an _ontology_ of classes describing a particular subject, (2) the creation of a _knowledge-acquisition tool_ for collecting knowledge, (3) the entering of specific instances of data and creation of a _knowledge base_, and (4) the execution of applications. The ontology defines the set of concepts and their relationships. The knowledge-acquisition tool is designed to be domain-specific, allowing domain experts to easily and naturally enter their knowledge of the area. The resulting knowledge base can then be used with a _problem-solving method_ to answer questions and solve problems regarding the domain. Finally, an _**application**_ is the end product created when the knowledge base is used in solving an end-user problem employing appropriate problem-solving, expert-system, or decision-support methods.

The main assumption of Protégé-2000 is that knowledge-based systems are usually very expensive to build and maintain. For example, the expectation is that knowledge-based system development is a team effort, including both developers and domain experts who may have less familiarity with computer software. Protégé-2000 is designed to guide developers and domain experts through the process of system development. Protégé-2000 is designed to allow developers to reuse domain ontologies and problem-solving methods, thereby shortening the time needed for development and program maintenance. Several applications can use the same domain ontology to solve different problems, and the same problem-solving method can be used with different ontologies. For more information about building knowledge-based systems and the Protégé-2000 approach, see Planning a Protégé-2000 Project.

Protégé-2000 is currently being used in clinical medicine and the biomedical sciences, although it can be used in any field where the concepts can be modeled as a class hierarchy. Throughout this online guide, you will see examples of a newspaper domain used to illustrate points of explanation; for more information about this running example, see an overview of the Newspaper Example.

Using this Guide

The Protégé-2000 User's Guide documents how to use Protégé-2000, an integrated knowledge-base development and management system. These pages include descriptions of the user interface along with step-by-step instructions for completing specific tasks. Screenshots of a Newspaper Example are used to illustrate the explanations through most of this guide's pages.

We assume that you are familiar with your operating-system platform and with using an Internet browser.

You can navigate among topics by using your browser's **Back** and **Forward** buttons, or by using the contents list of top-level topics in the left-hand frame. In addition, at the bottom of every page there are pointers to related topics. There is always one link designated as "Next": by continually following this link, you can visit every page. As a final aid for navigation, the main subsystems of Protégé-2000 include tables of contents that list all pages associated with it, as well as a table of contents for the project level. (Classes Table of Contents, Slots Table of Contents, Instances Table of Contents, Forms Table of Contents, and Project Table of Contents).

Next: What is Protégé-2000?

A Newspaper Example

Throughout this guide, we provide examples and screenshots from a fictitious "newspaper" example. We designed this example to be intuitive. This example is available under the examples directory in the Protégé-2000 directory.

There are a variety of possible uses for a knowledge base of newspaper data. Our example knowledge base includes:

- A list of all published articles, indicating when published, in what section, etc.
- Information about standard sections of the newspaper (Sports, Lifestyle, Business, etc.)
- Employee information
- Advertising information

There are a variety of applications that might use information in this knowledge base. For example, one could build:

- A system for retrieving, organizing and answering queries about published articles
- A system for analyzing advertisement revenues or pricing
- A system for reviewing the organization of the employees that makes sure that reporters are balanced appropriately among editors, and that all sections of the newspaper have a responsible editor

As this is an artificial example, we have not built any of these applications, but we hope this example will give a flavor of how knowledge bases, ontologies, and knowledge-acquisition tools can be designed with Protégé-2000.

Next: Project Table of Contents

Projects

Project Tasks

- **Creating a Project**
- **Opening a Project**
- **Saving a Project**
- **Renaming a Project**
- **Saving a Project in a New Format**
  - **Saving a Text Project**
  - **Saving a Database Project**
  - **Saving an RDF Project**
- **Importing a Project**
  - **Importing Text Files**
  - **Importing a Database Table**
  - **Importing RDF Files**
- **Including a Project**
- **Configuring a Project**
- **Generating HTML From a Project**

Window Management Tasks

- **Cascading Open Windows**
- **Closing All Windows**
- **Working With a Small Window**
- **Working With Notes**

Next:

Creating a Project

There are two ways to create a new Protégé-2000 project:

- when you start Protégé.
- from within the Protégé window

Creating a Project When You Start Protégé

To create a project when you start Protégé:

1. Start Protégé.
   A dialog box allows you to create a new project, open a recent project, or get help.
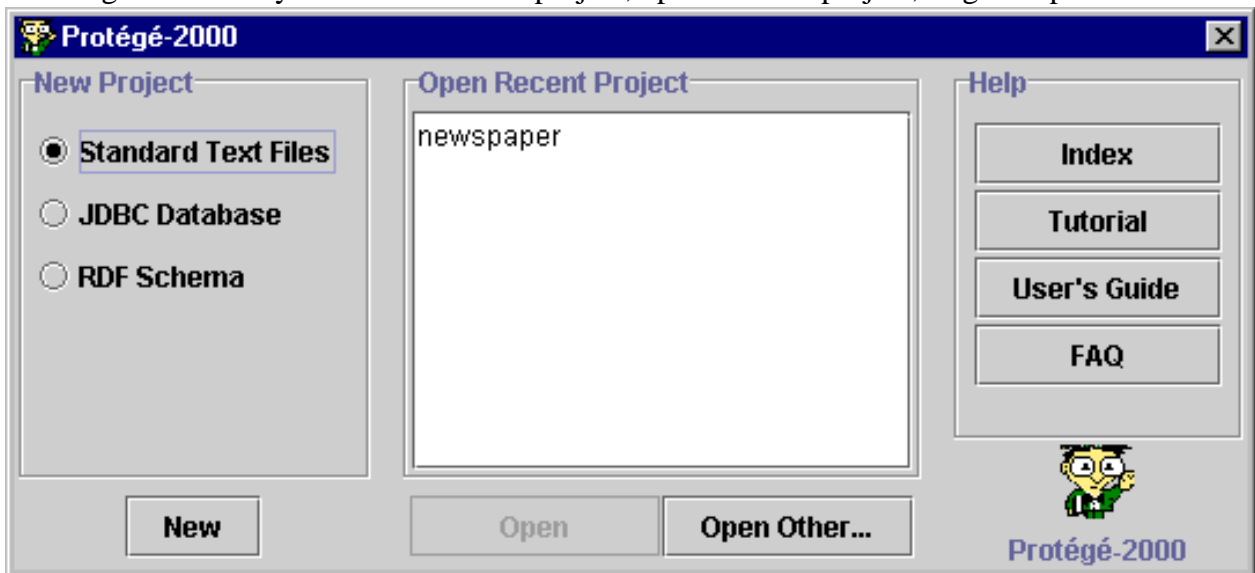
2. Unless you have a need for a special format for your files, make sure **Standard Text Files** is selected in the New Project field at the left of the window. This will create a project file in Protégé-2000 format. (See Project Structure, below, for more information about file formats.)
3. Click **New**.
   The Protégé window opens and the standard tabs become visible. An initialized knowledge base will be created which contains the system classes rooted in :THING. No instances will be created. You can now structure your project by creating your classes and slots. (See Creating a New Class and Creating a Slot for more information.)
4. To save the project to disk, select **Save** from the **Project** menu.
5. Enter a name for your project in the Project line of the dialog box. This is the name of your project (**.pprj**) file. Protégé also creates internal files for its own use. By default, these files are given the same name, with a different extension.

Creating a Project From the Protégé Window

To create a new Protégé-2000 project:

1. Select **New** from the **Project** menu *or* click the New Project button .

2. You will be prompted to save changes to the current file, if any. Click **Yes** to save changes, **No** if for some reason you do not want to modify the file (e.g., if you have been browsing or experimenting).
3. Select the format you want for your Protégé-2000 files from the Select Format dialog and click **OK**.
   Unless you have a need to create files in a special format, make sure **Standard Text Files** is selected. (See Project Structure, below, for more information about file formats.)
4. An initialized knowledge base will be created which contains the system classes rooted in :THING. No instances will be created. You can then structure your project by creating your classes and slots. (See Creating a New Class and Creating a Slot for more information.)
5. To save the project to disk, select **Save** from the **Project** menu.
6. Enter a name for your project in the Project line of the dialog box. This is the name of your project (.**pprj**) file.

Project Structure

Each time you create a project, you are given a choice of project formats. Unless you have a need for a specific structure (e.g., for exporting files), you should select Standard Text Files. No matter which format you choose, the information specific to the Protégé-2000 interface is saved in a **pprj** (**P**rotégé **pr**oject) file. You can create, open, and save your projects directly via the **pprj** file. You do not need to name or access any other files unless you wish to import a project.

Internally, Protégé maintains two files in addition to the **pprj** file; these files contain further information about the ontology and instances of the project. When you open a **pprj** file, Protégé-2000 automatically loads these files. By default, the additional files are saved in text format:

- a text file containing the class and slot information, given the extension **pont** (**P**rotégé **ont**ology).
- a text file containing the instance information, given the extension **pins** (**P**rotégé **ins**tances).

Whenever you are creating a project, you are given a choice of formats in which these files could be saved:

- **Standard Text Files** (the default) creates the project files in Protégé-2000 format as described above. These can also be viewed with any text editor or word processor.
- **JDBC Database** creates the project as a table in a JDBC database. To do this you must have a

database installed and configured on your system. See [Saving a Database Project](#) for more information.

- **Resource Description Framework (RDF)** saves the project in RDF format. See [Saving an RDF Project](#) as well as [RDF Support in Protégé-2000](#) for more information.

If your version of Protégé-2000 has been customized to support additional formats, then you will also see them in this list.

In all cases, Protégé-2000 still uses the **pprj** file to access the project. You should not see a difference within Protégé.

Next: [Opening a Project](#)

[Project Table of Contents](#)

Creating a New Class

To create a new class:

1. In the Class Relationship Pane, highlight the class that you want as the **superclass** of the new class.
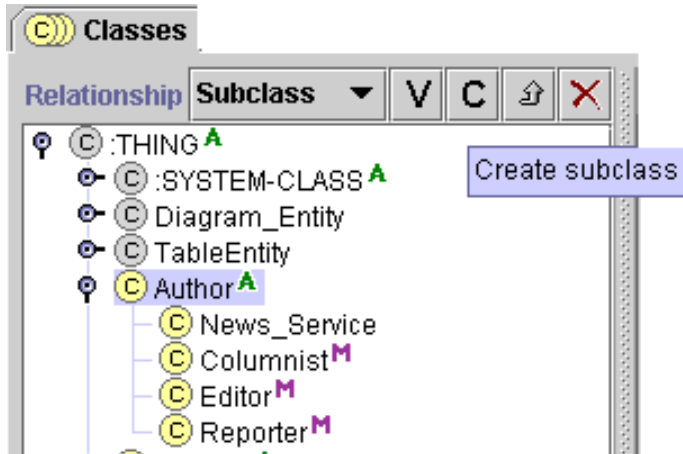
2. Click the **C(reate)** button, which appears as a **C** in the class buttons at the right of the Class Relationship Pane, *or* click the right mouse button and select Create subclass from the cascading class menu.
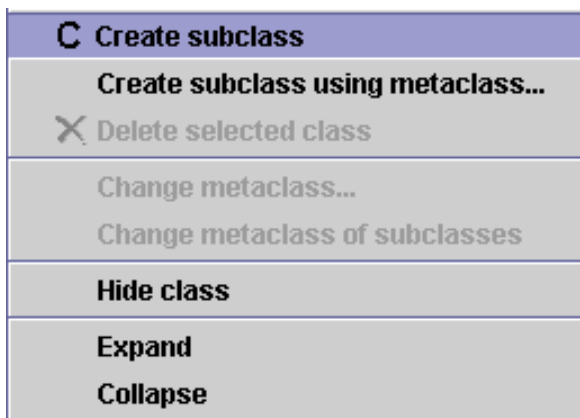
3. The new class will be added under the highlighted class. It will have a default name, such as project_CLASS_00001.

4. Use the Class Form to name the class, choose its role, create constraints, and create and edit slots. See Viewing a Class for more information.

Next: Deleting a Class

The Class Relationship Pane

The Class Relationship pane, in the upper left in the [Classes tab](#), displays the classes in the knowledge base as a tree. The Class Relationship pane includes four components:

1. The [Class Relationship menu](#) allows you to control the display of the classes in the [Class Window](#).
2. The [Class Buttons](#) allow you to create, edit, and delete classes in your knowledge base.
3. The [Class Window](#) displays your classes and allows you to rearrange your hierarchy using "drag-and-drop". A number of icons give additional information about your classes; see [Icons](#) for a description. See [Replacing a Superclass](#) for information on rearranging your hierarchy.
4. The Class Lookup Bar allows you to locate a class in the Class Relationship window by typing all or part of the class name and clicking the binoculars. See [Finding a Class](#) for more information.
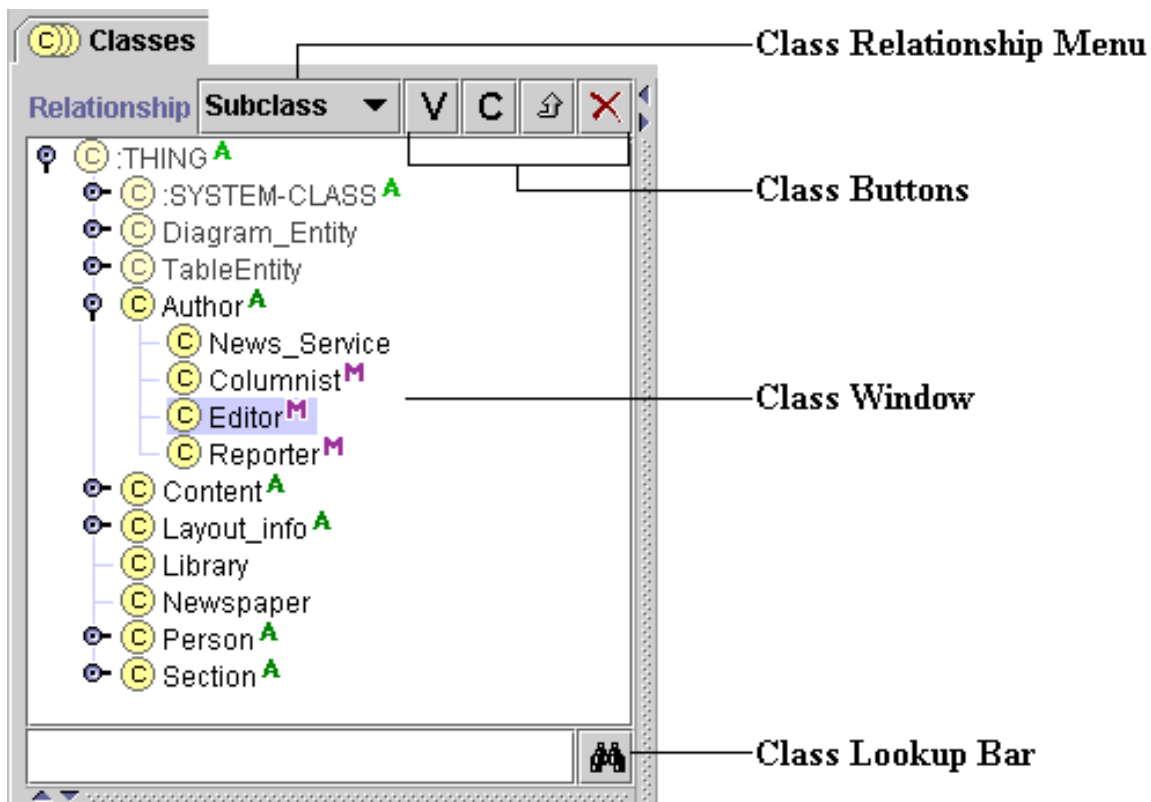
Any changes you make in this window, or other windows, take effect immediately. To make changes permanent, save your project by choosing **Save** from the **Project** menu.

Next: [The Class Buttons](#)

The Class Buttons

The Class buttons, V C ⇧ ✕, located at the top right of the [Class Relationship pane](#), allow you to view and edit, create, or delete a class. Note that these icons have a similar action wherever they appear; for example, see the [Instance Buttons](#).
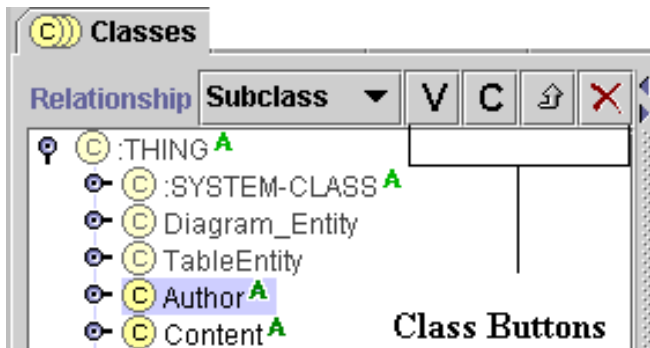
The buttons have the following actions:

V(iew) button: Click this button to open the [Class Form](#) for the highlighted class. This allows you to view or edit the class. See [Viewing a Class](#).

C(reate) button: Click this button to create a new class as a subclass of the highlighted class. See [Creating a New Class](#).

Back-references button: Click this button to view all the objects that reference the highlighted class. See [Viewing Back-References](#).

Delete button: Click this button to delete the highlighted class and all of its subclasses. See [Deleting a Class](#).

If the Create or Delete button is grayed out, this indicates that the current class cannot be edited or deleted. Such a class always has a gray ☐ icon to its left. Classes cannot be edited if they either are a system class or are included from another project.

Next: [The Class Window](#)

[Classes Table of Contents](#)

The Class Form

The Class form can be used to define and edit the attributes of the class selected in the Class Relationship pane. The Class Form can be viewed in several ways:

- In the Classes tab, if a single class is selected, the Class form is displayed at the right.
- In the Classes tab, the Class form can also be displayed separately in a free-standing window by clicking on the View V icon in the Class Relationship pane.
- In the Instances Tab, the Class form can be displayed by selecting a class in the Class pane and clicking the View V icon.

Whenever you enter changes into the Class form, they take effect immediately. To make the changes permanent, select **Save** from the **Project** menu.

There are six separate areas in the Class form:

1. The Note Icons allow you to add and remove notes.
2. The Class Name field allows you to name your class.
3. The Class Role menu allows you to choose whether your class is concrete or abstract.
4. The Class Constraints pane allows you to create, edit, add, and delete constraints for your class.
5. The Class Documentation pane allows you to add simple text notes explaining your class.
6. The Template Slots pane allows you to view the slot information for your class.

For more information on using these areas to create and view and edit classes, use the links in the list above, click on the desired area in the graphic below, or see Creating a Class and Viewing a Class.

Editor

Name
Editor

Role
Concrete

Documentation
Editors are responsible for the content of sections.

Constraints
⬧ editor-employees-salary-constrain

Template Slots

Name	Type	Cardinality	Other Facets
S responsible_for	Instance	multiple	classes={Employee}
S sections	Instance	multiple	classes={Section}
S byname	String	single	
S salary	Float	single	
S date_hired	String	single	
S current_job_title	String	single	
S other_information	String	single	
S name	String	single	
S phone_number	String	single	

Note Icons

The note icons, **C** ✗, at the upper right of the form allow you to add and remove yellow sticky notes to your class. The note is always displayed along with the form. For information on how to add notes to any frame (class, instance, or slot), see Working with Notes.

The Class Name Field

The Name field allows you to name your class. When a class is created, it is given a default name, such as CLASS_00001. You can change the name of a new or existing class by highlighting the name and overwriting it. The following rules apply to class names:

- The name must be a unique class name in the knowledge base.

- Class names are case sensitive.

A recommended convention is to make the first character of each word in a class name uppercase and the rest lowercase, and to separate words with an underscore.

The Class Role Menu

The Role menu allows you to choose the role of your class: concrete or abstract. Concrete classes may have direct instances, but do not have to; abstract classes cannot have direct instances. Protégé-2000 sets the role to **Concrete** by default.

Protégé-2000 does not impose any restrictions on the role of your classes. One common modeling convention is to make all leaf (bottom-level) classes concrete, and all interior (higher-level, non-leaf) classes abstract.

The Class Constraints Pane

Class constraints are defined programatically. See Constraints for more information.

The Class Documentation Pane

The Class Documentation pane allows you to enter a text description of the class, as well as any special notes. Filling in this field is optional but is recommended to aid in the maintenance of the knowledge base.

The Template Slots Pane

The Template Slots pane displays the direct and inherited slots for the selected class. Slots, which represent properties of your class, are a crucial part of your knowledge base. In the example, **Editor** has several slots which appear in the Template Slots pane. For a full description, see the separate Template Slots Pane topic.

Viewing Several Classes

To view the information for several classes at once, select the classes and click the View V class button to open the Class Form for each class. To highlight multiple classes, hold down the Ctrl key while clicking each class. To highlight a range of classes, click the first class, then hold down the Shift key and click the last class in the range.

Opening a new class form does not close the previous form. This allows you to compare the attributes for two or more classes. Edits can be made directly in any open Class Form.

If you have multiple forms open, you can manage them as follows:

- Cascade multiple forms by clicking the Cascade 🖼 button below the main menu bar, or by selecting **Cascade Windows** from the **Windows** menu..

- Close all open forms by clicking the **CloseAllWindows** 🖼 button below the main menu bar, or by selecting **Close All Windows** from the **Windows** menu.

Next: The Template Slots Pane

Classes Table of Contents

Viewing a Class

You can edit an existing class using the Class Form. You can access the Class Form in one of two ways:

- from the Classes Tab.
- by selecting a class in the Class pane at the Instances Tab and clicking the View V button.

To edit a class from the Classes Tab:

1. Select the class you want to edit in the Class Relationship pane in the Classes Tab. The current information for the highlighted class will be shown in the Class Form to the right.
2. Enter the updated information directly in the Class Form to the right *or* click the View V icon in the Class Relationship pane to show the same form as a free-standing window.

To edit a class from the Class pane at the Instances Tab:

1. Select the class you want to edit in the Class pane at the Instances Tab.
2. Click the View V icon at the upper right of the pane to open the Class Form.

Note: If the class appears as one of the Allowed Parents for a slot of type Class, you can also open the Class Form by going to an instance that has that slot and clicking on the View V icon. See Standard Widgets for more information.

Note: If the class is included from another project, it cannot be edited. Included classes are shown with a pale class icon to the left. See Including a Project for more information.

Using the Class Form

Editor

Name	Documentation	Constraints V C + −
Editor	Editors are responsible for the content of sections.	⬥ editor-employees-salary-constrair

Role

Concrete ▼

Template Slots   V V C ✕ + −

Name	Type	Cardinality	Other Facets
S responsible_for	Instance	multiple	classes={Employee}
S sections	Instance	multiple	classes={Section}
S byname	String	single	
S salary	Float	single	
S date_hired	String	single	
S current_job_title	String	single	
S other_information	String	single	
S name	String	single	
S phone_number	String	single	

Any changes you enter into the Class Form take effect immediately. To make the changes permanent, save the knowledge base by selecting **Save** from the **Project** menu.

To revert to the last saved version, close Protégé-2000 without saving changes. If you have made extensive changes to your knowledge base during the current session, you may wish to save before editing classes.

Changing the Name of a Class

To change the name of a class, edit the name in the **Name** field. The following rules apply to class names:

- The name must be a unique frame name in the knowledge base.

- Class names are case sensitive.

A recommended convention is to make the first character of each word in a class name uppercase and the rest lowercase, and to separate words by underscores.

Changing the Role of a Class

Select the new role from the Role menu. Concrete classes may have direct instances, but do not have to; abstract classes cannot have direct instances. Protégé-2000 does not impose any restrictions on the role of your classes.

Changing Constraints

Class constraints are defined programatically. See Constraints for more information.

Changing Class Documentation

To change the notes for a class, change the text directly in the **Class Documentation** pane.

Working with Class Slots

The Template Slots pane allows you to edit, create, add, and remove slots from your class.

Editing Class Slots

You can edit the slot in one of two ways:

- At the top level. Editing the slot at the top level affects the slot everywhere it appears, including the Slots Tab and any classes where the slot is attached.

- At the class. Editing the slot at the class level (called overriding) only affects the slot at the class and any subclasses. The slot remains unchanged at the Slots Tab and any unrelated classes where appears. A slot that has been overridden has an ° override icon in the left column.

To edit a slot at the top level:

1. Highlight the slot name in the Template Slots pane

2. Click the top-level View button at the top right of the pane. The Slots Form is displayed.

3. Use the Slots Form to edit slot properties such as **Name**, **Type**, **Cardinality**, and to add notes. See Viewing a Slot for more information on how to edit a slot.

To edit a slot at the class:

1. Highlight the slot name in the Template Slots pane

2. Click the Class-Level View button at the top right of the pane. The Slots Form is displayed.

3. Use the Slots Form to edit slot properties such as **Name**, **Type**, **Cardinality**, and to add notes. See

Viewing a Slot for more information on how to edit a slot.

Restrictions on Global and Inherited Classes

Your class can have slots that were inherited from a superclass. If the slot is defined globally or is inherited, only the following edits can be performed:

- For **Cardinality**, the slot may be changed from **Multiple** to **Single** (but not vice versa).

- For certain slot **Types**, some restrictions apply:

  - For a slot of type **Any**, the slot may be restricted to *one* of the other types (**Boolean, Class, Float, Instance, Integer, String,** or **Symbol**). Slots of type other than **Any** cannot have the type changed.

  - For a slot of type **Class**, the allowed parents can only be changed to subclasses of the allowed classes in the parent class.

  - For a slot of type **Instance**, the allowed classes can only be changed to subclasses of the allowed classes in the parent class.

- A **Minimum** value may be created or increased. The **Minimum** value is only available for types **Integer** or **Float.**

- A **Maximum** value may be created or decreased. The **Maximum** value is only available for types **Integer** or **Float.**

- The **Documentation** can be changed.

Direct slots, which were created or added at the level of the slot, have no editing restrictions.

If you have edited a slot at the class level, the slot is displayed with an override ° icon.

Creating a New Slot

You can define a new slot for your class.

1. Make sure the correct class is highlighted in the Class Relationship pane.

2. Click the Create C button at the top right of the pane. The Slots Form is displayed.

3. Use the Slots Form to edit slot properties such as **Name**, **Type**, **Cardinality**, and to add notes. See Creating a Slot for more information on how to create a slot.

A direct slot is displayed with a blue S icon.

Clearing Slot Overrides

If you have overriden a slot at the class, you can remove your overrides and use the top-level definition of the slot:

1. Highlight the slot name in the [Template Slots](#) pane

2. Click the Clear Overrides ![icon] button at the top right of the pane. Any changes you made at the class level are removed and the top-level definition of the slot is used.

Adding an Existing Slot to a Class

Once slots have been created, you can add them to more than one class. For example, the **Prototype_Newspaper** class has a **Weekday** slot, which can be used to choose among the days of the week. If you were creating a new type of employee who wrote a weekly feature, you might want to reuse this slot.

To choose a pre-existing slot to add to your class:

1. Make sure the correct class is highlighted in the Class Relationship pane.

2. Click the Add ![icon] button at the top right of the pane. The Select Slots dialog box displays all the slots you can add to the class.

3. Highlight the slot you wish to add to your class.

4. Click OK.

The new slot is added to the Template Slots pane. It is a directly attached slot and is displayed with a blue ![S icon] icon. You do not need to name the slot; however, you may need to override its facets. If you wish to override the facets on the slot, you may click the Top-Level View ![icon] or Class-Level View ![icon] button to display the [Slot Form](#). See [Viewing a Slot](#) for more information.

Removing a Slot

You can remove any direct slot that appears in the Template Slots pane. To remove a slot:

1. Highlight the slot
2. Click the Remove ![icon] button. The slot will be removed from the class.

Note that the slot has not been deleted from the knowledge base. It will still appear in the Select Slots dialog box (see [Adding a Slot](#), above) and the [Slots Tab](#). See [Deleting a Slot](#) for more information.

Other Class Editing Operations

Viewing Multiple Classes

To view or edit the information for several classes at once, select the classes one by one and click the View
V class button to open the Class Form for each class. Opening a new class form does not close the previous
form. This allows you to compare the attributes for two or more classes. Edits can be made directly in any
open Class Form.

Changing Superclasses

To change the superclasses of a class see: Adding a Superclass, Replacing a Superclass, and Removing a
Superclass.

Next: Viewing Class Relationships

Classes Table of Contents

Deleting a Class

To delete a class from the knowledge base:

1. Highlight the class you want to delete in the Class Relationship pane.
2. Click the ✖ icon from the class buttons *or* click the right mouse button and choose **Delete selected class** from the cascading class menu.

C Create subclass	
Create subclass using metaclass...	
✖ Delete selected class	
Change metaclass...	
Change metaclass of subclasses	
Hide class	
Expand	
Collapse	
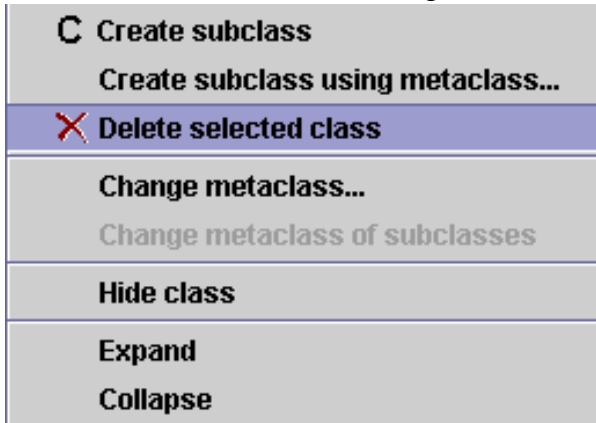
3. You will be prompted for confirmation. Click OK.

To delete multiple classes:

1. Highlight one or more classes in the Class Relationship pane. To highlight multiple classes, hold down the Ctrl key while clicking each class. To highlight a range of classes, click the first class, then hold down the Shift key and click the last class in the range.
   You cannot use the cascading menu to delete multiple classes.
2. Click the ✖ icon from the class buttons. You will be prompted for confirmation.

Deleting a class deletes the class and all of its subclasses. Once a class has been deleted it cannot be recovered.

However, if you close Protégé-2000 without saving changes, you will revert to the last saved version. If you have made extensive changes to your project during the current session, you may wish to save before deleting classes. To do this, select **Save** from the **Project** menu.

Classes cannot be deleted if they either are a system class or are included from another project. The class icon to the left of the class indicates the class type. See Including a Project for information about project inclusion.

Classes Table of Contents

The User Interface

- **The Classes Tab**
- **The Class Relationship Pane**
- **The Class Buttons**
- **The Class Window**
- **The Class Icons**
- **The Class Menu**
- **The Superclasses Pane**
- **The Class Form**
- **The Template Slots Pane**
- **The Template Slot Buttons**
- **The Back-References Pane**

Class Tasks

- **Creating a Class**
- **Deleting a Class**
- **Viewing a Class**
- **Viewing Class Relationships**
- **Finding a Class**
- **Hiding a Class**

Superclass Tasks

- **Replacing a Superclass**
- **Adding a Superclass**
- **Jumping to Another Superclass**
- **Removing a Superclass**

Metaclass Tasks

- **Understanding Metaclasses**
- **Creating a Metaclass**
- **Creating a Class Using a Metaclass**
- **Changing the Metaclass of a Class**
- **Changing the Metaclass of Subclasses**

Classes & Instances Tab

- **The Classes & Instances Tab**

Next:

Creating a New Slot

There are three ways to create a slot:

1. By clicking the Create **C** Slot button in the Slot pane of the Slots Tab. This creates a slot, but does not assign it to any class. The slot can later be added to one or more classes.
2. By right clicking the mouse button in the Slot pane of the Slots Tab and selecting **Create slot** from the Slot menu. This is identical to the previous operation.

C Create slot
C Create subslot
Create subslot using slot metaclass...
✕ Delete selected slots
Change slot metaclass...
Change slot metaclass of subslots
Expand
Collapse
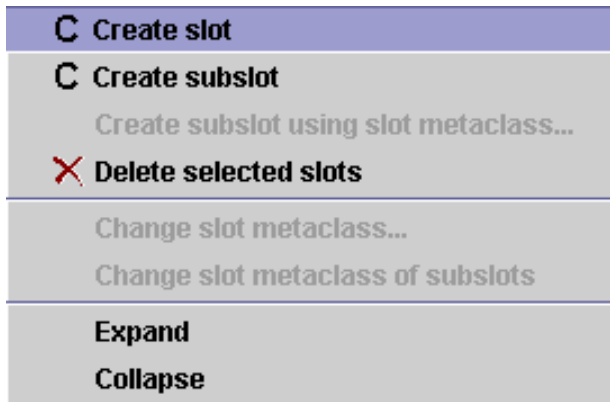
3. By selecting a class in the Classes tab and then clicking the Create **C** Template Slot button in the Template Slots pane. This creates a slot and attaches it to the selected class.

In either case, clicking the Create **C** button will create a new slot and display the Slot Form, which you can use to define the properties of the slot:

Once a slot has been created, you can define and edit its properties as described in [Editing Slot Properties](#).
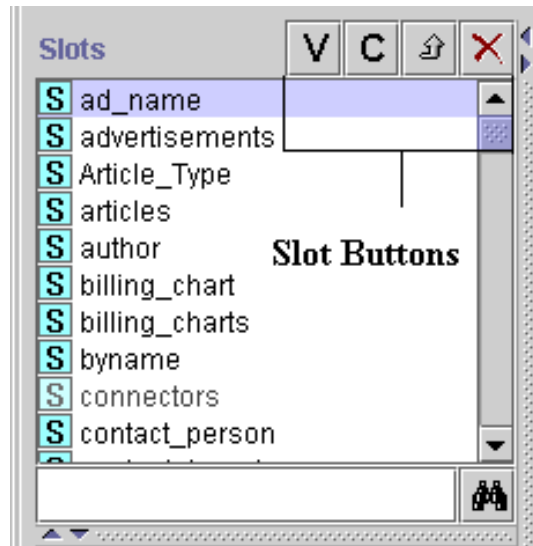

Next: [Viewing a Slot](#)

The Slot Buttons

The Slot buttons, V C ⬆ −, located at the top right of the Slot pane in the [Slots Tab](#), allow you to view and edit, create, or add, or remove a slot for the current class.

The buttons have the following actions:

V(iew) button: Click this button to open the [Slot Form](#) for the highlighted slot. You can also view a slot by double-clicking it in the Slot Pane. See [Viewing a Slot](#).

C(reate) button: Click this button to create a new slot. See [Creating a Slot](#).

Back-references button: Click this button to view all the objects that reference the highlighted slot. See [Viewing Back-References](#).

Remove button: Click this button to delete the highlighted slot from the project. See [Deleting a Slot](#).

You can also view, create, and edit slots from the [Template Slots pane](#). See the [Template Slot Buttons](#) for more information.

Next: [The Slot Menu](#)
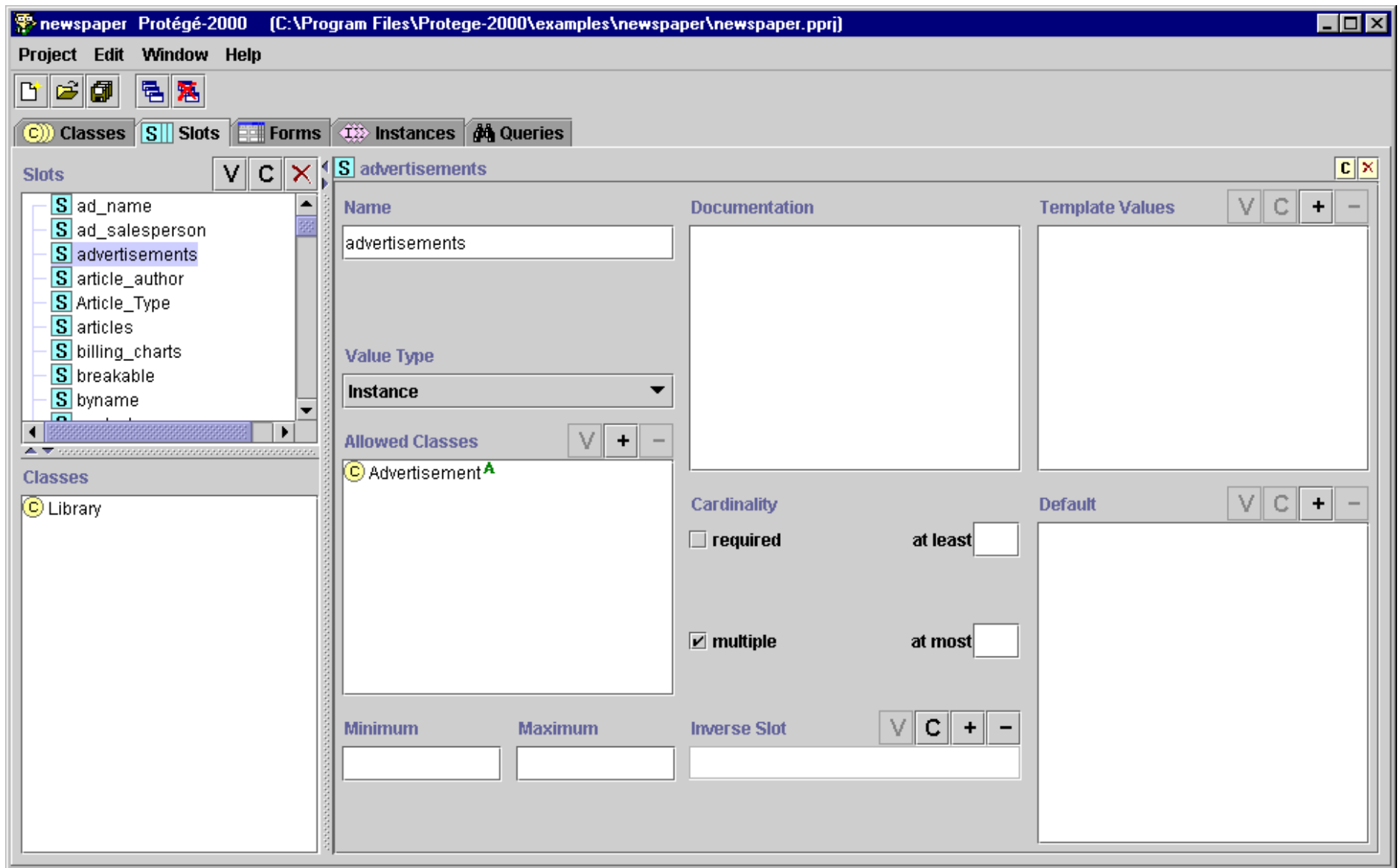
[Slots Table of Contents](#)

The Slots Tab

The Slots tab provides a single window in which you may view, create, and edit **slots**. Although slots are often thought of as being attached to a class, they can be defined and manipulated independently, and can exist without any relationship to classes.

An example of the Slots tab is shown below. The window consists of three panes:

1. The Slots pane in the upper left shows all the slots in the project and allows you to edit existing slots, create new slots, view back-references for slots, and delete slots. For more information, see the Slot Buttons.

2. The Classes Pane at the Slots Tab in the lower left shows the all the classes that have the slot attached. This is a view-only window. To view or manipulate classes, use the Classes Tab. As mentioned above, a slot can exist without being attached to any class.
Note: If you cannot see the Classes Pane, your window may be too small. You can see the pane by enlarging your window or by dragging the slider bar at the bottom of the Slot Menu. See Working With a Small Window for more information.

3. When a single slot is selected, the Edit pane on the right contains the Slot Form for the selected slot. The Slot Form allows you to: name the slot, choose its cardinality and value type, define constraints, defaults, and maximum and minimum values, as well as provide a brief description. The Slot Form can be displayed as a separate window by clicking the View V slot button in the Slots Pane. The Slot Form can also be accessed from the Templates Slots pane in the Classes Tab.

For information about the Slots Tab user interface and about accomplishing specific tasks, see the Slots Table of Contents.

Next: The Slot Buttons

Adding an Existing Slot to a Class

Once a slot has been created, you can attach it to one or more classes. For example, the **name** slot appears both at **Organization** and **Person**.

To choose a pre-existing slot to add to your class:

1. Make sure the correct class is highlighted in the [Class Relationship pane](#).
2. Click the Add ✛ button at the top right of the pane. The Select Slots form displays all the slots you can add to the class.
3. Highlight the slot you wish to add to your class.
4. Click OK.

The new slot is added to the Template Slots pane. It is a directly attached slot and is displayed with a blue [S] icon. You do not need to name the slot or define its facets. However, if you wish to edit override the slot, you may double-click it or click the top-level View ⋁ or the class-level View ⋁c button to display the [Slot Form](#). See [Viewing a Slot](#) and [Editing Slot Properties](#) for more information.

Next: [Clearing Overrides From a Slot](#)
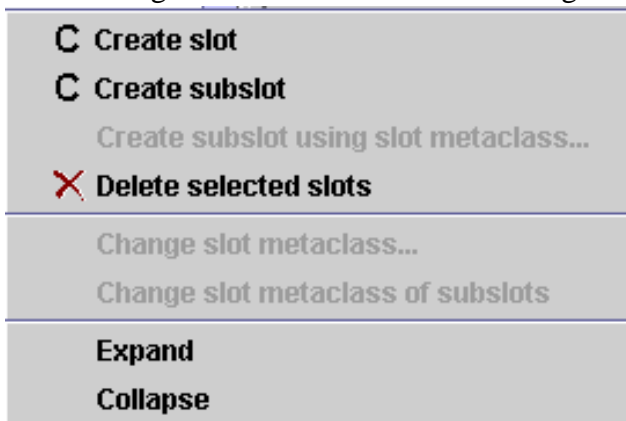
[Slots Table of Contents](#)

The Slot Menu

Whenever you have a slot selected in the Slot Form, you can access the cascading slot menu by clicking the right mouse button. This menu allows you to perform a number of slot-related tasks.

To access the slot menu:

1. Select a slot in the Slot Form.
2. Click the right mouse button. The cascading slot menu is displayed.

3. Make your selection and click the left mouse button.

The class menu allows you to perform the following tasks. Not all tasks are available at all times; tasks that cannot be performed are greyed.

- **C Create slot**: Creates a new slot. This operation is identical to clicking the **Create C** slot button. See Creating a slot for more information.
- **C Create subslot**: Creates a subslot subordinate to the highlighted slot. See Creating a Subslot for more information.
- **Create subslot using slot metaclass...**: (Note that metaclasses are an advanced feature; you should have a good understanding of Protégé before you use metaclasses.) If you have added slot metaclasses to your project, allows you to create a new subslot using a slot metaclass as a template.
- **✕ Delete selected slots**: Deletes the highlighted class and all of its subclasses, removing it from the current project. This operation is identical to the **Delete ✕** class button. See Deleting a Class for more information.
- **Change slot metaclass**: (Note that metaclasses are an advanced feature; you should have a good understanding of Protégé before you use metaclasses.) Changes the slot metaclass of the highlighted slot. See Changing the Metaslot of a Slot for more information.
- **Change slot metaclass of subslots**: (Note that metaclasses are an advanced feature; you should have a good understanding of Protégé before you use metaclasses.) Changes the metaslot of all subordinate slots to the metaslot of the highlighted slot.

- **Expand**: Shows all slots subordinate to the highlighted slot. This is a multi-level display operation that is more extensive than clicking the ⊙ icon, which only shows the next level of direct subslots.
- **Collapse**: Hides all slots subordinate to the highlighted slot. This is a multi-level display operation that is more extensive than clicking the ♀ icon to the left of the class.

Next:

The Classes Tab

The Classes tab provides a single window in which you may view, create, and edit **classes**, which model concepts in your domain. An example is shown below. The window consists of three panes:

1. The Class Relationship pane in the upper left shows classes in a hierarchy and allows you to edit, create, and delete new classes. It also allows you to rearrange the class hierarchy by dragging a class to a replacement superclass.
2. The Superclasses pane in the lower left shows the superclasses of the selected class and allows you to add and remove superclasses for a class, as well as jump to a different superclass by clicking on it.
   Note: If you cannot see the Superclasses Pane, your window may be too small. You can see the pane by enlarging your window or by dragging the slider bar at the bottom of the Class Relationship Pane. See Working With a Small Window for more information.
3. When a single class is selected, the Edit pane on the right contains the Class Form for the selected class. The Class Form allows you to: name the class, choose its role, define constraints, provide a brief note, and, most importantly, define and edit the template slots. The Class Form can also be displayed as a separate window by clicking the View V icon in the Class Relationship pane.

newspaper Protégé-2000   (C:\Program Files\Protege-2000\examples\newspaper\newspaper.pprj)

Project   Edit   Window   Help

Classes   **Slots**   **Forms**   **Instances**

Relationship   Subclass ▼   V C ⌂ ✕

- :THING **A**
  - :SYSTEM-CLASS **A**
  - Diagram_Entity
  - TableEntity
  - Author **A**
    - News_Service
    - Columnist **M**
    - Editor **M**
    - Reporter **M**
  - Content **A**
  - Layout_info **A**
  - Library
  - Newspaper
  - Person **A**
  - Section **A**

Class Relationship Pane

🔍

Superclasses   + −

- Author **A**
- Employee **A**

Superclasses Pane

Editor (instance of :STANDARD-CLASS)

Name   **Constraints**   V C + −   **Documentation**

Editor   Editors are responsible for the content of sections.

Role

Concrete ▼

Template Slots   V V₀ C ✖ + −

Name	Type	Cardinality	Default	Other Facets
S byname	String	Single		
S current_job_title	String	Single		
S date_hired	String	Single		
S name	String	Single		
S other_information	String	Single		
S phone_number	String	Single		
S salary	Float	Single		
S sections ○	Instance	Multiple		classes={Section}
S supervises	Instance	Multiple		classes={Employee}

Class Edit Pane
(displaying Class Form)

◄ ►

For information about the Classes tab user interface and about accomplishing specific tasks, see the Classes Table of Contents.

The Template Slot Buttons

The Template Slot buttons, V V C X + − , located at the top right of the Template Slots pane, allow you to view and edit, create, or add, or remove a slot for the current class.

You can also view, create, and delete slots, as well as view back-references, directly from the Slots Tab. See the Slot Buttons for more information.

The buttons have the following actions:

V

V(iew) Top-Level Slot button: Click this button to open the top-level Slot Form for the highlighted slot. If you make any overrides to the slot at the top level, they affect the slot everywhere it appears, including the Slots Tab and any classes where the slot is attached. See Viewing a Slot.

V

V(iew) Slot at Class button: Click this button to open the class-level Slot Form for the highlighted slot. If you make any overrides to the slot at the class level, they only affect the slot at the class and any subclasses. The slot remains unchanged at the Slots Tab and any unrelated classes where appears. A slot that has been overridden has an ° override icon in the left column. There are limits on the overrides that you can make to an inherited slot. See Viewing a Slot for more information.

C(reate) button: Click this button to create a new slot for the current class. See [Creating a Slot](#). If the Create **C** button is disabled, the selected class cannot be edited. Classes cannot be edited if they either are a system class or are included from another project. See [Including a Project](#) for information about project inclusion.

Clear Slot Overrides button: Click this button to remove any slot overrides. See [Clearing Overrides From a Slot](#) for more information.

Add button: Click this button to add a pre-existing slot to the current class. See [Adding a Slot](#).

Remove button: Click this button to remove the highlighted slot from the current class. The slot remains in the project and can be viewed via the [Slots Tab](#). If the Remove **–** button is disabled, the selected slot cannot be removed from this class. Slots that are inherited from a superclass class cannot be removed. See [Removing a Slot](#) for more information. To delete a slot from the project, use the Delete ✕ [slot button](#) at the [Slots Tab](#). See [Deleting a Slot](#) for more information.

The Template Slots Pane

The Template Slots pane at the bottom of the Class Form displays the direct and inherited slots for the selected class. Slots, which represent properties of your class, are a crucial part of your knowledge base.

The Template Slots pane provides the following information:

1. The Template Slot Buttons, which allow you to view, create, clear, and add slots.
2. The Slot Icons, which give information about the slot.
3. The Slot Summary Columns, which summarize the information which was entered or edited in the Slot Form or Slots Tab.

In the example, **Editor** has several slots which appear in the Template Slots pane.

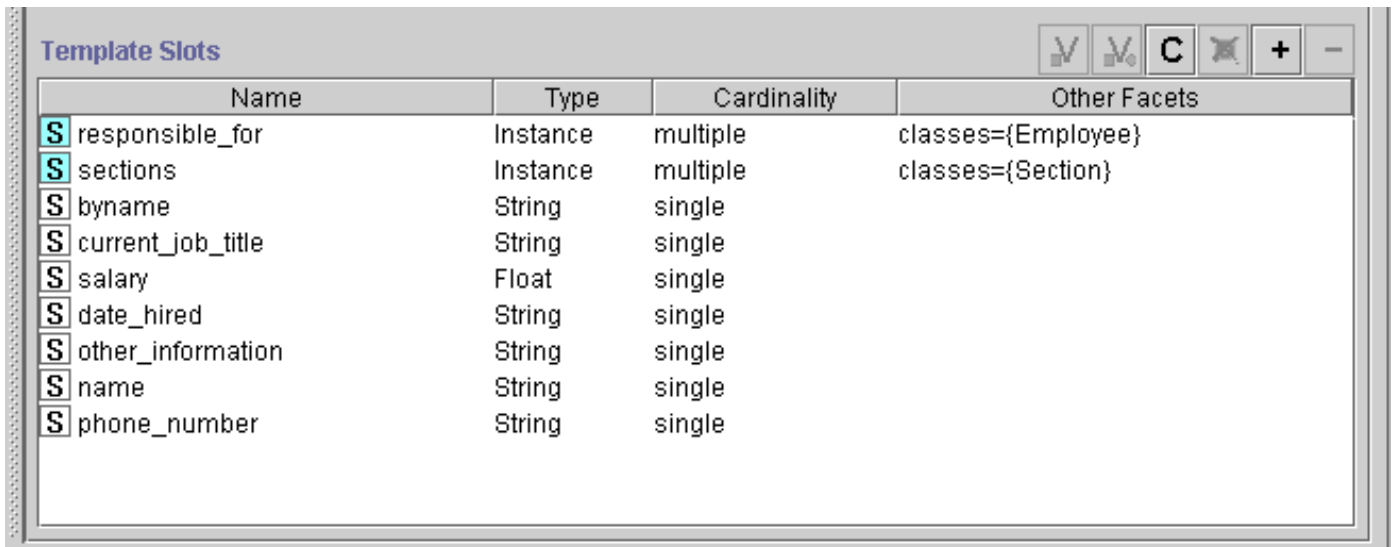| Template Slots			∀ ∀ C ✖ + −
Name	Type	Cardinality	Other Facets
S responsible_for	Instance	multiple	classes={Employee}
S sections	Instance	multiple	classes={Section}
S byname	String	single	
S current_job_title	String	single	
S salary	Float	single	
S date_hired	String	single	
S other_information	String	single	
S name	String	single	
S phone_number	String	single	

The Template Slot Buttons

The buttons at the upper right of the Template Slots pane have the following actions. See the separate Template Slot Buttons topic for more information:

You can also view, create, and delete slots, as well as view back-references, directly from the Slots Tab. See the Slot Buttons for more information.

The Slot Icons

The icons at the left of the column indicate where the slot was created.

S   A gray icon means that the slot is inherited from one of the selected class's ancestors. Inherited slots can be edited but not deleted. **byname** is an inherited slot.

A blue icon means that the slot is "direct", that is, it was created directly on the selected class. **responsible_for** is a direct slot.

An icon to the right of the column indicates a slot that has been modified:

An O (override) means that the slot has been modified at the class. If the slot was created directly and modified later, this icon is still displayed.

An I (inverse) means the slot has an inverse slot. Inverse slots allow you to link two slots in a reciprocal relationship. See Understanding Inverse Slots for more information.

Slot Summary Columns

The remaining columns in the Slot Templates Pane summarize the information that was entered or edited in the Slot Form or Slots Tab.

Column	Meaning
Slot Name	Displays the name of the slot.
Type	Indicates the kind of values that the slot may hold. Available types are: **Any**, **Boolean**, **Class**, **Float**, **Instance**, **Integer**, **String**, and **Symbol**. See The Value Type Menu for more information.
Cardinality	Indicates whether a slot value can consist of **Multiple** items or must be a **Single** item.
Other Facets	Displays any other facets defined in the Slot Form or Slots Tab. The content of this column depends on the type of the slot. For type Integer slots, for example, the range can be displayed in the last column. For type Instance slots the allowed classes are displayed. The **published_in** slot is an Instance slot with an allowed class of **Newspaper**. This column also indicates any required values or defaults.

Next: The Template Slot Buttons

Classes Table of Contents
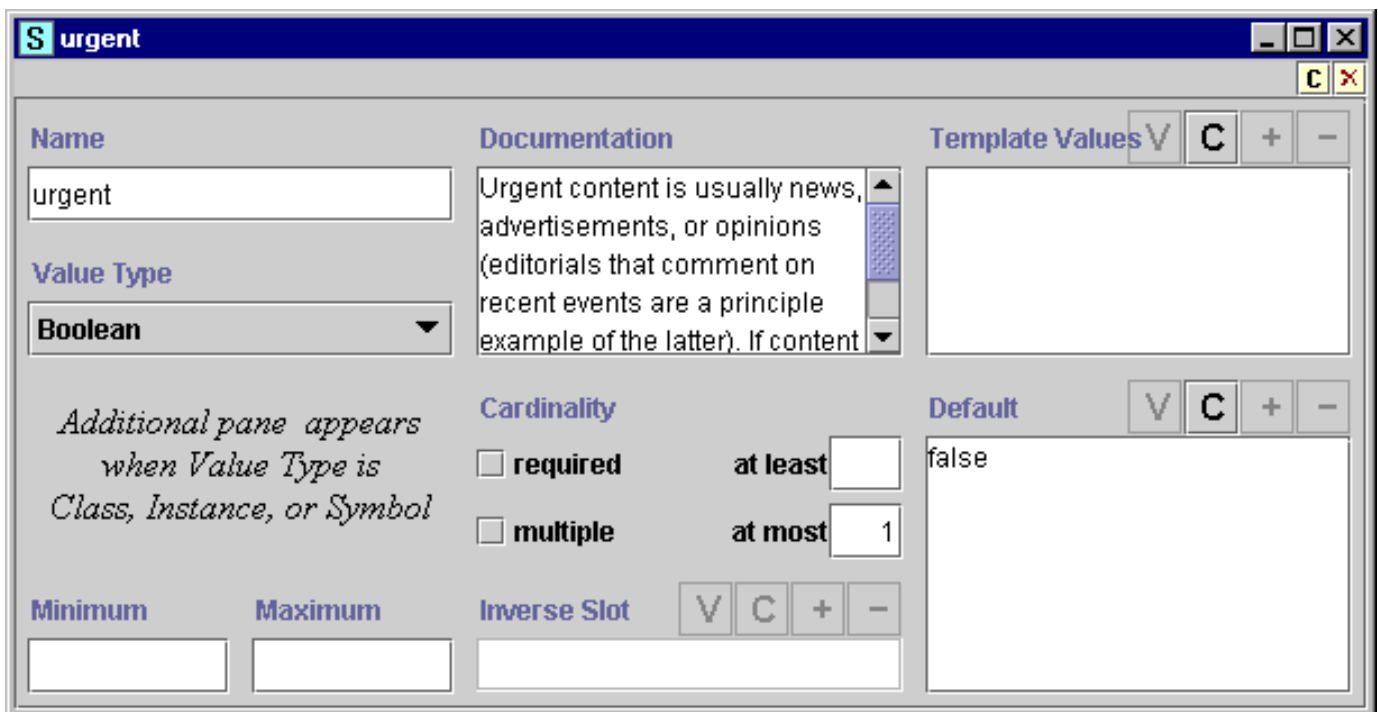
The Slot Form

The Slot Form can be used to define and edit the attributes of a slot. The Slot Form for the selected slot is displayed in the Slot Edit Pane at the right of the Slots Tab. The Slot Form can also be displayed as a free-standing window as follows:

- When you highlight a slot in the Slots Tab and then click the View **V** slot button.
- When you create a new slot for a highlighted class by clicking the Create **C** button in the Template Slots pane of the Classes tab.
- When you highlight a slot in the Template Slots pane and then click the Top-Level View **V** or the Class-Level View **V** Template Slot button.

In the example below, the slot **urgent** for the class **Article** was double-clicked in the Template Slots pane. The slot form displays a **Boolean** Type, with **Single** Cardinality, and a text documentation pane of the **urgent** slot.

A slot can be used for more than one class. Each slot has a top-level (system) description; the system description can be specialized for a specific class. The scope of your edits depends on how you access the Slot Form. See Viewing a Slot for more information:

- When you access the Slot Form via the Slots Tab or via the Top-Level View **V** button in the Template Slots pane of the Classes tab, you can view and edit the top-level description; these modifications affect the slot for any class where it occurs.

- When you access the Slot Form via the Class-Level View ![icon] button in the [Template Slots pane](#) of the [Classes tab](#), you can view and edit the slot properties for the currently highlighted class. Class-level modifications only affect the slot at the current class and its subclasses. This type of editing is called overriding.

The Slot Form displays the following information for a slot:

1. The [Note Icons](#), which allow you to add notes.
2. The [Name](#) of the slot.
3. The [Value Type](#) of the slot.
4. The [Cardinality](#) of the slot.
5. (optional) The [Minimum](#) and [Maximum](#) values for the slot. These fields are only relevant to certain slot types.
6. (optional) Any [Documentation](#) that has been entered for the slot.
7. (optional) Any [Inverse Slots](#) for the slot.
8. (optional) Any [Template Values](#) that have been defined for the slot.
9. (optional) Any [Defaults](#) that have been defined for the slot.

For certain value types, an additional pane appears below the **Value Type** pane:

1. For type Class, the Allowed Parents pane appears.
2. For type Instance, the Allowed Classes pane appears.
3. For type Symbol, the Allowed Symbols pane appears.

Note Icons

The note icons, ![icons], at the upper right of the form allow you to add and remove yellow sticky notes to your class. The note is always displayed along with the form. For information on how to add notes to any frame (class, instance, or slot), see [Working with Notes](#).

Slot Name

You can edit the slot name directly in the **Name** field. Slot names are case-sensitive. A recommended convention is to make slot names lowercase, and with words separated with an underscore (_).

Value Type

The **Type** of the slot determines the kind of values that the slot may hold. The available types are summarized in the following table:

Type	Description	Examples
Any	Any of the types below (logical Union)	
Boolean	Logical Value	True, False

Class	Class in the knowledge base	Organization
Float	Number with a decimal point	1.0, 3.4e10, -0.3e-3
Instance	Instance of a class in the knowledge base	instance_00010
Integer	Whole number	1, 2, -4
String	String of alphanumeric characters, possibly including spaces	"John Doe"
Symbol	List of values, which may not include spaces	red, blue and green

For a full description of the types, see The Value Type Menu.

Cardinality

The **Cardinality field** allows you to specify the number of values allowed or required for the slot. The default configuration allows the slot to have at most one value, that is, it can have one value or no value.

You can change the default by entering a positive whole number in the **at least** and/or **at most** options, or by selecting the **multiple** option with no **at most** value.

- **required/at least** sets the minimum number of values for the slot. To use this, enter a positive whole number in the **at least** entry bar. Most common is to set **at least** equal to one, which requires a value for the slot. For example, a setting of **at least** equal to 1 *and* **at most** equal to 1 means that the slot must have exactly one value. If you enter a value for **at least**, **required** is automatically selected. If you set **at least** greater than one, **multiple** is automatically selected.
- **multiple/at most** means that the slot can contain multiple values, but that there is a limit to the number of values. If you set **at most** greater than one, **multiple** is automatically selected.
- **multiple** with *no* **at most** value indicates that the value of a slot may contain any number of items. (For example, the slot **keywords** in the class **Article** allows multiple values. This means an instance of **Article** can have more than one keyword.) **multiple** is stored as a list, and duplicate values are allowed, although their use is uncommon.

Minimum (optional)

This field is applicable only to slots of type **Integer** or **Float**.

Minimum allows you to specify a minimum value for your slot. When an instance is created for a class with this slot, the value of the slot must be greater than or equal to the minimum. For example, a minimum of zero means instances cannot have negative values. Together, **Minimum** and **Maximum** can be used to define an allowable range.

When present, the **Minimum** value is displayed in the last column of the Template Slots pane.

Maximum (optional)

This field is applicable only to slots of type **Integer** or **Float**.

Maximum allows you to specify a maximum value for your slot. When an instance is created for a class with this slot, the value of the slot must be less than or equal to the maximum. Together, **Minimum** and **Maximum** can be used to define an allowable range.

When present, the **Maximum** value is displayed in the last column of the [Template Slots pane](#).

Documentation (optional)

The **Documentation** field allows you to enter a text description of the slot. Filling in this field is optional but is recommended to make maintaining the knowledge base easier.

See [The Slot Buttons](#) and [The Template Slot Buttons](#) for the operations you can perform on slots.

Inverse Slot (optional)

Only available for slots of type Class or Instance. Allows you to create a reciprocal relationship between two slots. If this relationship is set up correctly, assigning a value (i.e., a specific class or instance) to the slot for one instance automatically assigns the instance as a value to the appropriate inverse slot. For example, the "direct superclass/direct subclass" relationship is actually an inverse slot relationship. See

Inverse slots are designated by an Inverse icon to the right of the slot.

Template Values (optional)

Allows you to specify the value(s) for a slot at the class level. This value is filled in for all classes and instances that use or inherit the slot. A template value is a required value; it can *not* be changed or overridden at the instance level. For a value that can be overridden, use **Defaults** instead.

The number of **Template Values** should not exceed the **at most** value for the **[Cardinality](#)** of the slot.

When present, any **Template Values** are displayed in the Other Facets column of the [Template Slots pane](#) in the [Classes tab](#). They also appear automatically in the slot value field of the Class Form or Instance Form of any class or instance created with that slot.

Defaults (optional)

Allows you to specify the default value(s) for a slot. When an instance is created for a class that has a slot with a defined default, the default value is automatically entered as the value of the slot. The default value

can then be changed or overwritten.

The number of **Defaults** should not exceed the **at most** value for the **Cardinality** of the slot.

When present, any **Defaults** are displayed in the Other Facets column of the Template Slots pane in the Classes tab. They also appear automatically in the appropriate slot value field of any instance created with that default.

Viewing Several Slots

To view the information for several slots at once, select the slots at the Slots Tab and click the View ⋁ slot button to open the Slot Form for each class. To highlight multiple slots, hold down the Ctrl key while clicking each slot. To highlight a range of slots, click the first slot, then hold down the Shift key and click the last slot in the range. You can also view multiple slots using the Top-Level View ⋁ or Class-Level View ⋁ template slots buttons in the Template Slots pane.

Opening a new slot form does not close the previous form. This allows you to compare the attributes for two or more slots. Edits can be made directly in any open form.

If you have multiple forms open, you can manage them as follows:

- Cascade multiple forms by clicking the Cascade 🖿 button below the main menu bar, or by selecting **Cascade Windows** from the **Windows** menu..
- Close all open forms by clicking the **CloseAllWindows** 🗙 button below the main menu bar, or by selecting **Close All Windows** from the **Windows** menu.

Next: The Value Type Menu

Slots Table of Contents

Editing Slot Properties

You can edit the properties of a newly created or existing slot using the Slot Form.

Slot Name

To change the name of a slot, edit the text in the **Name** field. Slot names are case sensitive. To distinguish slot names from class names, a recommended convention is to make slot names lowercase.

Cardinality

To change whether or not a slot can be composed of more than one item, select a different options for the **Cardinality**. The default configuration allows the slot to have at most one value, that is, it can have one value or no value.

You can change the default by entering a positive whole number in the **at least** and/or **at most** options, or by selecting the **multiple** option with no **at most** value.

- To require a minimum number of values for the slot, enter a positive whole number in the **at least** entry bar. Setting **at least** equal to one requires a value for the slot. Setting **at least** equal to 1 *and* **at most** equal to 1 means that the slot must have exactly one value. If you enter a value for **at least**, **required** is automatically selected. If you set **at least** greater than one, **multiple** is automatically selected.
- To allow the slot to have multiple values, select **multiple**.
- To allow the slot to have multiple values, but to limit the number of values allowed, enter the maximum number of values in the **at most** entry var. If you set **at most** greater than one, **multiple** is automatically selected.

NOTE: If you are editing the slot for a specific class, and the slot is inherited, you cannot change the cardinality from **single** to **multiple**. The **at least** field must be greater than or equal to any inherited **at least**. The **at most** field must be less than or equal to any inherited **at most**.

Value Type

To change the value type of the slot, select a different type from the Value Type menu. The available types are summarized in the following table:

Type	Description	Examples
Any	Any of the types below (logical Union)	
Boolean	Logical Value	True, False
Class	Class in the knowledge base	Organization
Float	Number with a decimal point	1.0, 3.4e10, -0.3e-3
Instance	Instance of a class in the knowledge base	instance_00010
Integer	Whole number	1, 2, -4
String	List of alphanumeric characters, possibly including spaces	"John Doe"
Symbol	Enumerated list of values, which may not include spaces	red, blue and green

NOTE: If you are editing the slot from a specific class, and the slot is inherited, the following restrictions apply:

- For an inherited slot of type **Any**, you can select any *one* of the other types.
- For an inherited slot of a value type other than **Any**, you *cannot* change the type of the slot. However, for slots of type **Class**, **Instance**, or **Symbol**, you can change the choices in the associated **Allowed** menu.

When a value type of **Class**, **Instance**, or **Symbol** is selected, an additional pane appears below the Value Type menu, as follows:

Value Type	Associated Pane
Class	The **Allowed Parents** pane. See Allowed Parents in the Value Type menu for information on how to add and remove allowed parents.
Instance	The **Allowed Classes** pane. See Allowed Classes in the Value Type menu for information on how to add and remove allowed classes.
Symbol	The **Allowed Values** pane. See Allowed Values in the Value Type menu for information on how to add and remove allowed values.

For more information, see The Value Type Menu.

Defaults (optional)

Allows you to specify the default value(s) for a slot. When an instance is created for a class that has a slot with a defined default, the default value is automatically entered as the value of the slot. The default value can then be changed or overwritten.

For a slot with **Single** cardinality, **Defaults** must be a single value or no value; for a slot with **Multiple** cardinality, **Defaults** can be more than one item.

When present, any **Defaults** are displayed in the second-to-last column of the Template Slots pane in the Classes tab.

Minimum (optional)

For a slot of type **Integer** or **Float**, you can change the minimum value by typing the new value in the **Minimum** field. If you are editing the slot for a specific class, and the slot is inherited, the new minimum value must be *greater than or equal to* the inherited minimum.

Maximum (optional)

For a slot of type **Integer** or **Float**, you can change the maximum value by typing the new value in the **Maximum** field. If you are editing the slot for a specific class, and the slot is inherited, the new maximum value must be *less than or equal to* the inherited maximum.

Documentation (optional)

You can change the text description of the slot directly in the documentation field.

Next:
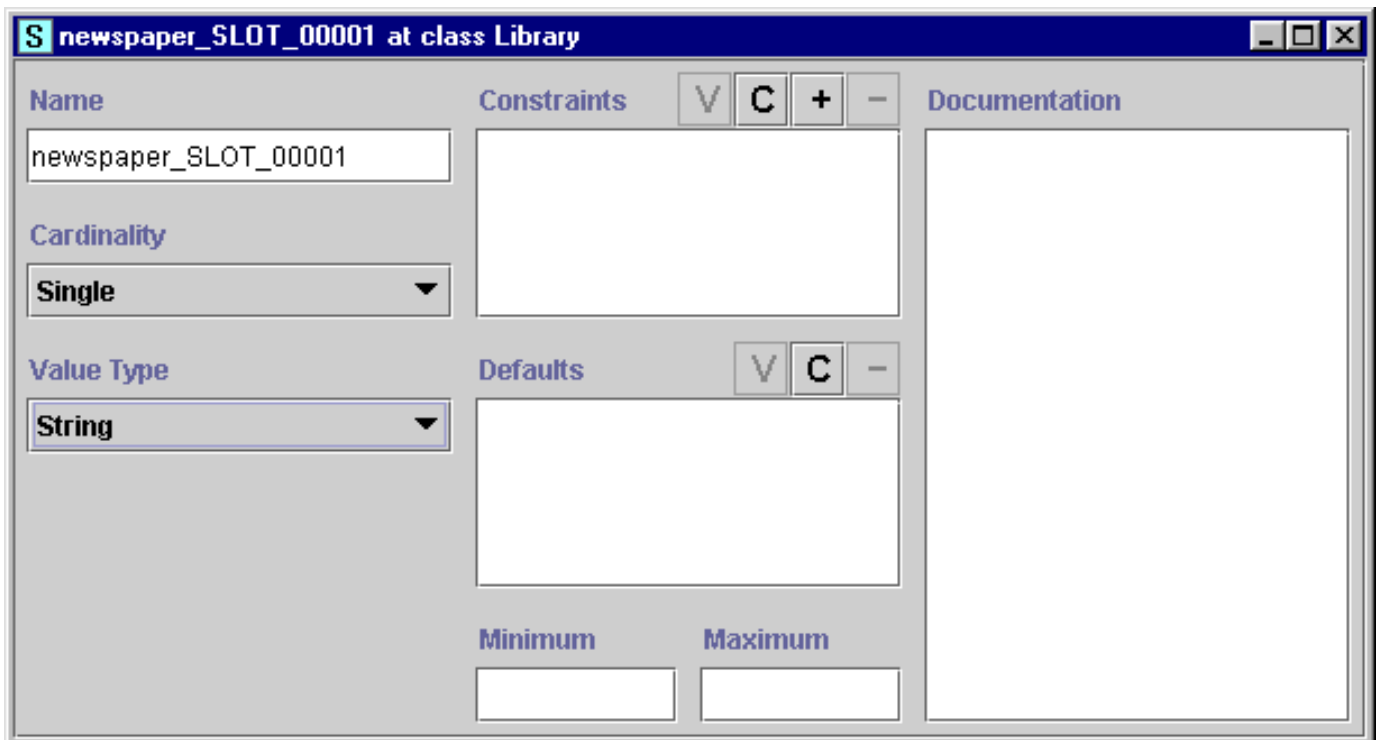
Viewing a Slot

You can view and edit a slot at two levels:

- View/edit the top-level properties of a slot. This changes the slot everywhere it appears, including the Slots Tab and all the classes that reference the slot.
- View/edit the slot at a class. This creates overrides that appear at the class and any subclasses. The slot remains unchanged at the Slots Tab, any superclasses, and any unrelated classes. A slot with overrides is shown with an ○ override icon in the Template Slots pane.

In either case, you can edit the properties of the slot directly in the Slot Form. See Editing Slot Properties for more information. Any changes you enter into the Slot Form take effect immediately. To make the changes permanent, save the project by selecting **Save** from the **Project** menu.

Viewing a Slot at the Top-Level

You can edit the top-level properties of a slot directly from the Slots Tab or by using the top-level View button at the Template Slots pane. Editing at the top-level changes the definition of the slot. The changes appear everywhere the slot occurs, including the Slots Tab and all the classes that reference the slot.

To edit the top-level properties of a slot from the **Slots Tab**:

1. Select the slot you wish to edit in the Slot pane of the [Slots Tab](#).
2. Click the View ⋎ [slot button](#). This opens the [Slot Form](#) for the selected slot.

To edit the top-level properties of a slot from the **Template Slots pane** in the [Classes Tab](#):

1. Select a class in the [Class Relationship pane](#) in the [Classes tab](#).
2. Select the slot you wish to edit in the [Template Slots pane](#).
3. Click the View Top-Level Slot ⋎ [Template Slot button](#) at the upper right of the template [Template Slots pane](#) *or* double-click the slot and make sure View top-level slot is selected in the Select Slot View dialog box, then click OK.
   This opens the [Slot Form](#) for the selected slot. Note that there may be restrictions on the edits that can be performed on an inherited slot.

Viewing a Slot at a Class

You can also edit a slot at the class level. Class-level modifications only affect the slot at the current class and its subclasses. To edit the slot properties for a specific class:

1. Select a class in the [Class Relationship pane](#) in the [Classes tab](#).
2. Select the slot you wish to edit in the [Template Slots pane](#).
3. Click the View Slot at Class ⋎ [Slot button](#) at the upper right of the [Template Slots pane](#) *or* double-click the slot and make sure View slot at class is selected in the Select Slot View dialog box, then click OK. This opens the [Slot Form](#) for the selected slot. Note that there may be restrictions on the edits that can be performed on an inherited slot.

A slot that has been edited at a class is shown with an override ⬡ icon in the [Template Slots pane](#).

Next: [Editing Slot Properties](#)

Slots Table of Contents

The User Interface

- **The Slots Tab**
- **The Slot Buttons**
- **The Slot Menu**
- **The Slot Form**
- **The Value Type Menu**

Slot Tasks

- **Creating a Slot**
- **Viewing a Slot**
- **Editing Slot Properties**
- **Removing a Slot From a Class**
- **Deleting a Slot From the Project**
- **Adding an Existing Slot to a Class**
- **Clearing Overrides From a Slot**
- **Understanding Inverse Slots**
- **Creating an Inverse Slot Relationship**
- **Creating a Subslot**

Slot Metaclass (Metaslot) Tasks

- **Understanding Metaslots**
- **Creating a Metaslot**
- **Setting the Default Metaslot**
- **Changing the Metaslot of a Slot**

Next: Forms Table of Contents

Importing a Project

Importing a project means to build a Protégé-2000 project from files which are not in Protégé-2000 format. For example, these might a .pont file from a previous version of Protégé or an RDF file.

You can import a project from the external formats that Protégé supports. The following formats can be imported. Each format is described in a separate help topic:

- Text Files: You can import a project from two text files describing the classes/slots and instances information. Importing a text project can be used, for example, for updating from Protégé/Win to Protégé-2000.
- Database Table: You can import a project from a table in a JDBC database.
- Resource Description Framework (RDF) Files: You can import a project from two RDF files that describe the classes/slots and instances information.

Protégé will generate default forms for all classes in the imported project.

Next: Importing Text Files

Project Table of Contents

Importing Text Files

To import a Protégé-2000 project from text files:

1. Select **Import Project...** from the **Project** menu.
2. Select *Standard Text Files* from the Select Format dialog box and click OK.

3. A **Text Files** dialog box is displayed.

Clips 6.0 Text Files

Included Projects

Classes file name

Instances file name

OK   Cancel
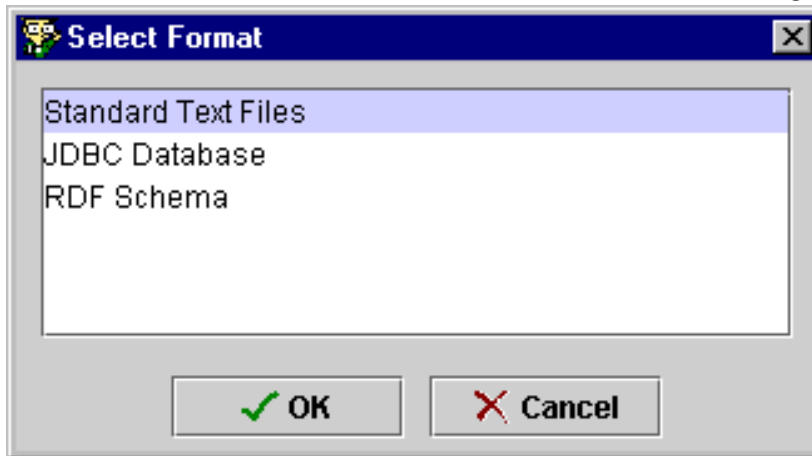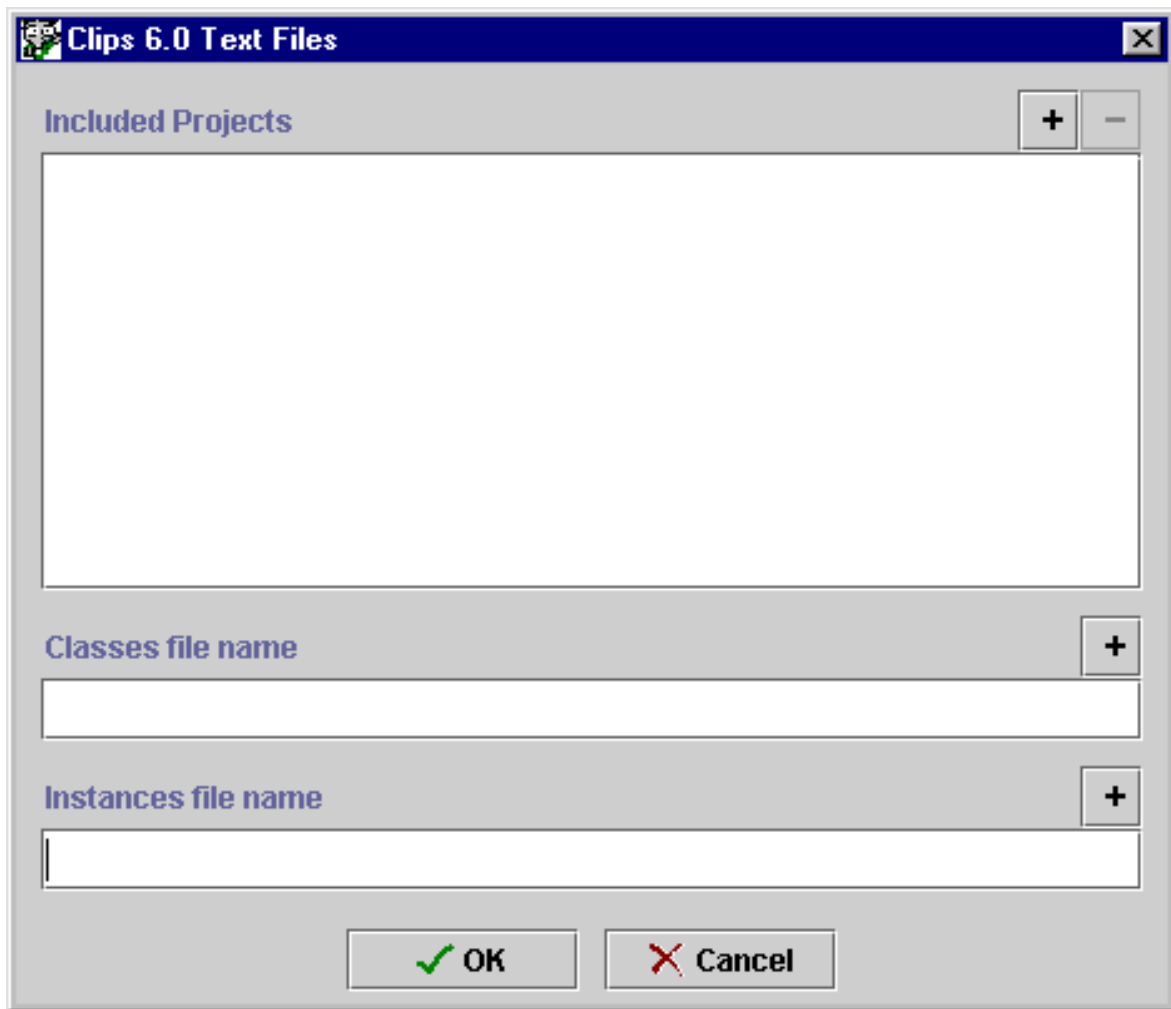
4. If you wish to include any existing Protégé-2000 projects in the project you are building, click the **+** button to the right of the **Included Projects** line and select your project. See Including a Project for more information.

5. Enter the location and name of the text file that contains the class information in the **Classes file name** line. Normally, this will be a file with the extension .pont. To browse for a file, click the **+** button.

Classes file name   **+**

c:\program files\protege-2000\examples\newspaper\newspaper.pont

6. Enter the location and name of a text file that contains the compatible instance information in the **Instances file name** line. Normally, this will be a file with the extension .pins. To browse for a file, click the **+** button.

Instances file name   **+**

c:\program files\protege-2000\examples\newspaper\newspaper.pins

7. Click OK.

Protégé-2000 will create a project file from your class and instance information. To save your project and give it a name, select **Save** from the **Project** menu. See [Saving a Project](#) for more information.

Next: [Importing a Database Table](#)

[Project Table of Contents](#)
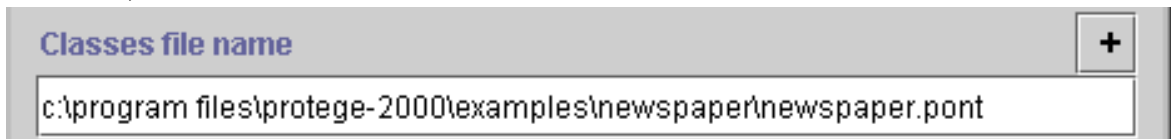
Importing a Database Table

1. Select **Import Project...** from the **Project** menu.
2. Select *JDBC Database* from the Select Format dialog box and click OK.

3. A **JDBC Database** dialog box is displayed.

4. Enter the name of your JDBC driver in the **JDBC Driver** line of the dialog box.
5. Enter the URL for your database in the **JDBC URL** line.
6. Enter the name of the table that contains the information you wish to import.
7. If your database requires a username and/or password, enter these in the appropriate lines of the dialog box.

Protégé-2000 will create a project file from your class and instance information. To save your project and give it a name, select **Save** from the **Project** menu. See Saving a Project for more information.
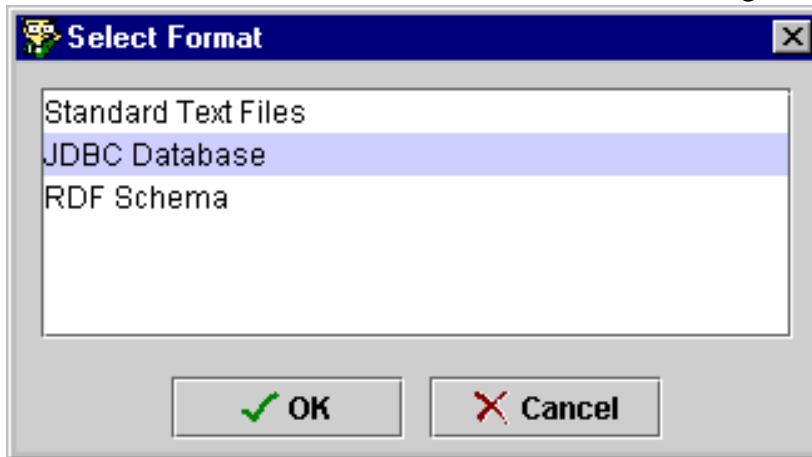
Next:

Importing RDF Files

You can convert a project in RDF format to Protégé-2000 by importing it. The original project is not altered. If you have no schema for your RDF (your instances do not have a declared type) the system will create a single class with the required properties and all imported instances will be instances of this class. You can then use Protégé-2000 to create a schema and to assign your existing instances to the appropriate classes.

To import existing RDF files:

1. Select **Import Project...** from the **Project** menu.
2. Select *Resource Description Framework (RDF)* from the Select Format dialog box and click OK.
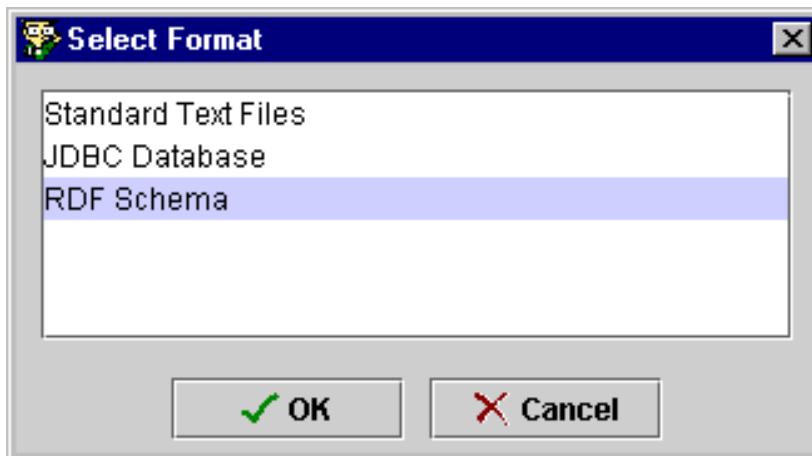
3. A **Resource Description Framework (RDF)** dialog box is displayed.

Resource Description Framework (RDF)

Included Projects + −

Classes file name +

C:\PROGRAM FILES\PROTEGE-2000\examples\rdf\MotorVehicleToImport.rdfs

Instances file name +

Namespace

http://www.rdfschema.org/mynamespace.rdf

✓ OK   ✗ Cancel
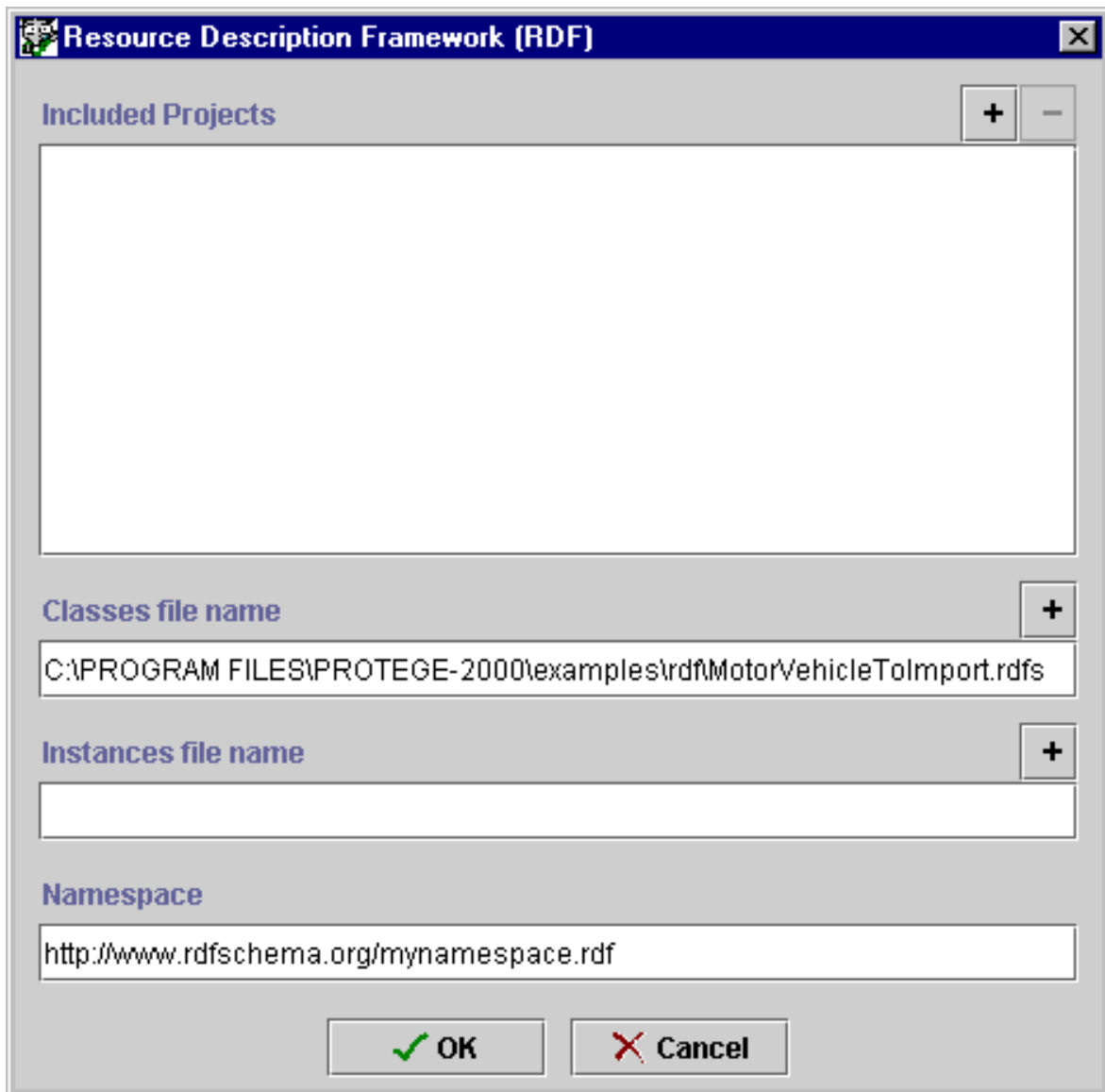
4. Enter the name of the file containing your RDF schema (if any) in the *Classes file name* field. To browse for a file, click the **+** button.
5. Enter the name of the file containing your RDF instance data in the *Instances file name* field. To browse for a file, click the **+** button.
6. Enter your namespace in the *Namespace* field.
7. Press the OK button and the system will load the schema and instances.

For more information about RDF files and Protégé, see RDF Support in Protégé-2000.

Saving a Database Project

You can save a Protégé-2000 project as a single table in a JDBC database. Before you do this, you will need to do the following:

- make sure you have a database program installed on your system
- make sure you have a JDBC driver that is compatible with your database and the version of the Java VM that you are using
- add the driver to Protégé's path by placing the jar file of the driver in the plugins subdirectory of the Protege-2000 directory.

Note: When a project is saved in a JDBC database, backend changes to the database (for example, changes to slots, classes, and instances) are committed as soon as they are made. However, changes to the User Interface (which is stored in the .pprj file) are not committed until you press the Save button.

Once you have set up your JDBC database, you can save your Protégé-2000 project as a database files in one of two ways:

- When the project is first created, by selecting **JDBC Database** in the Select Format dialog box.
- By selecting **Save in Format...** from the **Project** menu and then selecting **JDBC Database** in the Select Format dialog box.

After you have chosen your format, choose the name and location for the project as follows:

1. When you have selected **JDBC Database**, the Single Table JDBC Database dialog box is displayed:

2. To select or change the name or location of the **pprj** file, enter the information you want in the **Project** line of the dialog box. Be sure to give the file a **pprj** extension. To browse for a new location, cllick the ✛ button. If you do not specify a location for the project, by default, the project is saved in the directory where Protégé-2000 is installed.

3. Enter the class name of your JDBC driver in the **JDBC Driver** line of the dialog box. The class name can be found in the documentation provided with your particular driver.
(For Microsoft Access, use: sun.jdbc.odbc.JdbcOdbcDriver.)

4. Enter the URL for your database in the **JDBC URL** line.
(For Microsoft Access, navigate to the ODBC Data Source Administrator dialog via the Control Panel and add a User DSN for your Access database. If you're unfamiliar with adding a User DSN, please refer to Microsoft Windows Help. Your URL will look like: jdbc:odbc:<User DSN name>)

5. Protégé-2000 will automatically use the name of the existing **pprj** file as the name of the table in the database. If you wish to change this, type the new name for the database in the Table line of the dialog box.

6. If your database requires a username and/or password, enter these in the appropriate lines of the dialog box.

7. Click OK.

You can store multiple projects in the same database by giving their tables different names.

Next:

Saving an RDF Project

This version of Protégé-2000 provides support for creating and editing Resource Description Framework (RDF) schema and instance data. (For more information about RDF, see http://www.w3.org/RDF.) You can use Protégé-2000 to design RDF schema and create the corresponding instance data. You can also view and edit your existing RDF files in Protégé-2000.

You can save your Protégé-2000 project in RDF format in one of two ways:

- When the project is first created, by selecting **Resource Description Framework (RDF)** in the Select Format dialog box.
- By selecting **Save in Format...** from the **Project** menu and then selecting **Resource Description Framework (RDF)** in the Select Format dialog box. Subclasses of :THING from a non-RDF project will become subclasses of rdfs:Resource when you save as RDF.

After you have chosen your format, choose the name and location for the project as follows:

1. When you have selected **Resource Description Framework (RDF)**, the RDF dialog box is displayed:

2. To select or change the name or location of the **pprj** file, enter the information you want in the **Project** line of the dialog box. Be sure to give the file a **pprj** extension. To browse for a new location, cllick the **+** button. If you do not specify a location for the project, by default, the project is saved in the directory where Protégé-2000 is installed.

3. Protégé-2000 will create two files: one for the schema (classes and slots) and the other for the instances. By default, these use the name of the existing **pprj** file. If you wish to change the name

of the schema file, type the new name in the Classes file name line of the dialog box. Make sure to give the file an **.rdfs** extension.

4. If you wish to change the name of the instances file, type the new name in the Instances file name line of the dialog box. Make sure to give the file an **.rdf** extension.

5. Enter the RDF namespace for your project in the Namespace field. All classes, slots, ad instances i the project are placed in this namespace.

6. Click OK.

You should create new RDF classes as subclasses of `rdfs:Resource`.

[Project Table of Contents](#)

RDF Schema Support in Protégé-2000

For version 1.5 of Protégé-2000, we have re-implemented our RDF (Schema) support in order to make it more stable and to make it easier to implement other RDFS-based backends such as OIL and DAML. Furthermore, several features were added, such as

- in addition to a round-trip version, you can export as simple RDF(S) (i.e., Protégé specific facets are not encoded in additional triples);
- support for namespace abbreviations (frame names are never prefixed with complete URIs, only with abbreviations);
- when exporting a project that has included projects, you can put the included projects into different namespaces (see the section "Renaming the (default) namespaces of included projects" below).

Here you will find a comparison of the RDF(S) and Protégé-2000 models.

Importing, creating, and saving RDF(S) files

- To import your existing RDF(S) files into Protégé-2000 select the **Project | Import** menu item and select *RDF Schema* from the storage format dialog. Enter the name of the file containing your RDF schema and the name of the file containing your RDF instance data in the appropriate fields.
- To create a new RDF project, which will include the schema and the instance data, select the **Project | New** menu item and select *RDF Schema* from the storage format dialog.
- An existing Protégé-2000 project can be saved as an RDF(S) project by selecting the **Project | Save In Format** menu item and selecting *RDF Schema* from the storage format dialog. If you check **plain RDFS**, no Protégé specific facets are exported.
- You can use **Project | Save in Format...** to export a project using the old backend into a project using the new one (don't use **Project | Import** for this since the new backend uses different encodings for Protégé specific facets).

Renaming the (default) namespaces of included projects

This feature can be used to transform an existing (Standard Text File/CLIPS) project into an RDF(S) one where frames from included projects reside in *different* namespaces (you can also achieve the reverse effect, i.e. forcing all (or some) projects to share a namespace or have the same namespace as the main project). Here is how it works:

1. Transform the project into an RDF(S) one where *all* projects (main and included ones) have the *same* default namespace. You need to do this "bottom up", i.e., for the included projects first, and finally for the main one. Make sure you save all files with new file names (or in a new directory), or the main file does not load any more (since it will already try to load the new included projects)!
2. Load the new main project and click **Project | Save as ...**. If you have included projects that were loaded with the RDF(S) backend, you will see an **Advanced ...** button. Change some or all of the namespaces and save (the console window will show a remark if namespaces of included projects are renamed). Close the project.
3. For each of the included projects for which you have renamed the default namespace, load and change the namespace (via **Project | Save as ...**).
4. You can now load the main project.

Example

A simple example is in the *Protege-2000/examples/rdf* directory.

Known bugs and problems

- only the RDF Schema namespace http://www.w3.org/TR/1999/PR-rdf-schema-19990303# is supported when saving (on import, the new namespace is also recognized)
- rdfs:seeAlso, rdfs:isDefinedBy, rdfs:label, container, and reified statements are not supported
- multiple types for a single resource are not supported
- you cannot make changes to the standard meta classes (i.e., rdf:Resource, rdf:Property, rdfs:Class, etc.)

- XML Schema data types are not recognized as rdfs:Literal (but it is unclear if they should)

Opening a Project

You can open a existing Protégé-2000 project in one of two ways:

- when you start Protégé.
- from within the Protégé window

Opening a Project When You Start Protégé

To open an existing project when you first start Protégé-2000:

1. Start Protégé.
   A dialog box allows you to create a new project, open a recent project, or get help.
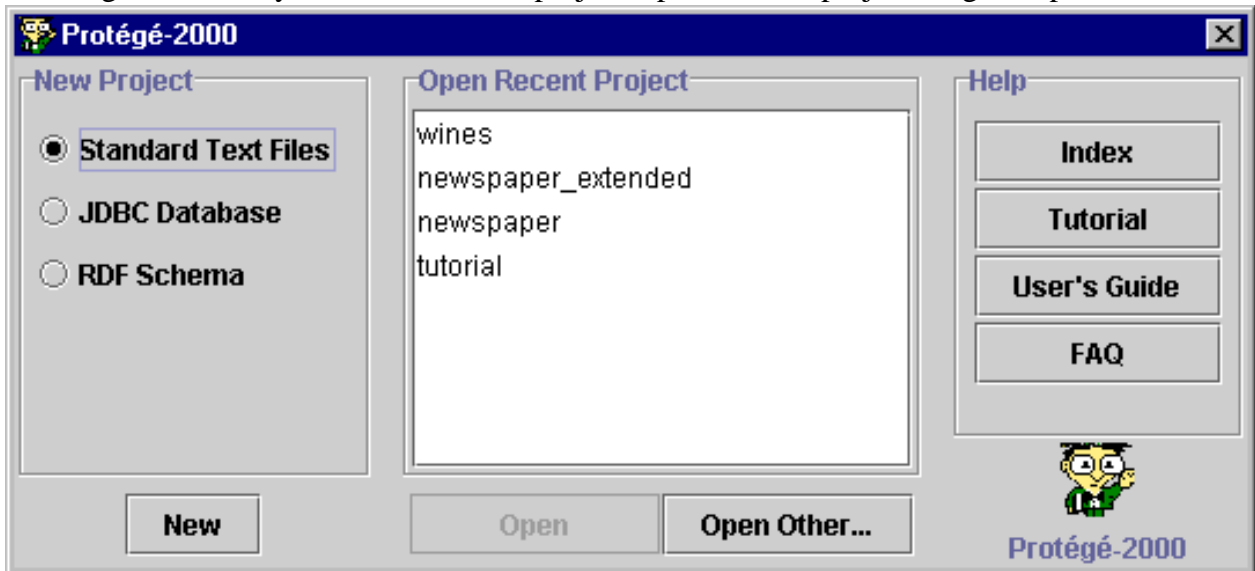
2. To select from the list of recent projects, highlight the project you want from the list and click OK. OR
3. To open a project that is not listed, click **Open Other...** Locate the pprj project you wish to open and click OK.

Opening a Project From Within Protégé

1. Select **Open** from the **Project** menu or click the Open Project button 📂. A Project dialog box allows you to select the project.
2. Select the **pprj** project you wish to open.
3. Click OK.

For Windows systems, you can also open a project by double-clicking on the **pprj** file in Windows Explorer.

No matter what format your project is in, you can create, open, and save your projects directly via the **pprj** file. You do not need to name or access other file types unless you wish to build a project.

To save the open project to disk after you have edited it, select **Save** from the **Project** menu.

Project Table of Contents

Saving a Project

To save a Protégé-2000 project:

1. Select **Save** from the **Project** menu or click the Save Project button ▣.
2. If this is a new project, you will be prompted for the name and location. You only need to choose a name and location for the .pprj file; the other files will be created automatically. If you do not specify a location for the project, by default, the project is saved in the directory where Protégé-2000 is installed.
3. Click OK.

See Renaming a Project for information on how to save the current project with a new name.

Next: Renaming a Project

Project Table of Contents

Renaming a Project

To save a Protégé-2000 project under a new name:

1. Select **Save As** from the **Project** menu.
2. Type the new name for the **pprj** file in the Project line of the window. If you wish to browse for a location, click the ✚ button.
3. By default, the new names are entered automatically for any additional files in the project (for example, **pont** and **pins** files for standard text format). You do not need to name these file types if you wish to use the new names. However, you can choose to give a different name to these files. You might want to do this, for example, if you are creating a new **pprj** file which still accesses the old **pont** and **pins** file names. In this case, enter the names you want for the **pont** and **pins** files.
4. Click OK.

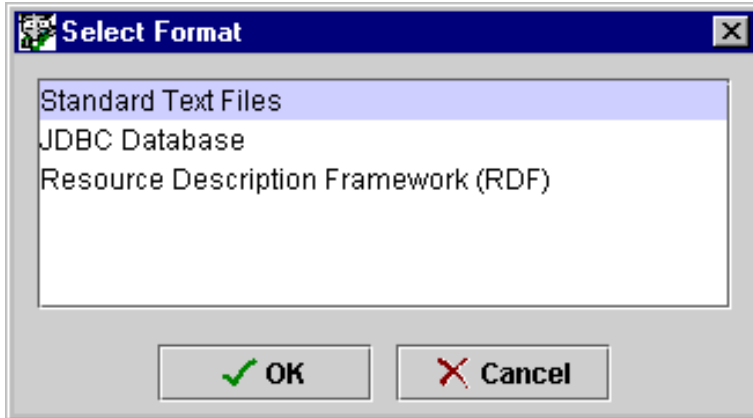You can use this feature to create multiple versions and backups of a project.

![icon] Saving a Project in a Different Format

To save a Protégé-2000 project in a different format:

1. Select **Save In Format** from the **Project** menu. A Select Format dialog is displayed.

2. Select the format you want. There are three basic formats that are always displayed:
   Standard Text Files (the default) creates project files in Protégé-2000 format. These can also be viewed with any text editor or word processor.
   JDBC Database creates the project as a table in a JDBC database. To do this you must have a database installed and configured on your system. See [Saving a Project as a Database](#) for more information.
   Resource Description Framework (RDF) saves the project in RDF format. See [RDF Support in Protégé-2000](#) for more information.
   If your version of Protégé-2000 has been customized to support additional formats, then you will also see them in this list.
3. Enter a name and location for your project in the Project line of the dialog box. This is the name of your project (.pprj) file. The same name will automatically be entered as the name for your ontology and instance files unless you change it. If you do not specify a path for the project, by default, the project is saved in the directory where Protégé-2000 is installed.
4. If you have chosen a non-text format, make sure the additional information needed by Protégé-2000 is correct. See [Saving a Database Project](#) or [Saving an RDF Project](#) for more information.
5. Click OK.

You can use this feature to create multiple versions and backups of a project.

Next: [Saving a Text Project](#)

[Project Table of Contents](#)

![icon] Saving a Text Project

You can save the class and instance information of any Protégé-2000 project in one of several text form. The User Interface information from the Forms Tab is always in the standard Protégé text-format.

Note: When a project is saved in text format, changes to the project are not committed until you click "Save".

You can save a Protégé-2000 project as text files in one of two ways:

- When the project is first created, by selecting **Standard Text Files** in the Select Format dialog box.
- By selecting **Save in Format...** from the **Project** menu and then selecting **Standard Text Files** in the Select Format dialog box.

After you have made your selection, choose the name and location for the project as follows:

1. When you have selected **Standard Text Files**, the Text Files dialog box is displayed:

2. To select or change the name or location of the **pprj** file, enter the information you want in the **Project** line of the dialog box. Be sure to give the file a **pprj** extension. To browse for a new location, cllick the **+** button. If you do not specify a location for the project, by default, the project is saved in the directory where Protégé-2000 is installed.
3. Protégé-2000 will automatically use the name of the **pprj** file as the name of the text

file that contains the class information. If you wish to change this, enter the name in the **Class file name** line. Be sure to give it a **pont** extension.
4. Protégé-2000 will automatically use the name of the **pprj** file as the name of the text file that contains the instances information. If you wish to change this, enter the name in the **Instance file name** line. Be sure to give it a **pins** extension.
5. Click OK.

Next:

Including a Project

You can include an existing project in your current Protégé-2000 project. This allows you to build a single large project from one or more smaller projects. Included classes, slots, and instances cannot be edited; included forms can be edited, however.

To include a project into an existing Protégé-2000 project:

1. Select **Include...** from the **Project** menu.
2. In the Project dialog box, browse to the location of the project you want to include.
3. Select the .pprj file of the desired project.
4. Click **Open**.

The project is imported into your current project. To show they cannot be edited, frames in the included project appear with pale icons: for example, ©, ◇, S.

Once a project has been included in another project, there is no way to remove it using the Protégé-2000 interface. It is possible to remove an included project by editing the "Project" instance in the **pprj** file.

You can see a list of all imported projects by selecting **Show Included Projects...** from the **Project** menu.

Next: <u>Configuring a Project</u>

<u>Project Table of Contents</u>

Configuring a Project

You can change the project configuration using the project **Configure** dialog box. This dialog box allows you to customize your project window. Some of the available options are:

- configuring the project tabs
- configuring the project options
- enabling journaling

Configuring the Project Tabs

To configure the project display:

1. Select **Configure...** from the **Project** menu.
   The Configure dialog box opens.
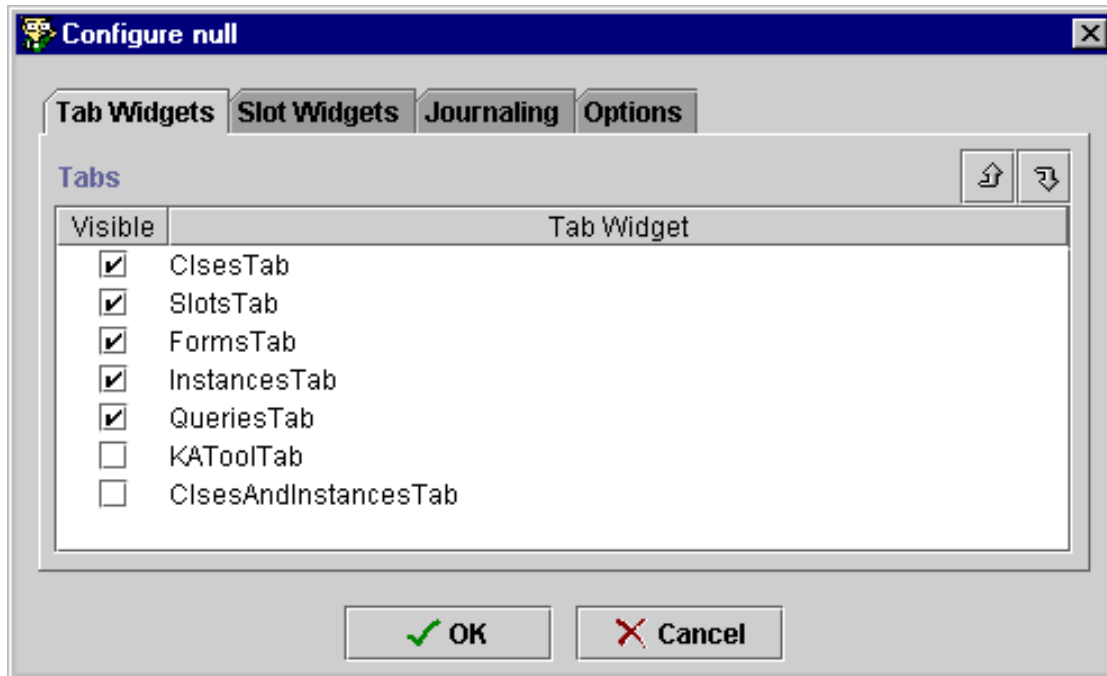
2. Make sure **Tab Widgets** is selected. It will show the tabs available for your project. Tabs with a mark in the box are visible; others are hidden. By default, a project displays the Clses Tab, Slot Tab, Forms Tab, Instances Tab, and Queries Tab. All projects also include the KAToolTab (Knowledge Acquistion Tool) and the ClsesandInstancesTab. Additional tabs may show in this window, depending on the project.
3. If you wish to *hide* a tab, click to remove the mark from in front of it. For example, if you are giving your project to someone to enter instances, you might wish to hide the other tabs.
4. If you wish to display a tab, click to add a mark in front of it. For example, you might want to display the Classes and Instances Tab, for a unified view of your classes and instances.
5. Click OK to close the dialog box and see your new configuration.

Configuring Options

To change project options:

1. Select **Configure...** from the **Project** menu.
2. Click the **Options** tab in the Configure dialog box.

Configure null ☒

| Tab Widgets | Slot Widgets | Journaling | **Options** |

☑ Display hidden classes
☑ Display abstract class icon
☑ Display multi-parent class icon
☐ Display confirmation dialog on remove
☑ Allow knowledge-base changes
☐ Update modification slots and facets

✓ OK   ✗ Cancel
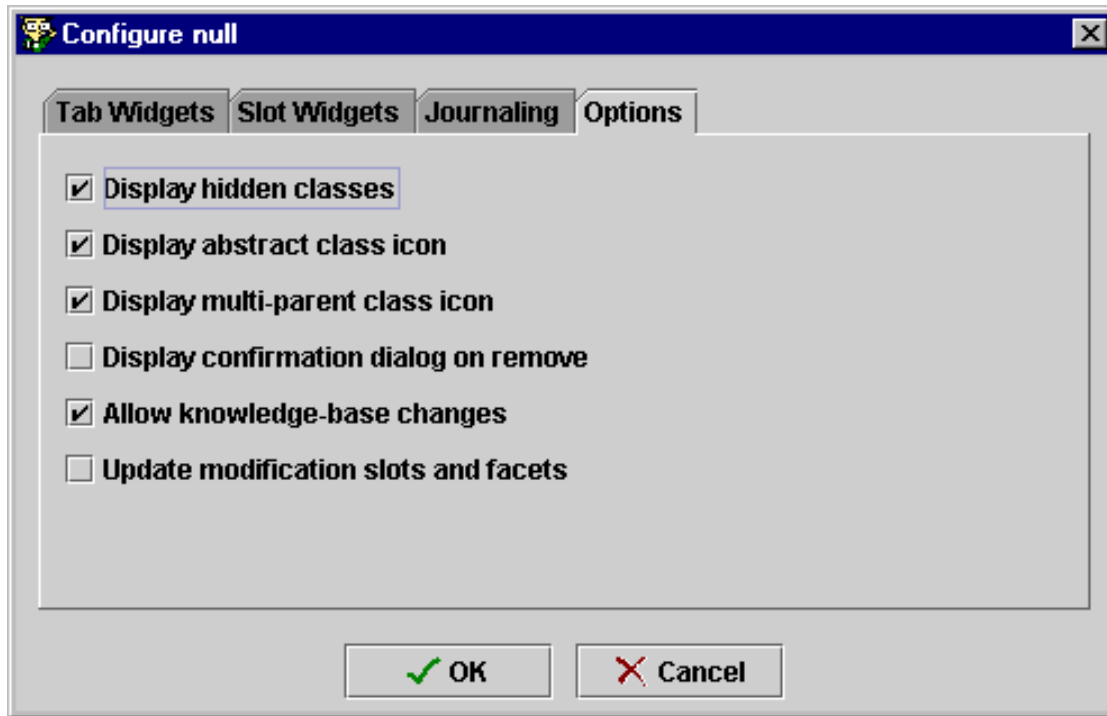
3. This tab allows you choose the following options:
   ○ Display hidden classes: Allows you to choose whether or not you can see the classes you hide using the right mouse button.
   ○ Display abstract class icon: Allows you to choose whether or not the ᴬ icon is shown on Abstract classes.
   ○ Display multi-parent class icon: Allows you to choose whether or not the ᴹ icon is shown on classes with multiple super classes.
   ○ Display confirmation dialog on remove: Allows you to add a warning when you remove a slot from a class. Recall that when you remove a slot, the slot remains in the project, but is simply removed from the current class. You always receive a warning on a delete.
   ○ Allow knowledge-base changes: Allows you to lock the knowledge base so it cannot be changed by the viewer.
     The remaining options are advanced and are not described here.
4. If you wish to select an option, click to mark it. For example, you might want to display the Classes and Instances Tab, for a unified view of your classes and instances.
5. If you wish to remove an option, click to remove the mark.
6. Click OK to close the dialog box and see your new configuration.

Enabling Journaling

Journaling allows you to keep a record of all the changes that you make to a project. Changes are stored in an ASCII (text) file, with one line for each change you make. The file is no

To enable journaling:

1. Click the Journaling tab.
2. Make sure that **Enabled** is selected.
3. If you want to change your user name, enter a new name in the **User** entry field.
4. Click OK.
   The journaling file is created in the same directory as the project, with a .pjrn extension. It can be read using any text editor.

Such a file might look like this:

```
Wed May 08 15:47:50 PDT 2002, Eliza, journaling enabled, source=edu.stanford.smi.protege.model.Project
Wed May 08 15:49:50 PDT 2002, Eliza, setOwnSlotValues, kb=newspaper, frame=instance_00043, lot=salary, values={50000.0}
Wed May 08 15:49:54 PDT 2002, Eliza, createClass, kb=newspaper, name=<null>, parents={Organization}, metaCls=:STANDARD-CLASS, isNew=true
```


Next:

Generating HTML From a Project

You can generate an HTML of a project. This allows you to view the class hierarchy and, optionally, all the instances. The output consists of an **index** page, which gives the class hierarchy for the project, including links to individual pages for each the class. Class pages include slot descriptions and optional instances. If instances are selected, they have individual pages, and appear in the index hierarchy and under each class.

You can generate HTML pages for the entire project, or restrict the output to a subset of the project.

To generate HTML for the entire project:

1. Select **Generate HTML...** from the **Project** menu.
   The **Generate HTML Options** dialog box is displayed.
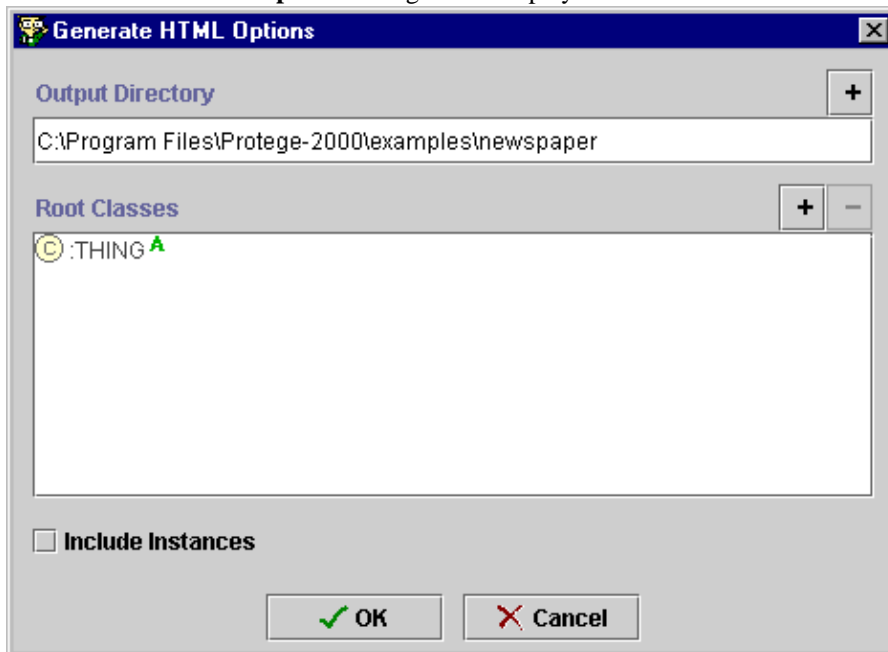
2. If you want to save the project in a different location from the default, type the new location in the Output Directory line or click the **+** button to navigate to the desired location.
3. If you want a page for each instance, make sure **Include Instances** is selected.
4. Click OK.
   The HTML pages are generated in the selected directory.

To restrict the output to a subset of the project:

1. Select **Generate HTML...** from the **Project** menu.
   The **Generate HTML Options** dialog box is displayed.
2. Choose the location where you want the project to be saved.
3. If you want a page for each instance, make sure **Include Instances** is selected.
4. To restrict the classes that will appear, first remove the :THING class from the **Root Classes** list by highlighting it and clicking the Remove **–** button. Removing the :THING class allows you to restrict the classes you want.
5. To choose a class as a root class, click the Add **+** button at the right of the **Root Classes** list.
   A Select Classes dialog box is displayed.
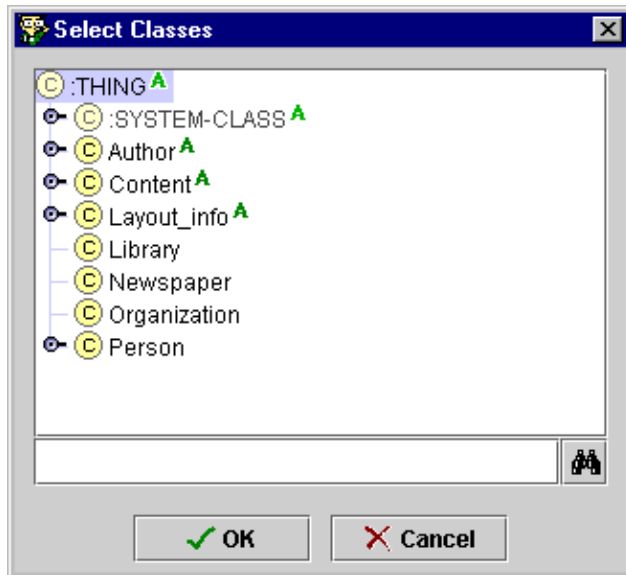
6. Highlight one or more classes you want as root classes. Selecting a class as a root class means that it and all its subordinate classes will be displayed in the HTML hierarchy. You can select multiple classes by holding down the Ctrl key.
7. Click OK.
8. The classes you selected are added to the **Root Classes** list. You can select additional classes by repeating steps 5 though 8.
9. When you have the classes you want as root classes, click OK.
   The HTML pages are generated in the selected directory.

The following shows examples of the index and class page output for the **newspaper** project, with **Layout_info** selected as the root class.

Page for the index:

Class Hierarchy for *newspaper* Project

- Layout_info
    - Billing_Chart
        Instances : *Weekday Ads*, *Saturday Ads*, *Sunday Ads*
    - Content_Layout
    - Prototype_Newspaper
        Instances : *Thursday*, *Tuesday*, *Friday*, *Wednesday*, *Monday*, *Saturday*, *Sunday*
    - Rectangle
    - Section
        Instances : *World News*, *Local News*, *Automotive*, *Lifestyle*, *Sports*, *Magazine*, *Science*, *Business*

Page for the billing chart class:

Project: newspaper

Class Billing_Chart

Concrete Class Extends
> Layout_info

Direct Instances:
> Weekday Ads, Saturday Ads, Sunday Ads

Direct Subclasses:
> None

Template Slots					
Slot name	**Documentation**	**Type**	**Allowed Values/Classes**	**Cardinality**	**Default**
cost_chart		Float		0:*	
name		String		0:1	

Return to class hierarchy

Next: Cascading Open Windows

Project Table of Contents

Cascading Open Windows

Protégé-2000 allows you to have multiple free-standing form windows open at a time. Each form displays the information for a **class**, **instance**, or **slot**. As you continue to work with a project, you may find that these form windows get hidden, or that the screen gets cluttered.

To cascade all open form windows:

1. Click the Cascade ⊞ button immediately below the main menu bar.

All open form windows will be brought to the front and cascaded.

Next: **Closing All Windows**

Project Table of Contents

Closing All Windows

Protégé-2000 allows you to have multiple free-standing form windows open at a time. Each form displays the information for a [class](#), [instance](#), or [slot](#). As you continue to work with a project, you may find that the screen gets cluttered.

To close all open form windows:

1. Click the CloseAllWindows ![icon] button immediately below the main menu bar.

All open form windows will closed. The main Protégé window will remain open.

Next: [Working With a Small Window](#)

[Project Table of Contents](#)

Working With a Small Window

Protégé-2000 displays a lot of information in many of its windows. In some cases, for example, if your window is sized to fit on a smaller screen, not all the information will appear. In this case you might have trouble finding some of the following:

- The Superclasses Pane at the Classes Tab
- The Classes Pane at the Slots Tab
- The Query Library Pane and the Search Results Pane at the Queries Tab

To view or enlarge these areas, try any of the following:

1. Enlarge the Protégé window by dragging it. A hidden section will usually appear if the window becomes large enough.
2. *OR* Drag the slider bar positioned where the hidden section should be. For example, to view the Search Results Pane, place the cursor over the slider bar at the right of the window, and drag the bar to the left. To view the Superclasses Pane, Query Library Pane, or Classes Pane, place the cursor over the slider bar at the bottom left of the window and drag the bar up.

3. *OR* If your window is very small, you can toggle between two panes by clicking on the arrows on the slider bar. For example, to toggle between the Query Pane and the Query Library Pane, click the up arrow on the slider bar to display only the Query Library Pane; click the down arrow to display only the Query Pane.

You can also customize the distance between any of the panes using the slider bars in the same way.


Working With Notes

Protégé-2000 allows you add a "yellow sticky" note to any *frame*, that is, a [class](#), [instance](#), or [slot](#). This allows you to add notes that are not part of the ontology structure. The note is always displayed along with the form.
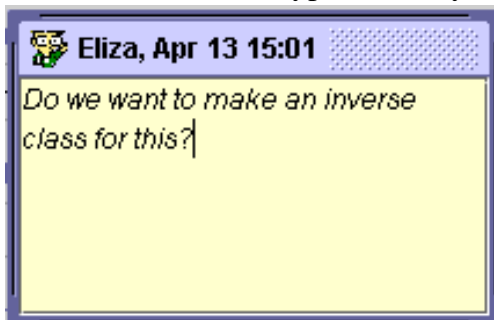
To add a note to a frame:

1. Click the Create Note **C** icon at the top right of the form for the frame (e.g., [Class Form](#), [Instances Form](#), etc.)
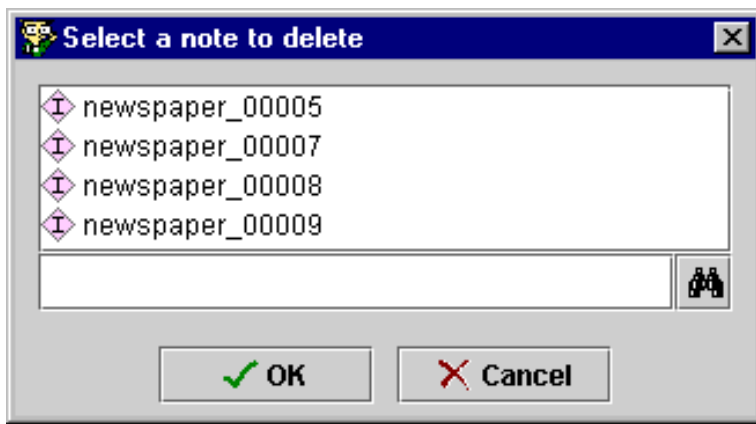   A yellow note is created above the form.

   Eliza, Apr 13 15:01
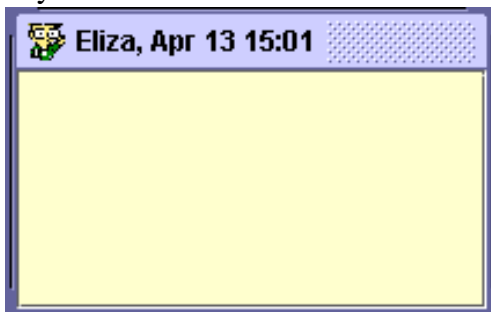
2. Click in the note and type the text you want.

   Eliza, Apr 13 15:01
   Do we want to make an inverse class for this?

3. As always, you can resize or reposition the note by dragging.

To remove a note:

1. Click the Delete Note ✖ icon at the top right of the form.
2. If there is only one note, you will be prompted to confirm the deletion. Click OK.
3. If there is more than one note, a list of notes is displayed.
   Select the note you want to delete and click OK.

Select a note to delete

⬦ⓘ newspaper_00005
⬦ⓘ newspaper_00007
⬦ⓘ newspaper_00008
⬦ⓘ newspaper_00009

✓ OK   ✕ Cancel

Next:

Viewing Class Relationships

The Class Relationship Menu allows you to choose which relationship between classes is represented in the Class Relationship pane. The following choices are available:

- **Superclass**: Displays all the classes in your project in a subclass-superclass relationship. This is the default.
- **Reference**: Displays all the slot reference relationships.
- **Slot reference relationships**: When a class is selected, each of its slots of type **Class** or **Instance** appears separately on the menu. Selecting the slot name displays classes which are reference through the slot.

The picture below shows the Class Relationship menu for the class **Editor** in the Newspaper example. **Editor** has two **Instance** slots, **responsible_for** and **sections**. Other classes might show more slots, or only the **Superclass** and **Reference** choices.
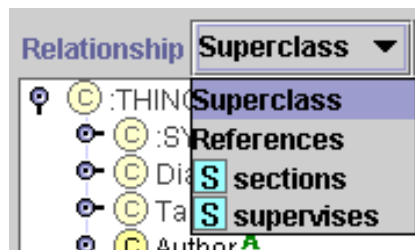
Certain actions can be performed in the **Superclass** relationship view *only*. These include creating a new class and rearranging the hierarchy using drag-and-drop.

The view in the Superclasses pane does not change to reflect the choice on the Relationship menu.

Superclass

The default **Superclass** relationship displays the subclass-superclass structure of your knowledge base. All the classes in the knowledge base appear in the displayed tree. (Some classes may be hidden. You can choose whether or not hidden classes are displayed in the Class Relationship pane by choosing **Configure** from the **Project** menu, going to the **Display** tab, and toggling the **Display Hidden Classes** option.) The **Superclass** view gives you an overall picture of the class structure. This choice is always shown on the menu.

Reference

The **Reference** relationship displays the hierarchy of classes, starting at the selected class, that are related to each other through any of the slots for the class. The **Reference** view thus shows the union of

all the [slot reference relationships](#).

Slot Reference Relationships

If you select a class in **Superclass** view, the Class Relationship menu will list all the slots of type **Class** or **Instance** for that slot. Selecting a slot from the menu will show the hierarchy of classes, starting at the selected class, that are related to each other through the slot selected on the menu.

For example, the class **Editor** in the **Newspaper** example has the **Instance** slot **supervises**. The only [allowed class](#) for this slot is the class **Employee**. This is what is shown when **supervises** is selected.

In more complex knowledge bases, the displayed tree might be more extensive. Multiple classes can be related to the original class through the slot. A class that is related to the selected class through the slot might itself have that slot, and then its related classes would appear in the tree. This allows for a complex hierarchy that diagrams the slot relationship between classes.

The different views can be read as "parent-relationship-child," where "relationship" is displayed on the **Relationship** menu. For example, in the **responsible_for** view, **Editor** - is **responsible for** - **Employee**. Similarly, in the **Superclass** view, **Employee** - is a **superclass** of - **Employee**.

It is also possible to view a list of all the frames that reference a highlighted class using the Back-References class button. See [Viewing Back-References](#) for more information.

Next: [Finding a Class](#)

The Class Icons

The icons in the Class Window give information about the structure of your knowledge base and the nature of your classes.

The icons to the left of the class name give information about the display of the class hierarchy:

This icon indicates that all direct subclasses of the class are displayed. You can click on this icon to hide the subclasses. In the example, the subclasses of **Content** are displayed. In some views, this is shown as a -.

This icon indicates that the class has direct subclasses which are not currently displayed. You can click on this icon to display the subclasses. In the example, **Layout_info** has subclasses which are not being displayed. In some views, this is shown as a +.

No icon The absence of a ♀ or a ☉ indicates that the class has no subclasses. In the example, **Library** has no subclasses.

The color of the C icons give information about the type of class:

The class can be edited.

The class is a metaclass.

A pale icon indicates that the class has been included from another project. Included classes cannot be edited. See [Including a Project](#) for more information about including a project.

Icons to the right of a class give additional information about the class. You can control the display of these icons by choosing **Configure** from the **Project** menu, going to the **Display** tab, and toggling the appropriate option:

The class is **abstract**. A class can have one of two roles: concrete or abstract. Concrete classes may have direct instances; abstract classes cannot have direct instances. No A icon means the class is **concrete**. See [The Class Form](#) for more information on the class role.

The class has multiple superclasses. Protégé-2000 allows classes to have more than one superclass. M means the class has multiple superclasses. No M icon means the class has only one superclass. See [Jumping to Another Superclass](#) for more information.

The class is hidden. No H icon means the class is not hidden. You can choose whether or not hidden classes are displayed in the Class Relationship pane by choosing **Configure** from the **Project** menu, going to the **Display** tab, and toggling the **Display Hidden Classes** option. You can use this, for example, if you want to restrict the user's view to a part of your knowledge base but want to retain the structure of the larger project.

Next:

Replacing a Superclass

You can replace a superclass of a class using drag-and-drop in the Class Relationship pane. To replace a superclass:

1. Select the subclass you want to move
2. Hold down the mouse button and drag the subclass on top of the new superclass.
3. Release the mouse. The subclass will now be located under the selected superclass. Note that the slots of the dragged class will automatically change to reflect the inheritance from the new superclass.

The following example shows how to make **Columnist** inherit from **Employee** rather than the class **Person**.

1. Highlight **Columnist** in the Class Relationship pane. Because **Columnist** has more than one superclass, you must make sure to select the copy of **Columnist** that is under **Employee**. (When a class has more than one superclass, you can use the Superclasses pane to select the copy that you want. See Jumping to a Different Superclass for more information.) Note that the Slots information for **Columnist** reflects the slots it inherits from **Person**.
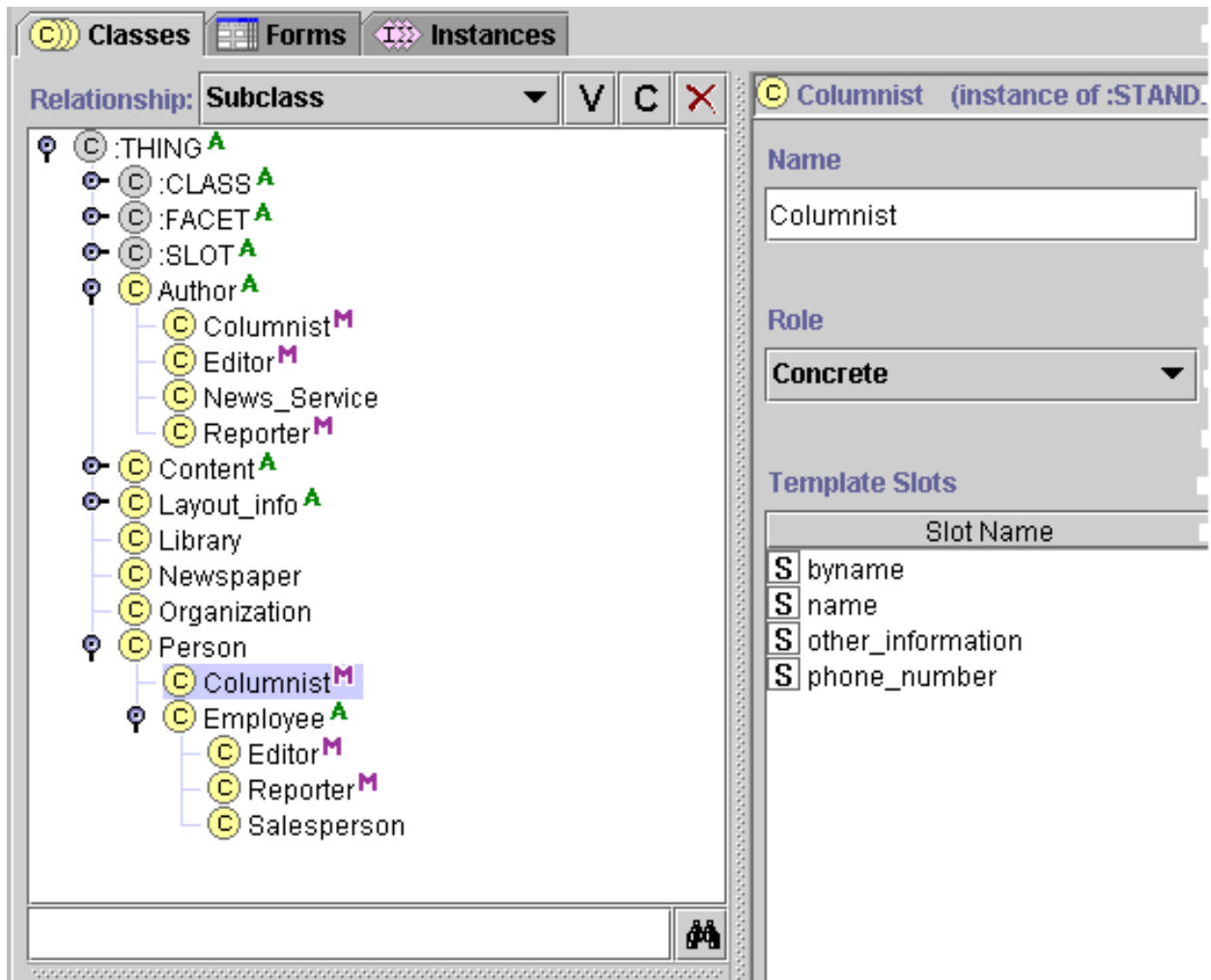
2. Hold down the mouse button and drag **Columnist** on top of the new superclass **Employee**.
3. Release the mouse button to drop the class. **Employee** is highlighted and the Class Relationship pane redisplays to show the new hierarchy. The dragged class and all of its subclasses are moved from the original superclass to the new superclass.
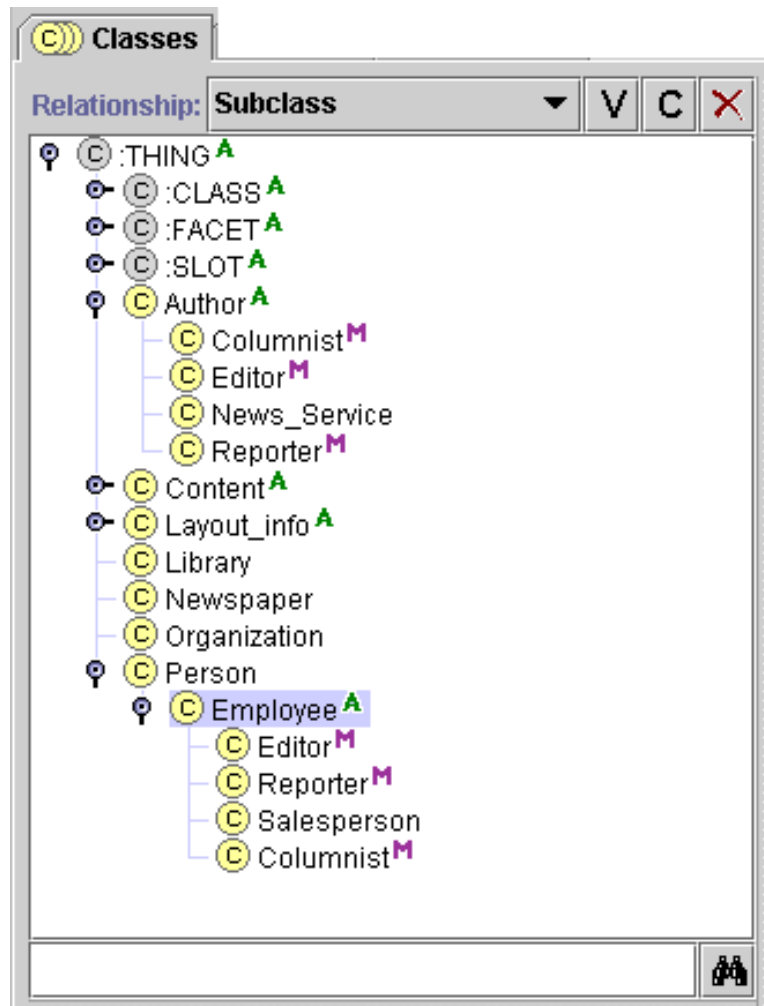
Notice that if you highlight **Columnist** after it has been repositioned, it now has three additional slots. These are the new slots it inherits from **Employee**. When you replace the superclass of a class, you will often lose and gain slots.

Next:

Finding a Class

To find a class in the Class Relationship pane:

1. Type all or part of the name of the class in the Class Lookup Bar at the bottom of the Class Relationship pane.

    > ad   🔍

2. Press Enter/Return or click the Find ☐ button.
    If there is one match the class will be highlighted. Otherwise, you will get a dialog box of all the matches, and you will be able to select one.

 Comparison is case-insensitive and position-independent. For example, in the Newspaper project, *ad* would find *Advertisement*, *Personals_Ad*, and *Standard_Ad*.

If a found class is not currently visible in the Class Relationship pane, Protégé-2000 automatically

scrolls to the correct location and expands the hierarchy to make the class visible.

If you are unable to find the class you are looking for, the class may be hidden. To verify that hidden classes are displayed, choose **Configure** from the **Project** menu, go to the **Options** tab, and make sure the **Display Hidden Classes** option is checked. When **Display Hidden Classes** is checked, hidden classes are shown with an **H** icon.

Next:

The Class Window

The Class Window in the Class Relationship Pane displays your classes and allows you to rearrange your hierarchy using "drag-and-drop". A number of icons give additional information about your classes; see The Class Icons for a description.

By default, the Class Window displays the class structure of your knowledge base. Subclasses appear below their superclasses and indented to the right. Classes with more than one superclass appear more than once in the tree. All classes are shown as descending, directly or indirectly, from the system class **:THING**. Icons show additional information about your class.

The currently selected class is shown with a highlight. Icons give information about the structure of your knowledge base and the nature of your classes. See The Class Icons for more information.

The Class Window allows you to perform the following tasks:

- You can **select** a class by clicking on it.
- You can **edit** the attributes of a class in the Class Form. When a class is selected, the Class Form appears directly in the Class Edit Pane in the Classes Tab, or you can open it for one or more selected classes by clicking the View ∨ Class Button. See Viewing a Class for more information.
- You can **create a new class** by clicking the Create C Class Button and editing the attributes in the Class Form in the . See Creating a New Class for more information.
- You can **delete** a class by clicking the Delete ✕ Class Button. See Deleting a Class for more information.
- You can **find** a class by typing all or part of the class name in the Class Lookup Bar and then clicking the Find 🔍 button. See Finding a Class for more information.
- You can access the cascading class menu by selecting a class and then clicking the right mouse button. See The Class Menu for more information.

You can also rearrange the superclass/subclass relationships:

- You can **move a class to another superclass** by selecting the class, dragging it directly over the new superclass, and then dropping it. See Replacing a Superclass for more information.
- You can **add an additional superclass to a class** by selecting the class, dragging it over the additional superclass, and holding down the Control key while you drop it. Classes with multiple superclasses are shown with an M icon. See Adding a Superclass for more information.

For additional superclass tasks you can perform, see the Superclasses Pane.

You can also choose to view classes according to Reference. See [Viewing Class Relationships](#) for more information.
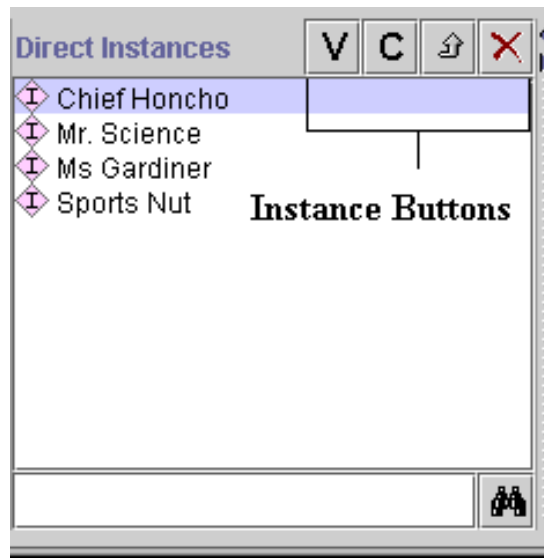

Next: [The Class Icons](#)

The Instance Buttons

The Instance buttons, **V C ⬆ ✕**, located at the top right of the [Direct Instances pane](#), allow you to view and edit, create, or delete a instance. Note that these icons have a similar action wherever they appear; for example, see the [Class Buttons](#).

The buttons have the following actions:

V
V(iew) button: Click this button to open the [Instances Form](#) for the highlighted class. See [Editing an Instance](#).

C
C(reate) button: Click this button to create a new instance for the class highlighted in the [Class Pane at the Instances Tab](#). See [Creating an Instance Directly](#).

⬆
Back-references button: Click this button to view all the objects that reference the highlighted instance. See [Viewing Back-References](#).

✕
Delete button: Click this button to delete the highlighted instance. See [Deleting an Instance](#).

If the View or Delete button is grayed out, this indicates that the current instance cannot be edited or deleted. Such an instance always has a gray ⬦ icon to its left. Instances cannot be edited if they are included from another project.

If the Create button is grayed out, no instances can be created for the class selected in the [Class Pane at the Instances Tab](#). This is the case if the class is abstract, shown by an ᴬ icon.

Next:

Viewing Back-References

In a complex project, it is possible to have many different relationships between classes, instances, and slots. The Back-References Pane allows you to view a list of all the frames that reference a highlighted item.

To view the Back-References Pane:

1. Highlight the item for which you wish to see a list of back references. You must make sure to go to the correct location:
   For a class, go to the Class Tab and highlight the item in the Class Relationship Pane.
   For a slot, go to the Slots Tab and highlight the item in the Slots Pane.
   For an instance, go to the Instances Tab, select the class where the instance appears in the Class Pane at the Instances Tab, then highlight the item in the Direct Instances Pane.
2. Click on the Back-References button at the top right of the pane.

For classes, it is also possible to view those classes that are related through a particular slot. See Viewing Class Relationships for more information.


Next: The Superclasses Pane

Constraints

Preliminary information about constraints can be found in the PAL section of the [Plugins Library](#)

The Class Pane at the Instances Tab

The Class Pane at the Instances Tab shows the classes in your knowledge base in a superclass/subclass relationship. By navigating through the classes, you can select the class whose instances you wish to see.
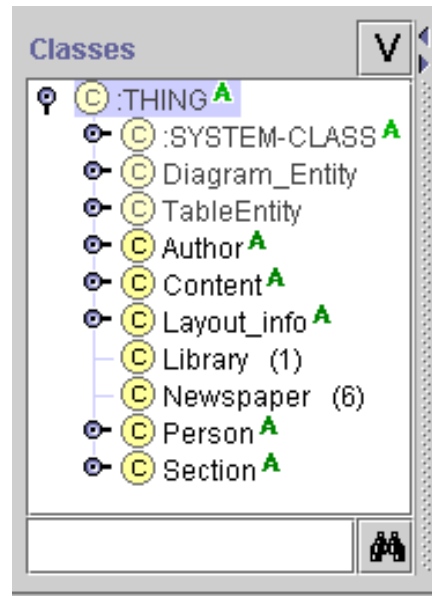
Subclasses appear below their superclasses and indented to the right. Classes with more than one superclass appear more than once in the tree. All classes are shown as descending, directly or indirectly, from the system class **:THING**.

The View button opens the Class Form for the highlighted class and allows you to view or edit the class properties. See Viewing a Class for more information.

The **numbers** to the right of a class give information about the number of direct instances that class has. For example, **Editor** has 4 instances. When a class with instances is selected, the instances are shown in the Direct Instances Pane. If there is no number, the class does not have any instances. Abstract classes (see below) cannot have instances; concrete classes may or may not have instances.

The Class Pane at the Instances Tab uses icons to give information about the structure of your knowledge base and the nature of your classes. See the Class Icons for more information.

Next: The Direct Instances Pane

Instances Table of Contents

![icon] The Instances Tab

The Instances tab provides a window in which you may view, create, and edit **instances**. (Classes model concepts in your domain, slots model properties of classes and any relationships between them, and instances model the actual data.)

An example of the Instances Tab is shown below. The window consists of three panes:

1. A [Class pane](#) at the upper left shows the classes in a superclass/subclass relationship. The Instances Tab lets you view classes, but you cannot edit or rearrange them. See the [Classes Tab](#) for information about working with classes.
2. The [Direct Instances pane](#) in the center shows all the direct instances, if any, for the selected class, and allows you to view, edit, create, and delete direct instances.
3. When a single instance is selected, the Edit pane on the right contains the [Instance Form](#) for the selected instance. The [Instance Form](#) displays all the slots which apply to the instance, and allows you to edit them. The [Instance Form](#) can also be displayed as a separate window by clicking the View V icon in the [Direct Instances pane](#).

[FrontPage Image Map Component]

For information about the Instances tab user interface and about accomplishing specific tasks, see the [Instances Table of Contents](#).
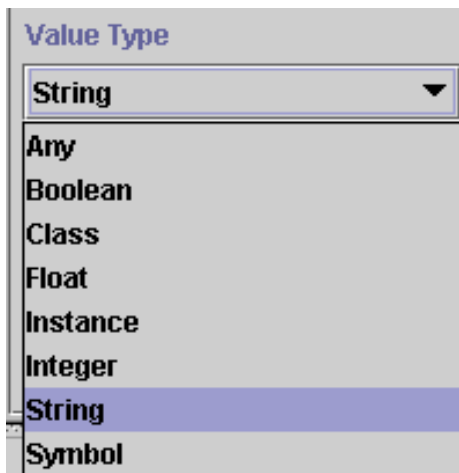

Next: [The Class Pane at the Instances Tab](#)

[Instances Table of Contents](#)

The Value Type Menu

The Value Type menu in the [Slot Form](#) allows you to select the slot type, which determines the kind of values that the slot may hold. When a value type of **Class**, **Instance**, or **Symbol** is selected, an additional pane appears below the **Value Type** menu.

When you are creating instances for a slot, the slot type also determines how the slot is displayed in the [Instances Form](#). See the [Standard Widgets](#) for more information about instances and type.

Any

A value type of **Any** means the slot can take any one of the other values: **Boolean**, **Class**, **Float**, **Instance**, **Integer**, **String**, or **Symbol**. If a class inherits a slot of type **Any**, then the slot may be modified by restricting it to a specific one of the other types. This is the only case where the actual value type of an inherited slot can be changed.

Any allows you to create a generic slot for a high-level class and then determine the actual value type at a lower level. For example, suppose you are modeling a taxonomy for all the vertebrates in an ecosystem, with specific species as the instances. You could create the slot **Diet** at the **Vertebrate** class level. Then for the subclass **Carnivore**, you could restrict the **Diet** slot to type **[Instance](#)**, so that you could select specific species already in your taxonomy as the diet. On the other hand, for the subclass **Herbivore**, you could restrict the **Diet** slot to type **[Symbol](#)** and list possible food plants by name.

A class with a slot of type **Any** cannot have instances.

Boolean

A **Boolean** slot can holds a logical Boolean value, that is, a value that is either true or false. An instance

of a class with a Boolean slot displays the slot as a checkbox. For example, the **Personals_Ad** class contains the Boolean slot **Urgent**; the **Silly** instance of **Personals_Ad** is not Urgent, while the **M137** instance is:

☐ Urgent   not **Urgent**

☑ Urgent   **Urgent**

Class

A slot of type **Class** has classes as values. More specifically, when **Class** is selected as the value type, the **Allowed Parents** pane is displayed. A slot or an instance then takes one of these classes or any of their subclasses as the value(s) of the slot.
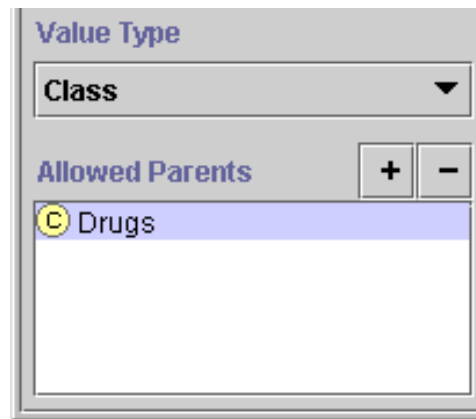
For example, suppose you have created a knowledge base which includes, among other information:

- the superclass **Drugs**, which contains prescription drugs as classes
- the concrete class **Patients**, which has instances which are patients

You can create the slot **medications** in the class **Patients** and make it of type **Class**, with **Drugs** as the **Allowed Parents**. Then when a user creates an instance, they can choose the medications from the list of prescription drugs that appear as subclasses of **Drugs**.

The Allowed Parents Pane

Buttons at the top of the **Allowed Parents** pane let you add and remove classes from the list of allowed parents for a slot of type **Class**.

To add allowed parents for a slot of type **Class**:

1. Click the Add ✛ button at the top of the **Allowed Parents** pane. A **Select Classes** window shows the list of classes in your knowledge base, in the usual subclass hierarchy.

2. Select one or more classes. To highlight multiple classes, hold down the Ctrl key while clicking each class. To highlight a range of classes, click the first class, then hold down the Shift key and click the last class in the range.
3. Click OK.

To remove a superclass from the list of allowed parents for a slot of type **Class**:

1. Select the classes to remove. To highlight multiple classes, hold down the Ctrl key while clicking each class. To highlight a range of classes, click the first class, then hold down the Shift key and click the last class in the range.

2. Click the Remove ⎯ button at the top of the **Allowed Parents** pane.

Float

A slot of type **Float** has numbers as values; these numbers may include a decimal point. Values of type **Float** are stored on your system as floating point values, and are only as accurate as your system.

When entering a **Float** value for an instance, you can use decimal point or exponential representation. You can enter positive and negative values. For example:

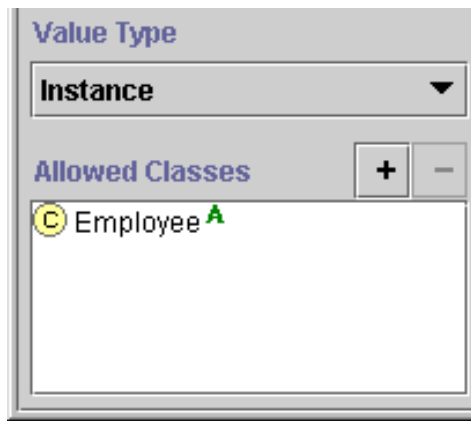| Representation	Description
1234.56	Standard decimal notation.
1.23456E3	Exponential notation, shorthand for $1.23456 \cdot 10^3$. Represents 1234.56
-1234.56	Negative number.
1.23456E-3	Exponential notation with a negative exponent, shorthand for $1.23456 \cdot 10^{-3}$. Represents 0.00123456.

For convenience in typing, users can enter a lower case **e** instead of an uppercase **E**.

For slots of type **Float**, you can also enter a **Minimum** and/or **Maximum** value. See The Slot Form for more information.
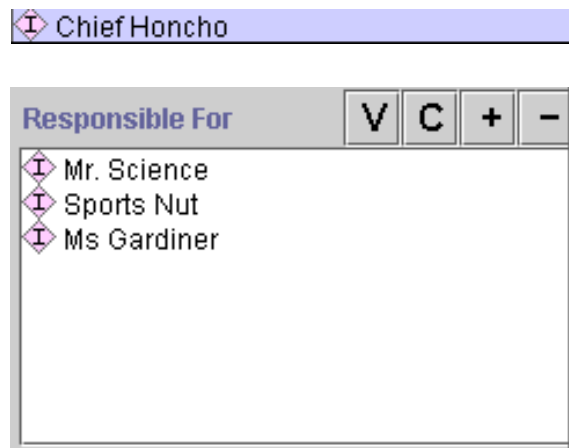
Instance

A slot of type **Instance** has instances as values. More specifically, when **Instance** is selected as the value type, the **Allowed Classes** pane is displayed. An instance of the class with this slot then takes instances of the allowed classes or their subclasses as the value(s) of the slot.

In the Newspaper example, the slot **responsible_for** in the class **Editor** takes instances of the class **Employee** as values. The instance **Chief_Honcho** is responsible for three instances which are descended from **Employee**: **Mr. Science**, **Sports Nut**, and **Ms Gardiner**.

The Allowed Classes Pane

Buttons at the top right of the **Allowed Classes** pane let you add and remove classes from the list of allowed classes for a slot of type **Instance**.

To add allowed classes for a slot of type **Instance**:

1. Click the Add ✚ button at the top of the **Allowed Classes** pane. A **Select Instances** window shows the list of classes in your knowledge base, in the usual subclass hierarchy.
2. Select one or more classes. To highlight multiple classes, hold down the Ctrl key while clicking each class. To highlight a range of classes, click the first class, then hold down the Shift key and click the last class in the range.
3. Click OK.

To remove a class from the list of allowed classes for a slot of type **Instance**:

1. Select the classes to remove. To highlight multiple classes, hold down the Ctrl key while clicking

each class. To highlight a range of classes, click the first class, then hold down the Shift key and click the last class in the range.

2. Click the Remove ⁻ button at the top of the **Allowed Classes** pane.

Integer

A slot of type **Integer** has numbers as values for an instance; these numbers cannot include a decimal point. Values of type **Integer** are stored on your system as integer values. You can enter positive and negative values.

For slots of type **Integer**, you can also enter a **Minimum** and/or **Maximum** value. See The Slot Form for more information.
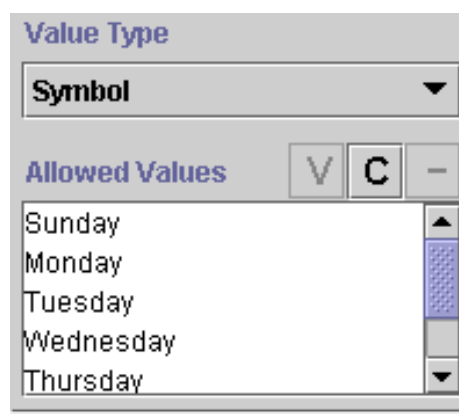
String

A slot of type **String** has text strings as values. You can enter ASCII characters for an instance, including upper and lowercase letters, numbers, the basic symbols on the keyboard, such as !, _, and %. **String** values can also include spaces.

The Symbol Value Type

A slot of type **Symbol** allows you to create a pre-set list of strings; an instance of a class chooses from among these strings. More specifically, when **Symbol** is selected as the value type, the **Allowed Values** pane is displayed. An instance then takes string(s) from among the allowed values as the value(s) of the slot.
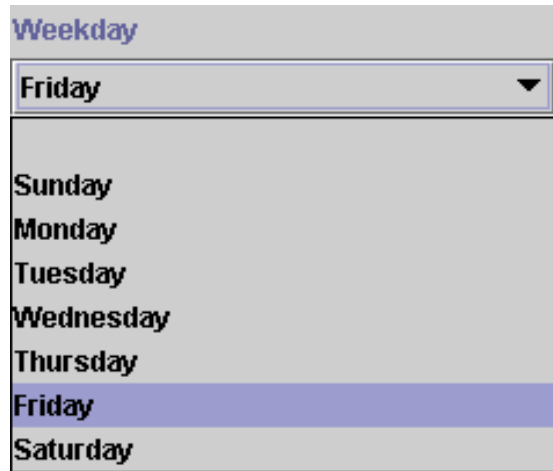

For example, the **weekday** slot at the class **Prototype_Newspaper** allows you to choose from among the seven days of the week: Sunday, Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, and Sunday. Then an instance has a pop-up menu that allows a user to choose one of the symbol values for

the slot:

The Allowed Values Pane

Buttons at the top right of the **Allowed Values** pane let you add, edit, and remove values from the list of allowed values for a slot of type **Symbol**.

To add a value to the list of available values for a slot:

1. Click the Create C button at the top of the **Allowed Values** pane.
2. Type the string you want in the **Create Symbol** window. You can enter ASCII characters, including upper and lowercase letters, numbers, and other common characters, such as !, _, and %.
3. Click OK.

To edit a pre-existing value:

1. Click the View V button at the top of the **Allowed Values** pane.
2. Edit the string in the **Edit Symbol** window.
3. Click OK.

To remove a value from the list of values:

1. Select the values to remove. To highlight multiple values, hold down the Ctrl key while clicking each value. To highlight a range of values, click the first value, then hold down the Shift key and click the last value in the range.
2. Click the Remove ⁻ button at the top of the **Allowed Values** pane.

Next:

Standard Fields

For each slot in the instance, the Instances Form displays a field where you can enter the information for that slot. The display and options for the field depend on the type of information that is included in the field.

Note: These topics describe the default fields only. You can select different formats for the fields using Forms.

Protégé-2000 provides the following fields for both **Single** and **Multiple** cardinality. For simplicity, only the **Single** cardinality is shown:

Field Type	Default Single Cardinality Field	Default Appearance
Boolean Field	A checkbox field that describes a slot as true or false for this instance	☑ Urgent
Class Field	A text display field and three buttons that allow you to specify a class as the value for this slot	Class V + −
Float Field	A text entry field that verifies that the entered value is a valid decimal number	Salary 12.50
Instance Field	A text display field and four buttons that allow you to specify an instance as the value for this slot	Section V C + − Lifestyle
Integer Field	A text entry field that verifies that the entered value is a valid whole number	Page Number 14
String Field	A text entry field	Headline Bank Outflanks Stubborn 90 Year Old Man
Symbol Field	A drop-down list that allows you to select from a preset list of values	Reading Level High_school ▼

Single vs. Multiple Fields

A multiple cardinality field is very similar to the corresponding single field. There are some common differences:

Single Field	Multiple Field
Only allows one value. Entering a new value means removing the old one.	Allows multiple values.

Some fields (e.g., Float, Integer, Symbol) do not have Field Buttons.	Always have Field Buttons which allow you to view, add or create, and delete values for the field.

To add a value, click on the Add **+** or Create **C** Field Button. The way you enter a value depends on the field type. For example, you may choose values from a list, or you may simply type your value in a pop-up entry window.
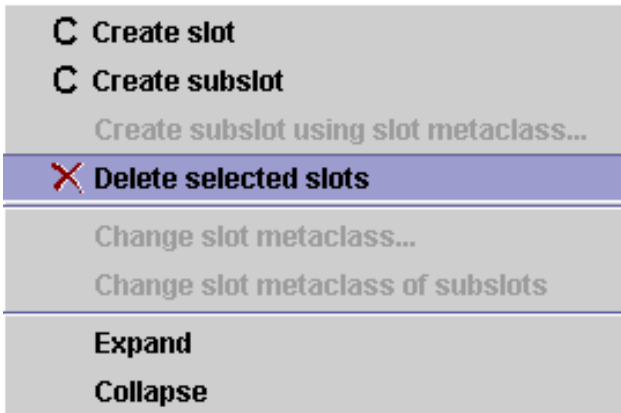

Next: The Boolean Fields

Deleting a Slot From the Project

You can delete a slot so that it is no longer part of the knowledge base. To do this:

1. In the Slots Tab, highlight the slot which you want to remove.
2. Click the Delete ✖ slot button *or* click the right mouse button and select **Delete selected slots** from the Slot menu.

   The slot no longer appears in the list of slots. The slot is also deleted from any classes where it appears.

To remove several slots at once:

1. In the Slots Tab, highlight the slots you wish to remove. To highlight multiple slots, hold down the Ctrl key while clicking each slot. To highlight a range of slots, click the first slot, then hold down the Shift key and click the last slot in the range.
2. Click the Delete ✖ slot button *or* click the right mouse button and select **Delete selected slots** from the Slot menu. The slots no longer appear in the list of slots and are deleted from any classes where they appear.

Deleting a slot cannot be undone. However, if you close Protégé-2000 without saving changes, you will revert to the last saved version. If you have made extensive changes to your project during the current session, you may wish to save before deleting slots. To do this, select **Save** from the **Project** menu.

You can also remove a slot from a specific class where it appears without deleting it from the project. See Removing a Slot for more information.

Next: Adding an Existing Slot to a Class

Slots Table of Contents

Adding a Superclass

You can add a superclass to a class in two ways: using the Superclasses pane or by drag-and-drop in the Class Relationship pane.

Adding a Superclass in the Superclasses Pane

Note: If you cannot see the Superclasses Pane, your window may be too small. You can see the pane by enlarging your window or by dragging the slider bar at the bottom of the Class Relationship Pane. See Working With a Small Window for more information.

To add a superclass using the Superclasses pane:

1. Select the subclass in the Class Relationship pane.

2. Click the Add ✛ button in the Superclasses pane immediately below the Class Relationship pane. A Select Classes window will appear.

3. Choose the additional superclass(es) you want from the Select Classes window. To highlight multiple classes, hold down the Ctrl key while clicking each class. To highlight a range of classes, click the first class, then hold down the Shift key and click the last class in the range.
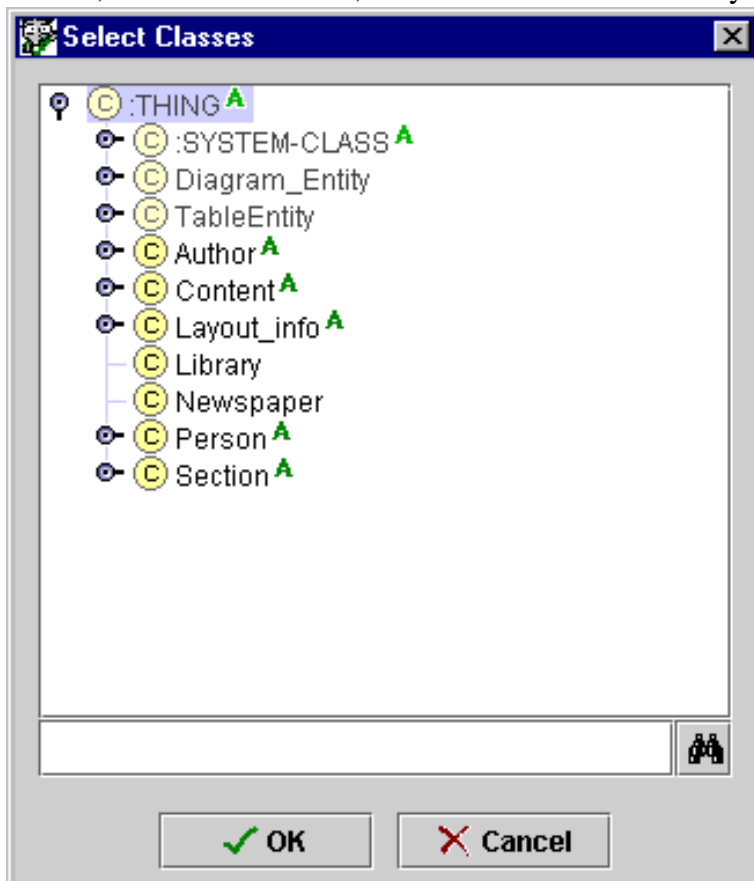
Adding a Superclass Using Drag-and-Drop

To add a superclass using drag-and-drop in the Class Relationship pane:

1. Select the subclass in the Class Relationship pane.
2. Hold down the mouse button and drag the subclass over the additional superclass.
3. Hold down the **Ctrl** key and release the mouse button to drop the subclass. The system will add the new superclass to the subclass.

Note that if you do not hold down the **Ctrl** key when releasing the mouse, the operation is interpreted as Replacing a Superclass.

If nothing happens when the class is dropped over a new superclass, the target class cannot be a superclass of the dragged class. For example, descendents of the dragged class are excluded.

Notes

Adding a superclass results in more than one superclass for a class, which then inherits the slots and facets of all of its superclasses. In the Class Relationship pane, a class with multiple superclasses is marked with and has multiple occurrences, one for each superclass. For example, the **Columnist** class in the **Newspaper** example has two occurrences in the Class Relationship pane: one with the superclass **Author**, and one with the superclass **Person**.

For information on replacing a superclass with a different superclass, see Replacing a Superclass.

Next: Jumping to Another Superclass

Classes Table of Contents

Removing a Superclass

To remove a superclass from a class:

1. Highlight the subclass in the [Class Relationship pane](#).
2. Highlight the superclass you want to remove in the [Superclasses pane](#) immediately below the [Class Relationship pane](#).
3. Click the Remove ▬ button at the top right of the [Superclasses pane](#).
   The superclass remains in the knowledge base, but the superclass/subclass link is broken.

You can only remove a superclass if the class has more than one parent. The subclass still appears in the knowledge base as a subclass of its other superclass(es). If the class has only one parent, the Remove ▬ button is disabled.

For example, to remove **Person** as a superclass of **Columnist** in the Newspaper knowledge base:

1. Highlight **Columnist** in the [Class Relationship pane](#)
2. Highlight **Person** in the [Superclasses pane](#).
3. Click the Remove ▬ button at the top right of the [Superclasses pane](#).

Columnist has multiple superclasses in this knowledge base, so it remains a subclass of **Author**. Note that a subclass such as **Columnist** loses its ᴹ icon when it no longer has multiple superclasses.

To delete a class from the knowledge base, highlight the class and click the Delete ✕ [class button](#). See [Deleting a Class](#).

Next: [Understanding Metaclasses](#)

[Classes Table of Contents](#)
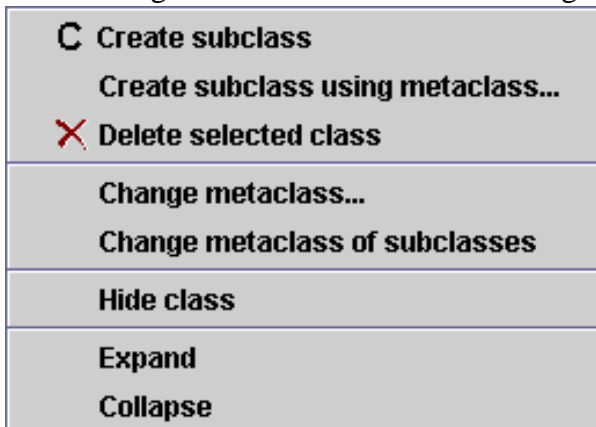
 The Class Menu

Whenever you have a class selected, you can access the cascading class menu by clicking the right mouse button. This menu allows you to perform a number of class-related tasks.

To access the class menu:

1. Select a class in the The Class Relationship Pane.
2. Click the right mouse button. The cascading class menu is displayed.

```
  C  Create subclass
     Create subclass using metaclass...
  ✕  Delete selected class

     Change metaclass...
     Change metaclass of subclasses

     Hide class

     Expand
     Collapse
```

3. Make your selection and click the left mouse button.

The class menu allows you to perform the following tasks. Not all tasks are available at all times; tasks that cannot be performed are greyed.

- **C Create subclass**: Creates a class subordinate to the highlighted class. This operation is identical to clicking the **Create C** class button. See Creating a Class for more information.
- **Create subclass using metaclass...**: (Note that metaclasses are an advanced feature; you should have a good understanding of Protégé before you use metaclasses.) If you have added metaclasses to your project, allows you to create a new class using a metaclass as a template. See Creating a Class Using a Metaclass for more information.
- **✕ Delete selected class**: Deletes the highlighted class and all of its subclasses, removing it from the current project. This operation is identical to the **Delete ✕** class button. See Deleting a Class for more information.
- **Change metaclass**: (Note that metaclasses are an advanced feature; you should have a good understanding of Protégé before you use metaclasses.) Changes the metaclass of the highlighted class. See Changing the Metaclass of a Class for more information.
- **Change metaclass of subclasses**: (Note that metaclasses are an advanced feature; you should have a good understanding of Protégé before you use metaclasses.) Changes the metaclass of all subordinate classes to the metaclass of the highlighted class.
- **Hide class**: Makes the highlighted class hidden. If the class is already hidden, the selection is **Make class visible**. See Hiding a Class for more information.

- **Expand**: Shows all classes subordinate to the highlighted class. This is a multi-level display operation that is more extensive than clicking the ⊙ icon, which only shows the next level of direct subclasses.
- **Collapse**: Hides all classes that are subordinate to the highlighted class. This is a multi-level display operation that is more extensive than clicking the ⊙ icon to the left of the class.


Next:

The Superclasses Pane

The Superclasses pane, in the lower left in the Classes tab, displays all the superclasses of the class currently selected in the Class Relationship Pane. This pane allows you to add and remove superclasses for a class, as well as to jump to another superclass in the Class Relationship pane and highlight a different occurrence of the same class.

Note: If you cannot see the Superclasses Pane, your window may be too small. You can see the pane by enlarging your window or by dragging the slider bar at the bottom of the Class Relationship Pane. See Working With a Small Window for more information.
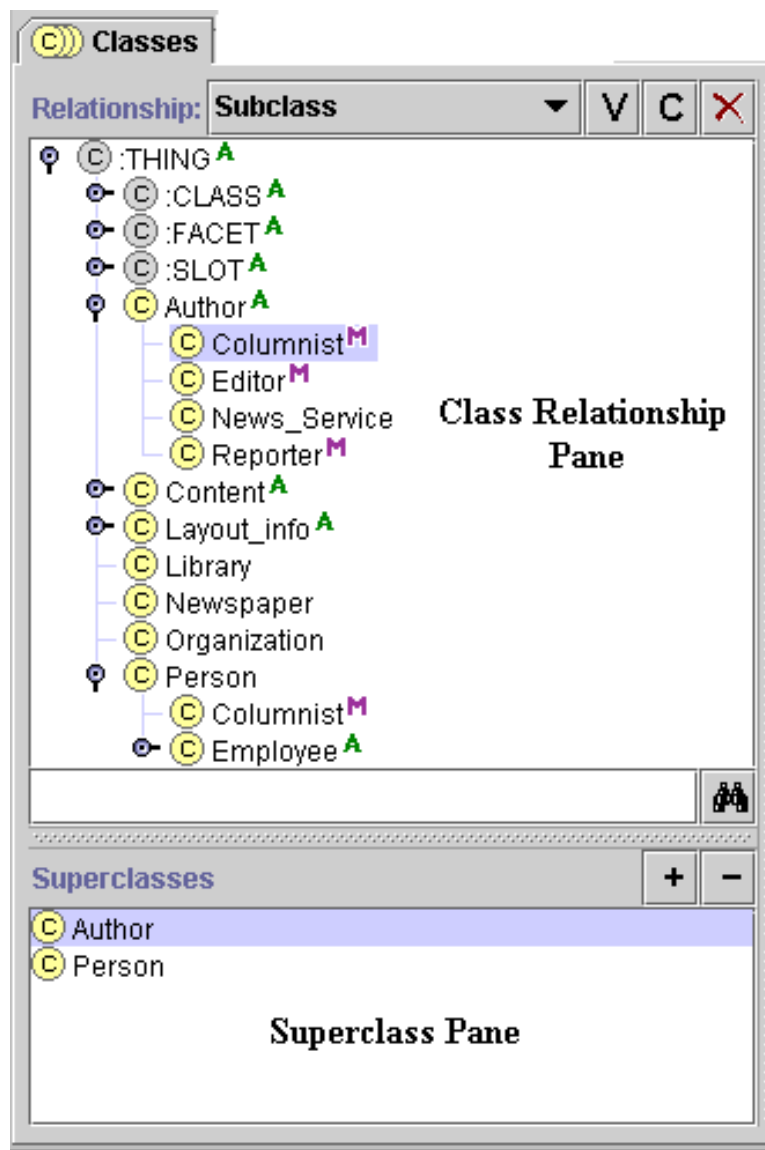
Viewing the List of Superclasses

To see a list of all superclasses of a given class, highlight any occurance of the class in the Class Relationship pane. The Superclasses pane automatically displays a list of all of the superclasses of the selected class.

Adding a Superclass

To add a superclass to the selected class, click the Add **+** button, highlight the class you want as an additional superclass in the Select Class window, and then click OK. See [Adding a Superclass](#) for more information.

Removing a Superclass

To remove a superclass from the list of superclasses for the selected class, highlight the superclass you want to remove and click the Remove button. The superclass/subclass relationship is broken, but the superclass is not deleted from the knowledge base. See [Removing a Superclass](#) for more information.

Jumping to Another Superclass

To jump to a different superclass in the [Class Relationship Pane](#), click on any superclass in the Superclasses pane. The highlight in the Class Relationship pane automatically moves to the occurrence of the class which is directly under the chosen superclass. If the occurrence is not currently visible in the Class Relationship pane, Protégé automatically scrolls to the correct location and expands the hierarchy to make the class visible. See [Jumping to Another Superclass](#) for more information.

Next: [The Class Form](#)

[Classes Table of Contents](#)

The Back-References Pane

In a complex project, it is possible to have many different relationships between classes, instances, and slots. The Back-References pane allows you to view a list of all the frames that reference the selected frame.

The following information is included in the Back-References pane:

- The View Button: Allows you to view the highlighted frame.
- The Frame column: Shows the name of each frame that references the item, with an icon indicating its type.
- The Slot Value column: Shows the slot in which the frame references the item.
- The Facet Value column: Shows any facet information for the slot.
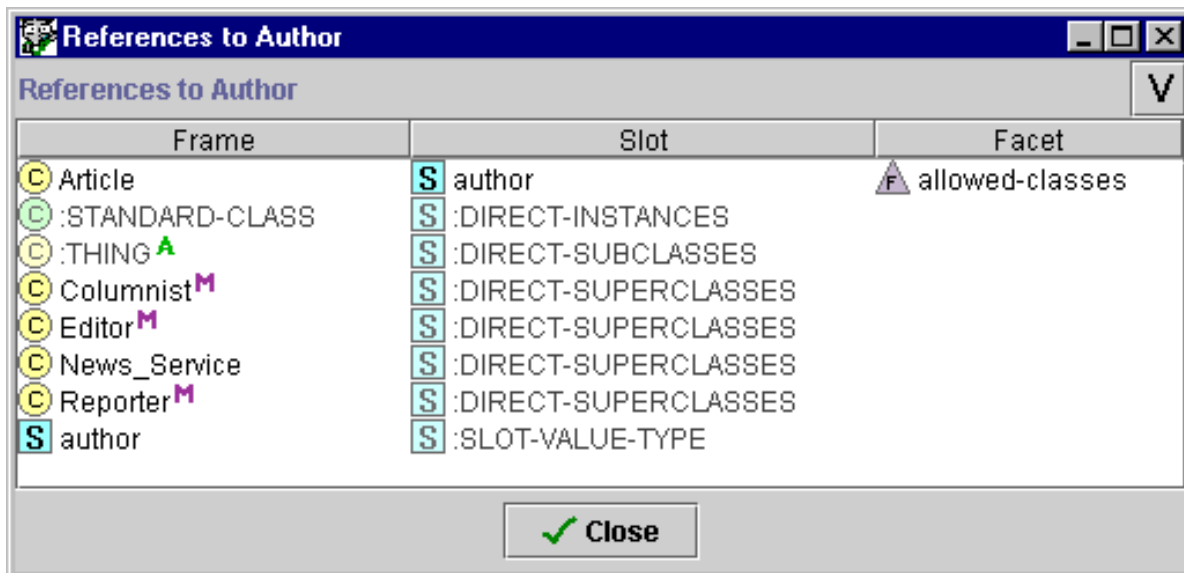
The Back-References pane can be viewed by selecting an item and clicking on the Back-References button. See Viewing Back-References for more information.

The View Button

V(iew) button: Click this button to open the form for the highlighted frame. Depending on the frame, this could be the Class Form, the Slot Form, or the Instances Form.

The Frame Column

This column shows the name of each frame that references the highlighted item, along with icons that

give further information about the referencing frame.

The icon to the left of the name indicates the type of frame. The following standard icons can appear:

	A yellow **C** icon indicates an editable class.
	A green **C** icon indicates a metaclass, that is a class that inherits from :CLASS.
	A pale **C** icon indicates a class that is imported from another project. Imported classes cannot be edited.
	A blue **S** icon indicates a direct slot.
	A gray **S** icon indicates an inherited slot.
	A pink **I** icon indicates an editable instance. For instances, the class where the instance appears is given in parentheses after the instance name.
	A gray **I** icon indicates an instance that cannot be edited. For instances, the class where the instance appears is given in parentheses after the instance name.

The icons (if any) to the right of the name give further information about the frame. The following standard icons can appear:

A	Indicates an abstract class.
M	Indicates a class with multiple superclasses.
H	Indicates a hidden class.

The Slot Value Column

For the frame in the Frame column, this column specifies the slot that actually references the selected item.

Note that Protégé-2000 provides standard slots which indicate the relationship between the selected item and the referencing frame:

DIRECT-INSTANCES	The selected item is a direct instance of the frame.
DIRECT-SUPERCLASSES	The selected item is a direct superclass of the frame.
DIRECT-SUBCLASSES	The selected item is a direct subclass of the frame.
SLOT-VALUE-TYPE	The frame is a slot and the selected item appears as one of its values. If the selected item is an instance, it may be a value. If the selected item is a class, it may be an Allowed Parent (for a slot of type Class) or an Allowed Class (for a slot of type Instance). See The Value Type menu for more information.

The name of any other slot in the project may also appear in this column.

The Facet Value Column

This column gives information about the facet value for certain types of back-references, and gives information about the slot relationships between classes.

For example, an item may appear as a value for a slot which is called by that frame. In the illustration above, the class **Author** appears as an Allowed Class for the slot **author**, as shown in the second row. In addition, the slot **author** appears as a slot for the class **Article**. Therefore, **Article** references **Author**. This reference appears in the fifth row in the example.

For an embedded reference of this type, the facet value column specifies whether the selected item appears as an Allowed Class or an Allowed Parent.

For classes, it is also possible to view those classes that are related through a particular slot using the Class Relationship Menu. See Viewing Class Relationships for more information.

Hiding a Class

You can make a class hidden using the Class Menu. You can use this, for example, if you want to restrict the user's view to a part of your knowledge base but want to retain the structure of the larger project. All subclasses of a hidden class are also hidden.

You can choose whether or not hidden classes are displayed in the Class Relationship pane by choosing **Configure** from the **Project** menu, going to the **Options** tab, and toggling the **Display Hidden Classes** option. See Configuring a Project for more information.

To hide a class:

1. Select the class in the Class Relationship Pane.
2. Click the right mouse button to display the Class Menu.
3. Select **Hide class** and click the left mouse button.
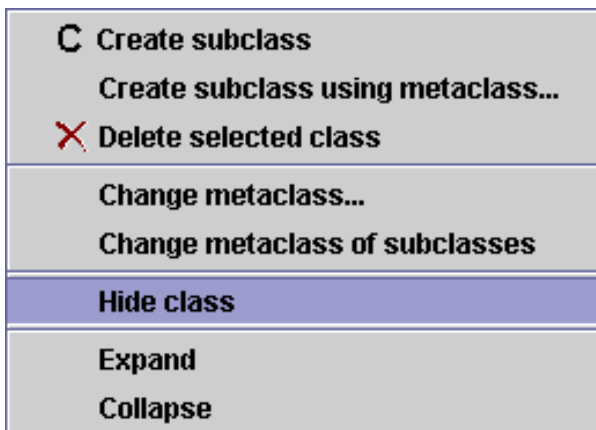
4. If hidden classes are visible in your project, you will see an **H** icon to indicate the class is hidden. If hidden classes are not visible in your project, you will no longer see the class.

To make a hidden class visible:

1. If hidden classes are not currently dsiplayed in the project, make them visible. To do this, choose **Configure** from the **Project** menu, select the **Options** tab, and make sure the **Display Hidden Classes** option is selected.
2. Select the hidden class you wish to make visible. You will know it is hidden because of the **H** icon.
3. Click the right mouse button to display the Class Menu.
4. Select **Make class visible** and click the left mouse button.
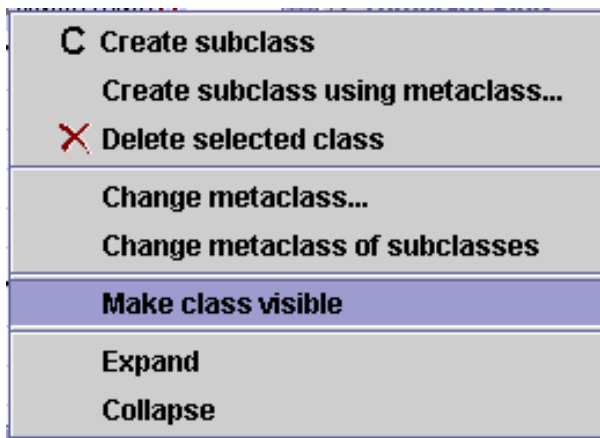
5. The **H** icon will be removed from your class. Now, if you choose not to display hidden classes, you will still be able to see the class, unless it has a superclass which is still hidden.


Jumping to Another Superclass

If a class has more than one superclass, you can use the Superclasses Pane to jump from one superclass to another in the Class Relationship Pane.

Note: If you cannot see the Superclasses Pane, your window may be too small. You can see the pane by enlarging your window or by dragging the slider bar at the bottom of the Class Relationship Pane. See Working With a Small Window for more information.

To jump to another superclass:

1. Highlight any occurrence of the class in the Class Relationship Pane.
   The superclasses of the class are displayed in the Superclasses Pane, located immediately below the Class Relationship Pane.
2. Click on any superclass in the Superclasses Pane.
   The highlight in the Class Relationship pane automatically moves to the occurrence of the class which is directly under the chosen superclass. If the copy is not currently visible in the Class Relationship pane, Protégé automatically scrolls to the correct location and expands the hierarchy to make the class visible.

In the illustration below, if you clicked on **Person** in the Superclasses pane, the highlight in the Class Relationship pane would jump to the copy of **Columnist** under **Person**. If the **Person** hierarchy was not currently expanded, it would expand to display the **Columnist** class.

Class Relationship Pane
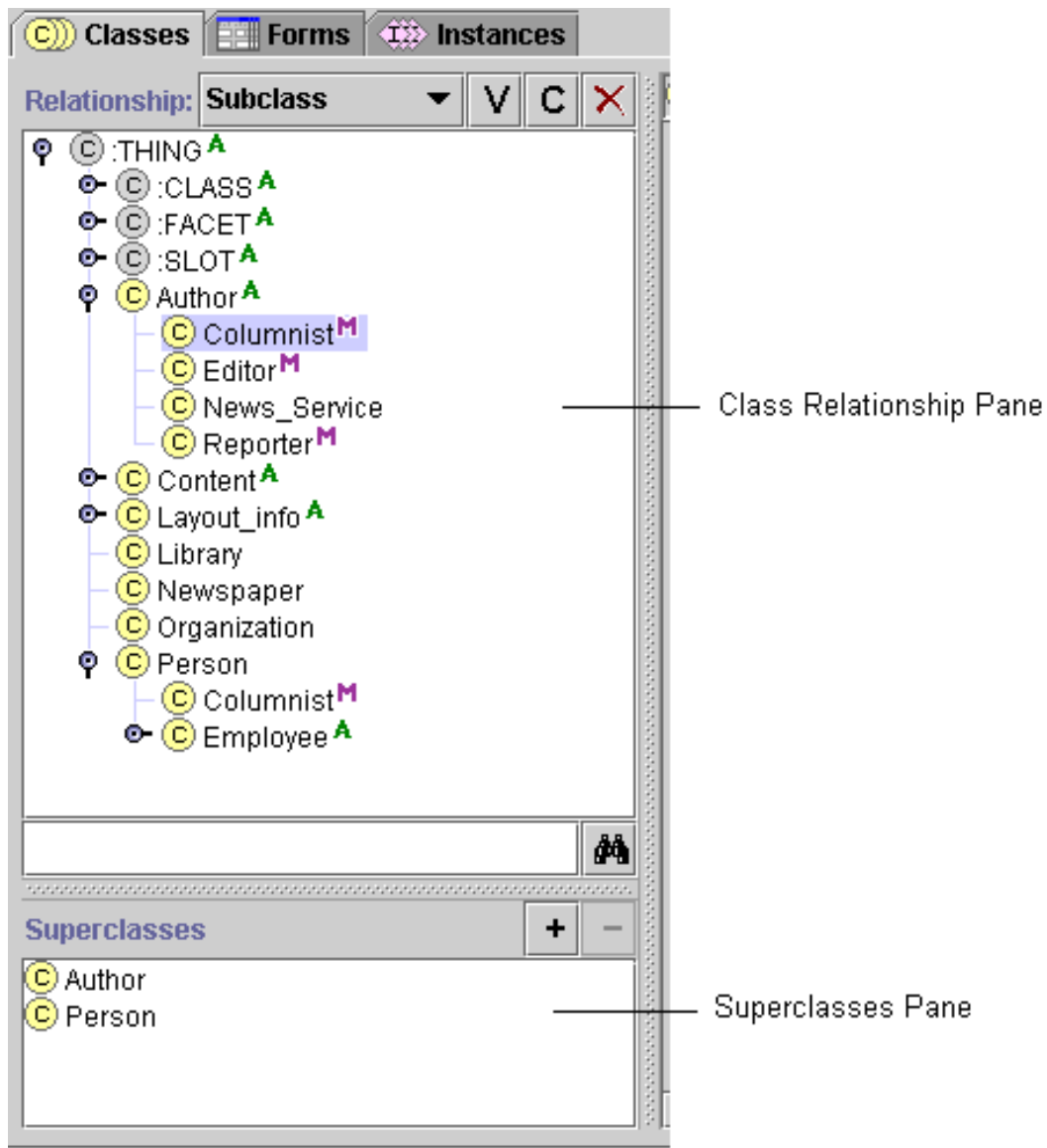
Superclasses Pane

Next:

Understanding Metaclasses

Metaclasses are part of the way Protégé internally handles and constructs classes. Although you might not use metaclasses directly in your project, Protégé uses its default metaclasses to build your classes and determine their properties. By creating your own metaclasses and defining their slots, you can create additional templates for your classes. This allows you to give your classes more complex properties and to greatly extend the power of your Protégé projects.

By creating a metaclass, you create a different template which you can use for selected classes. The new slots you create for the metaclass show up as widgets in the Class Form for any classes that use that metaclass. This allows you to attach additional information to your class at the class level. This is especially useful if you have several classes, each of which has a similar structure.

For example, suppose you are creating a knowledge base of wines and wineries. In this knowledge base, a type of wine is a class, while a specific label is an instance of that class. For your own reference, you could create a wine template that tells you, for each class of wine, which winery is, in your opinion, the best producer of that type of wine. Then, by applying the template to all the wine classes, you can store that information at the class level for each type of wine.

Metaclasses are part of the SYSTEM-CLASS hierarchy, which is included in every project. Metaclasses appear under the CLASS class. Every subclass of a metaclass is also a metaclass. All metaclasses appear with the green metaclass icon in the Class Relationship Pane.
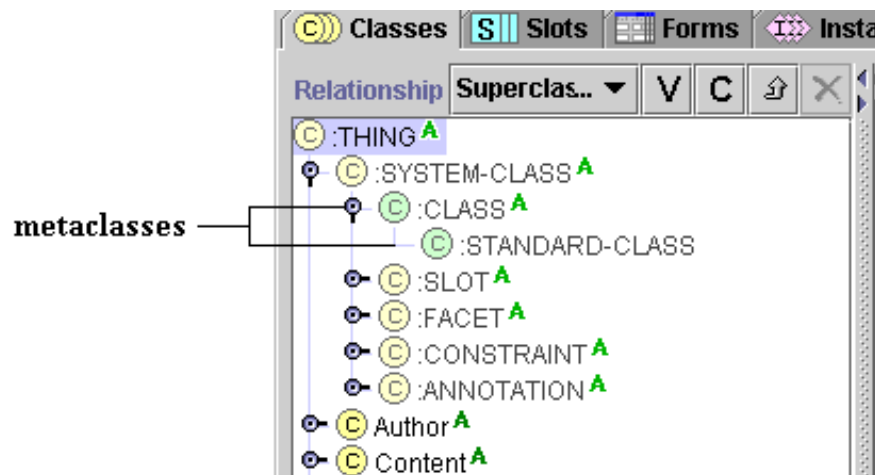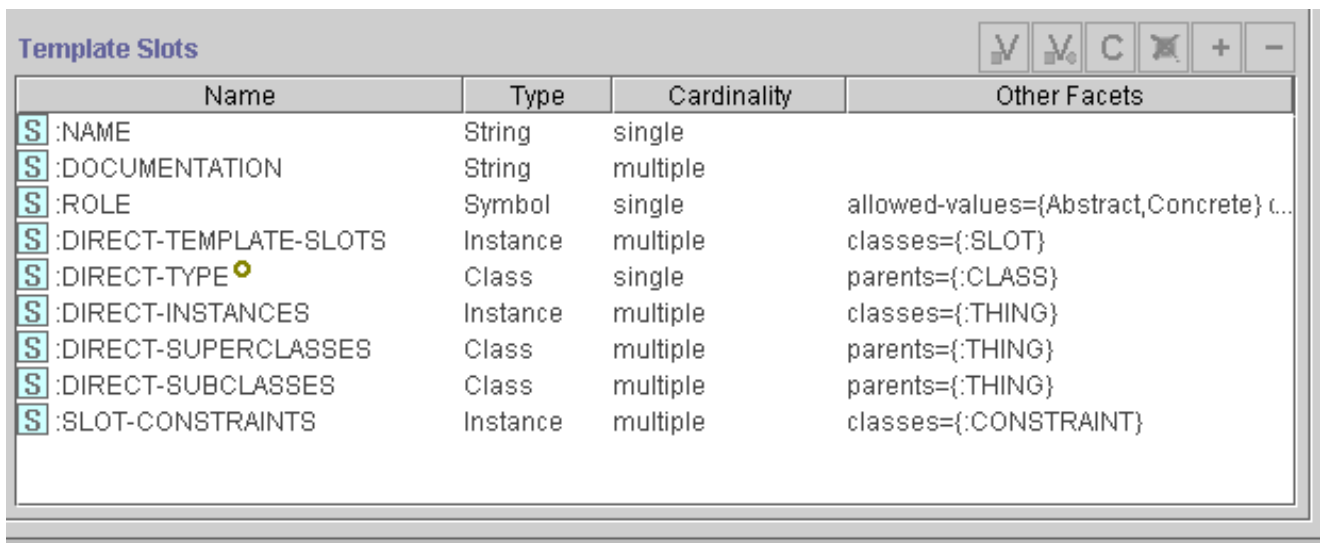

Looking at STANDARD-CLASS

By default, when a class is created as part of a project, Protégé treats that class as an instance of the metaclass STANDARD-CLASS. The properties of STANDARD-CLASS create the initial view of the class and determine the properties in the Class Form. You can look at STANDARD-CLASS to see how the slots for a metaclass translate to properties of a class.

Slots for STANDARD-CLASS

The slots for STANDARD-CLASS are shown in the Template Slots pane when you select the class. STANDARD-CLASS uses the standard slot types (String, Symbol, etc.) that are used for all classes, but does so in a fairly sophisticated way. The :NAME slot is a standard String slot which stores the class name. :DOCUMENTATION is also a String slot. The :ROLE slot controls the role of a slot and determines whether it is Abstract or Concrete, with a default of Concrete.

Template Slots

Name	Type	Cardinality	Other Facets
S :NAME	String	single	
S :DOCUMENTATION	String	multiple	
S :ROLE	Symbol	single	allowed-values={Abstract,Concrete} (...
S :DIRECT-TEMPLATE-SLOTS	Instance	multiple	classes={:SLOT}
S :DIRECT-TYPE	Class	single	parents={:CLASS}
S :DIRECT-INSTANCES	Instance	multiple	classes={:THING}
S :DIRECT-SUPERCLASSES	Class	multiple	parents={:THING}
S :DIRECT-SUBCLASSES	Class	multiple	parents={:THING}
S :SLOT-CONSTRAINTS	Instance	multiple	classes={:CONSTRAINT}

The other slots are more complex. For example, consider the :DIRECT-SUPERCLASSES slot. This slot keeps track of the direct superclasses of a class by storing them as a list of instances. Internal programmatic operations add and delete instances to from the appropriate slot value whenever you make changes to a Protégé project. In this way, Protégé represents the structure of your project using its tools for storing and representing structure, in much the same way you create the project structure.

Once you have created a metaclass, you can create all or some of the classes in your project using the new metaclass as a template instead of :STANDARD-CLASS. Clearly, creating a subclass of :STANDARD-CLASS and then creating new slots or overriding existing ones vastly extends the capabilities of your project.

STANDARD-CLASS in the Forms Tab

The widgets for each template slot on a metaclass can be viewed at the Forms Tab. In fact, internal to Protégé , the Class Form consists of widgets for :STANDARD-CLASS, and it could be modified using the Forms Tab. Look at the Forms Tab for :STANDARD-CLASS below and see how it reflects the template slots listed above. Note that the browser key is automatically set to :NAME.
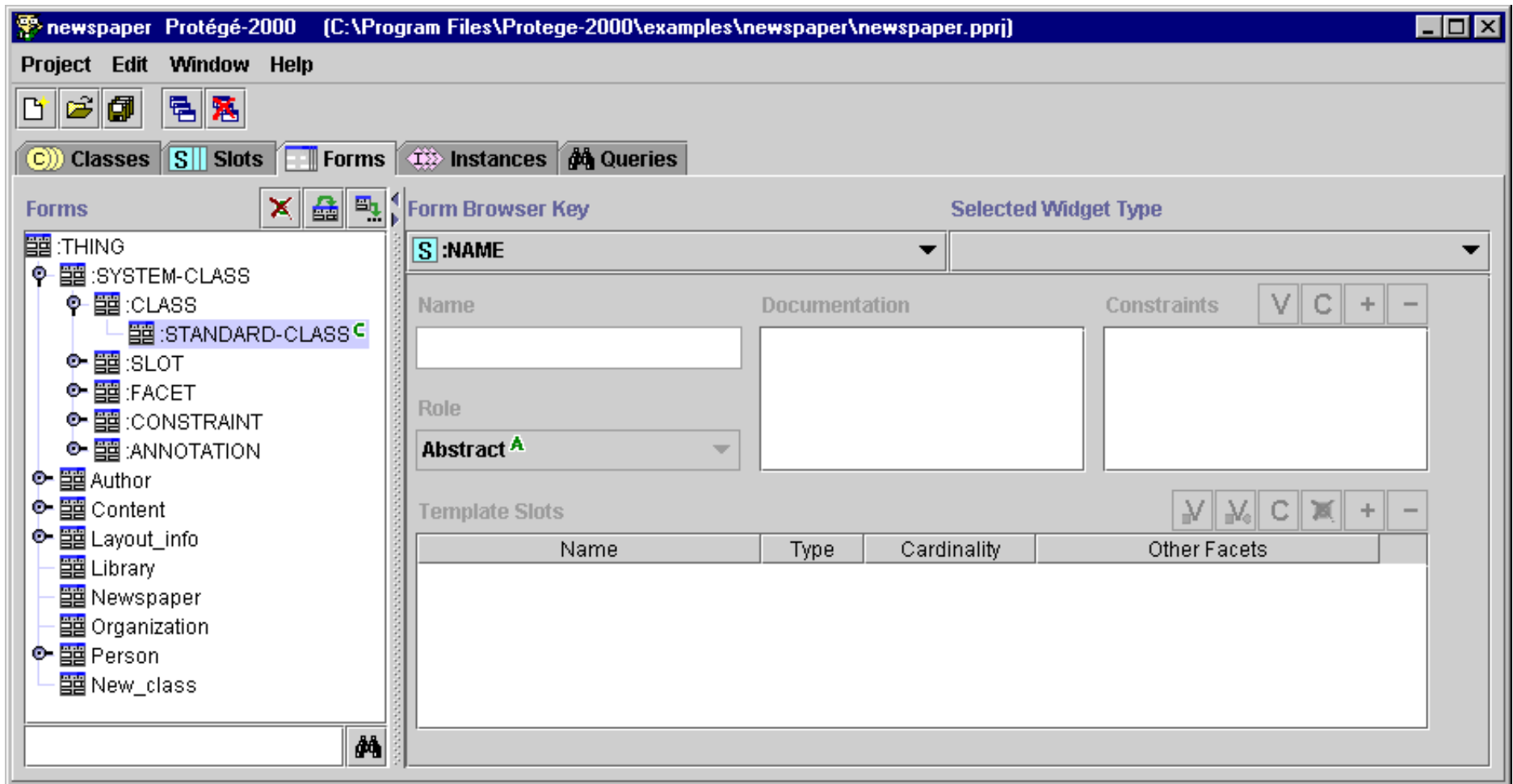
Class Form for a Class Created from STANDARD-CLASS

Since any new class is by default created using :STANDARD-CLASS, the Class Form itself reflects the structure and layout specified for :STANDARD-

CLASS at the Forms Tab. If you create a new project, then modify the Class Form at the Forms Tab, your modifications will show up for every class, whether existing or new.

Because of the power of metaclasses, you may want to be particularly cautious how much you experiment with a current project. It is a good idea to work on a copy.

Next:

Creating a Metaclass

Note: Before you create and use metaclasses, you should be confident with the basic Protégé interface and be comfortable designing a project, and creating and modifying classes, slots, forms and instances.

Despite the power of metaclasses, their basic interface is very simple. Creating a metaclass is almost identical to creating a class. You simply have to make sure the class is created subordinate to :CLASS. Every metaclass is subordinate to :CLASS. By default, every class subordinate to :CLASS is a metaclass, unless you change it. Frequently, it is desirable to create a metaclass subordinate to :STANDARD-CLASS, so that the classes created using the metaclass will have the various properties defined by the :STANDARD-CLASS slots. Without these properties, you cannot name the class or add template slots.

To create a new class as a metaclass:

1. Click on the desired superclass in the Class Relationship pane. The selected superclass must itself be a metaclass, as indicated by a green class icon ©. As mentioned above, this will be true only if the selected superclass is subordinate to :CLASS.
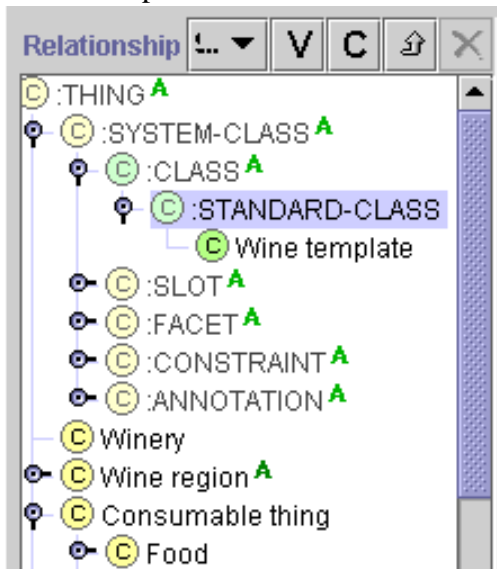
2. Click the **C(reate)** button at the right of the Class Relationship Pane. The new class will be added under the highlighted class. A green class © icon will indicate that it is a metaclass. It will inherit the properties of the selected metaclass, including any template slots.
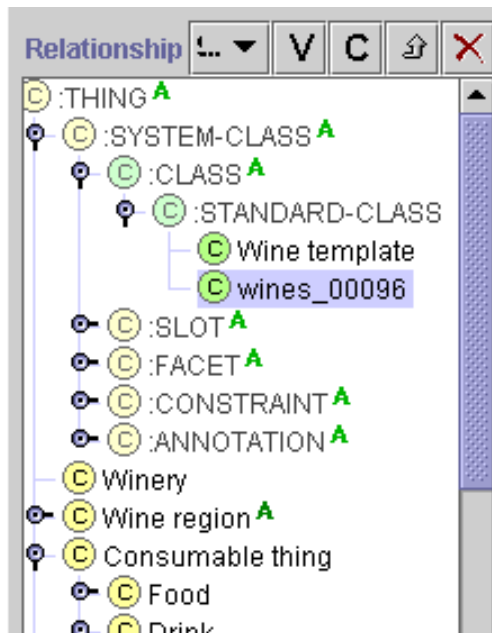
3. Use the Class Form to name the class, create constraints, and create and edit slots.

For example, click the **C(reate)** C template slots button at the right of the Template Slots pane to create a new slot. This slot will show up as an instance widget for any class you create using the metaclass.

Next:

Creating a Class Using a Metaclass

Note: Before you create and use metaclasses, you should be confident with the basic Protégé interface and be comfortable designing a project, and creating and modifying classes, slots, forms and instances.

Once you have added one or more metaclasses to your project, either by creating them directly, or by including a project that already has metaclasses, you can select the metaclass you want to use to create a class. When you create a class using a non-standard metaclass, the class is given the attributes specified by the selected metaclass. Unless you make changes, every new subclass you create for the class will also use the selected metaclass.

This example uses the wines project, which includes a Wine template that supplies metaclass structure.

If you create a class using the Create **C** button, you will use the metaclass of the direct superclass as the metaclass for the new class. Unless you have made changes, this is the default :STANDARD-CLASS.

To create a class using a non-standard metaclass:

1. In the Class Relationship Pane, find the class that you want as the superclass of the new class.
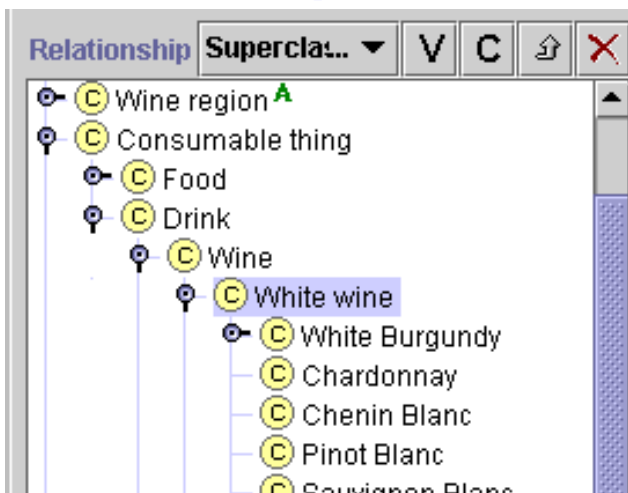2. Click the right mouse button and select "Create subclass using metaclass..."
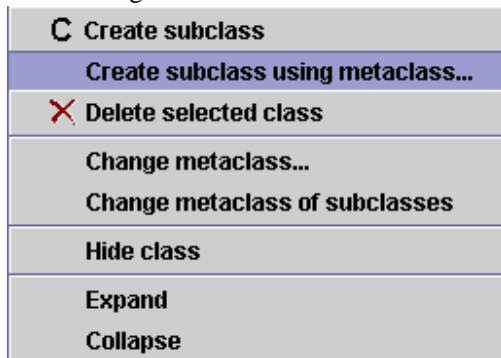
3. A dialog box displays the Concrete metaclasses, which are the classes you can use when creating a class. You cannot create a class as an instance of an Abstract metaclass.
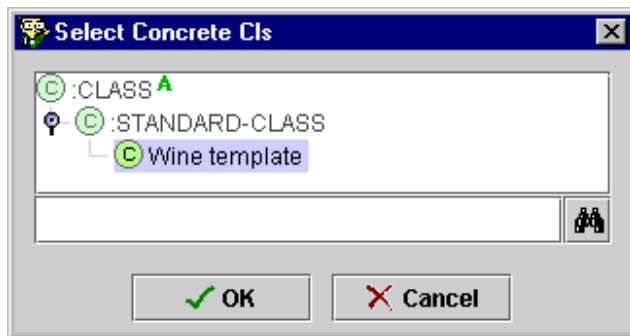
4. Select the metaclass that has the properties that you want and click OK.

The new class will be added under the highlighted class. It will have the Class Form determined by the selected metaclass. The name of the metaclass you used is displayed at the top of the Class Form, immediately after the name of the class. The Class Form may include additional widgets that are not part of :STANDARD-CLASS.
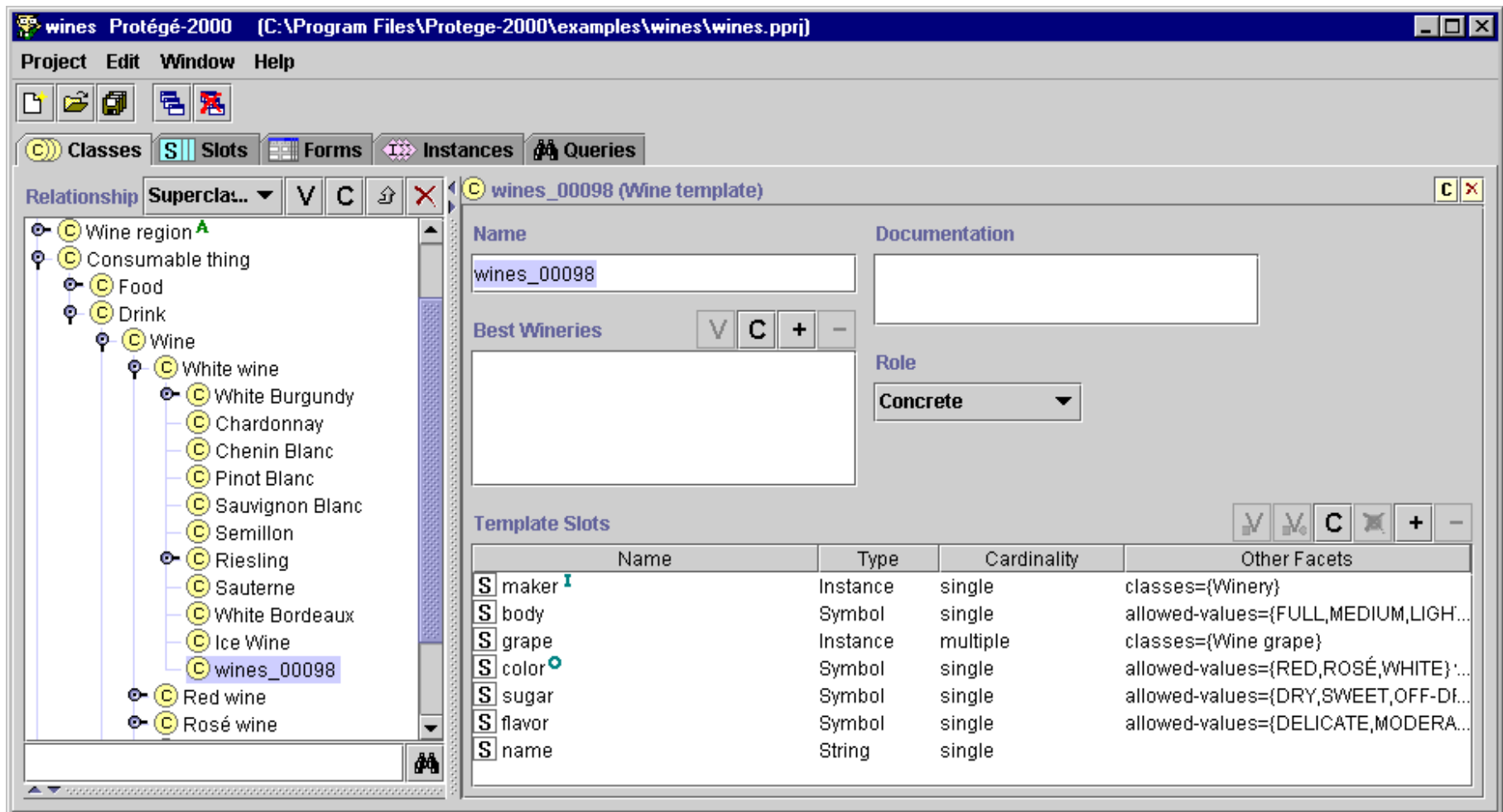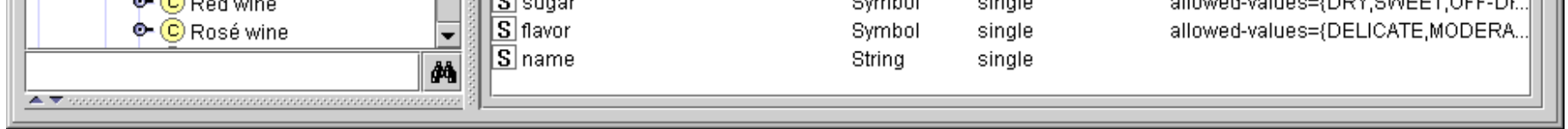
5. Use the Class Form to name the class, choose its role, create constraints, and create and edit slots.

Next: Changing the Metaclass of a Class

Classes Table of Contents

Changing the Metaclass of a Class

Note: Before you create and use metaclasses, you should be confident with the basic Protégé interface and be comfortable designing a project, and creating and modifying classes, slots, forms and instances.

For any existing class in your project, you can change the metaclass you use for that class. This gives the class and the Class Form the attributes defined by the new metaclass. New classes that you create as subclasses of the class will also use the new metaclass. However, existing subclasses will continue to use their previously assigned metaclass unless you specifically change it.

This example uses the wines project, which includes a Wine template that supplies metaclass structure.

To change the metaclass of an existing class:

1. In the Class Relationship Pane, click on the class that you want to change.

2. Click the right mouse button and select "Change metaclass..."
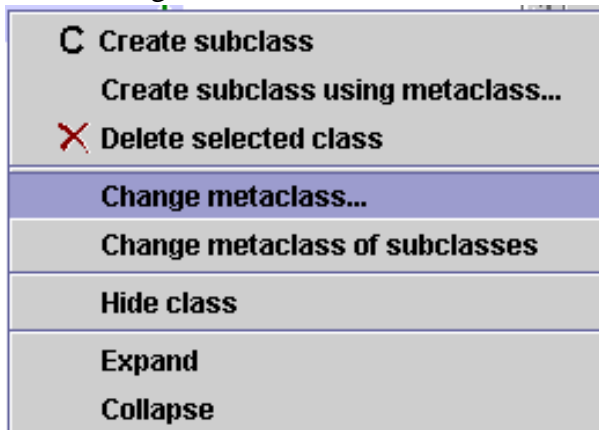
3. A dialog box displays the Concrete metaclasses, which are the classes you can use. You cannot use the Abstract metaclasses in this way. Note that if the class currently uses a non-standard class, you can revert to :STANDARD-CLASS.

4. Select the metaclass that has the properties that you want and click OK.
   The highlighted class will now have the Class Form and properties determined by the selected metaclass.

Next:

Changing the Metaclass of Subclasses

Note: Before you create and use metaclasses, you should be confident with the basic Protégé interface and be comfortable designing a project, and creating and modifying classes, slots, forms and instances.

When you change the metaclass of a class, all of its subclasses will retain their previous metaclass. This ensures that you do not propagate changes when you do not want to. However, often you will want existing subclasses to have the same metaclass as their parents. Protégé provides an option to quickly change the metaclasses of all the classes subordinate to a given class. Note that you only have to do this once; by default, new classes use the metaclass of their direct superclass.

You can only change the metaclasses to match the metaclass of the selected class. If the selected class has :STANDARD-CLASS as its metaclass, all subclasses will lose any additional metaclass information.

To do work with metaclasses, you must have added one or more metaclasses to your project, either by creating them directly, or by including a project that already has metaclasses. This example uses the wines project, which includes a Wine template that supplies metaclass structure.
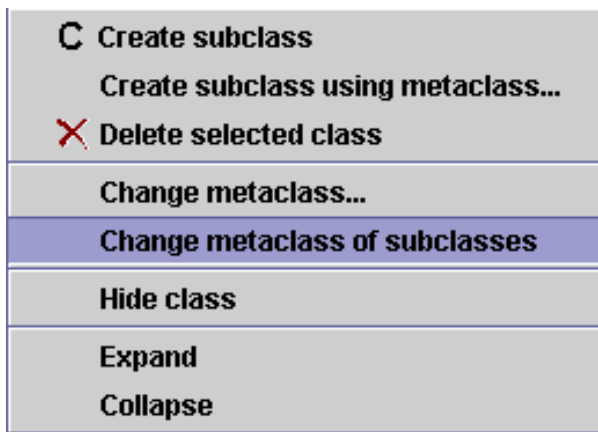
To change the metaclass of all existing subclasses of a given class:

1. In the Class Relationship Pane, find the class whose sublcasses you want to change.
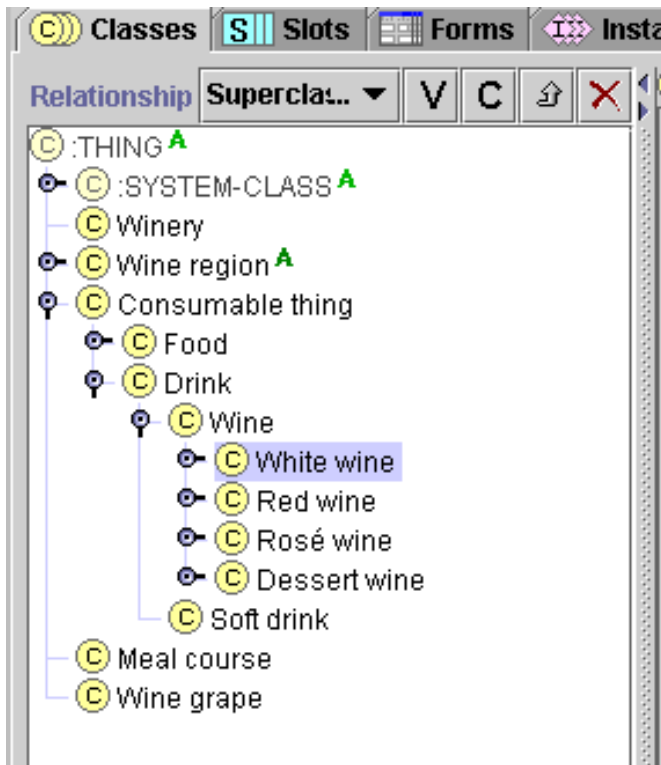
2. Click the right mouse button and select "Change metaclass of subclasses."

C	**Create subclass**
	Create subclass using metaclass...
✕	**Delete selected class**
	Change metaclass...
	Change metaclass of subclasses
	Hide class
	Expand
	Collapse

3. A dialog box pops up for verification. Click OK to continue.

Change metaclass of all subclasses of White Burgundy to Wine template

✓ OK   ✕ Cancel
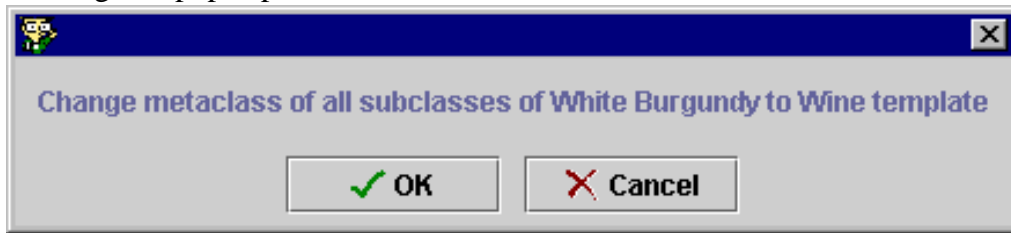
4. The change is made and all subordinate classes now have the metaclass of the selected class.

The Classes & Instances Tab

The Classes & Instances tab is an optional tab that combines the functionality of the Classes Tab and the Instances Tab into a single window. In this window, you may view, create, and edit classes *and* instances.

To display the Classes & Instances tab, use the Configure Project dialog box.
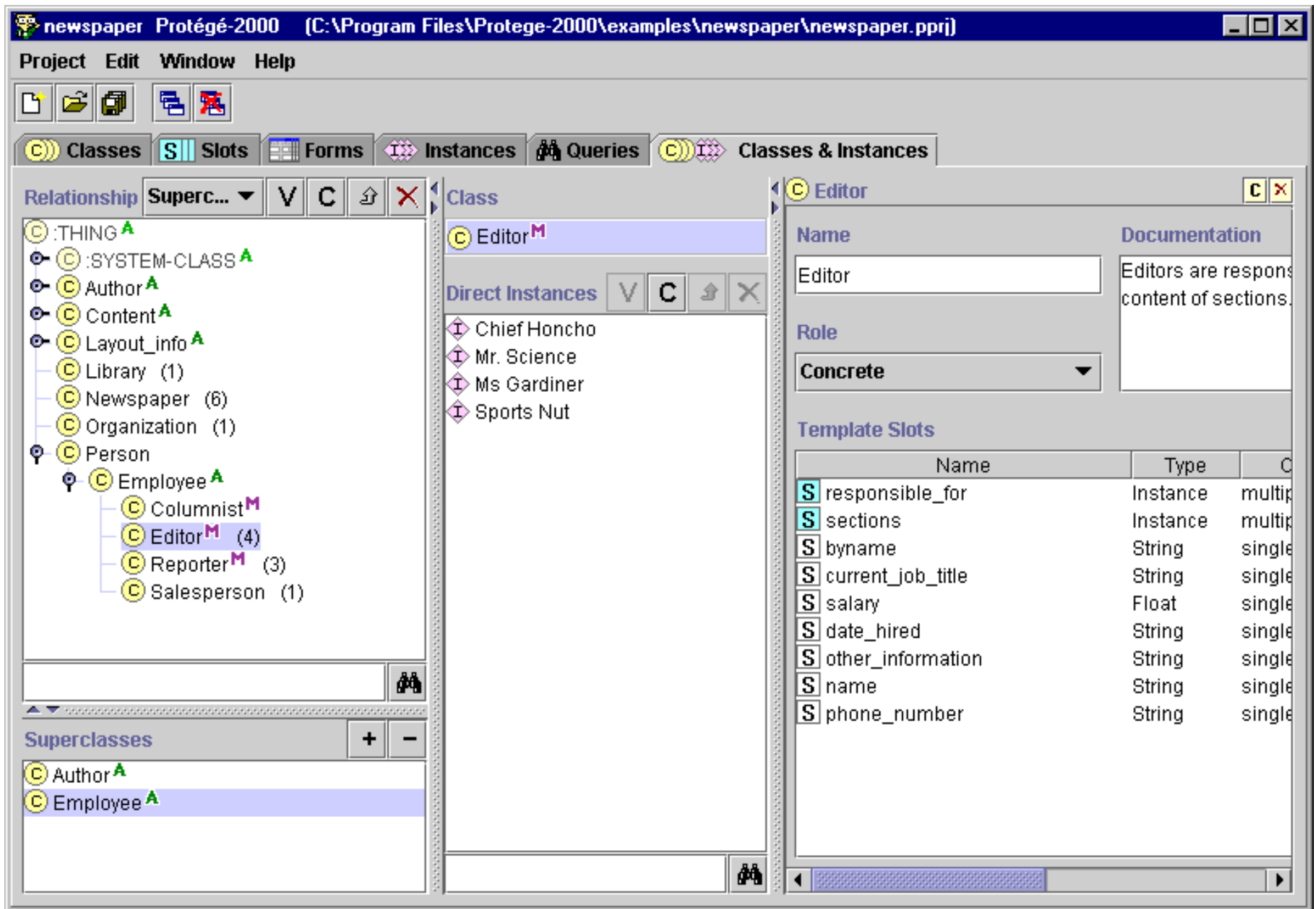
The Classes & Instances tab includes the following components:

1. A Class Relationship pane in the upper left, which shows the class hierarchy and allows you to create, delete, and edit classes. Note that this pane has all the class buttons and drop-down menu functionality that is available in the Class Relationship Pane at the Classes Tab
2. A Superclasses pane in the lower left which allows you to perform superclass operations. See the Superclasses Pane for more information.
3. A Class bar, which indicates the currently selected class.
4. A Direct Instances pane, which provides all the functionality of the Instances Pane, allowing you to create and delete instances.
5. A Form Pane on the right, which displays the information for the current selection. When a single class is selected, the Form pane contains the Class Form for the selected class. When a single instance is selected, the Form pane contains the Instances Form for the selected instance.

For information about the Classes tab user interface and accomplishing class-related tasks, see the Classes Table of Contents. For information about the Instances tab user interface and accomplishing instances-related tasks, see the Instances Table of Contents.

Next: Slots Table of Contents

Classes Table of Contents

See also: Instances Table of Contents

Clearing Overrides From a Slot

You can remove any class-level overrides from a slot using the Clear Slot Overrides ✖ Template Slot button in the Template Slots pane.

Note: Recall that a slot that has been edited at a class is shown with an override ⊙ icon. Class-level modifications only affect the slot at the current class and its subclasses. The top-level properties of the slot remain unchanged.

To remove the slot overrides at a class:

1. Select a class in the Class Relationship pane in the Classes tab.
2. Select the slot whose overrides you wish to remove in the Template Slots pane. Overrides are indicated by the override ⊙ icon.
3. Click the Clear Slot Overrides ✖ Template Slot button at the upper right of the Template Slots pane. The overrides will be cleared from the slot at this class and all of its subclasses. The slot will now match the top-level definition. Any overrides created at an unrelated class will still exist.
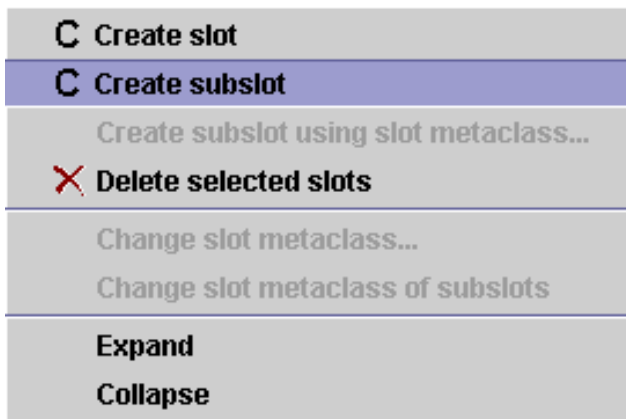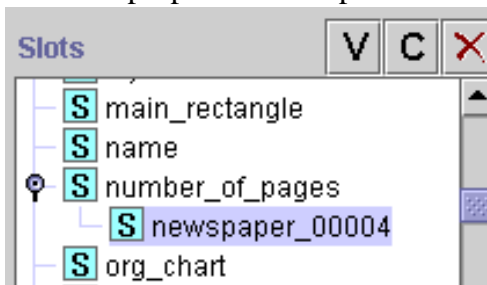

Next:

Creating a Subslot

Normally, whenever you create a new slot, it is a top-level slot. However, you can also create a subslot of an existing slot. You might do this, for example, when you want two slots which share information, but where one slot (the subslot) is more restrictive.

To create a subslot of a slot:

1. In the Slot Tab, select the slot that you want as the parent of your subslot.
2. Click the right mouse button.
3. Select **Create subslot** from the slot menu.

   C **Create slot**
   C **Create subslot**
   Create subslot using slot metaclass...
   ✕ **Delete selected slots**
   Change slot metaclass...
   Change slot metaclass of subslots
   Expand
   Collapse

4. The new slot is created as a subslot of the selected slot. Except for the name, when created, it has the same properties as its parent slot.

   Slots V C ✕
   S main_rectangle
   S name
   S number_of_pages
     S newspaper_00004
   S org_chart

5. You can use the Slot Form to edit the properties of the new slot.

Next:

Changing the Metaslot of a Slot

Note: Before you create and use metaslots, you should be confident with the basic Protégé interface and be comfortable designing a project, and creating and modifying classes, slots, forms and instances.

If you have multiple metaslots in your project, then for any existing slot, you can change the metaslot you use for that slot. This gives the slot and the Slot Form the attributes defined by the new metaslot. Subslots that you create subordinate to the slot will also use the new metaslot. However, existing subslots will continue to use their previously assigned metaslot unless you specifically change it.

To change the slot metaclass (metaslot) of an existing slot:

1. In the Slots Tab, click on the slot that you want to change.
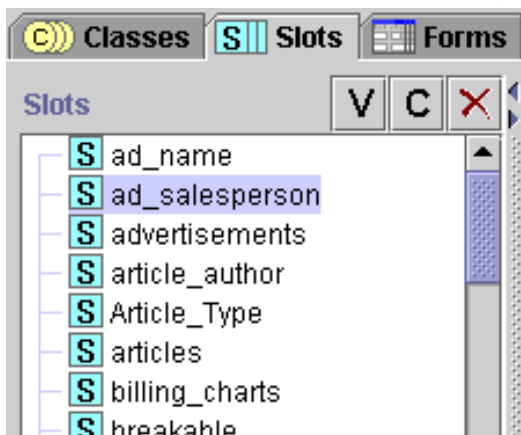
2. Click the right mouse button and select **Change slot metaclass...**.
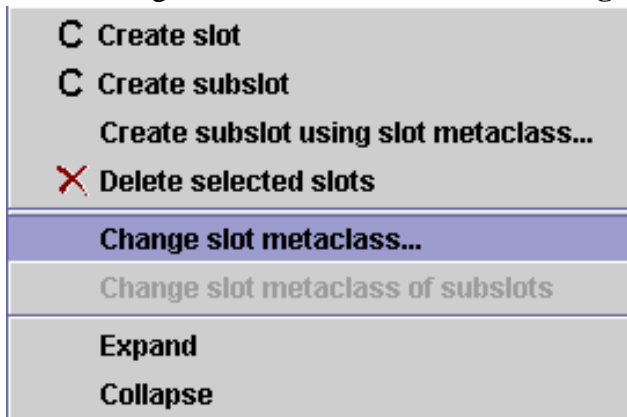
3. A dialog box displays the slot metaclasses. Note that if the class currently uses a non-standard class, you can revert to :STANDARD-SLOT.
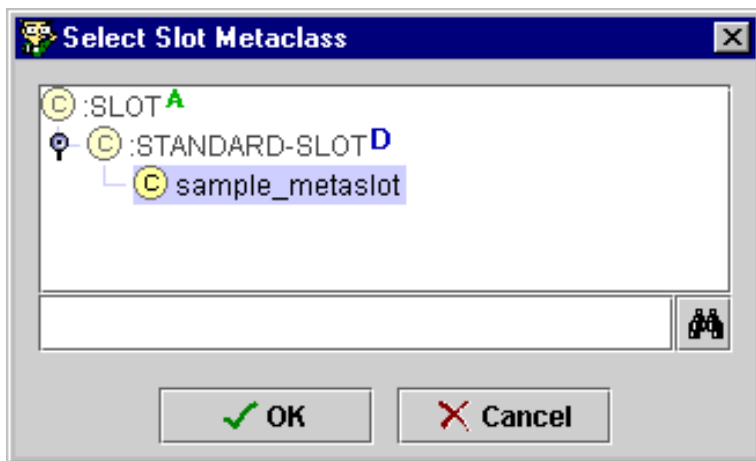
4. Select the metaslot that has the properties that you want and click OK.
   The highlighted slot will now have the Slot Form and properties determined by the selected
   metaslot.

Next:

Removing a Slot From a Class

You can remove a slot from a class without deleting it from the knowledge base. To do this:

1. In the Class Relationship pane, select the class which has the slot you want to remove. The slots for the class will be displayed in the Template Slots pane. Note that the slot must be a direct slot for the class; you may not remove an inherited slot.
2. Highlight the slot you wish to remove in the Template Slots pane.
3. Click the Remove ▬ Template Slot button. The slot no longer appears among the slots for the class. The slot is also removed from any subclasses of the class.

To remove several slots at once:

1. In the Class Relationship pane, select the class which has the slot you want to remove.
2. Highlight the slots you wish to remove in the Template Slots pane. To highlight multiple slots, hold down the Ctrl key while clicking each slot. To highlight a range of slots, click the first slot, then hold down the Shift key and click the last slot in the range.
3. Click the Remove ▬ Template Slot button. The slots no longer appear among the slots for the class. The slots are also removed from any subclasses of the class.

Removing a slot cannot be undone. However, if you close Protégé-2000 without saving changes, you will revert to the last saved version. If you have made extensive changes to your project during the current session, you may wish to save before deleting slots. To do this, select **Save** from the **Project** menu.

You can also **delete** a slot from the entire knowledge base using the Slots tab. See Deleting a Slot for more information.

Next: Deleting a Slot

Slots Table of Contents

Understanding Inverse Slots

The **Inverse Slot** widget in the [The Slot Form](#) allows you to create a reciprocal relationships between two slots, so that as one slot is filled in at one instance, its inverse slot is automatically filled in at another instance, according to the relationship you specified.

This section gives an overview of how to analyze a project and design inverse slots. For information on how to use the user interface, see [Creating an Inverse Slot Relationship](#).

For example, suppose you want to keep track of which editor(s) edit which section(s). There is already a slot **sections** at the class **Editor** which takes as values instances in the class **Section**. You could create an inverse slot **editor** at the class **Section** which takes as values instances in the class **Editor**. Now, at the instance **Mr. Science** you assign the value **Science** to the slot **sections**. Then, at the instance **Science**, **Mr. Science** automatically appears as a value for the slot **editor**.

This section considers the most common relationship, an inverse slot relationship between two instances.

To design and create inverse slots:

1. First identify the inverse relationship.
2. Second, analyze the Protégé components of the relationship.
3. Finally create or assign the slots to the appropriate classes and create the inverse slot relationship between them.

Identify the Relationship

This is a reciprocal relationship that always occurs: for example, if Mr. Science edits the Science section, the Science section is edited by Mr. Science. Different types of inverse relationships are possible. Take the following criteria into consideration:

1. You can restrict the relationship to a one-to-one correspondence *OR* you can allow one or both of the slots to have multiple values. For example, Chief Honcho edits a number of sections.
2. The relationship can be between instances in different classes *OR* instances in the same class. For example, *likes* and *is_liked_by* would be a relationship where the slot and its inverse slot are both attached to **Person**.
3. A slot can be its own inverse. *shares_ an_ office_ with* would be a candidate for an inverse relationship where the inverse slot was identical with the original slot. This is a stronger tie than simply having instances in the same class.
4. The relationship must be reciprocal.

Analyze the Relationship in Protégé

Once you have identified the relationship in your ontology, you need to analyze it in terms of your Protégé project. For an inverse slot relationship between Instance slots, determine the following:

- What Protégé class(es) do you want for the first set of instances? Make sure to choose a class or classes that are appropriate for the project. It is most important to choose a class list that is not too small; that is, you need to select a list of classes such that every possible instance is in one of those classes. At the same time, for simplicity you want the class list to be focused on the instances and not be too broad.
  For example, in the **newspaper** project, one of the reporters, Larry Tennis-Nut, is an eager beaver and sometimes edits the Sports section. It would make sense to expand the list of classes to **Editor** and **Reporter**. However, no salesmen or columnists are allowed to edit a section. Therefore, while it would be possible to simply use the class **Employee**, it is cleaner not to. Call this first list of classes **Class_List_A**. It may well be a single class.
- If the second set of instances is different from the first, perform the same analysis for the second set of instances. Here, we will restrict ourselves to instances in the class **Section**.
  Call this second list of classes **Class_List_B**. It may well be a single class. It might also be identical to **Class_List_A.**

Based on this information, here is what you need to do in your Protégé project:

1. Create a slot of type Instance which takes values in **Class_List_A**. We'll call this *original_slot*.
2. Create another slot of type Instance which takes values in **Class_List_B**. We'll call this *inverse_slot*.
   If the information is important in your ontology, it is likely that one or both of these slots already exist in the project.
3. Assign your Protégé slots to the appropriate classes. First, assign *original_slot* to every class in **Class_List_B**.
   Recall that the target values of *original_slot* are in **Class_List_A**. Assigning the slot to **Class_List_B** is what creates the cross-connection between the classes.
4. Similarly, assign *inverse_slot* to every class in **Class_List_A**.
5. Make *inverse_slot* the inverse of *original_slot*. You can actually do this as soon as both slots are created.

Note that if you create an inverse slot relationship after some instances have been created, existing instances will not display the inverse slot information.

Create the Inverse Slot Relationship

For information on how to use the Protégé interface to make an inverse slot, see [Creating an Inverse Slot Relationship](#).

Next:

Creating an Inverse Slot Relationship

The Inverse Slots widget on the Slot Form allows you to create an inverse relationship between two slots of type Class or Instance. For this to work correctly, your two slots and the classes where they appear must be designed appropriately. For slots of type Instance, the following situation provides optimal results:

- All classes where the first slot is attached appear in the Allowed Classes pane for the second slot.
- All classes where the second slot is attached appear in the Allowed Classes pane for the first slot.

However, depending on the structure of your knowledge base, this may not be appropriate or feasible. Other situations are also valid. For example, suppose slots A and B are inverse slots. If one of the allowed classes for slot A does not have B as one of its attached slots, all that happens is that inverse slot value does not get created for those instances. The inverse relationship will still work normally for those allowed classes that do have slot B attached.

A slot can have at most one inverse slot. For more information on designing an inverse slot relationship, see Understanding Inverse Slots.

Note that if you create an inverse slot relationship after instances have been created, existing instances will not display the inverse slot information.

You can create an inverse slot relationship in one of two ways:

- by linking two existing slots
- by creating a new slot as the inverse of an existing slot

You can also modify inverse slot relationships:

- by removing the inverse slot link between two slots
- by replacing an inverse slot with another slot

Linking Two Existing Slots

To create an inverse slot relationship between two existing slots, the slots you choose must be of value type Class or Instance. To create the relationship:

1. Open the Slot Form for one of the two slots. You can do this from the in the Class Tab, or from the Slots Tab. Which of the two slots you choose does not matter.

2. Click the Add ✛ button at the right of the Inverse Slot field. If the button is grayed out, you cannot create an inverse slot for the selected slot. Make sure the slot is of type Class or Instance.

3. Select the slot you want to use from the Select Slot dialog box. Only slots of type Class or Instance will

appear.

4. Click OK. The two slots will now be inverses of each other. You can view the Slot Form of the second slot by double-clicking on its name in the Inverse Slot field. You will see that it already has the first slot filled in as its inverse slot. It is also good to verify that the Allowed Classes of each slot contains all the possible classes where the other slot will appear.

Creating a New Inverse Slot

To create a new slot and make it the inverse of a slot:

1. Open the Slot Form for the slot you want to create an inverse for. You can do this from the in the Class Tab, or from the Slots Tab. Which of the two slots you choose does not matter.
2. Click the Create **C** button at the right of the Inverse Slot form. If the button is grayed out, you cannot create an inverse slot for the selected slot. Make sure it is of type Class or Instance.
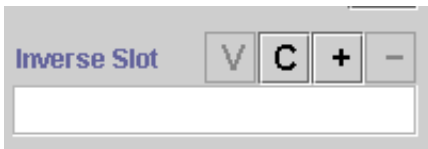
3. A new class of type Instance is created. The Slot Form for the new class opens automatically.

4. Name the slot.
5. To ensure that the reciprocal relationship works correctly, add the classes where your original slot appears to the Allowed Classes field.
6. Make any other changes to the slot definition.

7. Attach the new slot to the classes where you wish it to appear. Remember that attaching a slot to a class also attaches it to all subclasses of the class.

Removing an Inverse Slot Relationship

Removing the inverse slot relationship removes the link between the two classes. Note that if you remove an inverse slot relationship after instances have been created, existing values that were created by the inverse slot relationship are *not* removed. Removing an inverse slot only affects the values of new instances. To remove the inverse slot relationship between two slots:

1. Highlight the Inverse Slot field in the Slot Form for either one of the two slots.
2. Click the Remove ‐ button at the right of the Inverse Slot field. The slot relationship will be removed.

Replacing an Inverse Slot

Replacing the inverse slot relationship at a slot removes the link to the original inverse slot and makes a link with a different, existing, slot. Note that if you replace an inverse slot relationship after instances have been created, existing values that reflect the original (replaced) inverse slot relationship are retained. Replacing an inverse slot only affects the values of new instance. To replace an inverse relationship:

1. Select the slot you want to have a different inverse.
2. Click the Add **+** button at the right of the Inverse Slot field.
3. Select the slot you want from the Select Slot dialog box.
4. Click OK. The new slot becomes the inverse slot of the current slot. The original inverse slot is now unlinked from the slot where you made the change and no longer has an inverse slot.

Next:

![icon] Understanding Metaslots

Metaslots, or slot metaclasses, are similar to metaclasses, except that they determine the properties of slots instead of classes. A slot metaclass is a metaclass that can be used as a template to determine the way Protégé constructs a slot. By creating a new metaslot, you can create a different template for selected slots. This allows you to attach additional information to your slot at the class level. This is especially useful if you have several slots, each of which has a similar structure.

If you have multiple metaslots in your project, Protégé allows you to choose which metaslot you want to use as the default for every new slot you create. This feature is not available for classes in Protégé-2000, since it is more useful for slots.

Metaslots are part of the :SLOT hierarchy in the Class Tab, which is included in every project. Every subclass of a metaslot is also a metaslot.

Note that metaslots appear at the Class Tab. This is because every slot is an *instance* of a slot meta*class*. Slots are *not* values or subslots of a global slot.
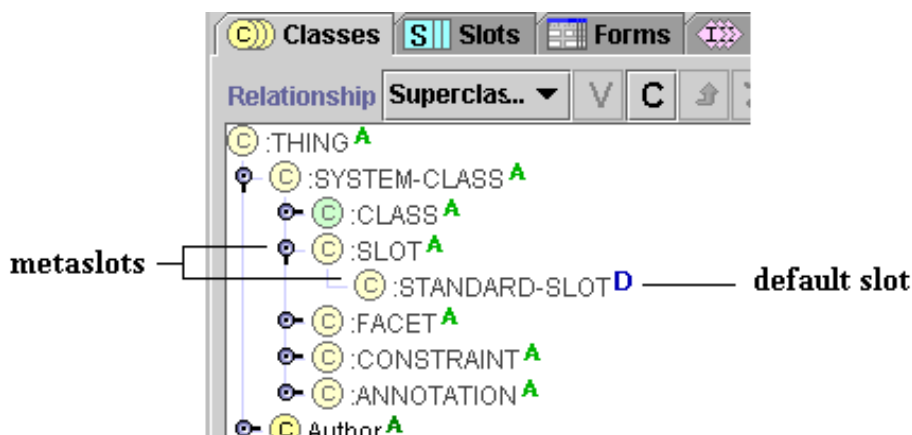
STANDARD-SLOT

By default, when a slot is created as part of a project, Protégé treats that slot as an instance of the slot metaclass :STANDARD-SLOT. The properties of :STANDARD-SLOT create the initial view of the slot and determine the properties in the Slot Form. Just as with :STANDARD-CLASS, you can look at :STANDARD-SLOT to see how the slots for a metaslot translate to properties of a slot.

In a new project, :STANDARD-SLOT has a Default ![D] icon to indicate that it is the default metaslot used for creating new slots. If you create a new metaslot and change the default, the new default will have the ![D] icon.

Slots for :STANDARD-SLOT

Just as with :STANDARD-CLASS, the slots for :STANDARD-SLOT translate into fields in the Slot Form, as well as other associated properties of the slot. Various slots of :STANDARD-SLOT create the entry fields and any default values for Value Type, Cardinality, Maximum and Minimum, etc. The correspondence between slots for a

metaclass and fields in the Class Form is explained in more detail in [Understanding Metaclasses](#).

Because of the power of metaclasses, you may want to be particularly cautious how much you experiment with a current project. It is a good idea to work on a copy.

Next: [Creating a Metaslot](#)

Creating a Metaslot

Note: Before you create and use metaslots (slot metaclasses), you should be confident with the basic Protégé interface and be comfortable designing a project, and creating and modifying classes, slots, forms and instances.

Creating a metaslot is almost identical to creating a metaclass. You simply have to make sure the class is created subordinate to :SLOT. Every metaslot is subordinate to :SLOT. By default, every class subordinate to :SLOT is a metaslot, unless you change it. Frequently, it is desirable to create a metaslot subordinate to :STANDARD-SLOT, so that the slot created using the metaslot will have the various properties defined by :STANDARD-SLOT.

To create a new metaslot:

1. Click on the desired superclass in the Class Relationship pane. The selected superclass must itself be a metaslot. As mentioned above, this will be true only if the selected superclass is subordinate to :SLOT.
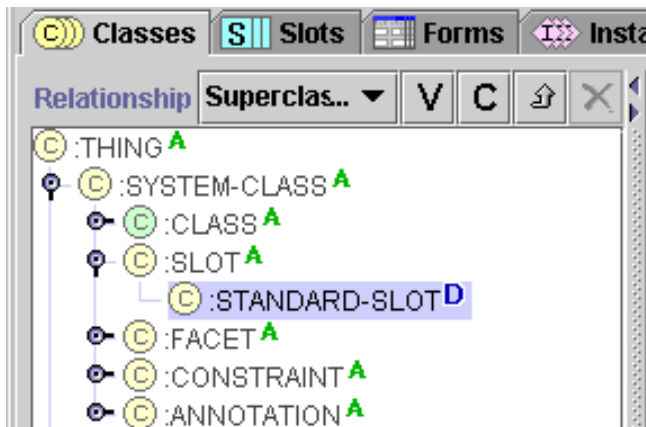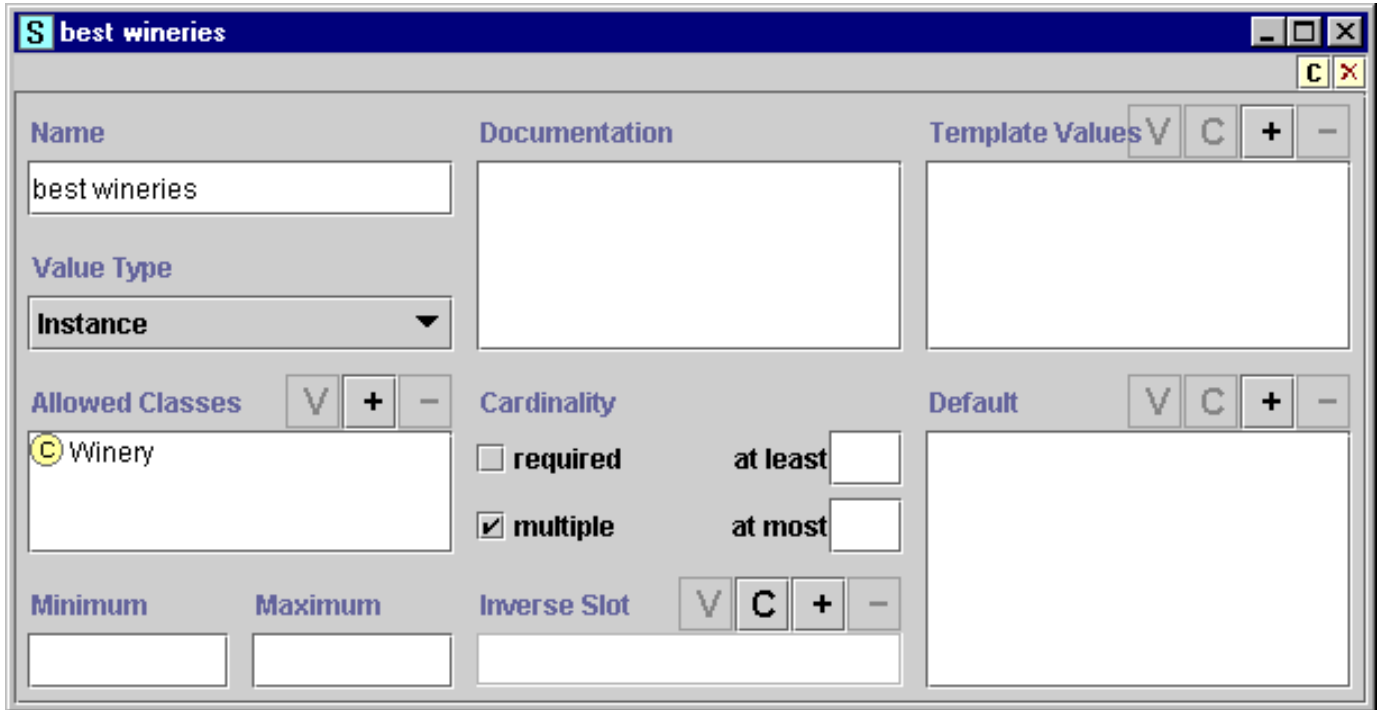
2. Click the **C(reate)** **C** button at the right of the Class Relationship Pane, or click the right mouse button and select **Create subclass** from the Class Menu. The new class will be added under the highlighted class. It will inherit the properties of the selected metaclass.
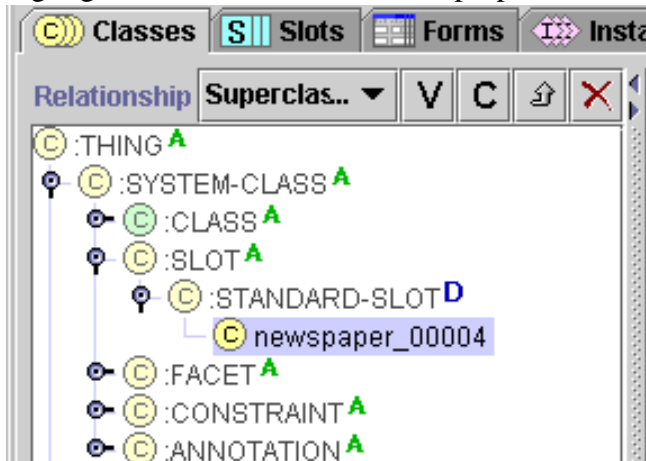
3. Use the Class Form to name the metaslot, create constraints, and create and edit slots.

For example, click the **C(reate)** C template slots button at the right of the Template Slots pane to create a new slot. This slot will show up as an instance widget for any slot you create using the metaslot.

Next:

Setting the Default Metaslot

Note: Before you create and use metaslots (slot metaclasses), you should be confident with the basic Protégé interface and be comfortable designing a project, and creating and modifying classes, slots, forms and instances.

If you have multiple metaslots in your project, Protégé allows you to choose which metaslot you want to use as the default for every new slot you create. The default metaslot is denoted by a Default **D** icon at the right.

To set a metaslot as the default:

1. Click on the metaslot you want to use in the Class Relationship pane. This must be a class that is subordinate to :SLOT.
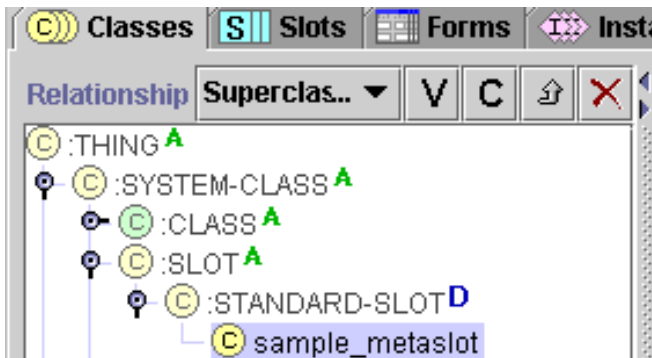
2. Click the right mouse button and select **Set as default slot metaclass** from the Class Menu. This option is available only when a metaslot is selected.
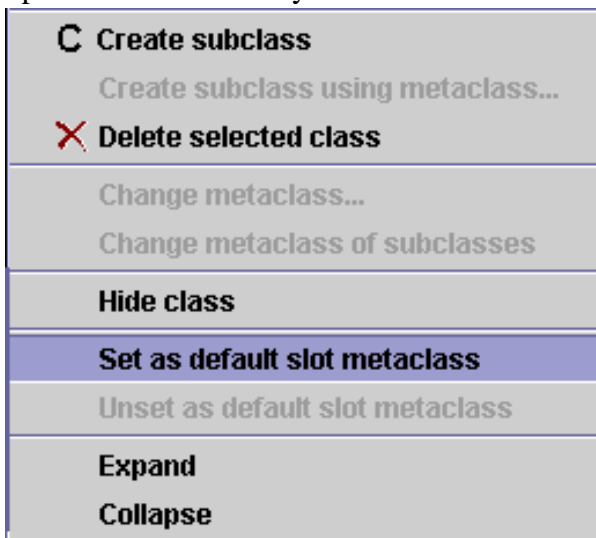
3. The selected metaslot becomes the default, as indicated by the **D** icon. Now, unless you choose otherwise, every new slot you create will use the new default slot as its template and inherit the properties defined by the default.
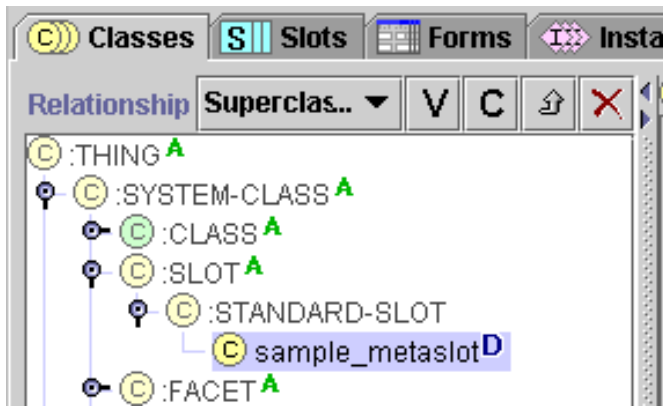
It is also possible to remove the default property from a metaslot without designating a new metaslot. To do this:

1. Select the default metaslot in the Class Relationship pane.
2. Click the right mouse button and select **Unset as default slot metaclass**.

If you do this, no default metaslot will be designated. In this case, if you have only one concrete metaslot, it will be used as the default. If you have more than one concrete metaslot, then you will be prompted to choose a metaslot each time you create a new slot.

Next:

Forms Table of Contents

- **Understanding Forms**

The User Interface:

- **The Forms Tab**
- **The Forms Pane**
- **The Form Buttons**
- **The Form Edit Pane**
    - **The Browser Key Menu**
    - **The Widget Type Menu**
    - **The Widget Configuration Dialog**
    - **The Form Configuration Dialog**
- **Default Widget Types**
- **Additional Widget Types**
    - **InstanceRowWidget**
    - **InstanceTableWidget**
    - **ContainsWidget**
    - **SliderWidget**
    - **ImageMapWidget**
- **Advanced Widget Types**

Forms Tasks:

- **Changing Form Characteristics**
- **Modifying a Widget's Appearance**
- **Selecting a Widget Display**
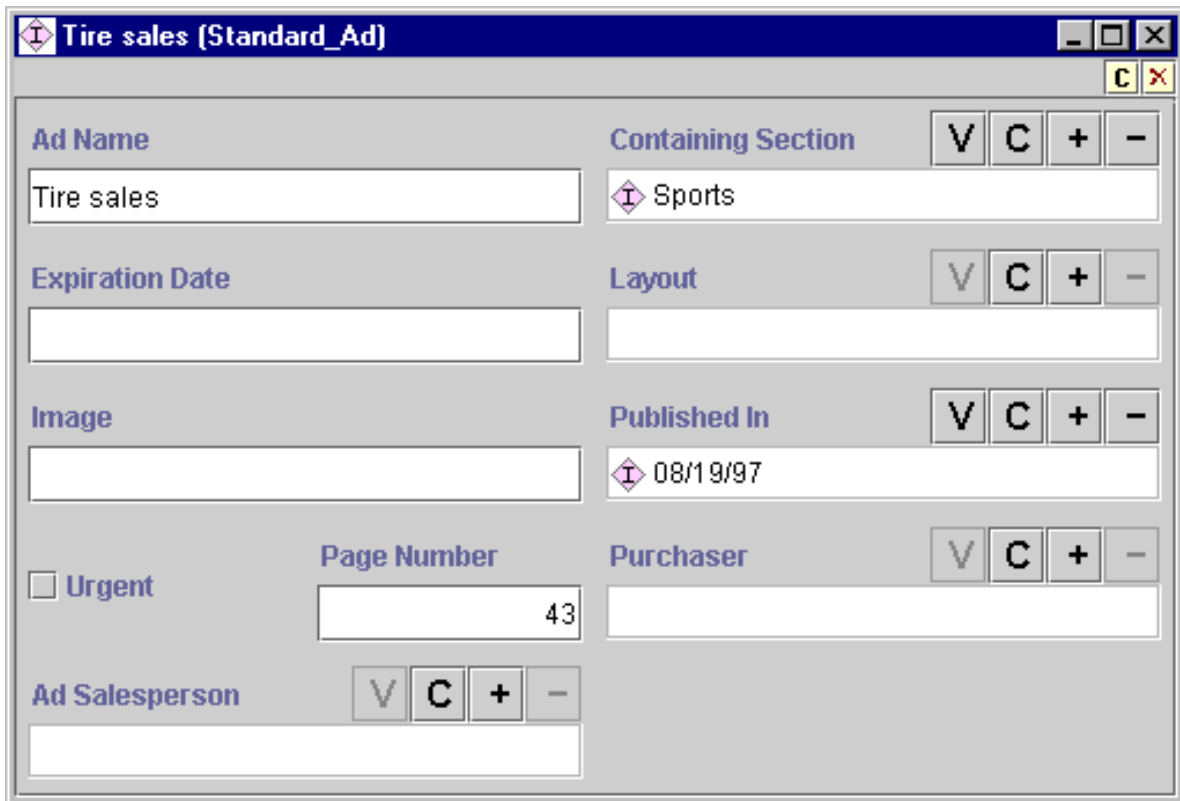
The Instances Form

The Instances form can be used to define and edit the slot attributes of the instance selected in the Direct Instances pane. If a single class is selected, the Instances form is displayed at the right of the Instances tab. The Instances form is also displayed separately in a free-standing window whenever you click on the View V Instance button in the Direct Instances pane. Whenever you enter changes into the Instances form, they take effect immediately. To make the changes permanent, select **Save** from the **Project** menu.
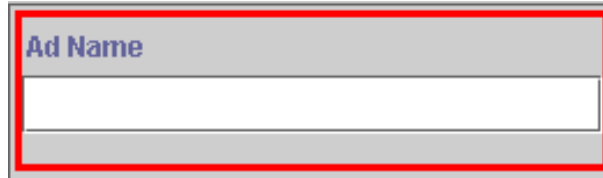
For each slot in the instance, the Instances Form displays a field where you can enter the information for that slot. The display and options for the field depend on the type of information that is included in the field.

If the type of information you enter is not appropriate for the slot, the entry will be shown in red when you leave the slot. For example, entering a decimal number in an integer slot is not allowed. Values that violate other slot conditions, such as values less than the minimum or greater than the maximum, will also be shown in red.

If a value is required for the slot and there is currently no value, the field is outlined in red.

Note Icons

The note icons, **C** ✕, at the upper right of the form allow you to add and remove yellow sticky notes to your class. The note is always displayed along with the form. For information on how to add notes to any frame (class, instance, or slot), see [Working with Notes](#).

Next: [The Field Buttons](#)

The Query Library Pane

The Query Library allows you to retrieve saved queries.

Note: If you are working on a smaller screen, you may not see this pane. To view or enlarge the Library Pane, drag the slider bar near the bottom of the Query Pane (above the Query Library Pane), or click the up button at the bottom of the Query Pane. See Working With a Small Window for more information.
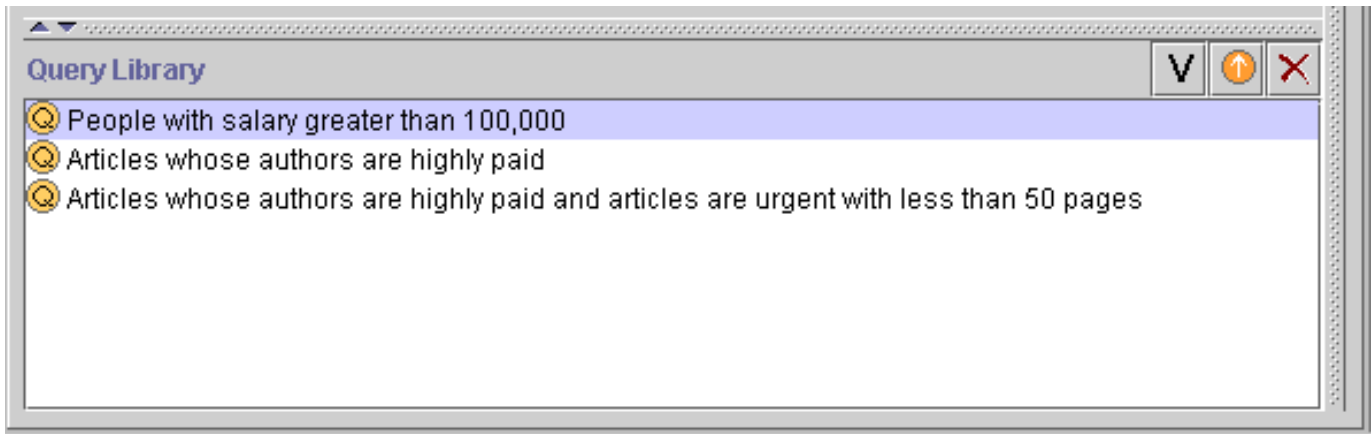
The Query Library Pane has the following components:

1. A list of all saved queries.
2. Buttons that allow you to view, retrieve, or delete a query.

Only the buttons are described further.

The buttons at the right of the pane have the following actions:

View button: Click this button to view the details for a selected query.

Retrieve button: Click this button to retrieve the query, which puts the query in the Query Pane ready for modification or for finding instances.

Delete button: Click this button to delete the highlighted querie(s).

Next: Creating a Simple Query

Queries Table of Contents

The Search Results Pane

The Search Results Pane at the right of the Query Tab displays the query results when you click **Find**.

Note: If you are working on a smaller screen, you may not see this pane. To view or enlarge the Search Results Pane, drag the slider bar at the right side of the Query Pane (to the left of the Search Results Pane). See Working With a Small Window for more information.
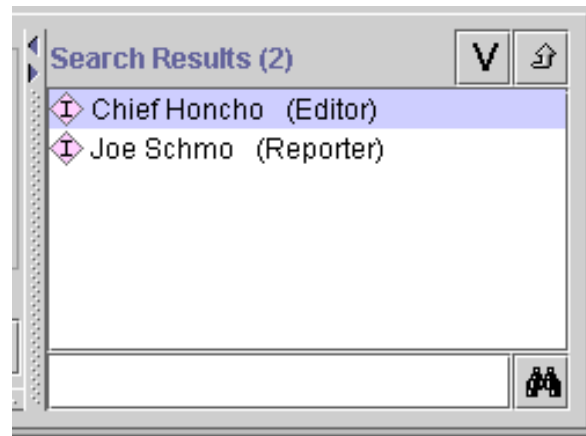
The Search Results Pane has the following components:

1. A list box which shows all instances that match the most recent query.
2. Two buttons which allow you to get more information about a selected instance.
3. A Find bar which allows you to find an instance in a long list.

These components are fairly simple, so only the buttons are described further.

The buttons at the right of the pane have the following actions:

V   **View button:** Click this button to open the Instances Form for a selected instance

          Back References button: Click this button to view all references to the selected instance.

Next: The Query Library Pane

Queries Table of Contents

The Queries Tab

The Queries tab allows you to create, run, and save **queries**, which are a way to select instances from your project base on the values of one or more slots. Queries are not part of your knowledge base, but are a way to identify the instances in your project, based on class and slot properties.

The Queries Tab consists of three panes:

1. The Query Pane, where you enter or modify your query. You can also combine multiple queries.
2. The Search Results Pane, which displays the query results when you click **Find**.
3. The Query Library, which allows you to save and retrieve queries. Saved queries can be modified or chained.

Note: If you are working on a smaller screen, you may not see all of these panes. To view or enlarge the Query Library pane, drag the slider bar near the bottom of the Query Tab. To view or enlarge the Search Results Pane, drag the slider bar at the right side of the window. See Working With a Small Window for more information.

Next:

The Query Pane

The Query Pane allows you to enter, modify, or save your queries.

The Query Pane has the following components:

1. One or more query bars, where you can construct a query by selecting a class, slot, criterion, and value.
2. The Combined Query buttons, which allow you to add and remove additional query bars, and to specify how they are combined.

3. The [Find button](#), which allows you to run your query.
4. The [Query Name bar and Save Query button](#), which allow you to name and save a query.

Query Bar(s)

Each query bar has:

1. A Class entry field, which allows you to specify a class by clicking the Select Class ✚ button. If a class is already shown, using the Select Class ✚ button replaces it. You can also view the selected class by clicking the View ∨ button, or remove it by clicking the Remove ▬ button.

2. A Slot entry field, which allows you to specify a slot by clicking the Select Slot ✚ button. If a slot is already shown, using the Select Slot ✚ button replaces it. You can also view the selected slot by clicking the View ∨ button, or remove it by clicking the Remove ▬ button.

3. A criteria menu, which allows you to set the criteria for the query based on the slot.

4. A value entry bar or menu, which allows you to set the value for comparison. The entry type is based on the slot value type.

Choices for the criteria menu and value are as follows:

Slot Type	Criteria Menu	Action
Boolean	**is**	Verifies whether or not the Boolean value is **true** or **false**, as selected from the menu at the right.
Class	**contains** / does not contain	The selected criterion is compared to a class selected at the right. To select a class, click the Select Class ✚ button. To remove a class, click the Remove ▬ button. Only one class can be selected.
Float	**is** / is greater than / is less than	The selected criterion is compared to the value typed in the Float entry bar at the right.

Instance		The selected criterion is compared to an instance selected at the right. To select an instance, click the Select Instance **+** button. To remove an instance, click the Remove **−** button. Only one instance can be selected.
	contains	
	does not contain	

Instance

> contains
> does not contain

The selected criterion is compared to an instance selected at the right. To select an instance, click the Select Instance **+** button. To remove an instance, click the Remove **−** button. Only one instance can be selected.

Integer

> is
> is greater than
> is less than
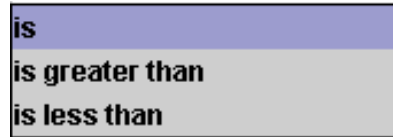
The selected criterion is compared to the value typed in the Integer entry bar at the right.

String

> contains
> does not contain
> is
> is not
> begins with
> ends with
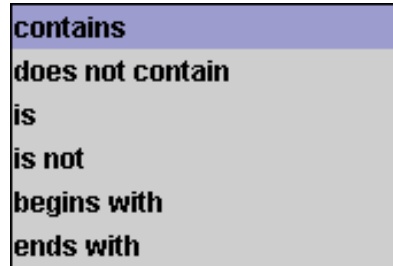
The selected criterion is compared to all or part of a string typed in the entry bar to the right. For example, selecting **begins with** and typing M in the String entry bar will find all instances of the selected class/slot combination that begin with M.

Symbol

> is
> is not

The Symbol entry bar to the right becomes a drop-down menu that displays all possible values for the slot. The selected criterion is compared to the value.

Combined Query Buttons

More	Click this button to create an additional query bar, which can then be set with class, slot, and criterion.
Fewer	If there are two are more query bars, click this button to remove the bottommost query bar.
Clear	Click this button to clear all query bars and reduce the Query Pane to a single, blank query bar.
Match All	For two or more query bars, click this button to specify that any instance found must match all the criteria (the intersection or AND) specified in the query bars.
Match Any	For two or more query bars, click this button to specify that any instance found must match at least one of the criteria (the union or OR) specified in the query bars.

Find Button

| **Find** | When the query has been set up as desired, click this button to find all instances that match the selected criteria. |

Query Name and Save Query

The Query Name bar at the bottom of the pane allows you name a query by typing any string.

 When the query has been set up as desired, click this button to save the query in the [Query Library](#) using the name in the Query Name bar.

Next: [The Search Results Pane](#)

[Queries Table of Contents](#)

The Direct Instances Pane

The Direct Instances pane shows all the direct instances, if any, for the class selected in the [Class Pane at the Instances Tab](), and allows you to view, edit, create, and delete direct instances.

A *direct instance* is an instance that has been created directly under the class. However, because of inheritance, an instance of a class is also an instance of all that class's superclasses. Therefore, an instance can be an instance of many classes; however, it can only be a *direct* instance of a single class.

This pane has three components:

1. The [Instance Buttons]() allow you to create, edit, and delete instances.
2. The [Instances Window]() displays an alphabetical list of the instances for the selected class.
3. The Instance Lookup Bar allows you to locate an instance in the Instances Window by typing the instance name and clicking the binoculars. See [Finding an Instance]() for more information.
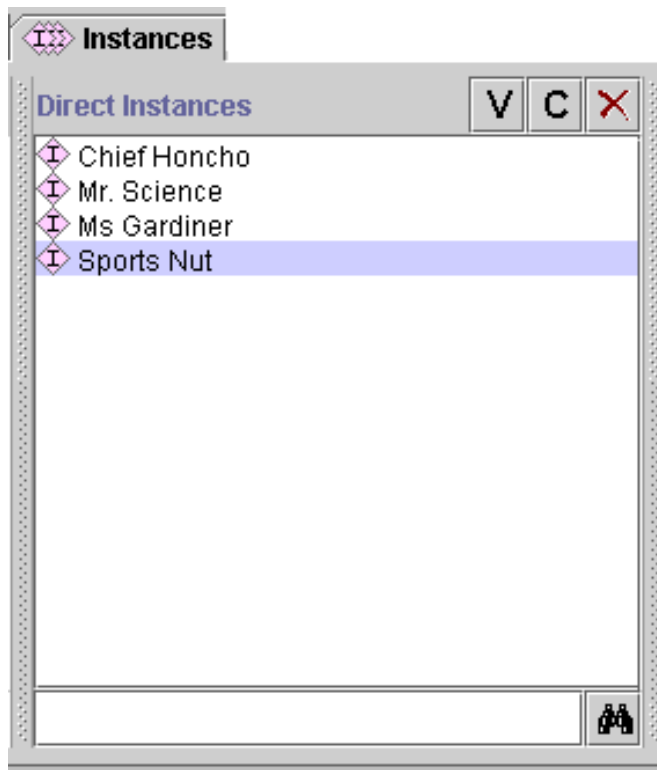
The Instances Window

When a single class is selected in the [Class Pane at the Instances Tab](), the Instances Window shows all the instances belonging to that class. Instances are displayed in alphabetical order by the text in the browser key, which is the slot that has been designated as the identifying slot for the class. If no browser

key has been selected, Protégé-2000 uses a default key, <instance name>, which is displayed as a number, such as newspaper_0017. See The Browser Key Menu for more information.

When a single instance is selected, the information for that instance is shown in the Instances Form to the right.

The color of the icon to the left of the instance name gives information about the instance:

The instance can be edited. You can enter your edits directly to the right, or click the View V button to open the Instances Form.

The instance cannot be edited. Instances cannot be edited if they are included from another project. See Including a Project for more information.

Next: The Instance Buttons

Instances Table of Contents

The Field Buttons

Some fields have field buttons, **V C + –** , located at the top right of the field. These buttons allow you to view and edit, create, add, or delete a value for the field. The buttons available depend on the type of field.

Integer Multiple	V	C	–
12			
144			

Section	V	C	+	–
Lifestyle				

The following field types have buttons:

- Multiple fields of any type, that is, fields that allow more than one value.
- Single Class fields and Instance fields.

The buttons have the following actions:

	Button Name	Fields Where it Appears	Action

V	**V(iew) button**	All multiple fields Class fields and Instance fields	Click this button to open a dialog box or form that allows you to edit the highlighted value. For [Instance fields](#), any edits you make will take effect everywhere the instance occurs. For single [Class](#) and [Instance fields](#), the value does not need to be highlighted.
C	**C(reate) button**	Multiple Float, Multiple Integer, and Multiple String fields All Instance fields	For Float, Integer, and String fields, click this button to open a form where you can enter a new value. For [Instance fields](#), click this button to create a new instance. See [Creating an Instance From a Field](#).
+	**Add button**	All multiple fields Class and Instance fields	Click this button to add a value. For multiple fields, this adds an additional value to the list; note that you cannot add the same value twice. For single [Class fields](#) and [Instance fields](#), this lets you choose from a list of pre-existing values.
–	**Remove button**	All multiple fields Class and Instance fields	Click this button to remove a value. For multiple fields, this removes the highlighted value from the field. For single [Class fields](#) and [Instance fields](#), this clears the field, but does *not* remove the selected class or instance from the knowledge base.

For multiple fields, the View and Delete buttons are grayed out when none of the values are highlighted.

Next: [The Standard Fields](#)

Instances Table of Contents

The User Interface:

Basic Tasks:

Finding an Instance

To find an instance in the Direct Instances pane:

1. Select the class that contains the instance in the Class Pane at the Instances Tab.
2. Type all or part of the browser text for the instance in the Instance Lookup Bar at the bottom of the Direct Instances pane.

3. Press Enter/Return or click the Find ⚲ button.

If there is only one match, the selection in the the Direct Instances pane will move to the instance which contains the entered text. If there is more than one match, a dialog will be displayed asking you to choose the instance you want.

Comparison is case-insensitive and position-independent. In the following list, *si* would find *Mir Doomed by Collision* and *Sunburned Swans Sing Sad Song.*
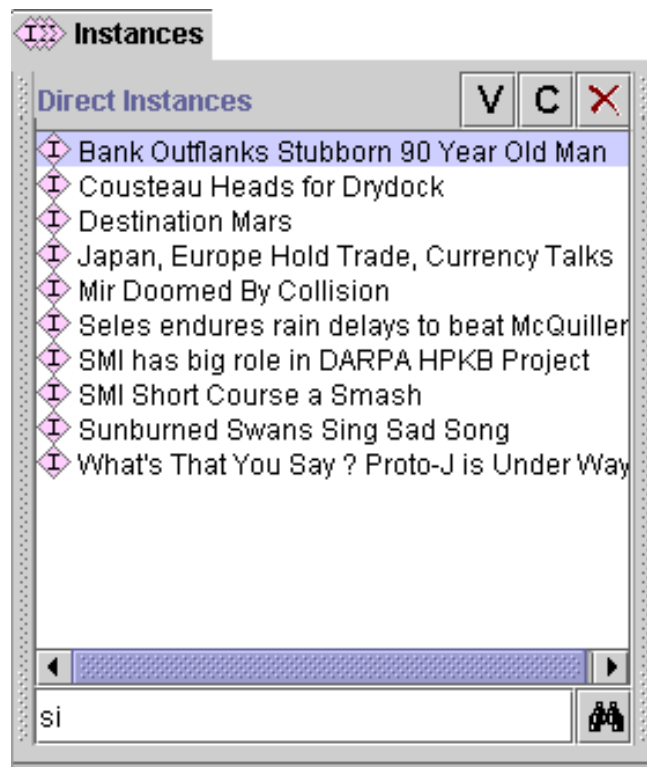
If you are unable to find the instance you are looking for, it may be in a different class. You can use the lookup bar at the bottom of the Class Pane at the Instances Tab to look for classes.

Next:

![Troll mascot icon] # The Browser Key Menu

The Browser Key menu allows you to choose the browser key, that is the slot that is used to identify the different instances when they are displayed in a list. For example, the form for **Prototype_Newspaper** has the browser key set to **weekday**. This means that, in the [Direct Instances Pane](), the different instances are listed by the value in the **weekday** field.
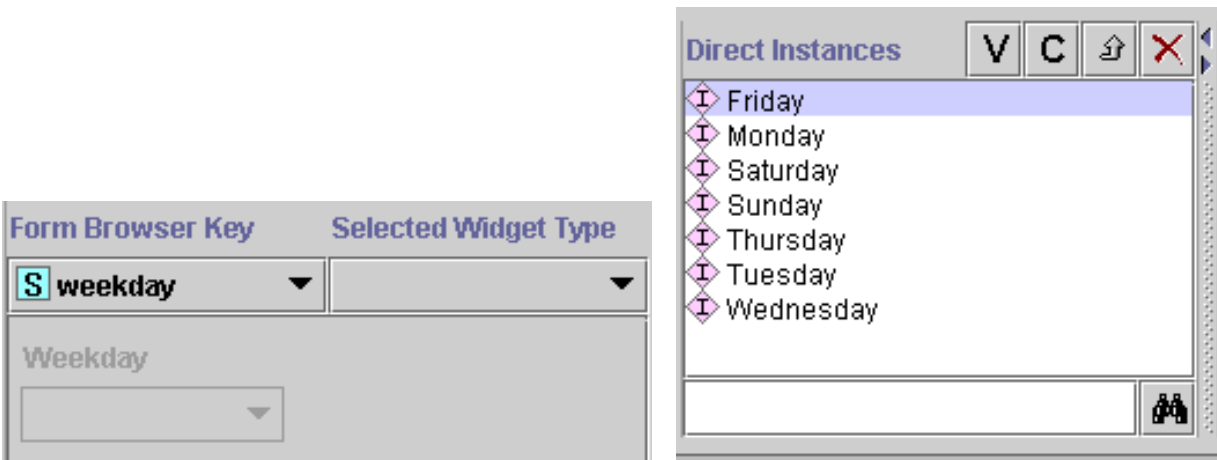
If you do not select a slot to use as a browser key, Protégé-2000 uses a default key, **<instance name>**, such as newspaper_0017. It is usually helpful to set a browser key.

Browser keys are inherited from the parent class.

Next: [The Widget Type Menu](#)

[Forms Table of Contents](#)

The Class Fields

A class slot contains values which are classes. The class fields allow you to enter these as values for a slot.

Class/Single Field

An instance with a **Class/Single** slot shows a display pane with three buttons that let you edit, add, and remove classes. The value of this slot is a single class:

To **edit** a class:

1. Click the View V button to open the [Class Form](#) for the class that is displayed or selected in the entry bar.

If there is no class, the View button is grayed out. For information on editing a class, see [Viewing a Class](#).

To **select** a class as the value of the slot, or to change the existing value:

1. Click the Add **+** button. A Select Classes dialog box is displayed.
2. Select a single class as the value of the slot.
3. Click OK.
4. The class you selected is shown in the display bar. If another class was displayed previously, it has been removed.

To **remove** a class:

1. Click the Remove **−** button. The display bar is cleared.

Class/Multiple Field

An instance with a **Class/Multiple** slot shows a display pane with three buttons that let you edit, add, and remove classes. The value of this slot is a list of classes:
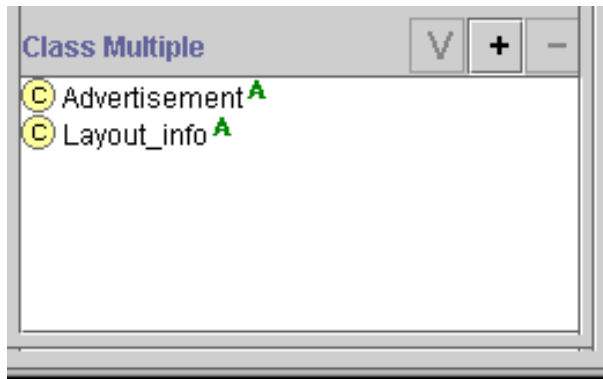
To **edit** a class:

1. Click the View ∨ button to open the [Class Form](#) for the class that is displayed or selected in the entry bar.

If there is no class, the View button is grayed out. For information on editing a class, see [Viewing a Class](#).

To **add** a class:

1. Click the Add **+** button. A Select Classes dialog box is displayed.
2. Select the class(es) you want as the value of the slot. You can select multiple classes by holding down the Ctrl key as you click.
3. Click OK.
4. The class(es) you selected is shown in the display bar.

To **remove** a class:

1. Highlight the class(es) you want to remove.
2. Click the Remove **–** button. The selected classes are removed.

Next: [The Float Fields](#)

[Instances Table of Contents](#)

The Instance Fields

Instance/Single Field

An instance with a **Instance/Single** slot shows an entry field and four buttons that let you edit, create, add, and remove instances. The value of this slot is a single instance:

To **edit** an instance:

1. Click the View **V** button to open the [Instances Form](#) for the instance that is displayed or selected.
2. Enter the updated information directly in the [Instances Form](#).

Any changes you enter into the [Instances Form](#) take effect immediately. Note that the instance value is a reference to the selected instance. This means that if you edit an instance in *any* location, the changes will appear in all the locations where it is referenced.

If no instance is currently displayed, the View button is grayed out.

To **create** a new instance:

1. Click the Create **C** button. The [Instances Form](#) is displayed.
2. Fill in the form for the new instance.

This creates a new instance as a direct instance of the class that you select. See [Creating an Instance From a Field](#) for more information.

To **add** an existing instance to the value of the slot:

1. Click the Add **+** button. A Select Instances dialog box is displayed.
2. In the Allowed Classes pane, select the class where the instance you want is located.
3. Select a single instance in the [Direct Instances pane](#).
4. Click OK. The instance you selected is shown in the display bar. If another instance was displayed previously, it has been removed.

This selects a pre-existing instance as the value of the field. This instance can also be viewed in the [Direct Instances pane](#) and may also appear as a value for other instances. Note that *any edits you make to this instance will appear in all locations where this instance occurs.* You should be sure that you want the change to be global before editing an existing instance. In some cases, it may be more appropriate to create a new instance instead.

To **remove** an instance:

1. Click the Remove **–** button. The selected instance is removed.

This removes the instance as a value of the current field, but does not remove it from the project. The instance can still be viewed via the [Direct Instances pane](#) for the correct class, and still appears in any other field where it has been selected. To delete an instance from the knowledge base, use the Delete button in the [Direct Instances pane](#). See [Deleting an Instance](#) for more information.

Instance/Multiple Field

An instance with a **Instance/Multiple** slot also shows a list with four buttons that let you edit, create, add, and remove instances. The value of this slot is a list of instances:

To **edit** an instance:

1. Click the View **V** button to open the [Instances Form](#) for the instance that is displayed or selected.
2. Enter the updated information directly in the [Instances Form](#).

Any changes you enter into the [Instances Form](#) take effect immediately. Note that the instance value is a reference to the selected instance. This means that if you edit an instance in *any* location, the changes will appear in all the locations where it is referenced.

To **create** an instance:

1. Click the Create **C** button. The [Instances Form](#) is displayed.

2. Fill in the form for the new instance.

See [Creating an Instance From a Field](#) for more information.

To **select** an instance as the value of the slot:

1. Click the Add ✚ button. A Select Instances dialog box is displayed.
2. In the Allowed Classes pane, select the class where the instance(s) you want are located.
3. Select the instance(s) you want to add. You can select multiple instances by holding down the Ctrl key as you click.
4. Click OK. The instance(s) you selected are shown in the display bar.

To **remove** an instance:

1. Highlight the instance(s) you want to remove.

2. Click the Remove ▬ button. The selected values are removed.

Next: [The Integer Fields](#)

[Instances Table of Contents](#)

Creating an Instance From a Field

You can create an instance from any Instance field in the Instances Form. Creating an instance from an Instance field creates the instance in one of the Allowed Classes for the slot associated with the field.

To create a new instance from an Instance field:

1. In the Class Pane at the Instances Tab, highlight a class which has a slot of type **Instance**. For example, to create an instance for the class **Section**, you could select the **Article** class, which has a **Section** slot.
2. In the Direct Instances Pane, select an instance.
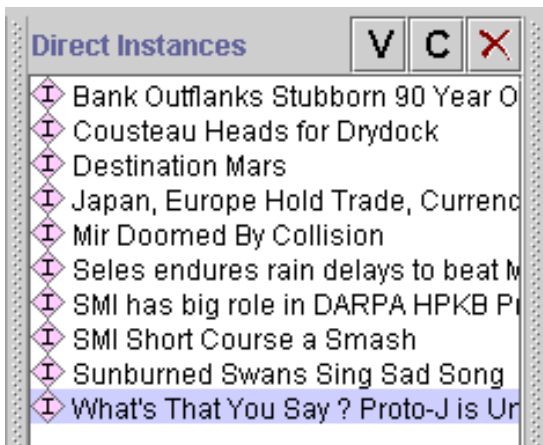
3. In the Instances Form, go to the field for the Instance slot. Here is the field for the slot **Section**:

4. Click the Create **C** button at the top right of the field. If there is more than one Allowed Class for the slot, a **Select Concrete CIs** dialog box is displayed. Select the class you want and click OK. If there is only one Allowed Class, you can skip this step.
5. An Instances Form will appear for the new instance. It will have a default name, such as project_INSTANCE_00001.
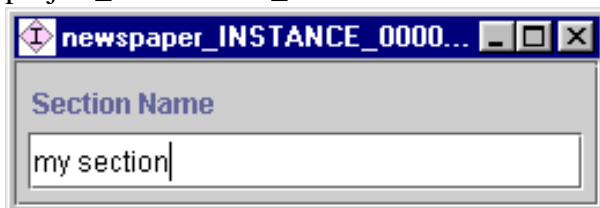
6. Use the Instances Form to fill in the slots for the instance.
7. The new instance is added as a slot value for the instance you selected in step 2.
8. To view the instance, you can select the class it appears in in the Class Pane at the Instances Tab. The new instance then appears in the Direct Instances Pane for that class.

Note: The slot value is a reference to the selected instance. This means that if you edit the instance in any location, the changes will appear in all the locations where it is referenced.

You can also create an instance by clicking the Create ⟳ instance button at the top of the Direct Instances Pane. See Creating an Instance Directly for more information.

Next: <u>Viewing an Instance</u>

The Boolean Fields

Boolean/Single Field

An instance with a **Boolean/Single** slot displays the slot as a checkbox. To set or change the value for a Boolean/single slot, click in the box to add or remove the checkmark.

For example, the **Personals_Ad** class contains the Boolean slot **Urgent**; the **Silly** instance of **Personals_Ad** is not Urgent, while the **M137** instance is:

☐ Urgent	not **Urgent**
☑ Urgent	**Urgent**

Next:

The Float Fields

A float slot contains values which are floating point numbers. Float numbers may include a decimal point. Values of type **Float** are stored on your system as floating point values, and are only as accurate as your system. When entering a **Float** value for an instance, you can use decimal point or exponential representation. You can enter positive and negative values. For example:

Representation	Description
1234.56	Standard decimal notation.
1.23456E3	Exponential notation, shorthand for $1.23456 \cdot 10^3$. Represents 1234.56
-1234.56	Negative number.
1.23456E-3	Exponential notation with a negative exponent, shorthand for $1.23456 \cdot 10^{-3}$. Represents 0.00123456.

For convenience in typing, you can enter a lower case **e** instead of an uppercase **E**.

Float/Single Field

An instance with a **Float/Single** slot shows an entry bar:

To edit the value, simply click in the entry field and make your edits.

To set the value of a **Float/Single** slot, simply enter the number you want in the entry bar.

To clear a **Float/Single** slot, simply delete the value in the entry bar.

Note: The information in a **Float** field must be a floating point number. If you enter text or other invalid information, the entered text will be displayed in red and will not be saved by the system. The previous value (if any) will be used instead.

Invalid entry in a Float field

Float/Multiple Field

An instance with a **Float/Multiple** slot shows a display list with three buttons:
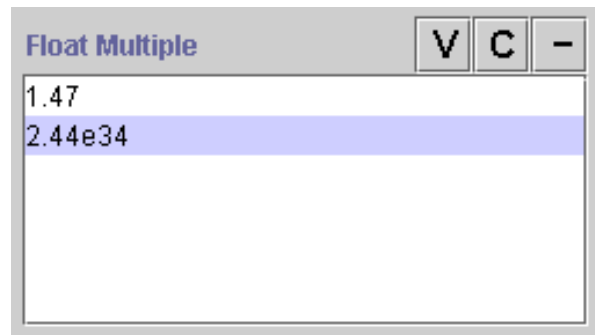
To **edit** a pre-existing value for a **Float/Multiple** slot:

1. Select the value you want to edit.
2. Click the View **V** button. An Edit Float Value dialog box is displayed.
3. Edit the number directly in the dialog box.
4. Click OK.

To **add** a value to a **Float/Multiple** slot:

1. Click the Create **C** button. A Create Float Value dialog box is displayed.
2. Enter the number you want in the entry bar. The number may include a decimal point.
3. Click OK.

To remove a value from a **Float/Multiple** slot:

1. Select the value(s) you want to remove. You can select multiple values by holding down the Ctrl key.
2. Click the Remove **–** button. The selected values are removed.

Next:

The Integer Fields

Integer/Single Field

An instance with a **Integer/Single** slot shows an entry field:

Page Number

14

To **edit** the value for an **Integer/Single** slot, simply click in the entry field and make your edits.

To **create** a value for an **Integer/Single** slot, simply enter a whole number in the entry field.

To **clear** an **Integer/Single** slot, simply delete the value in the entry field.

Note: The information in a **Integer** field must be a whole number. If you enter text or other invalid information, the entered value will be displayed in red and will not be saved by the system. The previous value (if any) will be used instead.

Number Of Pages

12.5

Invalid entry in an Integer field

Integer/Multiple Field

An instance with a **Integer/Multiple** slot shows a display list with three buttons:

Integer Multiple   V C –

12
144

To **edit** a pre-existing value for an **Integer/Multiple** slot:

1. Select the value you want to edit.
2. Click the View **V** button. An Edit Integer Value dialog box is displayed.
3. Edit the number directly in the dialog box.
4. Click OK.

To **add** a value to an **Integer/Multiple** slot:

1. Click the Create **C** button. A Create Integer Value dialog box is displayed.
2. Enter a whole number in the entry bar. You can enter positive or negative values.
3. Click OK.

To **remove** a value from an **Integer/Multiple** slot:

1. Select the value(s) you want to remove. You can select multiple values by holding down the Ctrl key.
2. Click the Remove **−** button. The selected values are removed.

Next: [The String Fields](#)

The String Fields

String/Single Field

An instance with a **String/Single** slot shows an entry field:

Headline

Bank Outflanks Stubborn 90 Year Old Man

To set the value of the slot, simply type in the slot. You can enter ASCII characters for a string, including upper and lowercase letters, numbers, and the basic symbols on the keyboard, such as !, _, and %. **String** values can also include spaces.

You can edit the string directly in the slot. To clear the slot, simply delete the current text.

String/Multiple Field

An instance with a **String/Multiple** slot shows a display pane with three buttons:

Keywords   V C –

Marine Biology
Jacques Cousteau

To **edit** a pre-existing value for a **String/Multiple** slot:

1. Select the value you want to edit.
2. Click the View V button. An Edit String Value dialog box is displayed.
3. Edit the string directly in the entry field in the dialog box.
4. Click OK.

To **create** a value for a **String/Multiple** slot:

1. Click the Create C button. A Create String Value dialog box is displayed.
2. Enter a string in the entry field. You can enter ASCII characters, including upper and lowercase

letters, numbers, and the basic symbols on the keyboard, such as !, _, and %. **String** values can also include spaces.

3. Click OK.

To remove a value from an **String/Multiple** slot:

1. Select the value(s) you want to remove. You can select multiple values by holding down the Ctrl key.

2. Click the Remove ▬ button. The selected values are removed.

Next:

The Symbol Fields

Symbol/Single Field

An instance with a **Symbol/Single** slot shows a drop-down list:

To set the value of the slot, click on the list and double-click to select your choice.

To change the value of the slot, make a different choice from the list. To clear the slot, select the blank value at the top of the list.

Symbol/Multiple Field

An instance with a **Symbol/Multiple** slot displays multiple values.

Creating a Instance Directly

Creating instances is part of the iterative process of developing a Protégé-2000 project. Before you create instances, you need to have created and organized the classes and slots that model your knowledge base structure.

Creating instances can give you valuable information about the project structure and its applicability. As you add instances to your project, you may expose missing or redundant areas which tell you that you need to redesign some of the class/slot structure. However, as you move classes and create and delete slots, you may lose information in your instances. In addition, it is difficult to split a single instance into two or more instances. Therefore, you should not add an extensive base of instances until you believe the structure of your project is fairly stable.

There are two ways to create an instance:

- You can create an instance directly, from the Direct Instances pane. This is described below.
- You can create an instance from an Instance Field for another instance. See Creating an Instance From a Field.

Creating an instance from the Direct Instances pane creates the instance directly in the selected class. To create a new instance from the Direct Instances pane:

1. In the Class Pane at the Instances Tab, highlight the class where you want the instance to appear.
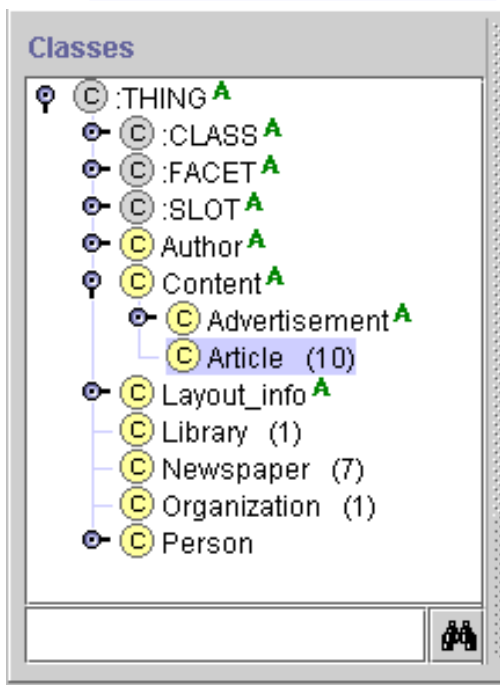
2. Click the **C(reate)** button, which appears as a **C** in the instance buttons at the right of the Direct Instances pane. The new instance will appear in the Direct Instances pane. It will have a default

name, such as project_INSTANCE_00001.
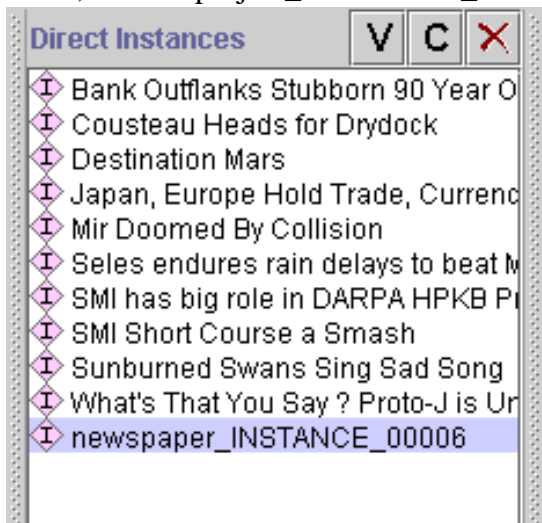
3. Use the Instances Form at the right to fill in the slots for the instance. Any required fields are outlined in red.

Next:

Viewing and Editing an Instance

To view and edit an existing instance:

1. In the Class Relationship pane at the Instances Tab, select the class which contains the instance you want to edit. If the class is not currently displayed, you may need to navigate to it via the class hierarchy. The instances for the selected class are displayed in the Direct Instances pane.
2. Select the instance you want to edit in the Direct Instances pane. The current information for the highlighted instance will be shown in the Instances Form to the right.
3. Enter the updated information directly in the Instances Form to the right *or* click the View **V** button in the Direct Instances pane to show the same form as a free-standing window.

Any changes you enter into the Instances Form take effect immediately. To make the changes permanent, save the knowledge base by selecting **Save** from the **Project** menu. To revert to the last saved version, close Protégé-2000 without saving changes. If you have made extensive changes to your knowledge base during the current session, you may wish to save before editing instances.

You can also edit an instance from a field. See Instance Fields for more information.

Next: Deleting an Instance

Instances Table of Contents

Deleting an Instance

To delete an instance from the knowledge base:

1. In the Class Relationship pane at the Instances Tab, select the class which contains the instance you want to delete. If the class is not currently displayed, you may need to navigate to it via the class hierarchy. The instances for the selected class are displayed in the Direct Instances pane.
2. Select the instance you want to delete in the Direct Instances pane. To highlight multiple instances for the same class, hold down the Ctrl key while clicking each class. To highlight a range of instances, click the first instance, then hold down the Shift key and click the last instance in the range.
3. Click the ✗ icon from the instance buttons. You will be prompted for confirmation.

Once an instance has been deleted it cannot be recovered. However, if you close Protégé-2000 without saving changes, you will revert to the last saved version. If you have made extensive changes to your project during the current session, you may wish to save before deleting instances. To do this, select **Save** from the **Project** menu.

You can also remove an instance value from a slot. This does not remove the instance from the knowledge base. See Instance Fields in the Standard Fields for more information.

Next: Finding an Instance

Instances Table of Contents

Changing the Class of an Instance

You can change the class of an instance using drag-and-drop in the Instances Tab. To change the class of an instance:

1. Select the class that contains the instance in the Class Pane at the Instances Tab.
2. Select the instance you want to move in the Direct Instances pane.
3. Hold down the mouse button and drag the instance from the Direct Instances pane to the Class Pane at the Instances Tab until it is on top of the desired class.
4. Release the mouse. The instance will now have the selected class as its direct type. Note that the slots of the dragged instance will automatically change to reflect the inheritance from the new class.

Note: When you drag an instance to a new class, the fields in the instance will change to match the slots in the class. If you drag an instance, you may lose some of the information in your instance. This loss is permanent. You *cannot* restore the information by dragging the instance back to the original class. In addition, any changes you make will be evident wherever that instance is referenced -- for example, if that instance appears as the value of a field, the change will also occur in the field. Therefore, you should be sure you want to make these changes before you drag an instance to a new class. You can use the backreferences ⬆ button to see where this instance is referenced before changing its class. You may also wish to save a backup of the project before you move instances. See Renaming a Project for more information.

Queries Table of Contents

The User Interface

- **The Queries Tab**
- **The Query Pane**
- **The Search Results Pane**
- **The Query Library Pane**

Query Tasks

- **Creating a Query**
- **Creating a Complex Query**
- **Running a Query**
- **Saving a Query**
- **Retrieving a Query**
- **Clearing a Query**

Next: Extending Protégé-2000

Creating a Simple Query

The Queries Tab allows you to query your project and locate all instances that match the criteria you specify. You can create a simple query, or combine multiple criteria to restrict or expand your results.

You can create a simple query in one of three ways:

- You can specify a class, select one of the slots attached to the class, and then specify criteria based on the slot type. Running the query will find the instances that match your criteria.
- You can create a query based solely on a slot, without selecting a class. Running the query will find the instances that match your criteria.
- You can create a query based solely on a class, without selecting a slot or any criteria. Running the query will find all instances of the selected class *and* all of its subclasses.

To create a simple query:

1. If you know the class you wish to specify, click the Select Class **+** button above the Class entry bar. (If you wish to create a query based solely on a slot, start at Step 3.)
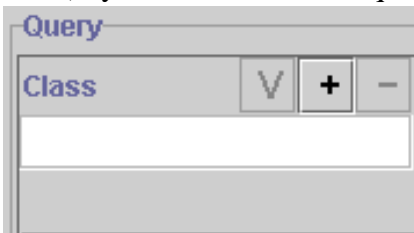
2. Select the class you want from the Select Classes dialog box, then click OK. (If you wish to create a query solely on that class, you are now ready to run it.)
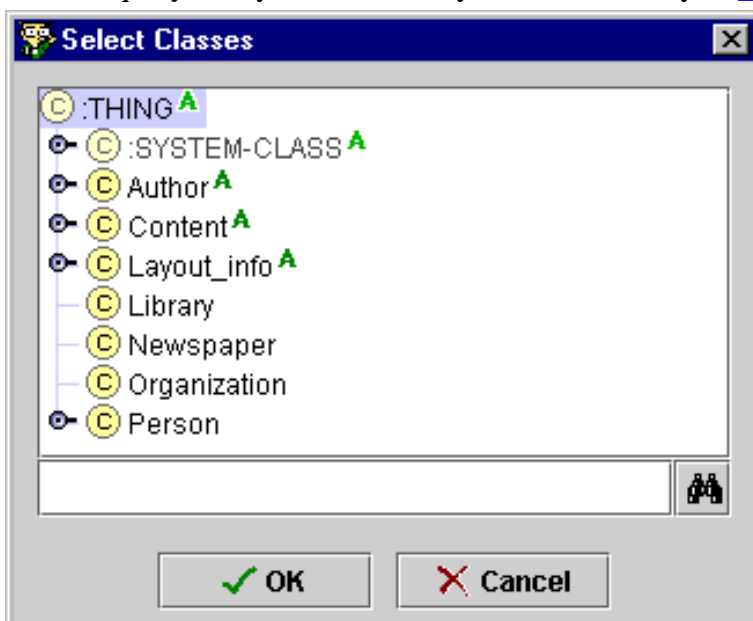
   The class is now displayed in the Class entry bar.

3. Click the Select Slot **+** button above the Slot entry bar.

Slot	V	+	−

4. Select a slot from the Select Slot dialog box. If you selected a class, only slots attached to that class are shown. If no class is selected, all slots are available. Click OK

Select Slot

- S ad_name
- S ad_salesperson
- S advertisements
- S article_author
- S Article_Type
- S articles
- S billing_charts

[✓ OK] [✗ Cancel]
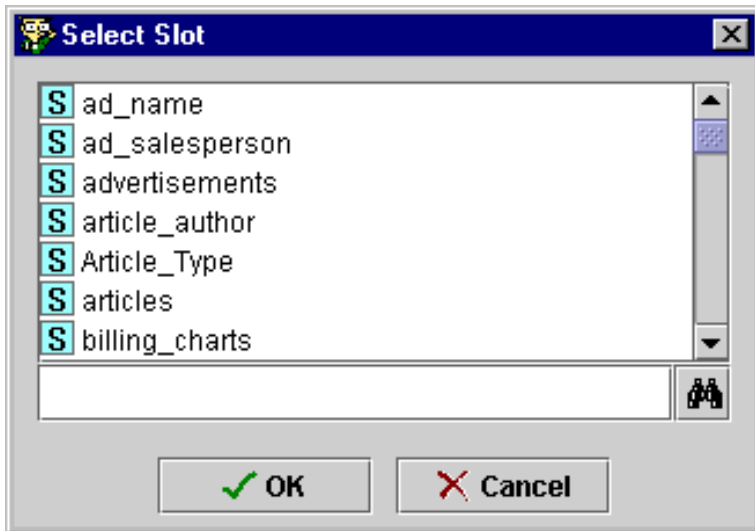
5. Make a selection from the pull-down criteria menu. This menu becomes active after you select a slot or a class; the choices are based on the slot's Value Type.
6. Enter a value for comparison at the entry bar to the right. The entry method for this value depends on the slot value type.

Note: As mentioned above, you can create a query based solely on a slot; to do this, skip Steps 1 and 2. You can also select the slot first and then the class; this is useful if you remember the slot name, but are not certain of the classes that it is attached to. To do this, select the slot as in Steps 3 and 4, then select as class as in Steps 1 and 2. In this case, the Select Classes dialog box will show only those classes that have the selected slot.

***

Next:

Running a Query

The Find button at the bottom of the Query Pane allows you to run your queries and find all instances that match the criteria currently in the Query Pane.

To run a query:

1. Make sure that the query is filled out as desired. See Creating a Query or Creating a Complex Query.
2. Click **Find**. The matching instances are displayed in the Search Results Pane.

Next: Saving a Query

Queries Table of Contents

Creating a Complex Query

The More button at the bottom of the Query Pane allows you to combine multiple query bars into a complex query. To create a complex query:

1. Fill in the first query bar as described in **Creating a Simple Query**.
2. Click **More**. An additional query bar is displayed.
3. Fill in the new query bar. You may use the same class or slot, but do not have to.
4. Continue to click the More button to create query bars as necessary to add more criteria to your query.
5. To specify that all your criterion should be matched by each instance found (Boolean AND), click **Match All**. To find all instances that match at least one query bar (Boolean OR), click **Match Any**.
6. To run your query, click **Find**.

If you are creating a complex query, and wish to remove the criteria in the last (bottommost) query bar:

1. Click **Fewer**. The bottommost query bar is removed.

Next: **Running a Query**

Saving a Query

Once you have a set up a query the way you like, as described in **Creating a Query** or **Creating a Complex Query**, you can save it in the **Query Library** to retrieve it later. This is especially useful for complex queries or for queries that you use often.

To save a query:

1. Make sure that the query is filled out as desired.
2. Enter a name for the query in the **Query Name** entry bar. You can use any name. It is a good idea to use a name that describes the query, such as Articles that are urgent.
3. Click the Save Query 🔽 button. The query is saved in the Query Library.

See **Retrieving a Query** for information on running a saved query.

Next: **Retrieving a Query**

Queries Table of Contents

Retrieving a Query

Once you have saved a query in the **Query Library**, you can retrieve it at any time.

To retrieve a query:

1. Select the query you want from the list in the **Query Library**. (If you are working on a smaller screen, you may not see this pane. To view or enlarge the Library Pane, drag the slider bar near the bottom of the Query Pane or click the up arrow button at the bottom of the Query Pane.)
2. Click the Retrieve Query ⬆ button. Any query information currently in the Query Pane is cleared and the highlighted query is loaded into the query pane.
3. If you wish to make any changes, or combine this query with additional query information, you can modify it. See **Creating a Query** and **Creating a Complex Query** for more information.
4. Click **Find** to run the query and see all matching instances.

Next: **Clearing a Query**

Queries Table of Contents

Clearing a Query

You can clear all the information in the Query Pane. This allows you to start or load a new query.

To clear a query:

1. Click the Clear button.
   All query information in the Query Pane is cleared. If there are multiple query bars, they are removed, so that only one query bar is shown.

Next: Glossary

Extending Protégé-2000

Protégé-2000 includes an application programmer interface, the Protégé API, that allows a Java programmer to extend the Protégé system. The API provides the Java packages and classes for complex operations such as creating new widgets. The documentation for this package is at http://protege.stanford.edu/doc/pdk/index.html

---

Next: Glossary

Understanding Forms

A Protégé-2000 project contains classes and instances. The Classes Tab is used by domain experts and knowledge engineers to create ontologies -- that is, to design a set of classes that describe the problem domain. The Instances Tab is used to acquire instances and is typically used by people who are neither domain experts nor knowledge engineers. For example, a team of medical specialists could carefully craft an ontology dealing with blood diseases. Then, nurses at a hospital could enter instances to describe each patient.

How do end users enter the instances? They fill out forms. Protégé-2000 generates a default form for every class. Protégé will attempt to create an initial, useful set of forms based on its knowledge of the ontology. These default user interfaces are not always user-friendly, however.

Forms Management solves this problem. Using the Forms Tab a developer can customize the way the forms look and feel. The forms you create for your classes at the Forms Tab are then used to enter instances at the Instances Tab. The Forms Tab also allows you to create several user interfaces for the same ontology -- forms for knowledge acquisition can be customized for groups of users.
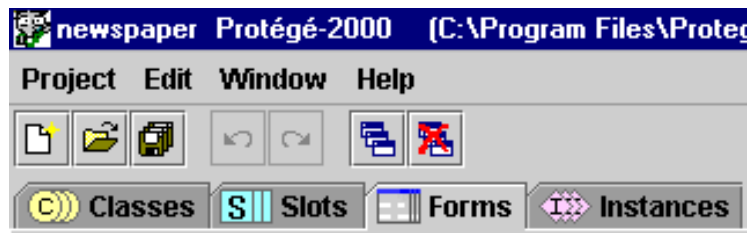
Each form contains a number of user-interface widgets, which correspond to slots in the class. The currently selected widget, if any, is outlined in blue. Widgets are translated to entry fields in the Instances Tab, and control how users will enter information as instances. Each slot in the class is associated with a user-interface widget on the form.

Next: The Forms Tab

Forms Table of Contents

The Forms Tab

The Forms tab provides a single window in which you may view and edit prototype **forms**. The forms you design and create at the Forms tab can be viewed in their final format at the Instances tab. End users will use the finished forms to enter instances into the knowledge base.

The Forms tab window consists of two panes:

1. The Forms pane at the left shows the hierarchy of all the classes and allows you to clear all customizations from your form as well as to create a form with one of the following preset layouts: the default layout, the layout of a form that has been created for a parent, or the layout of a form for another class.
2. The Form Edit pane shows the layout of the form associated to the selected class. Each slot in the class is associated with a user-interface widget on the form. If you have not created or modified the form, Protégé-2000 uses a default layout based on the slot type and cardinality.

As mentioned above, the forms you design will appear at the Instances tab. The following picture shows the same form as it appears under Instances:

Next:

The Forms Pane

The Forms pane at the left of the [Forms Tab](#) displays all the classes in your project. The Forms Pane has three components:

1. The [Forms Window](#) displays the hierarchy of the classes in your project. Icons give more information about the form associated with the class. You cannot change the hierarchy in this window. To edit the hierarchy, use the [Classes Tab](#).
2. The [Form Buttons](#) at the top right of the pane allow you to clear all customizations from your form as well as to create a form with a preset layout.
3. The Forms Lookup Bar allows you to locate a form in the Forms Window by typing all or part of the name and clicking the binoculars.

![Forms pane showing a hierarchy: :THING containing :SYSTEM-CLASS, Diagram_Entity, TableEntity, Author, Content, Layout_info, Library, Newspaper, Person, Section]
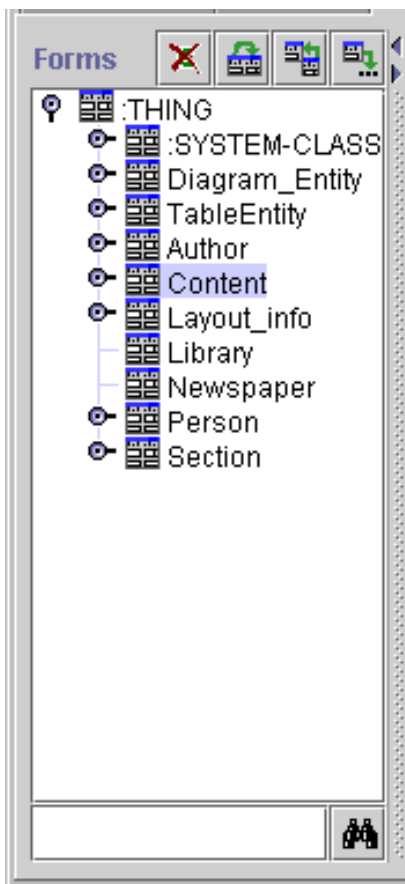
The Forms Window

This displays all the forms in the project, by the name of their associated class. When a single form is selected in this window, its associated form is displayed in the Form Edit pane to the right. Every class, concrete or abstract, has an associated form. For a concrete class, a user defines an instance of the class by filling out the class's form. For an abstract class, the form is only used to define layout information

which can then be inherited by other classes. If you have not created or modified the form, Protégé-2000 uses a default layout based on the slot type and cardinality.

As usual, icons to the left of the name give information about the display of the hierarchy:

	This icon indicates that all subclasses of the class are displayed. In some views, this is shown as a -.
	This icon indicates that the class has subclasses which are not currently displayed. In some views, this is shown as a +.
No icon	The absence of a or a indicates that the class has no subclasses.

The following icon to the right of a class form gives information about the form associated with the class:

	The form has been customized. You can customize a form by altering its widgets, for example: moving or resizing a widget, changing a widget's label, or selecting which buttons appear for the widget. See [Modifying a Widget's Appearance](#) for more information.
	You can clear all customizations for the selected form by clicking the Remove Customizations ✗ Form Button at the top right of the Forms Window. This will return the form to the default layout based on slot type and cardinality.

Next:

The Form Buttons

The Form Buttons at the top right of the [Forms Pane](#) allow you to change the layout of the selected form.

✗	The Remove Customizations button allows you to clear all customizations for the selected class form. This will return the form to the default layout, which has the following characteristics: 1. Widgets appear in an order based on size, type, and label. 2. Each widget is a standard type, based on the slot's **type** and **cardinality**. 3. Each widget is a standard size for that widget type. In addition, when you remove customizations, any labels you have changed will be returned to their default labels, and any hidden widgets will be displayed. However, if you have changed the browser key, that will not be returned to the default.
	The Default Layout button will return the selected class form to the default layout, based on the current information. If you have renamed any of the widget labels, or hidden any widgets, those changes will be retained. Any changes to the browser key will also be retained.
	The Layout Like Form button allows you to choose a form from a dialog box of all forms and use that as a basis for your layout. In this case, any widgets the two forms have in common are laid out in the same way. Any widgets that appear in your current form that do not appear in the form you choose as a base are displayed in a standard layout below the chosen form. Any widgets that appear in the base form that are not in the current form appear as blank space in the new form.

 You can also customize a form in a number of ways, by customizing global properties as well as individual widgets on the form. See the [Forms Table of Contents](#) for more information.

Next: [The Form Edit Pane](#)

[Forms Table of Contents](#)

The Form Edit Pane

The Form Edit Pane shows the layout of the form associated to the selected class and allows you to edit the form's layout.

Components

The Form Edit Pane has the following components:

1. The Browser Key menu, which allows you to choose the browser key for the form.
2. The Widget Type menu, which allows you to choose a widget type for the currently selected widget, if any.
3. A number of user-interface widgets. The currently selected widget, if any, is outlined in blue. Widgets are translated to entry fields in the Instances Tab, and control how users will enter information as instances. Each slot in the class is associated with a user-interface widget on the form.

In addition, double-clicking on the Form Edit Pane brings up a dialog box as follows:

1. Double-clicking on any widget brings up the Widget Configuration dialog for that widget, which allows you to view widget information and edit the widget's configuration, depending on the widget type.
2. Double-clicking on the background of the Form Edit Pane brings up the Form Configuration dialog, which allows you to view the widget type for each widget, redisplay hidden widgets, and select which widget, if any, takes up most of the horizontal and/or vertical space as the form expands.

Form Browser Key	Selected Widget Type
S name ▼	InstanceListWidget ▼

Name	Salary
Current Job Title	Phone Number
Date Hired	Other Information
Byname	
Sections V C + −	Supervises V C + −
Currently Selected Widget	

Default Layout

If you have not created or modified the form, Protégé-2000 uses a default layout, as follows:

1. Widgets appear in an order based on size and type.
2. Each widget is a standard type, based on the slot's **type** and **cardinality**.
3. Each widget is a standard size for that widget type.

The layout that you create in the Form Edit Pane appears in the [Instances Form](). Users use the form to enter knowledge into the knowledge base as instances.

The example below shows the default form for Newspaper, which has 4 slots. There are 3 different slot-value types and both single and mutliple cardinality slots.

You can use the Form Edit Pane to design an interface which allows easy entry of the disparate types of information represented by the different types of slots.

There are a number of basic editing tools provided on the Form Edit Pane. Some of the actions you can perform with widgets are:

1. Moving widgets to a different location and/or resize them.
2. Hiding or redisplaying a widget.
3. Selecting a different type of widget from the Select Widget Type menu.
4. For widgets with buttons, choosing which buttons are displayed.

Note: It is also possible to create customized widgets using the programmer's interface.

You can also make changes to the global properties of the form, including:

1. Selecting a browser key, which is used to identify the different instances when they are displayed in a list.
2. Selecting a single widget that will expand to take up most of the form, rather than having all widgets resize proportionally.

Next: The Browser Key Menu

Forms Table of Contents

The Widget Type Menu

The Widget Type menu allows you to choose the widget type for the currently selected widget. The choices for type of widget depend on the [value type](#) of the slot. Each value type has a default widget for the slot, and a preset list of possible widget types based on the slot's value type.

Clicking on the menu displays a list of possible options for the currently selected widget. For example, for a simple text entry widget, you can select between a TextFieldWidget and a TextAreaWidget.
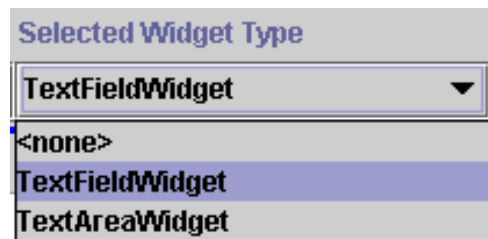
Making a new selection will change the appearance of the widget on the form. For example, the **urgent** slot is a slot of type Boolean. By default it is set to display as an **CheckBoxWidget**, which gives a drop-down list users can select from. By selecting **ComboBoxWidget** from the Widget Type menu and resizing it, you can have the widget display as a drop-down list with the choices **true** or **false**.
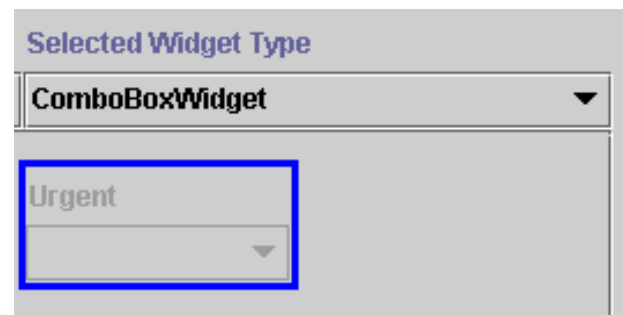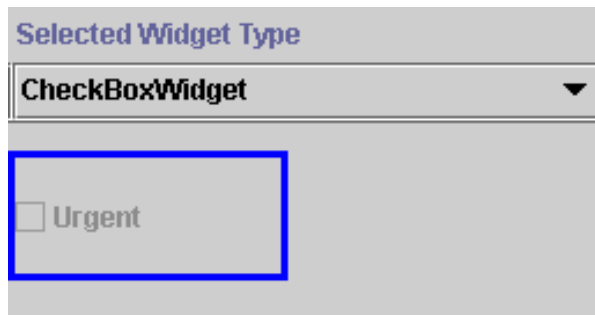
Setting the Widget Type

To set a widget type for a slot:

1. Select the name of the class whose form you wish to edit in the [Forms pane](#) at the left of the [Forms Tab](#).
2. Click the widget you wish to edit in the Form Edit Pane. The widget will be highlighted with a blue outline.
3. Click the Selected Widget Type menu. A list of possible widget types is displayed.
4. Select the type of widget you wish to use.

If you do not select a widget type, Protégé-2000 uses a default type, based on the type of the slot.

If you select **<none>** from the Widget Type menu, the widget is removed from the [Form Edit Pane](#) and is not displayed in the [Instances Form](#) for that class. To redisplay a widget that has been removed by choosing **<none>**, open the [Form Configuration dialog](#) by double-clicking on the background of the [Form Edit Pane](#) and choose a different display type for the widget.

Next: [The Widget Configuration Dialog](#)

The Widget Configuration Dialog

The Widget Configuration dialog box allows you to:

1. Change the label of any widget.
2. Change the widget type for any widget, including redisplaying currently hidden widgets.
3. Select a single widget that takes up most of the horizontal and/or vertical space as the form expands.

The General Tab

The General tab displays the name of the class and slot that the widget is associated to, as well as the label that is used for the widget on the form. The example below shows the General tab for the Name widget in the Editor form.

For simple widgets, the General tab is the only tab that is available.

To change the label of a widget, type a new label in the entry box and click OK.

Other Tabs

For complex widgets, such as widgets with multiple cardinality, other tabs may appear in the Widget Configuration dialog box. For example, the dialog box for a standard InstanceList Widget also has a Buttons tab. This tab allows you to choose which buttons to display on the form.
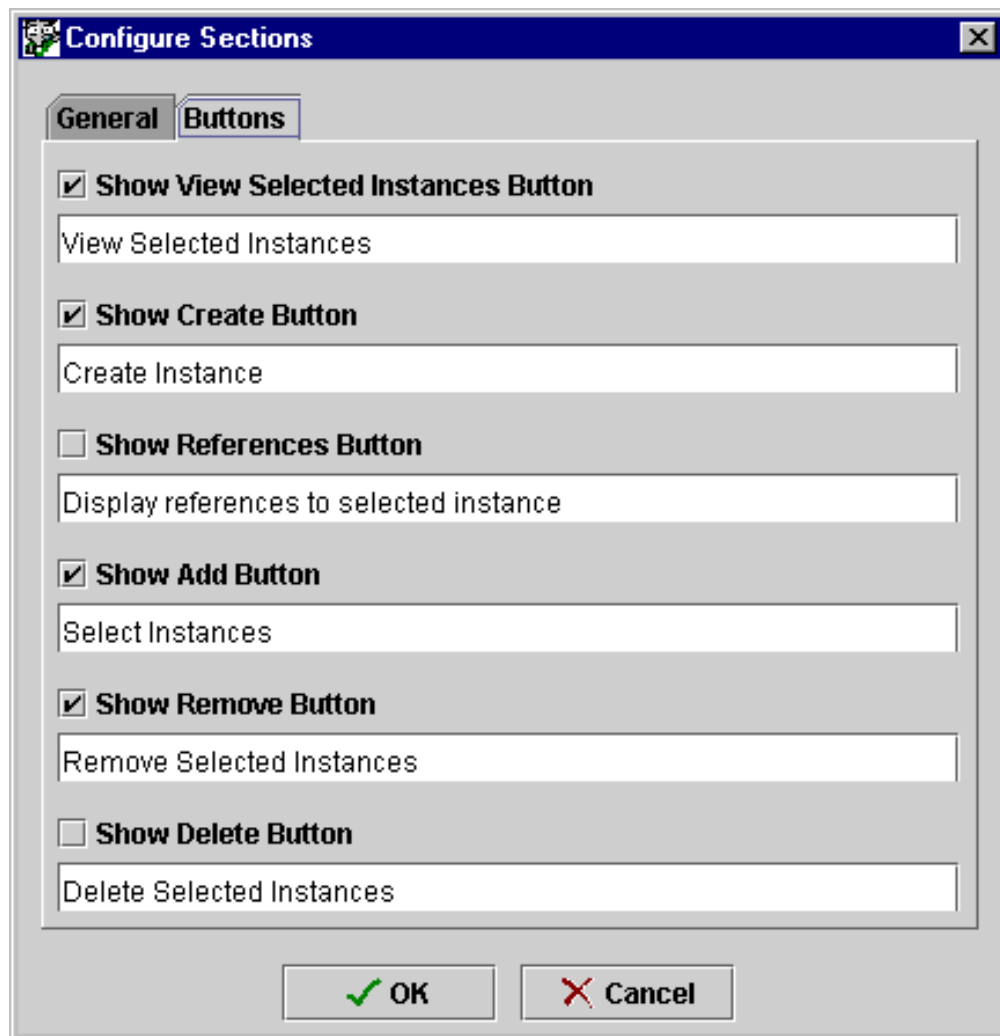
Next:

The Form Configuration Dialog Box

The Form Configuration dialog box allows you to:

1. View the list of available widgets for the currently select form.
2. Change the widget type for any widget, including redisplaying currently hidden widgets.
3. Select a single widget that takes up extra horizontal and/or vertical space as the form expands.

The Widget Tab

The Widget tab displays all the widget information for the form. For example, the Widget tab in the Form Configuration dialog box for Editor lists all the slots for Editor with the currently selected widget type.
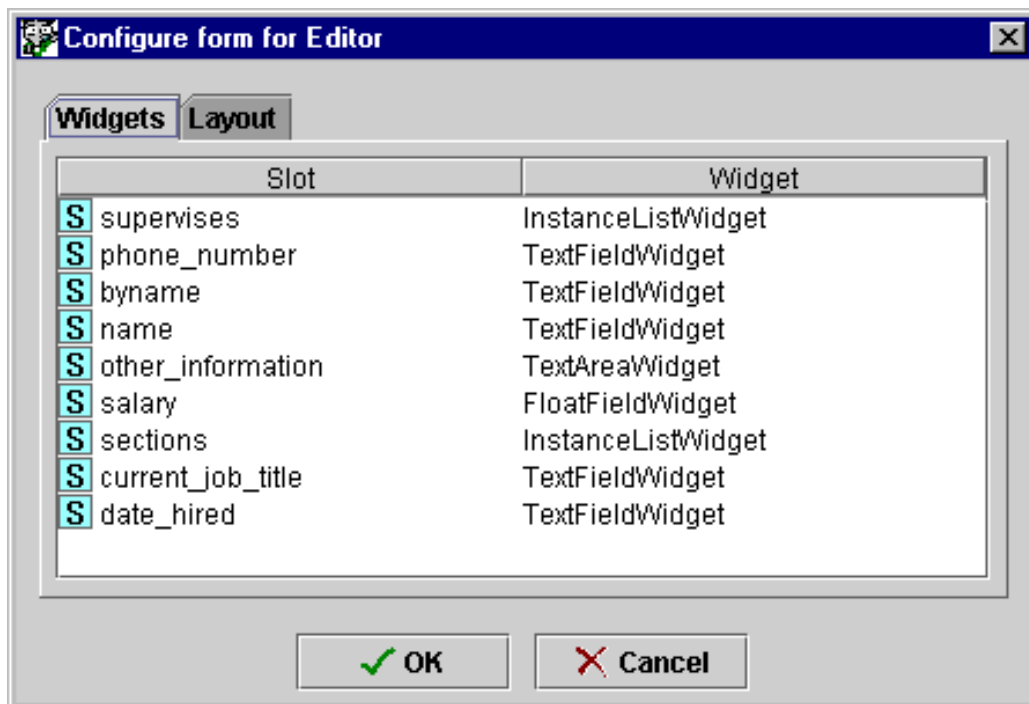
To change the widget type for the slot:

1. Click on the widget name in the widget column. A menu shows the list of standard widget types for the slot.
2. Select the widget you want.
3. Click OK.

If a widget has been hidden by choosing **<none>** from the Widget Type menu, you can redisplay the widget by selecting a different widget type under the Widget tab in the Form Configuration dialog box.

The Layout Tab

The Layout tab displays the current layout information. You can use the Layout tab to choose a widget that will expand when the form expands as the Protégé-2000 window is resized. This can be useful, for example, when you have a complex table widget that you wish to have take up most of the form. Clicking on one of the menus displays a list of all the widgets. You can choose a widget that expands vertically and a widget that expands horizontally. If you wish, these can be the same widget. Selecting **<all>** from the menu means that each widget is resized an equal amount.

Next:

Default Widget Types

The Form Edit Pane displays a widget for each slot in the class. In the Instances Form, these widgets are displayed as fields where users can enter the information for that slot. The possible display and options for the widget depend on the type of information that is included in the slot. For more information on the different types of slots and fields, see the Instances Table of Contents.

Default Widgets

When Protégé-2000 generates a default layout, it creates a default widget for each slot. These widgets are described below.

Note: These topics describe the default widgets only. You can select different formats for the widgets using the Widget Type Menu.

Protégé-2000 provides the following widgets for both **Single** and **Multiple** cardinality. For simplicity, only the **Single** cardinality is shown:

Widget Type	Default Single Cardinality Widget	Default Field Appearance
Boolean	A checkbox widget that describes a slot as true or false for this instance	☑ Urgent
Class	A text display widget and three buttons that allows the end-user to specify a class as the value for this slot	Class V + −
Float	A text entry widget that verifies that the value entered by an end-user is a valid decimal number	Salary 12.50
Instance	A text display widget and four buttons that allow an end-user to specify an instance as the value for this slot	Section V C + − Lifestyle
Integer	A text entry widget that verifies that the value entered by an end-user is a valid whole number	Page Number 14
String	A text entry widget	Headline Bank Outflanks Stubborn 90 Year Old Man
Symbol	A drop-down list that allows the end-user to select from a preset list of values	Reading Level High_school ▼

Single vs. Multiple Default Widget

A multiple cardinality widget is very similar to the corresponding single widget.

Single Widget	Multiple Widgets
Some widgets (e.g., Float, Integer, Symbol) do not have Field Buttons.	Always have Field Buttons which allow end users to view, add or create, and delete values for the field. You can control the display of the buttons using the Widget Configuration dialog box.

Next: Additional Widget Types

Forms Table of Contents

Additional Widget Types

When Protégé-2000 generates a default layout, it creates a default widget for each slot. However, the Widget Type Menu allows you to to change the widget type for the currently selected widget. The choices for type of widget depend on the value type of the slot. Some value types have only one possible widget.

The following widget types are available:

- InstanceRowWidget, which allows you to see a summary of all the slots of a single Instance value
- InstanceTableWidget, which allows you to see a summary of all the slots of a list of multiple Instance values
- ContainsWidget, which displays the form for single or multiple Instance or Class values
- SliderWidget, which displays a slider for a bounded integer value
- ImageMapWidget, which allows the user to select an icon for a Symbol value

To change a widget, first select the widget you want to change, then select the new widget type from the Widget Type Menu.

Next: InstanceRowWidget

Forms Table of Contents

InstanceRowWidget

The InstanceRowWidget is an optional widget for slots of type Instance with single cardinality (**at most** equal to 1). The default widget, InstanceFieldWidget, shows the name of the Instance that is selected as the value of the slot. The InstanceRowWidget actually shows the slot values for the widget.

InstanceFieldWidget
(Default for Instance, Single)

InstanceRowWidget

Creating an InstanceRowWidget

To make an InstanceRowWidget, first select the widget you want to change, then select InstanceRowWidget from the Widget Type Menu.

Customizing an InstanceRowWidget

When you select InstanceRowWidget from the Widget Type Menu, you will usually have to resize the widget.

In addition, you might not wish to use all the slots, and you might want to change the order in which they appear. You can make these changes using the Form Configuration dialog box. Note that the current implementation does not allow you to change the order directly by dragging, but you can reorder the slots by first removing them from the widget, and then adding them back in the order in which you wish them to appear.

To add or remove slots, change their order, or change the name under which they are displayed:

1. Double-click on the widget to open the Form Configuration dialog box.
2. Click the **Define columns of instance table** tab.

3. To make any modifications, you must click in the **Column Name** of the slot you wish to modify. (The top row, **Slot Name**, is a property of the slot and *cannot* be modified.)

4. To remove a column, click in its **Column Name** and click the Remove ‾ button.

Configure Contact Person   ✕

| In-place Editing Characteristics | Warnings And Suggestions | Action Buttons |
| General | Define columns of instance table | New instances |

Selected Slots   + −

	other_information	phone_number	name
Slot Name	other_information	phone_number	name
Column Name	other_information	phone_number	name
Text Color			

✓ OK   ✕ Cancel

5. To rename a column, edit the text in **Column Name**.

6. To add a column, click the Add **+** button.
   The column will be added after all the existing columns.

7. To reorder columns, remove all the columns except the one you want to appear first, then add the other columns. For instance, to make **name** appear first in the example, you would delete the **other_information** and **phone_number** columns, then add any columns you wanted back in.

Configure Contact Person

	name	phone_number
Slot Name	name	phone_number
Column Name	name	phone_number
Text Color		

Next:

Forms Table of Contents

![icon] InstanceRowWidget and InstanceTableWidget

The InstanceRowWidget is an optional widget for slots of type Instance with multiple cardinality (**at most** not specified or greater than 1). The default widget, InstanceListWidget, shows the names of the Instances that are selected as the values of the slot. The InstanceTableWidget actually shows the slot values for each slot.

The InstanceTableWidget is very similar to the InstanceRowWidget, except that it is available for Instance slots with multiple cardinality. Therefore, instead of just one row, the InstanceTableWidget can display multiple rows of information.

Articles											
containi...	article_...	layout	expirati...	text	Article_...	urgent	publish...	reading...	keywords	headline	page_n...
Lifestyle	AP		7/01/97	DALLAS (	News	false			banks	Bank Outf	5
Science	Joe Schm			Sojourner		false	08/22/97		space, sc	Destinatic	2
Local Nev			10/01/97	(Stanford		false	08/21/97	College	SMI	SMI Short	8

Creating an InstanceTableWidget

To make an InstanceTableWidget, first select the widget you want to change, then select InstanceTableWidget from the Widget Type Menu.

Customizing an InstanceTableWidget

When you select InstanceTableWidget from the Widget Type Menu, you will usually have to resize the widget.

In addition, you might not wish to use all the slots, and you might want to change the order in which they appear. You can make these changes using the Form Configuration dialog box. Note that the current implementation does not allow you to change the order directly by dragging, but you can reorder the slots by first removing them from the widget, and then adding them back in the order in which you wish them to appear.
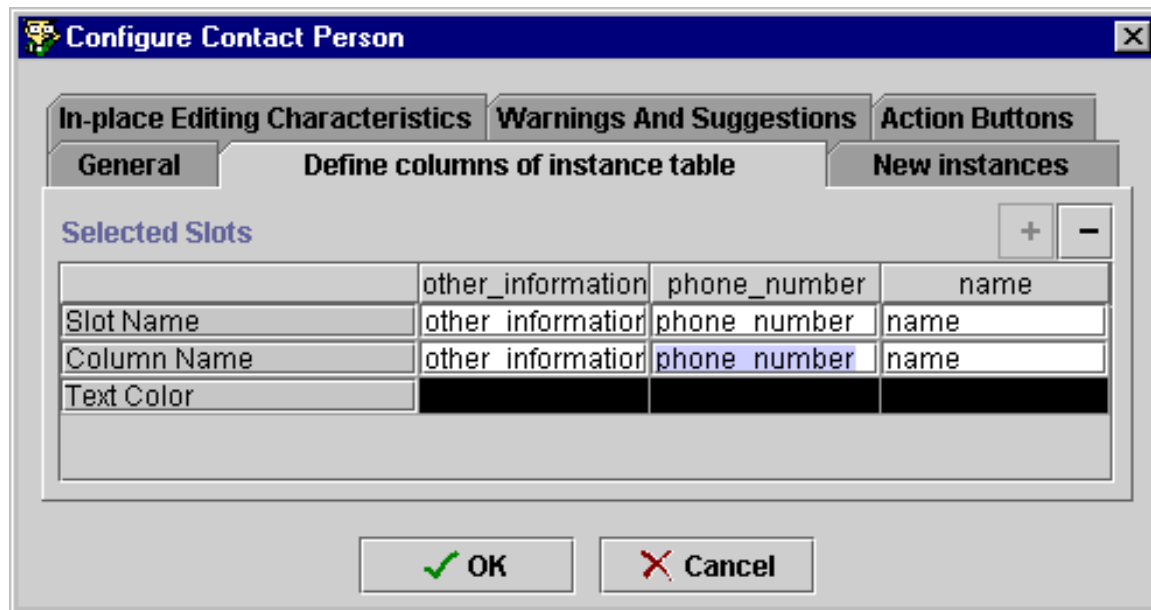
To add or remove slots, change their order, or change the name under which they are displayed:

1. Double-click on the widget to open the Form Configuration dialog box.
2. Click the **Define columns of instance table** tab.

Configure Contact Person

| In-place Editing Characteristics | Warnings And Suggestions | Action Buttons |
| General | Define columns of instance table | New instances |

Selected Slots + −

	other_information	phone_number	name
Slot Name	other_information	phone_number	name
Column Name	other_information	phone_number	name
Text Color			

✓ OK   ✗ Cancel

3. To make any modifications, you must click in the **Column Name** of the slot you wish to modify. (The top row, **Slot Name**, is a property of the slot and *cannot* be modified.)

4. To remove a column, click in its **Column Name** and click the Remove ▬ button.

Configure Contact Person

| In-place Editing Characteristics | Warnings And Suggestions | Action Buttons |
| General | Define columns of instance table | New instances |

Selected Slots + −

	other_information	phone_number	name
Slot Name	other_information	phone_number	name
Column Name	other_information	phone_number	name
Text Color			

✓ OK   ✗ Cancel

5. To rename a column, edit the text in **Column Name**.

6. To add a column, click the Add **+** button.
   The column will be added after all the existing columns.

7. To reorder columns, remove all the columns except the one you want to appear first, then add the other columns. For instance, to make **name** appear first in the example, you would delete the **other_information** and **phone_number** columns, then add any columns you wanted back in.

Next:

ContainsWidget

The ContainsWidget is an optional widget for slots of type Instance. The default widget, InstanceFieldWidget or InstanceListWidget, shows the name of the Instance(s) that are selected as the value of the slot. The ContainsWidget actually contains a version of the form for each instance that is a value of the slot.

A ContainsWidget is useful primarily for a Class or Instance slot where the form for the Allowed Classes is fairly simple, and where the cardinality is single or small. For a widget that provides a summary of selected fields, see the InstanceRowWidget or the InstanceTableWidget.

Creating a ContainsWidget

To make a ContainsWidget, first select the widget you want to change, then select ContainsWidget from the Widget Type Menu.

Next: SliderWidget

Forms Table of Contents

![icon] SliderWidget

The SliderWidget is available for only slots of type Integer that are *bounded*, that is, they have both a Maximum and Minimum value. In this case, you can create a widget that looks like a slider bar, with the possible values shown as ticks on the bar. The value of the selection is shown in the right and changes automatically as you move the slider.
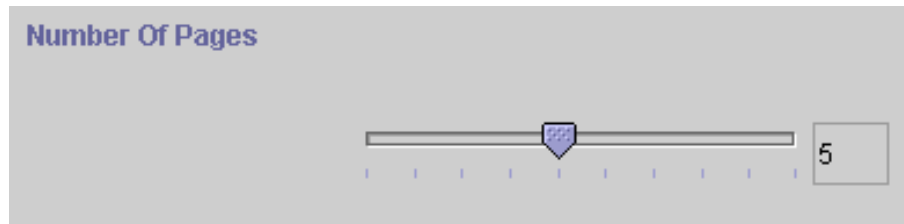
Creating a SliderWidget

To make a SliderWidget, first select the widget you want to change, then select SliderWidget from the Widget Type Menu.

When you first create a SliderWidget, it will probably have a very ugly display. You must enlarge the widget to make the display useful.

A slider widget in need of editing.

Next: ImageMapWidget

Forms Table of Contents

ImageMapWidget

The ImageMapWidget allows you to associate images to the different values of a slot of type Symbol. The user can then select the symbol by clicking on the image corresponding to it. The selected images must be rectangular.

Creating an ImageMapWidget

To make an ImageMapWidget, first select the widget you want to change, then select ImageMapWidget from the Widget Type Menu.
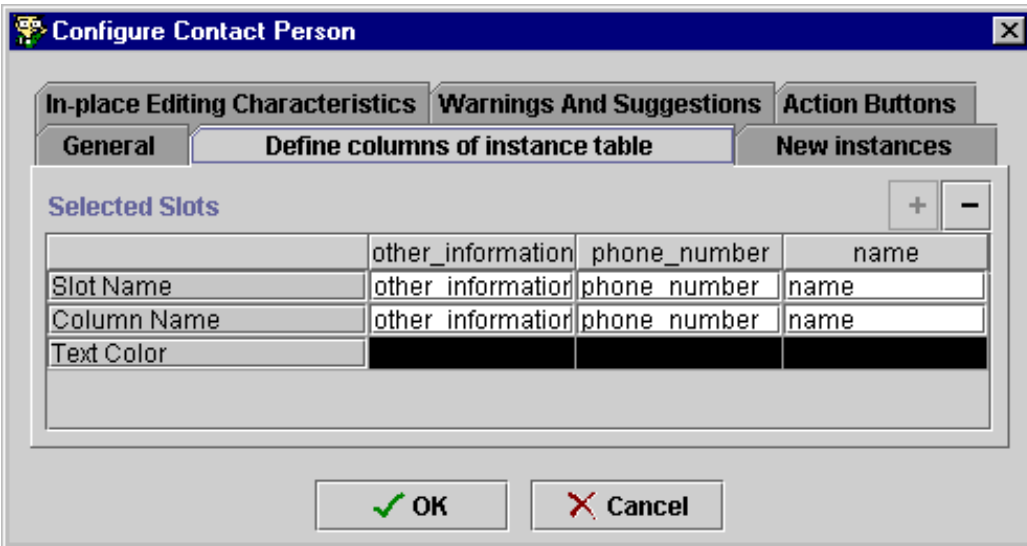
Configuring an ImageMapWidget

To configure an ImageMapWidget:

1. Double-click on the widget to open the Form Configuration dialog box.
2. Click the **Configure ImageMap** tab.
3. Select the Image Location for the first image.
4. Click the **Define Rectangles** tab.
5. Using the Currently Selected Symbol menu, select the symbol value you want to correspond to the first image.
6. Draw a rectangle to position the image on the display. The image in Image Location should appear within the rectangle you draw.
7. Repeat steps 2 - 6 for each image/symbol pair.
8. Click OK.

The images you have chosen should appear in the Forms window.

Next: Advanced Widget Types

Advanced Widget Types

In addition to the included widget types, you can add the capacity for more complex widgets to your Protégé-2000 project by loading special projects that extend the possible widgets. Protégé-2000 includes the following projects, in the **newspaper** folder, that allow you to extend the types of widget you can use:

- a **table** project, which allows you to display information stored as a function of two classes in tabular format
- a **diagram** project, which allows you to display relational information in the form of a diagram or graph.

To use these widgets, you must not only include the related project, but you must add other information to your project and, in the case of tables, make sure your project is actually constructed to support the widget. The additional work required, however, allows you to create an interface which conveys complex information in a visual and immediate way. See the tutorial for more information.

Next: Changing Form Characteristics

Forms Table of Contents

Changing Form Characteristics

You can change the following global properties of a form:

- [Change the browser key](#), that is the slot that is used to identify the different instances when they are displayed in a list.
- [Change the layout](#) by selecting a single widget that takes up most of the horizontal and/or vertical space as the form expands.

Setting the Browser Key

The browser key identifies the different instances when they are displayed in a list. For example, the form for **Prototype_Newspaper** has the browser key set to **weekday**. This means that, in the [Direct Instances Pane](#), the different instances are listed by the value in the **weekday** field.
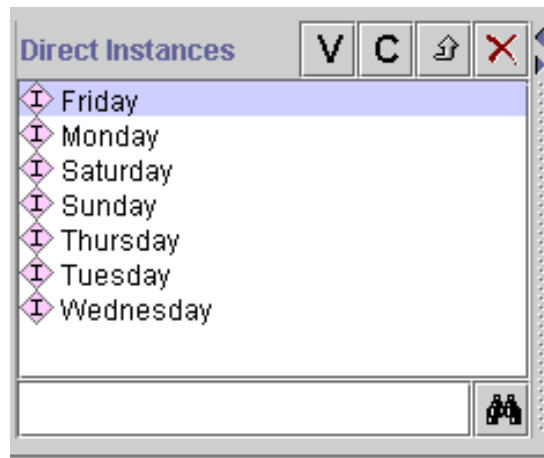
If you do not select a slot to use as a browser key, Protégé-2000 uses a default key, **<instance name>**, such as newspaper_0017. It is usually helpful to set a browser key.

To set a browser key for a class:

1. Select the name of the class whose form you wish to edit in the [Forms pane](#) at the left of the [Forms Tab](#).
2. Click the [Browser Key menu](#). A list of slots is displayed.
3. Select the slot you wish to use as a browser key.

Changing the Layout

You can choose a widget that will expand as the form expands whenever the Protégé-2000 window is

resized. This can be useful, for example, when you have a complex table widget that you wish to have take up most of the form. To do this:

1. Double-click anywhere on the background of the form, that is, on an area that does not contain a widget. The Form Configuration dialog box is displayed.
2. Click on the Layout tab.
3. To select a widget that will take up most the vertical room when the form expands, select the widget name from the Fill Vertical Space With: menu. Selecting **<all>** means each widget will expand equally.
4. To select a widget that will take up most the horizontal room when the form expands, select the widget name from the Fill Horizontal Space With: menu. You can select the same widget or a different one. Selecting **<all>** means each widget will expand equally.

To reset the form so that no widget expands more than the others, make sure that **<all>** is selected in both menus.

Note: For additional tasks that change a forms appearance, such as resizing, moving, changing, or hiding a widget, see the Forms Table of Contents.

Next: Modifying a Widget's Appearance

Forms Table of Contents

Modifying a Widget's Appearance

There are a number of ways to change the appearance of a widget. The most common ways are:

- changing the widget's label
- resizing the widget
- repositioning the widget on the form

For some widgets, you can also change which buttons are displayed and other widget-specific properties.

Changing a Widget's Label

Each widget has a label -- a string of text displayed in the left hand corner of the widget. The default layout uses a version of the slot name for the widget label, where underscores are replaced by blanks and the first letters of the resulting words are capitalized. When end users actually enter instances, they use the label to determine the type of information they should enter. To change the label:

1. Double-click directly on the widget. The Widget Configuration dialog box is displayed. (Note: If the Form Configuration dialog box is displayed, you have accidentally clicked somewhere on the background of the form. Close the dialog box and click again.)
2. Enter the label you want in the label field of the Widget Configuration dialog box. You can use a multi-word phrase.
3. Click OK.

Resizing a Widget

You can easily make a widget smaller or larger by resizing it. To do this:

1. Click on the widget to select it. A blue boundary appears around the border of the widget.
2. Click and drag anywhere on the boundary to resize the widget. Most widgets can only be resized horizontally (narrower or wider). You can resize cardinality Multiple widgets both horizontally and vertically.

Note: You can select a single widget that will take up most of the form as the form is resized on the screen by end users. See Changing a Form's Layout for more information.

Moving a Widget to a New Location

To move a widget on the form:

1. Click on the widget to select it. A blue boundary appears around the border of the widget.
2. Click in the interior of the widget and hold down the mouse button.
3. Drag the widget to the desired location.

Note: To hide a widget so that it won't be displayed on the form, use the [Widget Type Menu](#). See Hiding a Widget for more information.

Choosing a Widget's Buttons

Some widgets, for example, all cardinality Multiple widgets, have buttons displayed on the upper right corner of the widget. You can turn the display of these buttons on and off individually. To choose which buttons are displayed for a widget:

1. Double-click directly on the widget. The [Widget Configuration](#) dialog box is displayed. (Note: If the [Form Configuration](#) dialog box is displayed, you have accidentally clicked somewhere on the background of the form. Close the dialog box and click again.)
2. If the widget has buttons, there will be a Buttons tab in the [Widget Configuration](#) dialog box. Click this tab.
3. Select the buttons you want to display by making sure there is a check in the appropriate check boxes.
4. Click OK.

Selecting a Widget Display

Some slot types have multiple widget configurations that can be selected from the Widget Type Menu. For example, a text entry slot can have either a TextFieldWidget or a TextAreaWidget. A Boolean slot can have either a CheckBoxWidget (as shown above) or a ComboBoxWidget. A Symbol slot can have various widgets, including a ComboBoxWidget or an ImageMaptoSymbolWidget. By experimenting with these widgets, you can learn how to use them to get the desired display.

You can also remove a widget from the form by hiding it completely, or redisplay a hidden widget.

Selecting a Different Display

To select a different display for your widget from one of the available configurations:

1. Select the widget you wish to change by clicking it once. The widget is outlined in blue.
2. Select a different display from the Widget Type Menu.

Hiding a Widget

To hide a widget:

1. Select the widget you wish to hide by clicking it once. The widget is outlined in blue.
2. Select **<none>** from the Widget Type Menu. The selected widget is removed from the form.

Redisplaying a Hidden Widget

To redisplay a widget which is no longer visible on the form:

1. Double-click on an empty part of the form to display the Form Configuration dialog box.
2. Select the widget you want to redisplay by clicking on its name at the Widget tab.
3. Click on the text **<none>** under the Widget column.
4. Select a different display type for the Widget.
5. Click OK.

Glossary

This Glossary defines terms used in this description of Protégé-2000. It is not meant to be an exhaustive list of terminology associated with object-oriented modeling.

Abstract Class

An abstract class cannot have instances and is identified with an A icon.

Allowed Classes

A constraint on the values of a type-instance slot. The value of the slot can only be an instance of the class (or any of its children) in the Allowed Classes list.

Application

The program that combines the Protégé-2000 knowledge base with a problem-solving method so that end users can use a Protégé-2000 knowledge base to solve a problem.

Boolean

A type of slot with a true or false value that appears as a checkbox in Protégé-2000. A checked box equals a true value.

Browser Key

The browser key of a class is one of its slots whose value is displayed when instances of this class are referred to by other instances' forms.

Cardinality

A slot facet that describes whether the slot has just one value (*single*) or more than one value (*multiple*). In Protégé-2000, Single is the default.

Class

An abstract representation of a concept in a domain as a collection of related classes. For example, a medical model might have *protocol*, *guidelines*, and *patient data* as classes. A class appears with one of the following icons in Protégé-2000: C, C, or C. A class can have a set of slots that represent the attributes of the class.

Classes Tab

The Protégé-2000 part used to create, view, revise, and save classes.

Concrete Class

Concrete classes can have instances and are identified by the lack of an A.

Direct Slot

A slot attached directly to a class (in contrast to a slot which is inherited).

Domain

> A particular field of knowledge, such as breast cancer.

Facets

> The attributes of a slot. Some facets depend on the value of the type facet. For example, an integer slot type has facets for Minimum and Maximum.

Float

> A positive or negative real numeric value (e.g., 1.0, 3.4e10, -0.3e-3) used as a slot value.

Forms Tab

> The Protégé-2000 part used to create the forms for acquiring instances of classes. It may also be used to view, revise, and save the forms.

Inheritance

> A parent-child (superclass-subclass) relationship between two classes. A child (subclass) inherits the slots of its parent classes (superclasses).

Inherited Slot

> A slot that is attached to a class via inheritance from a parent class.

Instance (KB value)

> Concrete occurrence of information about a domain that is entered into a knowledge base. For example, Fran Smith might be an instance for a Name slot. An instances is entered via a form generated by Protégé-2000.

Instance (slot type)

> A type of slot whose value is the instance of a class.

Instances Tab

> The Protégé-2000 part used to acquire instances of classes. It may also be used to view, revise, and save the instances.

Integer

> A positive or negative whole number (e.g., 1, 2, -4) used as a slot value.

Knowledge-acquisition tool

> A tool used to build a knowledge base by acquiring instances. In Protégé-2000, the forms comprise the KA tool.

Knowledge base (KB)

> A set of instances of classes which may be used by PSMs.

Knowledge-based system

> A computer system that includes a knowledge base about a domain and programs that include rules for processing the knowledge and for solving problems relating to the domain.

Ontology
A model of a particular field of knowledge - the concepts and their attributes, as well as the relationships between the concepts. In Protégé-2000, an ontology is represented as a set of classes with their associated slots.

.pins file
A Protégé-2000 file in clips format that contains instances.

.pont file
A Protégé-2000 file in clips format that contains an ontology.

.pprj file
A Protégé-2000 file that contains a project. A project file contains the customized form information and references to external sources of the domain information.

Problem-solving method (PSM)
A computer program used in conjunction with a knowledge base to answer questions or solve problems.

Slot
An attribute of a class. For example, a *physician class* might have *name*, *title*, and *phone number* as slots.

Slots Tab
The Protégé-2000 part that allows you to create, view, edit, and delete slots.

Symbol
An enumerated list of slot values, such as red, blue, green.

Type
A slot facet that identifies the kind of values a slot may have - Any, boolean, float, instance, integer, string, or symbol.

What is Protégé-2000?

Protégé-2000 is an integrated software tool used by system developers and domain experts to develop *knowledge-based systems*. Applications developed with Protégé-2000 are used in problem-solving and decision-making in a particular *domain*.

While our earlier Protégé/Win, like a classical database system, separately defined *classes* of information (schema) and stored *instances* of these classes, Protégé-2000 makes it easier to work simultaneously with both classes and instances. Thus, a singular instance can be used on the level of a class definition, and a class can be stored as an instance. Similarly, *slots,* which earlier were employed only within classes, are now elevated to the same level as classes. With this new knowledge model we also facilitate conformance to the *Open Knowledge Base Connectivity (OKBC)* protocol for accessing knowledge bases stored in knowledge representation systems. Finally, *applications* on top of these components are also executed within the integrated Protégé-2000 environment.

The Protégé-2000 tool accesses all of these parts through a uniform GUI (graphical user interface) whose top-level consists of overlapping tabs for compact presentation of the parts and for convenient co-editing between them. This "tabbed" top-level design permits an integration of (1) the modeling of an *ontology* of classes describing a particular subject, (2) the creation of a *knowledge-acquisition tool* for collecting knowledge, (3) the entering of specific instances of data and creation of a *knowledge base*, and (4) the execution of applications. The ontology defines the set of concepts and their relationships. The knowledge-acquisition tool is designed to be domain-specific, allowing domain experts to easily and naturally enter their knowledge of the area. The resulting knowledge base can then be used with a *problem-solving method* to answer questions and solve problems regarding the domain. Finally, an *application* is the end product created when the knowledge base is used in solving an end-user problem employing appropriate problem-solving, expert-system, or decision-support methods.

The main assumption of Protégé-2000 is that knowledge-based systems are usually very expensive to build and maintain. For example, the expectation is that knowledge-based system development is a team effort, including both developers and domain experts who may have less familiarity with computer software. Protégé-2000 is designed to guide developers and domain experts through the process of system development. Protégé-2000 is designed to allow developers to reuse domain ontologies and problem-solving methods, thereby shortening the time needed for development and program maintenance. Several applications can use the same domain ontology to solve different problems, and the same problem-solving method can be used with different ontologies. For more information about building knowledge-based systems and the Protégé-2000 approach, see Planning a Protégé-2000 Project.

Protégé-2000 is currently being used in clinical medicine and the biomedical sciences, although it can be used in any field where the concepts can be modeled as a class hierarchy. Throughout this online guide, you will see examples of a newspaper domain used to illustrate points of explanation; for more information about this running example, see an overview of the Newspaper Example.

Next:

Planning a Protégé-2000 Project

The development of a successful knowledge-based system built with Protégé-2000 is more of an art than a science. Nonetheless, we can suggest a standard pattern of use that new users should follow to avoid some possible problems of systems development. Protégé-2000 is designed to support *iterative development*, where there are cycles of revision to the ontologies and other components of the knowledge-based system. Therefore developers should not expect to "complete" ontology development without considering other aspects of the process.

For the development of a successful Protégé-2000 project, we would recommend the following steps:

1. Plan for the application and expected uses of the knowledge base. This usually means working with domain experts that have a set of problems that could be solved with knowledge-base technology.
2. Build an initial small ontology of classes and slots, as explained in the Classes and Slots strand.
3. When you have built this ontology (and later when you have extended it or opened it from file), you can directly view forms for entering instance knowledge into the ontology, because Protégé-2000 generates initial forms "on the fly", in its role as a KA-tool generator.
4. You use these forms for acquiring slot values of your test instances, as explained in the Instances strand. At this point, it is usually appropriate to show the ontology and the filled-out instance forms to the domain experts or your expected users. This inevitably leads to a set of revisions, both to the ontology (2.) and to the forms (5.). Note that ontology modifications can be expensive, since some sorts of change could force rebuilding some or all of the knowledge base.
5. Customize the forms to a refined knowledge-acquisition tool, as explained in the Forms strand. While constructing this customized version of the KA-subtool, further design problems in the original ontology may become apparent. If necessary revise the ontology and repeat at 4.
6. With your domain experts, build a somewhat larger knowledge-base that can be tested with your application or problem-solving method.
7. Test the full application with your end-users. This step can lead to further revisions to the ontology and the KA-subtool.

The picture below shows this typical pattern of use for the Protégé-2000 subsystems. The black arrows indicate the forward progression through the process, while the blue dotted arrows show places where revisions are usually necessary (either to the ontology or the knowledge-acquisition tool).
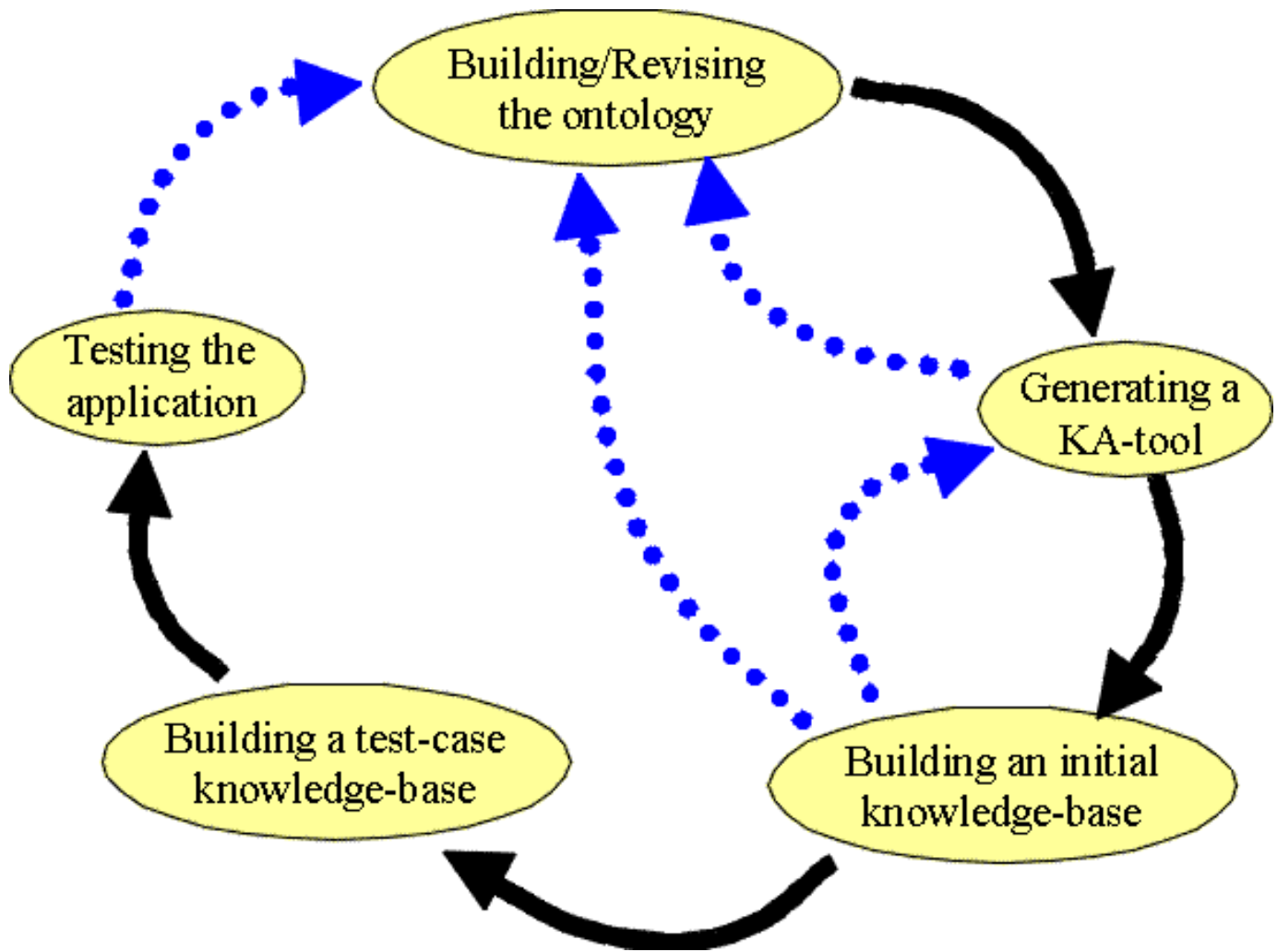
At the heart of a successful Protégé-2000 project is the design of the class and slot structure of the ontology. In particular, the model you use in building your ontology must balance the needs of the domain expert when building a knowledge base (at knowledge-acquisition time) against the requirements of your problem-solving method or application (at run-time). Hopefully, these are not too contradictory! Ontology developers should therefore both:

- Model the domain with a set of problems and a problem-solving method in mind.
- Design the ontology so that it can be used to generate and customize an appropriate KA-tool for a specific set of users.

As a simple example from the Newspaper Example, a problem to be solved could be finding all advertisements that are more expensive than some threshold. For this problem, one should build into the ontology a class for advertisements that includes price and date published. If this information were spread out over all publication issues, then it would be more difficult for the problem-solver to access all instances of advertisements and their prices.