Онтологический инжиниринг знаний в системе *PROTÉGÉ*

МЕТОДИЧЕСКОЕ ПОСОБИЕ

ОГЛАВЛЕНИЕ

Введение

Технология

Данное методическое пособие представляет собой введение в технологию создания баз знаний на основе фреймовой модели при помощи платформо-независимой расширяемой среды Protégé, позволяющей пользователю быстро и интуитивно приступить к созданию своих онтологий.

По мере прочтения на простом примере будет показано, как создавать, модифицировать и сохранять проекты, используя фреймовую модель, а также как создавать классы, экземпляры классов, слоты и другие объекты, являющиеся основой базы знаний.

Что такое онтология?

Онтология описывает основные концепции (положения) предметной области и определяет отношения между ними.

Процесс построения онтологий состоит из создания следующих блоков:

- Классов и их свойств (classes, properties).
- Свойств каждой концепции, описывающих различные функциональные возможности и атрибуты концепции (слоты (slots), иногда называемые роли).
- Ограничения по слотам (также известных как аспекты/грани (slot facets), иногда называемые ограничения ролей).

Онтология вместе с множеством индивидуальных экземпляров классов составляют базу знаний.

В литературе по искусственному интеллекту содержится много определений понятия онтологии, многие из которых противоречат друг другу. В этой статье **онтология** – формальное явное описание понятий в рассматриваемой предметной области (**классов,** иногда их называют **понятиями**), свойств каждого понятия, описывающих различные свойства и атрибуты понятия (**слотов** (иногда их называют **ролями** или **свойствами**)), и ограничений, наложенных на слоты (**фацетов,** иногда их называют **ограничениями ролей**). Онтология вместе с набором индивидуальных **экземпляров** классов образует **базу знаний**. В действительности, трудно определить, где кончается онтология и где начинается база знаний.

Зачем создавать онтологию?

В последние годы разработка онтологий – явное формальное описание терминов предметной области и отношений между ними – переходит из мира лабораторий по искусственному интеллекту на рабочие столы экспертов по предметным областям. Во всемирной паутине онтологии стали обычным явлением. Онтологии в сети варьируются от больших таксономий, категоризирующих веб-сайты (как на сайте Yahoo!), до категоризаций продаваемых товаров и их характеристик (как на сайте Amazon.com). Во многих дисциплинах сейчас разрабатываются стандартные онтологии, которые могут использоваться экспертами по предметным областям для совместного использования и аннотирования информации в своей области. Например, в области медицины созданы большие стандартные, структурированные словари, такие как SNOMED и семантическая сеть Системы Унифицированного Медицинского Языка (the Unified Medical Language System). Также появляются обширные общецелевые онтологии. Например, Программа ООН по развитию (the United Nations Development Program) и компания Dun & Bradstreet объединили усилия для разработки онтологии UNSPSC, которая предоставляет терминологию товаров и услуг (http://www.unspsc.org/).

Онтология определяет общий словарь для ученых, которым нужно совместно использовать информацию в предметной области. Она включает машинно-интерпретируемые формулировки основных понятий предметной области и отношения между ними.

Почему возникает потребность в разработке онтологии? Вот некоторые причины:

- Для совместного использования людьми или программными агентами общего понимания структуры информации.
- Для возможности повторного использования знаний в предметной области.
- Для того чтобы сделать допущения в предметной области явными.
- Для отделения знаний в предметной области от оперативных знаний.
- Для анализа знаний в предметной области.

Совместное использование людьми или программными агентами общего понимания структуры информации является одной из наиболее общих целей разработки онтологий. К примеру, пусть несколько различных веб-сайтов содержат информацию по медицине или предоставляют информацию о платных медицинских услугах, оплачиваемых через Интернет. Если эти веб-сайты совместно используют и публикуют одну и ту же базовую онтологию терминов, которыми они все пользуются, то компьютерные агенты могут извлекать информацию из этих различных сайтов и накапливать ее. Агенты могут использовать накопленную

информацию для ответов на запросы пользователей или как входные данные для других приложений.

Обеспечение возможности использования знаний предметной области стало одной из движущих сил недавнего всплеска в изучении онтологий. Например, для моделей многих различных предметных областей необходимо сформулировать понятие времени. Это представление включает понятие временных интервалов, моментов времени, относительных мер времени и т.д. Если одна группа ученых детально разработает такую онтологию, то другие могут просто повторно использовать ее в своих предметных областях. Кроме того, если нам нужно создать большую онтологию, мы можем интегрировать несколько существующих онтологий, описывающих части большой предметной области. Мы также можем повторно использовать основную онтологию, такую как UNSPSC, и расширить ее для описания интересующей нас предметной области.

Создание явных допущений в предметной области, лежащих в основе реализации, дает возможность легко изменить эти допущения при изменении наших знаний о предметной области. Жесткое кодирование предположений о мире на языке программирования приводит к тому, что эти предположения не только сложно найти и понять, но и также сложно изменить, особенно непрограммисту. Кроме того, явные спецификации знаний в предметной области полезны для новых пользователей, которые должны узнать значения терминов предметной области.

Отделение знаний предметной области от оперативных знаний – это еще один вариант общего применения онтологий. Мы можем описать задачу конфигурирования продукта из его компонентов в соответствии с требуемой спецификацией и внедрить программу, которая делает эту конфигурацию независимой от продукта и самих компонентов. После этого мы можем разработать онтологию компонентов и характеристик ЭВМ и применить этот алгоритм для конфигурирования нестандартных ЭВМ. Мы также можем использовать тот же алгоритм для конфигурирования лифтов, если мы предоставим ему онтологию компонентов лифта.

Анализ знаний в предметной области возможен, когда имеется декларативная спецификация терминов. Формальный анализ терминов чрезвычайно ценен как при попытке повторного использования существующих онтологий, так и при их расширении.

Часто онтология предметной области сама по себе не является целью. Разработка онтологии сродни определению набора данных и их структуры для использования другими программами. Методы решения задач, доменно-независимые приложения и программные агенты используют в качестве данных онтологии и базы знаний, построенные на основе этих онтологий.

Как создать онтологию?

Строго говоря, единого универсального подхода к созданию онтологий, который бы привел к однозначно успешному результату не существует. Процесс создания онтологий обычно является итеративным, т.е. сначала создается черновой набросок, а затем по мере необходимости происходит возврат для определения деталей, и так продолжается до тех пор, пока онтология не будет отражать концепцию предметной области с определенной степенью.

Практически, создание онтологий включает:

1. Определение классов в онтологии,
2. Организация классов в некоторую иерархию (базовый класс –> подкласс),
3. Определение слотов и их допустимых значений,
4. Заполнение значений слотов для экземпляров классов.

Как определить, правильно ли создана онтология?

Для любой предметной области может существовать бесчисленное количество онтологий; ведь каждая новая онтология – это всего лишь еще один из способов структурирования концепций и отношений между ними. Однако существуют несколько простых принципов, которые могут помочь при принятии решений о том, как создавать те или иные онтологии:

- Не может быть только одного способа описания модели предметной области – всегда есть жизнеспособная альтернатива. Лучшее решение почти всегда будет зависеть от того, какая система разрабатывается и от возможных будущих изменений в системе.
- Процесс разработки обязательно должен быть итеративным.
- Концепции в онтологии должны быть максимально близки к объектам (логическим или физическим) и отношениям между ними в интересуемой области знаний. При правильном моделировании, онтология может быть представлена предложениями, где вероятней всего в качестве существительных будут объекты, а отношений – глаголы.

С чего начать?

Для начала необходимо понять, для чего должна быть использована онтология и как примерно выглядел бы ее детальный и общий вариант. Затем среди многих альтернатив вы должны будете выбрать ту, которая будет лучше всего работать для намеченной задачи, а также будет наиболее

интуитивна, расширяема, поддерживаема. При этом необходимо помнить, что онтология представляет предметную область в реальном окружающем мире, и потому понятия в онтологии также должны отражать эту реальность. После того как вы сделаете черновой вариант онтологии, вы можете проверить ее и откорректировать, используя Protégé, или путем обсуждения с экспертами в исследуемой области (как результат, почти наверняка придется пересмотреть черновой вариант). Такой процесс итеративной коррекции, вероятней всего, будет продолжаться на протяжении всего жизненного цикла онтологии.

Разработка простейшей системы

Основные положения

Предположим, что мы хотим разработать систему, которая помогает управлять стоимостью и организацией печатного издания (для простоты можно взять некую газету). Система должна отвечать на следующие вопросы:

- Кто ответственный за каждый раздел в газете?
- Каково содержимое каждой статьи в разделе и кто автор?
- Перед кем отчитывается каждый автор?
- Каково расположение и расходы на каждую статью?

После того как мы определились с идеей, мы можем расписать некоторые из важных положений системы. Сюда могут войти: основные концепции и их свойства, а также отношения между ними. Для начала мы можем просто определить термины, независимо от роли, которую они могут играть в онтологии.

Итак, в любой газете есть разделы. Каждый раздел имеет содержимое, например, статьи, реклама и т.д. и ответственного редактора. У каждой статьи есть автор, который может быть как работником газеты, так и быть приглашенным со стороны. Для каждого автора, работающего в газете, мы хотим знать его имя и зарплату, а также перед кем он отчитывается.

По мере определения понятий, мы неявно определяем рамки нашей онтологии, а именно, что мы должны будем включить в нашу модель, а что нет. К примеру, при начальном рассмотрении термина «работник», мы, возможно, хотели бы включить в это понятие вахтера или водителя грузовика из службы доставки. Однако, подумав, мы могли осознать, что хотим чтобы наша онтология была сфокусирована на производственных затратах, связанных напрямую с тем что, как и где написано в газете. Таким образом, мы решаем не включать вахтера и т.п. в область рассмотрения.

Получив достаточно полный список терминов, мы можем разделить эти понятия по категориям в зависимости от их функции в онтологии. Понятия (концепции/термины предметной области), являющиеся объектами, такие как статья или автор, будут представлены в виде классов. Свойства

классов, такие как содержимое раздела или зарплата, могут быть представлены как слоты, а ограничения на свойства или отношения между классами как грани/аспекты (slot facets).

Определив основные понятия, теперь мы можем показать, как создавать и структурировать их, используя систему Protégé.

Создание проекта

Перед началом работы, вы должны создать новый проект в системе Protégé. Для этого:

1. Запустите Protégé. Если у вас уже открыт проект, просто сохранитесь и перезапустите программу. После того как программа запустилась, появляется диалог приветствия, предлагающий создать новый проект, открыть последний проект или посмотреть документацию.

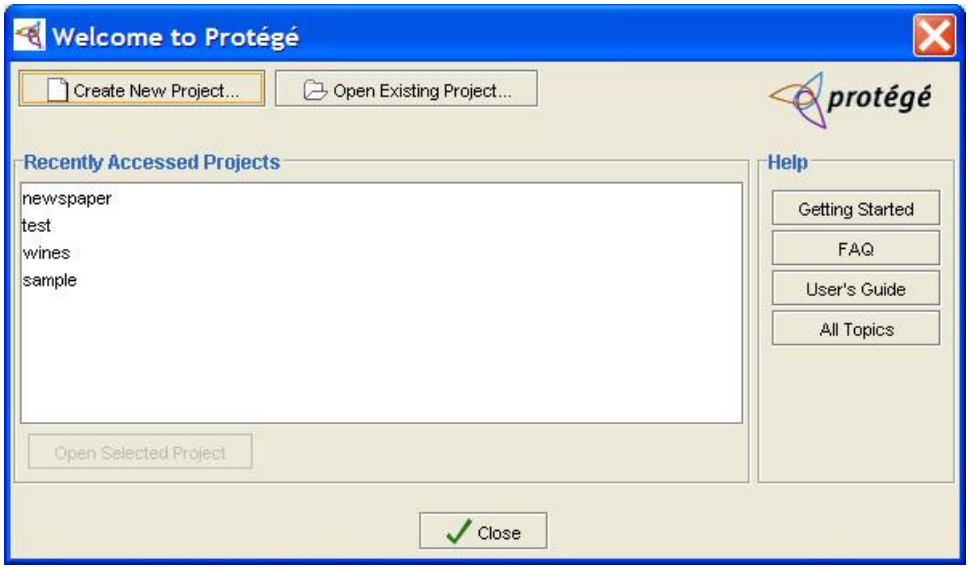

Рисунок 1. Окно приветствия.

2. Щелкните мышкой по кнопке **Create New Project...**. Появится диалоговое окно "Create New Project", позволяющее выбрать тип проекта. Если у вас нет необходимости в специальном формате для ваших файлов, просто нажмите кнопку **Finish** – будет выбран формат файла по умолчанию Protege Files (.pont and pins).

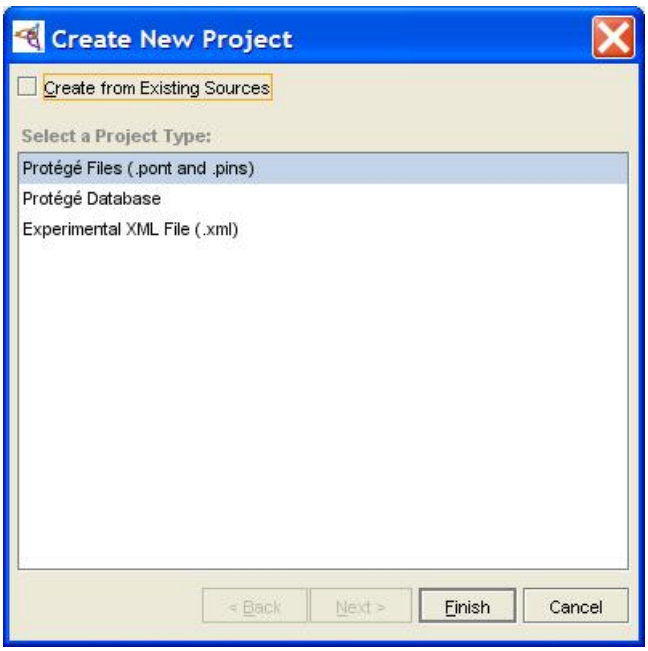

Рисунок 2. Окно создания нового проекта.

3. Откроется окно проекта Protégé. Новый проект всегда открывается в области просмотра классов (Classes view). Видно, что в этой области на данный момент находятся только внутренние системные класс Protégé: THING и SYSTEM-CLASS. Никаких экземпляров классов создано к этому моменту не будет.

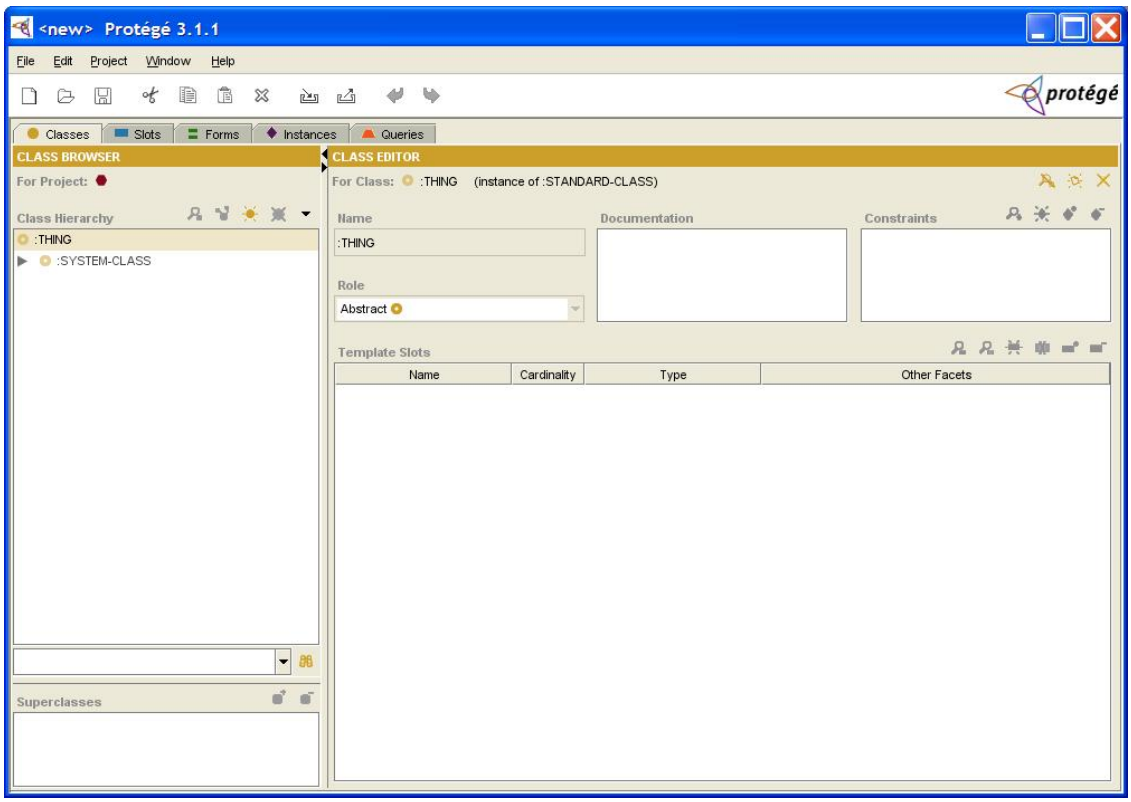

Рисунок 3. Область просмотра классов.

Сохранение проекта

Во время работы с программой вы можете захотеть сохранить промежуточные изменения, для этого:

1. Щелкните кнопку сохранить проект , вы также можете выбрать пункт **Save project** из меню файл **File**.

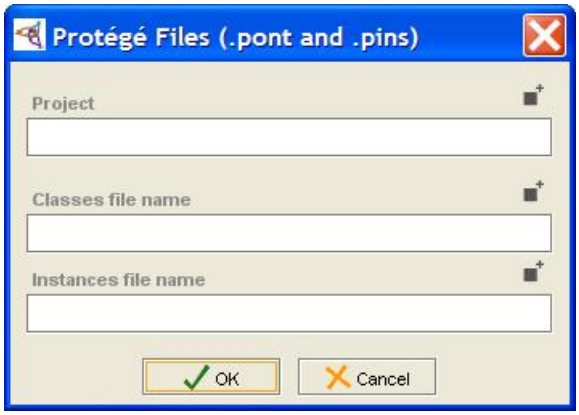

Рисунок 4. Окно сохранения проекта.

2. Для того чтобы указать место, куда будет сохранен Ваш проект, нажмите кнопку чуть выше самой верхней строчки (напротив надписи Project). В открывшемся диалоговом окне, перейдите по нужному пути в файловой системе (или создайте каталог, где будут храниться данные проекта и откройте созданную папку).

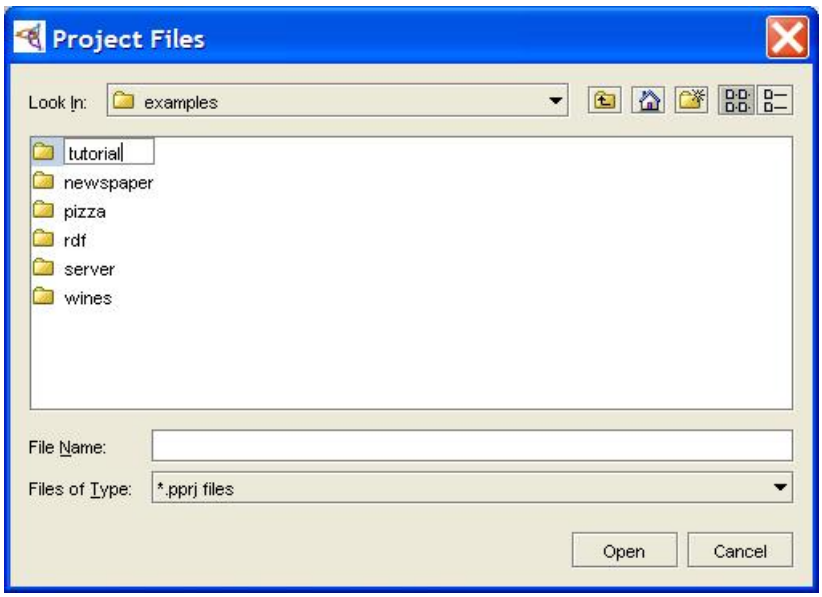

Рисунок 5. Окно выбора имени файла проекта.

3. Введите имя файла проекта (например, "tutorial").

4. Нажмите кнопку **Select**.

5. Вы вернетесь в окно сохранения файлов проекта, нажмите **OK** и проект будет сохранен.

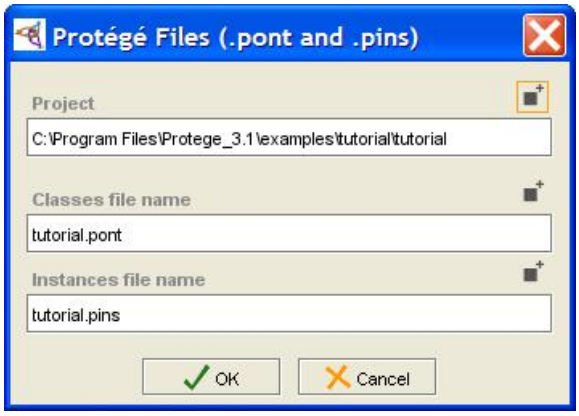

Рисунок 6. Завершение сохранение проекта.

Создание классов

Основное окно программы Protégé состоит из закладок (tabs) которые отображают различные аспекты модели знаний. Наиболее важной закладкой, когда вы только начинаете делать проект, является закладка классов (Classes). Обычно классы соответствуют объектам или типам объектов, в некой предметной области. В нашем примере с газетой, классы будут включать в себя людей, а именно, редакторов, репортеров, агентов по продаже, а также компоненты расположения информации газеты, такие как разделы, кроме того, содержимое газеты (реклама и статьи) будет также представлено в виде объектов.

Классы в Protégé отображаются в виде иерархии наследования (inheritance hierarchy), которая располагается в области просмотра называемой Class Browser (или навигатор классов) в левой части закладки классов. Свойства классов выбранных в текущий момент в навигаторе, будут отображены в редакторе классов справа. Ниже вы узнаете, как создавать классы, подклассы, изменять иерархию классов, создавать абстрактные классы и добавлять дополнительные базовые классы к существующим классам.

Создание класса “Корреспондент”

Мы хотим знать происхождение каждой статьи, и потому мы начнем с задания типов сотрудников или служб, которые могут создавать статьи. Для начала создадим новый класс “корреспондент” (Columnist):

1. Выберите закладку классов.
2. Найдите область в навигаторе классов (Class Browser), где отображается иерархия классов (Class Hierarchy, в окне Protégé слева). Эта область отображает иерархию классов, с выделенным текущим выбранным классом.

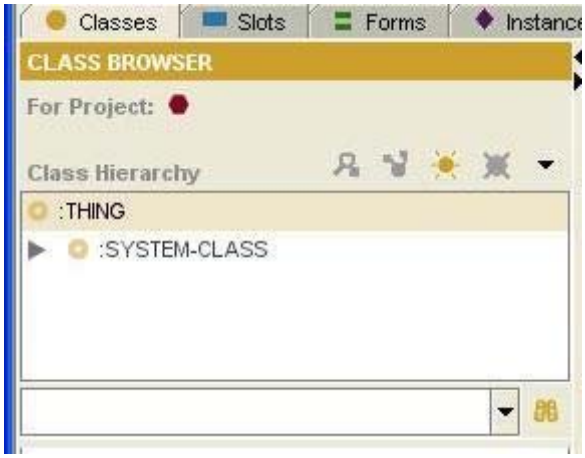

Рисунок 7. Область отображения иерархии классов.

3. Проверьте: по умолчанию класс :THING (вещь, нечто) должен быть выделен. Почти все классы в данном примере будут созданы на уровень ниже класса THING. Другой класс :SYSTEM_CLASS используется для определения структур различных форм Protégé.

4. Нажмите кнопку создать класс (Create Class)  в верхнем правом углу навигатора классов. Новый класс будет создан со стандартным именем (основанном на имени проекта). В нашем случае “tutorial_Class_0”. Вы можете увидеть, что имя нового класса в навигаторе классов после создания будет выделено, для указания того, что этот класс выбран в данный момент.

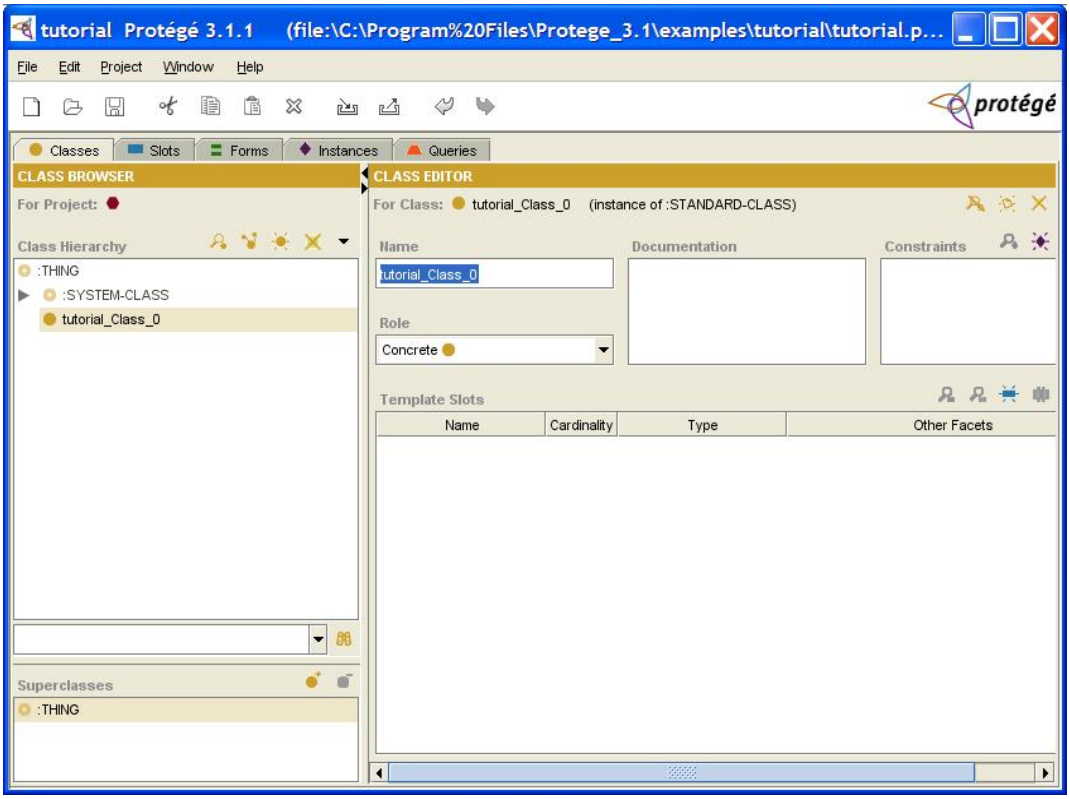

Рисунок 8. Редактор класса.

5. В активном поле редактора классов введите “Columnist”. В системе Protégé приняты правила наименования, когда первая буква в каждом слове в имени класса пишется в верхнем регистре, а остальные буквы в нижнем, при этом слова разделяются символом подчеркивания.
6. Нажмите ввод или щелкните мышью на отображаемый класс, чтобы подтвердить и отобразить свои изменения.
7. Если при изменении имени класса у вас возникли проблемы, посмотрите в панель редактора классов справа в главном окне Protégé. Стандартное имя нового класса должно быть отображено и выделено в поле Name. Если правильное стандартное имя отображается, но не выделено, просто щелкните на поле **Name** мышкой, для того чтобы его отредактировать. Если имя неправильное, тогда скорей всего вы выбрали неверный класс в области отображения иерархии классов (Class Hierarchy), щелкните на нужном классе.

Создание класса “Автор”

Автором может быть любой возможный источник информации для статьи, такой как новостная служба или корреспондент. Для того чтобы создать класс автор:

1. Выделите класс :THING. Если вы этого не сделаете, то вы создадите класс, который будет подклассом класса Columnist.
2. Нажмите кнопку создать класс (Create class)  и наберите с клавиатуры имя класса: Author.
3. Нажмите ввод для завершения создания класса.

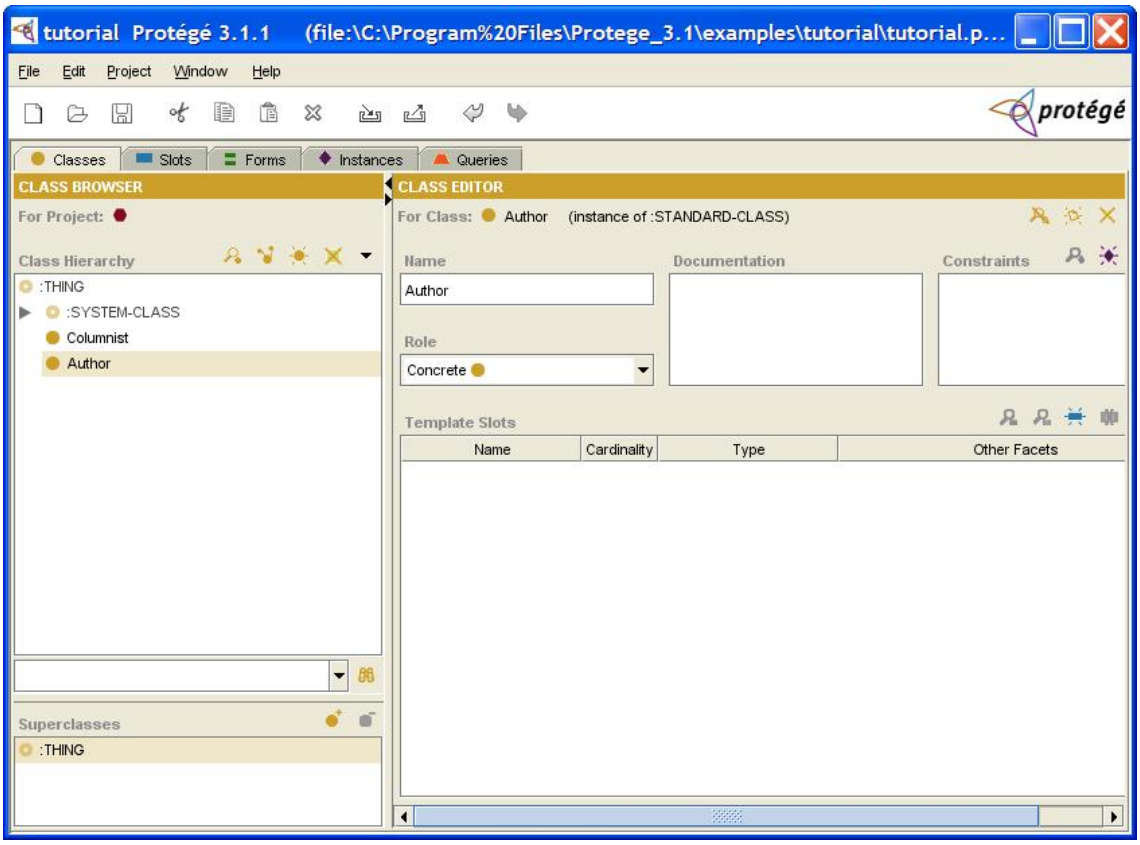

Рисунок 9. создание класса "Автор".

Создание подклассов класса “Автор”

Теперь мы хотим добавить больше источников статей (служба новостей и редактор), которые мы создадим как подклассы класса автор (Author).

1. Выберите класс “Author” в области отображения иерархии классов.
2. Нажмите кнопку “создать класс” (Create class) и переименуйте новый класс в News_Service (служба новостей). Помните, что когда бы вы не создавали новый класс, он будет создан как подкласс текущего выбранного класса. Также заметьте, что когда вы создаете первый

подкласс класса, то иконки ▼или ▶ появляются слева от него. Вы можете использовать эти иконки для того чтобы показать или скрыть подклассы класса. Продолжая разрабатывать онтологию, создадим еще один подкласс класса Автор (Author).

3. Выберите класс Автор (Author) в навигаторе классов (Class Browser).
4. Нажмите кнопку **Create Class** и переименуйте созданный класс в **Editor** (редактор).

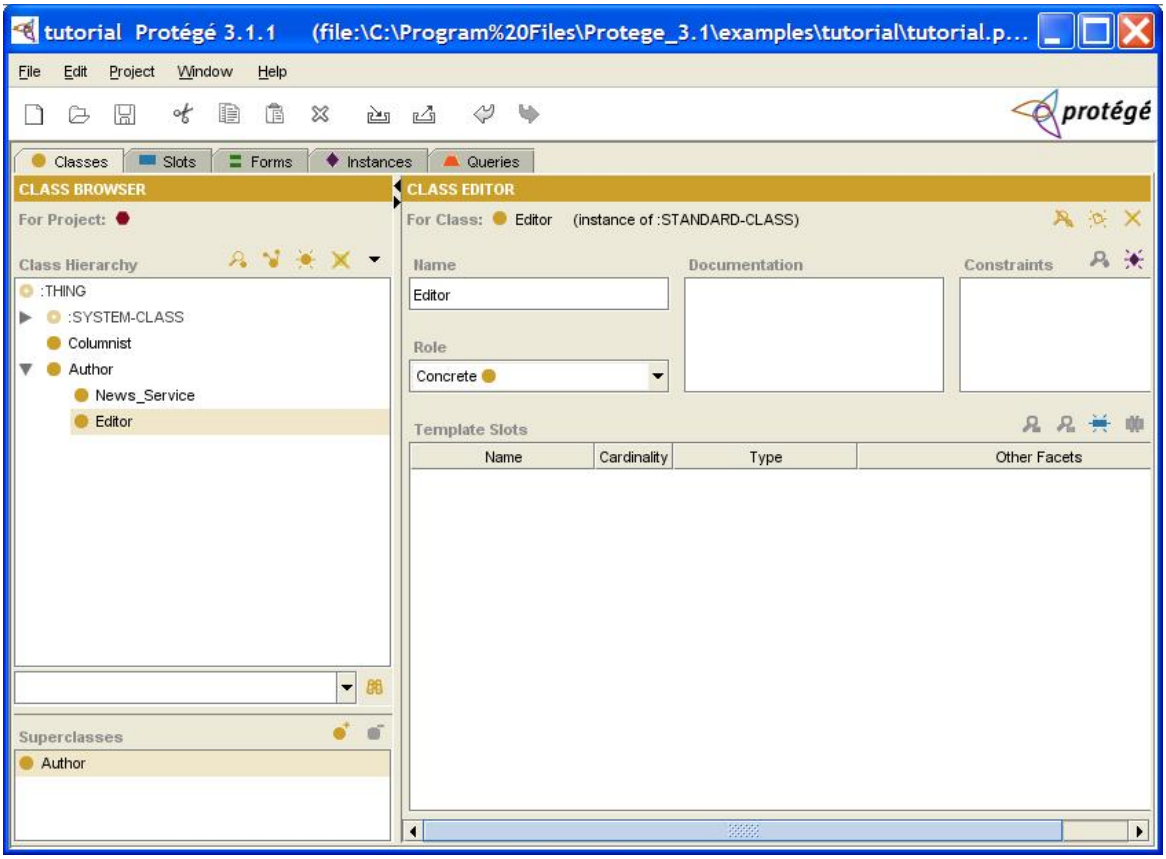

Рисунок 10. Создание класса "Редактор".

Изменение иерархии классов

На данной стадии разработки, можно заметить что “Автор” (Author) и “Корреспондент” (Columnist) находятся на одном уровне в создаваемой иерархии, в то время как “Служба новостей” (News_Service) и “Редактор” (Editor) являются подклассами класса Автор (Author). С точки зрения концепции и служба новостей, и редактор, и корреспондент, все могут являться авторами (источниками) статей, т.е. понятие “Автор” является общим для всех этих трех классов. Значит, текущее расположение в иерархии противоречит принципам хорошо спроектированных онтологий - все классы,

имеющие одно и то же более общее понятие, должны находиться на одном уровне в иерархии.

Таким образом, мы хотим модифицировать нашу иерархию, чтобы класс “Корреспондент” (Columnist) стал подклассом класса Автор (Author), правильно отражая структуру предметной области.

Для этого необходимо сделать следующее:

1. Щелкните на классе Columnist (Корреспондент) и перетащите (drag-n-drop) его на класс Author (Автор).

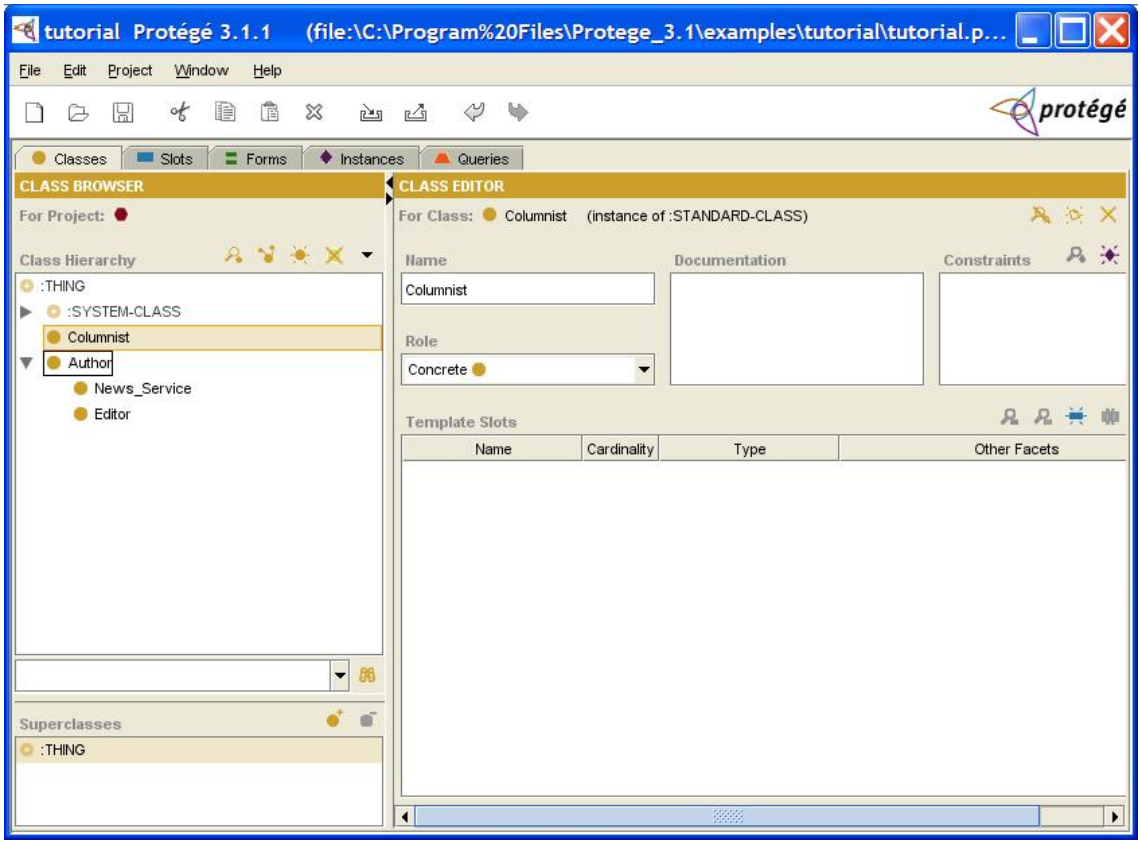

Рисунок 11. Изменение иерархии классов.

2. Класс **Columnist** будет удален с предыдущего места в иерархии и создан в новом, но уже как подкласс Author.

В данном случае, ошибка была введена явно, для примера. Однако, при создании собственных онтологий, в процессе разработки могут открываться отличия или сходства между классами, которые не были явно видны в начале и вам вероятней всего придется часто использовать перемещение, создание, удаление классов, для того чтобы создавать иерархии, оптимально отражающие положение вещей в предметной области.

Создание абстрактных классов

В системе Protégé классы могут быть как конкретными (Concrete), на основе таких классов система может непосредственно создавать готовые экземпляры, так и абстрактными, у таких классов не может быть экземпляров. По умолчанию, при создании класса выбирается тип класса как “конкретный класс”. Так как в нашей системе понятие автора является скорее обобществляющим, нежели связанным с какой конкретной сущностью, мы делаем вывод что класс Автор (Author) не может сам по себе иметь экземпляров без более детального определения (к примеру, является ли автор службой новостей или корреспондентом). Поэтому мы решаем сделать класс Автор (Author) абстрактным.

1. Выберите класс “Author” в области иерархии классов (Class Hierarchy). Далее в редакторе классов (справа от навигатора классов), найдите меню Роль (Role), оно должно находиться прямо под именем класса.

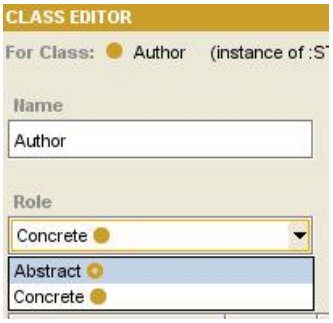

Рисунок 12. Выбор меню Role.

2. Щелкните по списку в меню Role и выберите “Abstract”.
3. Заметьте, что когда вы меняете роль класса, иконка в перед именем класса меняется на . Такая иконка означает, что класс является абстрактным (соответственно означает конкретный класс).

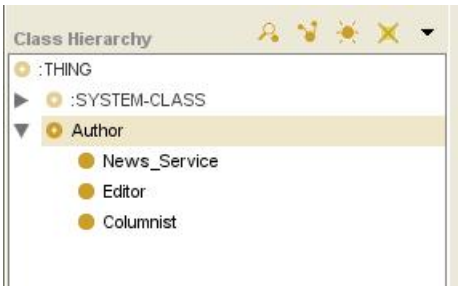

Рисунок 13. Маркировка абстрактных классов.

Создание класса “Работник”

Теперь настало время создать класс “работник” (Employee). Этот класс подразумевает под собой любого работника газеты, независимо от того является ли он автором или нет. Заметим, однако, что нам интересны только те работники, которые как-то привлечены к созданию и управлению непосредственным содержимым газеты. Здесь стоит еще раз отметить, что одним из выборов который делает разработчик при проектировании онтологий, является отсечение лишних сущностей. Хотя изначально может казаться, что многие классы являются важными, все же необходимо следить затем, чтобы создаваемая иерархия не становилась слишком сложной. Применительно к нашему случаю, это означает, что хотя технически вахтер и является служащим компании, мы не будем создавать такой подкласс.

1. Выберите класс :THING в навигаторе классов. Несмотря на то, что некоторые авторы являются служащими газеты, мы не хотим, чтобы класс “Employee” (работник) был подклассом класса Автор (Author).
2. Нажмите кнопку **Create Class** и переименуйте вновь созданный класс в **Employee**.

Рисунок 14. Добавление класса Employee.

Добавление дополнительного базового класса к существующему подклассу

Как было упомянуто выше, мы хотим чтобы “корреспондент” стал “работником”. Но поскольку мы уже создали такой класс, мы не хотим создавать его снова как подкласс класса “работник”. Вместо этого мы можем сделать так, чтобы существующий класс “корреспондент” (Columnist) стал подклассом “Employee” (работника). Для этого нужно сделать следующее:

1. Выберите класс Columnist (корреспондент) в навигаторе классов.

Рисунок 15. Выделение класса в навигаторе иерархии классов.

2. Найдите панель базовых классов (Superclasses) в нижней левой части окна системы Protégé (под навигатором классов). Заметьте, что когда выбран класс Columnist (корреспондент), его базовый класс (Автор) отображается в панели Superclasses.

Рисунок 16. Панель базовых классов.

3. Нажмите кнопку **Add Superclass** (добавить базовый класс), в верхнем правом углу панели базовых классов (Superclasses). Появится диалоговое окно, отображающее все классы, созданные на тот момент времени, в виде иерархии.

Рисунок 17. Выбор базового класса.

4. Выберите класс “Employee” (работник) и нажмите **OK**. Теперь класс “корреспондент” (Columnist) имеет два базовых класса (Автор и Работник). Оба класса показаны в панели базовых классов.

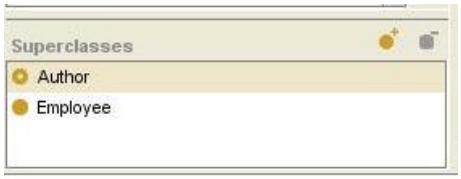

Рисунок 18. Обновленный список базовых классов.

5. Заметьте, что рядом с именем класса “Employee” появилась иконка ▶. Щелкните по ней, для того чтобы развернуть дерево подклассов и увидеть детей класса “работник” (Employee). Как видим, класс “корреспондент” теперь присутствует в двух местах в навигаторе классов: как подкласс класса Автор (Author) и еще раз как подкласс класса “Работник” (Employee).

Рисунок 19. Обновленная иерархия классов.

Добавление базового класса с помощью перетаскивания (drag-n-drop)

Существует еще одна возможность задания базового класса – при помощи механизма перетаскивания (drag-n-drop):

1. Выберите класс Редактор (Editor) в навигаторе классов.
2. Удерживая нажатой левую кнопку мыши, перетащите класс Editor (редактор), так чтобы он находился над классом Employee (работник). Класс Employee автоматически будет выделен.
3. Перед тем как отпустить кнопку мыши, нажмите дополнительно клавишу Ctrl, затем отпустите кнопку мыши, для того чтобы перенести класс.

Рисунок 20. Добавление базового класса с помощью перетаскивания.

Для удаления базового класса от некоторого подкласса, выберите класс, который должен быть удален в панели отображения базовых классов (Superclasses), и нажмите кнопку удаления базового класса (**Remove Superclass**).

Теперь вы готовы к тому чтобы добавить несколько атрибутов (свойств) к созданным классам. В следующей секции будет показано, как это можно сделать с помощью механизма слотов (slots).

Создание слотов

Как вы могли убедиться выше, в системе Protégé под классами понимаются конкретные понятия (концепции) предметной области, такие как

редактор или корреспондент. В то же время классы это больше чем объекты, объединенные в иерархию. Они также могут иметь атрибуты (свойства), к примеру, имя, номер телефона или уровень зарплаты и отношения между ними, такие как Автор Статьи.

Атрибуты и отношения класса описываются конструкцией под названием слот. В данном разделе будет показано, как создавать слоты, привязывать слоты к классам, описывать отношения между классами, а также будет описан механизм наследования слотов.

Создание слота (используя закладку слоты (Slots tab))

Для создания слота есть несколько способов. Один из них – это создать слот используя закладку “Slots”, а затем связать его с одним или более классами. Вернемся к нашему примеру, для того, чтобы создать слот name, используя закладку Slots, необходимо:

1. Щелкнуть на закладку Slots. Заметьте, что расположение элементов управления на закладке слотов, схоже с закладкой классов, а именно, готовые слоты отображаются слева в области просмотра, а редактирование слотов возможно с помощью редактора (справа).

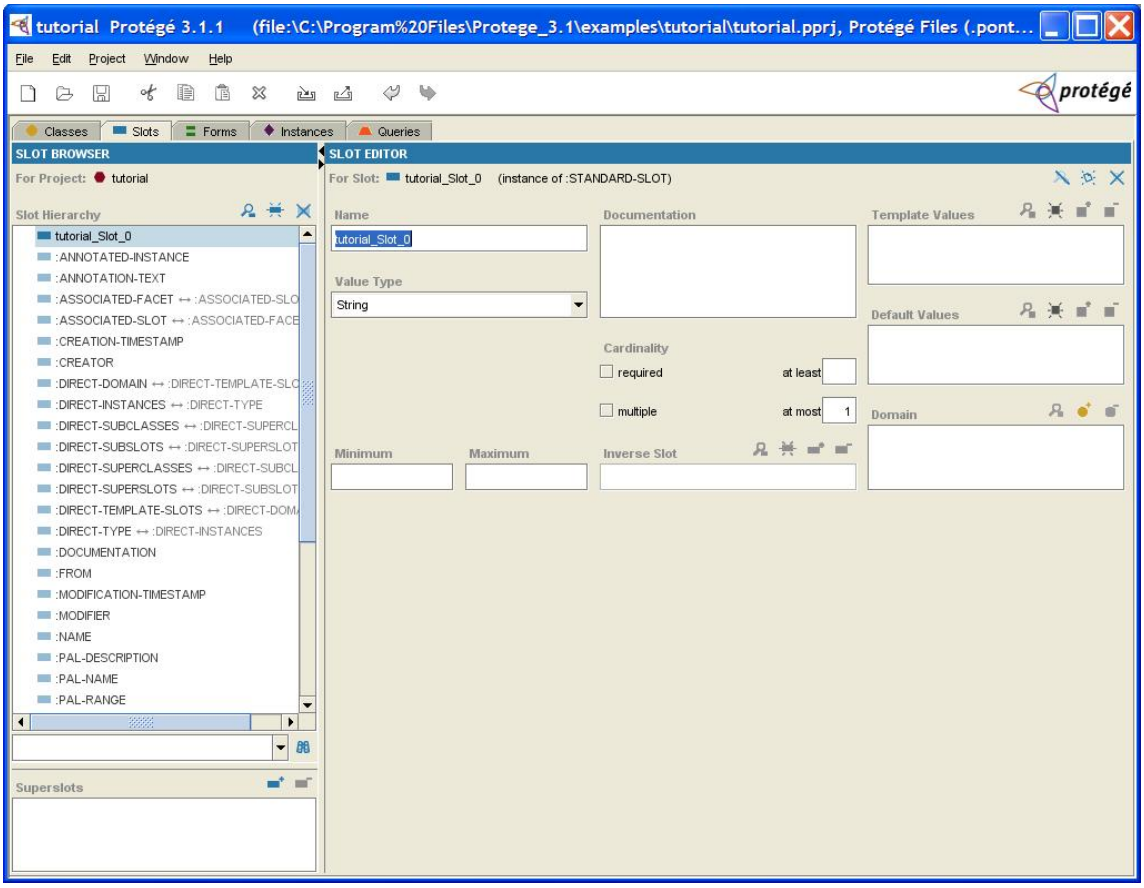

Рисунок 21. Добавление слота.

2. Нажмите кнопку создать слот (Create Slot ) в правом верхнем углу панели отображения иерархии слотов (Slot Hierarchy). Будет создан новый слот. Также как и при создании класса, ему присваевается стандартное имя, в нашем случае “tutorial_Slot_0” (имя будет автоматически выделено, после создания слота).
3. Перед переименованием слота, убедитесь, что стандартное имя выделено в редакторе слота. Наберите новое имя слота (в нашем случае name). Рекомендованные правила наименования, таковы, что имя слота должно быть написано в нижнем регистре, при этом разные слова разделяются подчеркиванием. Такое наименование (классы с большой буквы, слоты с маленькой) помогает отличить классы от слотов в созданной онтологии.
4. Заметьте, что слот имеет по умолчанию тип значения String (строка). Тип накладывает ограничения на то, какие значения может принимать слот. Строковый слот, к примеру, может принимать в качестве значений алфавитно-цифровые строки (включая пробелы).

Для этого простого слота мы не будем менять никаких аспектов/граней (facets) в редакторе слотов.

Связывание слота с классом

Все, что мы пока сделали, это определили общий атрибут **name (имя)**. Для того чтобы действительно задействовать его в нашей онтологии, мы должны привязать его к классу. К примеру, мы хотим, чтобы каждый из экземпляров подклассов класса Автор имел имя.

Вернемся к закладке классов и откроем на редактирование класс Автор (Author). Любой из атрибутов, который вы создаете или связываете с классом, будет отображаться в редакторе классов, справа от навигатора классов. Мы уже использовали редактор классов для смены имени нового класса, а также для изменения роли класса Автор. Теперь мы будем использовать редактор классов для просмотра и именования слотов. Для того чтобы связать слот name с классом:

1. Щелкните на закладке классов.
2. Выделите класс Автор в панели отображения иерархии классов (Class Hierarchy). Посмотрите на редактор классов (справа), в этой области отображается поле имя, роль класса, а также документация и ограничения (constraints). Под этими полями расположена, панель шаблонов слотов (Template slots), которая занимает всю оставшуюся нижнюю часть редактора классов. Эта область показывает слоты, связанные с классом. На текущий момент она пуста.

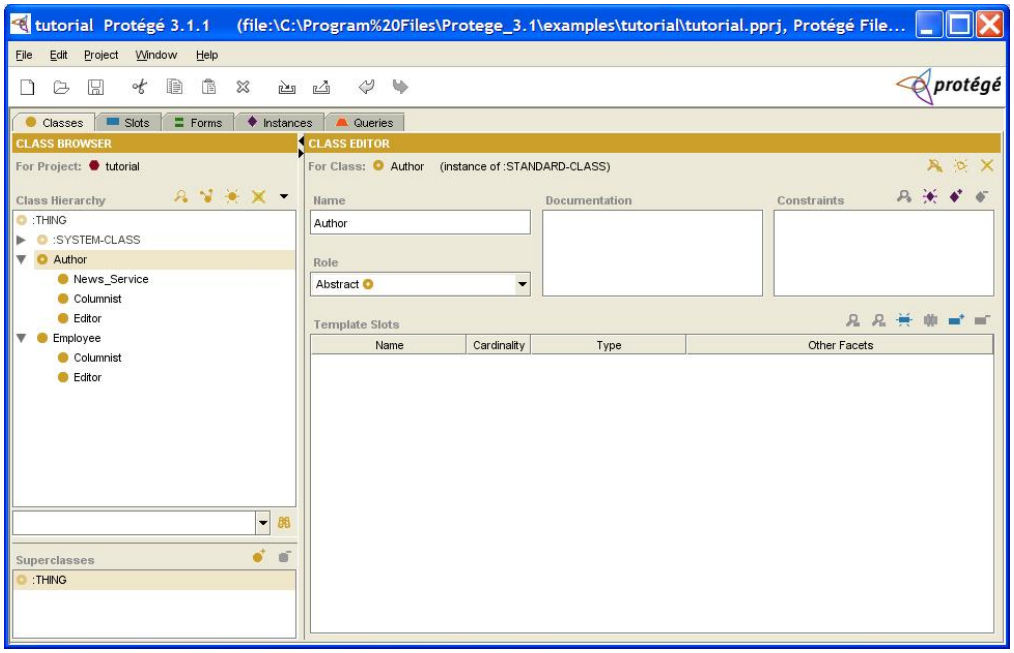

Рисунок 22. Связывание слота с классом.

3. Для добавления слотов к классу, нажмите кнопку **Add Slot** . Кнопки управления слотами находятся в верхнем правом углу панели шаблонов слотов (Template slots).

Рисунок 23. Панель инструментов шаблонов слотов.

4. После того как вы нажмете кнопку, появится диалог выбора слота, в котором будет отображен список всех доступных слотов в вашем проекте (в алфавитном порядке, за исключением системных классов Protégé, которые будут видны в самом низу списка).

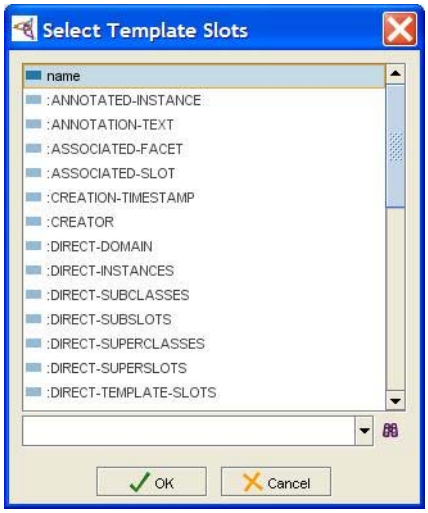

Рисунок 24. Список доступных слотов.

5. Выберите name и нажмите **OK**.

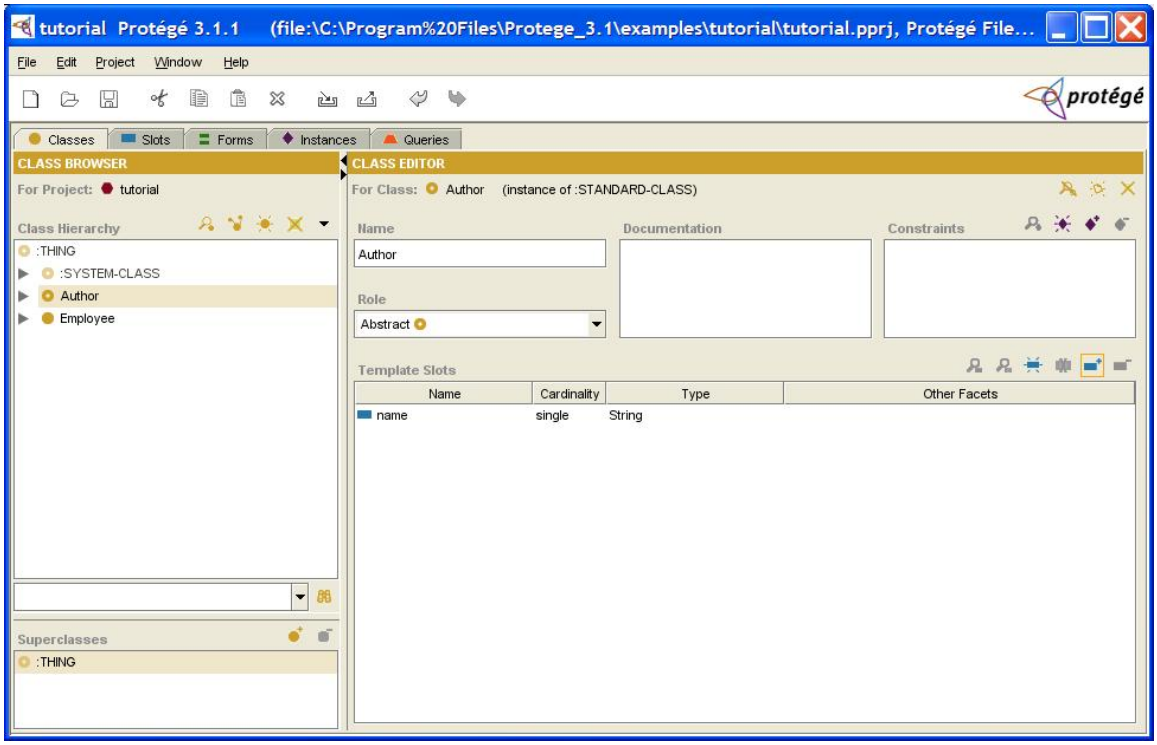

Рисунок 25. Новый слот в списке шаблонов.

Если вы посмотрите теперь на панель шаблонов слотов (Template slots), то увидите, что слот name был добавлен в список, и вместе с ним отображаются его свойства, в нашем случае это мощность (количество элементов типа) и сам тип (строка, String).

Создание слота из закладки классов

Переключение между закладками классов и слотов может показаться утомительным. И так как слоты есть свойства класса, их можно создавать проще, непосредственно с закладки классов.

Попытаемся создать слот для класса Employee, для этого:

1. Выберите класс Employee в панели иерархии классов.
2. Нажмите кнопку создать слот (Create Slot), в правом углу панели шаблонов слотов, будет вызвано окно добавления слота:

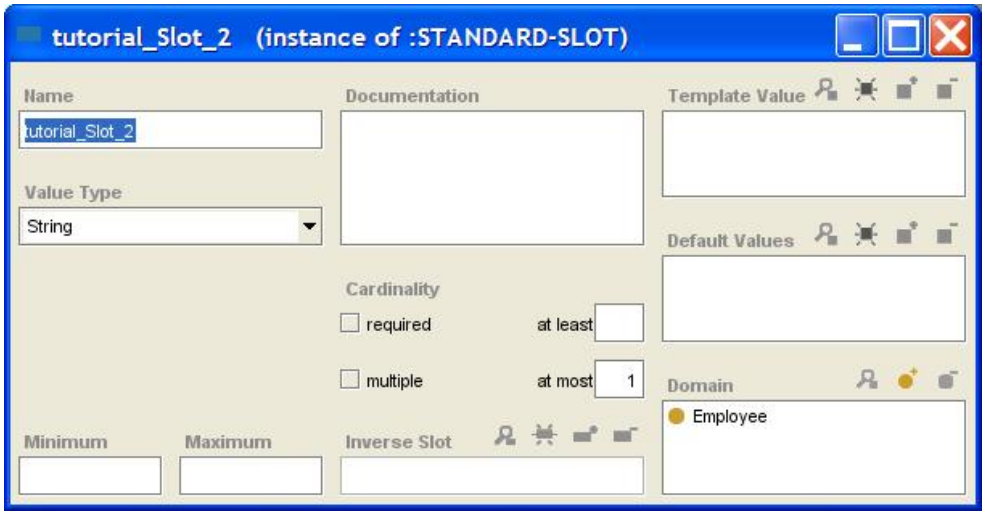

Рисунок 26. Окно создания слота.

3. Наберите salary (зарплата) в поле имя (Name), нажмите ввод.
4. Вернитесь в главное окно (при этом не обязательно закрывать окно редактирования, т.е. можно оставить его открытым и вернуться туда позже для редактирования свойств слота). Заметьте, что теперь новый слот показывается в панели шаблонов слотов, когда выбран класс “Работник” (Employee).

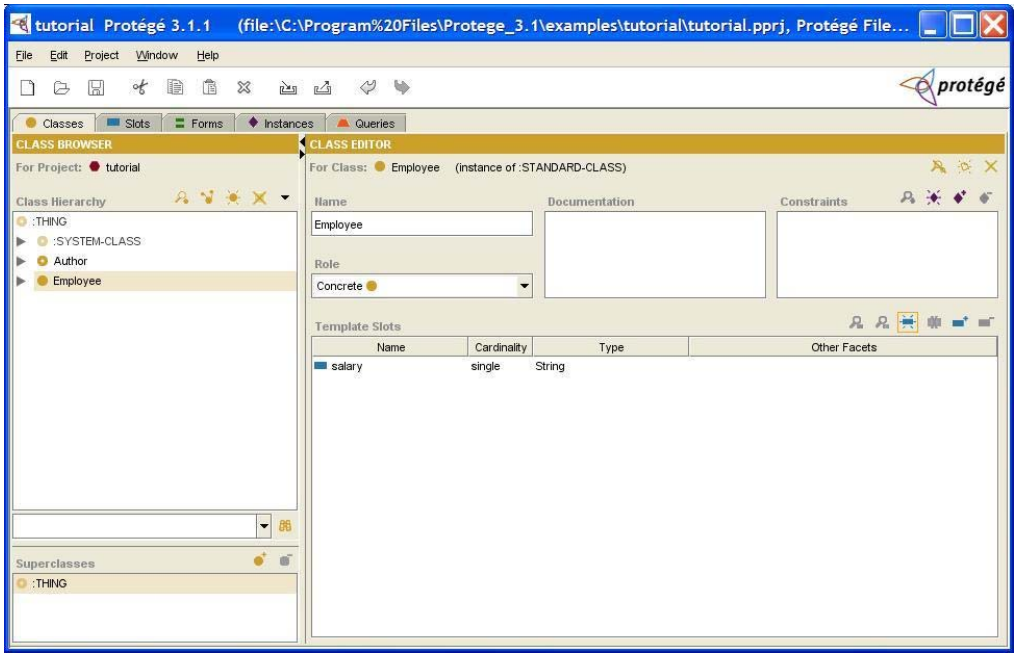

Рисунок 27. Новый слот в списке шаблонов.

Слоты и наследование

Мы не должны добавлять слот **name** (имя) к любому классу, где мы хотим его видеть. В смысле того, что любой подкласс класса автоматически наследует все слоты базового класса. К примеру, если вы выберите класс Служба новостей (News_Service), то увидите, что:

- Слот **name** (имя) уже связан с этим классом через механизм наследования.
- При этом иконка для слота отличается от той, которая использовалась для класса Автор (Author), а именно, для наследованных слотов используется иконка .

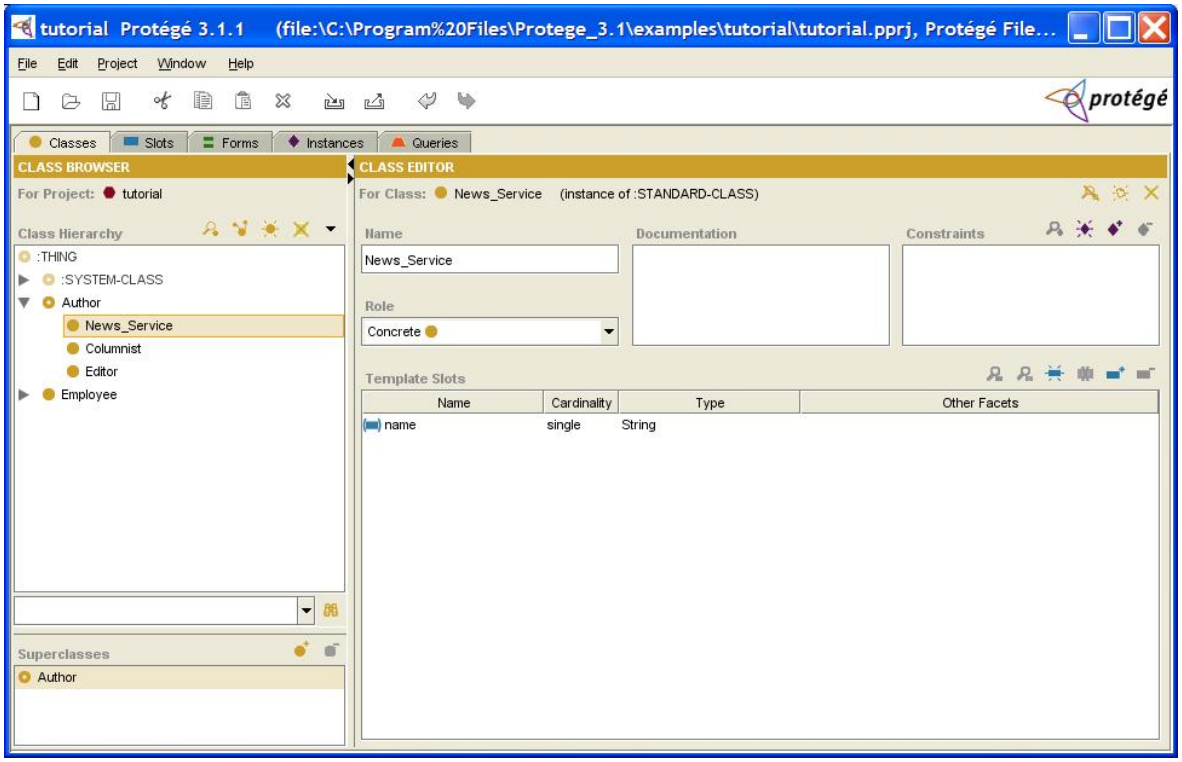

Рисунок 28. Отображение наследованных слотов.

Подклассы более чем с одним базовым классом наследуют слоты от всех базовых классов. К примеру, если Вы выберите класс Editor (редактор), то увидите, что он наследует слот **name** (имя) от Автора, и слот зарплата (salary) от Работника. Множественное наследование одна из основополагающих возможностей Protégé.

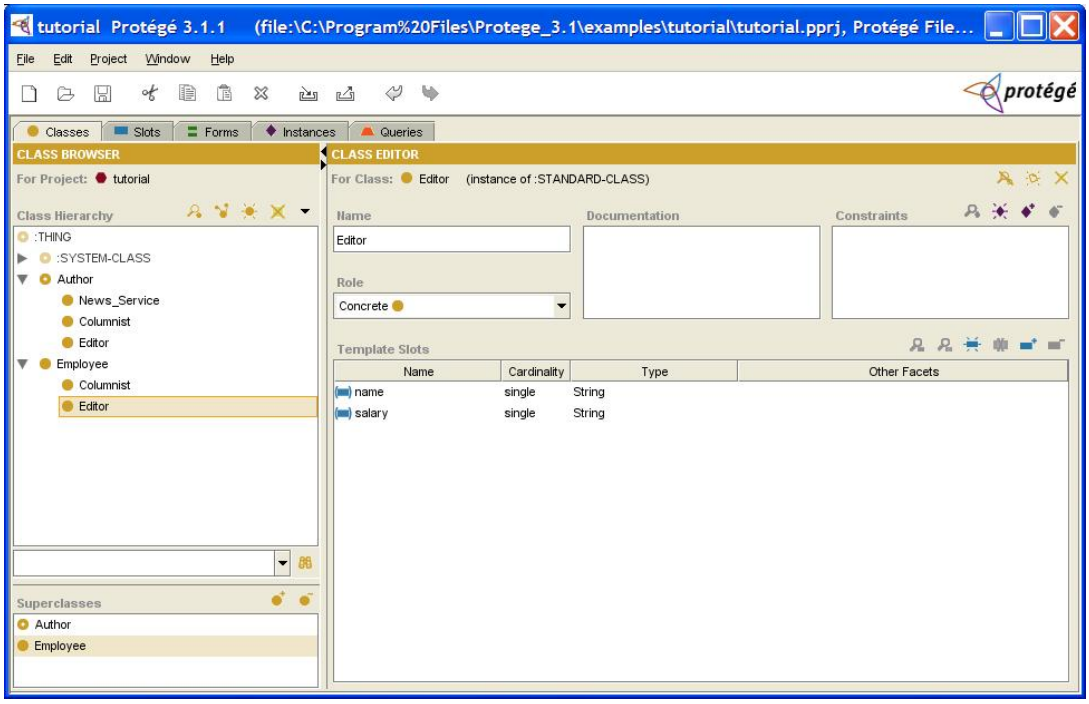

Рисунок 29. Пример множественного наследования.

Создание аспектов/граней (facets) слота

Слоты, которые были созданы на предыдущем шаге, очень простые. Однако, слоты сами по себе, тоже могут иметь свойства. К примеру, зарплата всегда является числом. Вы также можете использовать слоты для задания отношений между классами. Свойства слота, называемые аспектами/гранями (facets), могут быть созданы, как на закладке классов (используя диалог спецификации слота), так и на закладке слотов (используя редактор слота).

Создание аспектов слота “зарплата”

Мы можем определить несколько аспектов для слота “зарплата”, который был создан ранее.

1. Выберите класс “работник” (Employee) в панели иерархии классов.
2. Щелкните два раза на слоте “зарплата” в панели шаблонов слотов (Template slots), для того чтобы открыть форму выбора вида слота. Когда вы редактируете слот, Вы можете выбрать, будут ли изменения применяться к слоту и всем классам, связанным со слотом (вверх по

иерархии до самого верхнего класса), или вы просто хотите чтобы изменения коснулись текущего класса и всех его детей.

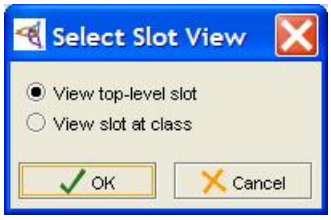

Рисунок 30. Выбор формы отображения слота.

3. В нашем случае, мы хотим просмотреть и отредактировать слот верхнего уровня. Потому убедитесь, что режим просмотра слотов верхнего уровня (**View top-level slot**) выбран и нажмите **OK**. При этом изменение определения слота будет затрагивать всю онтологию.

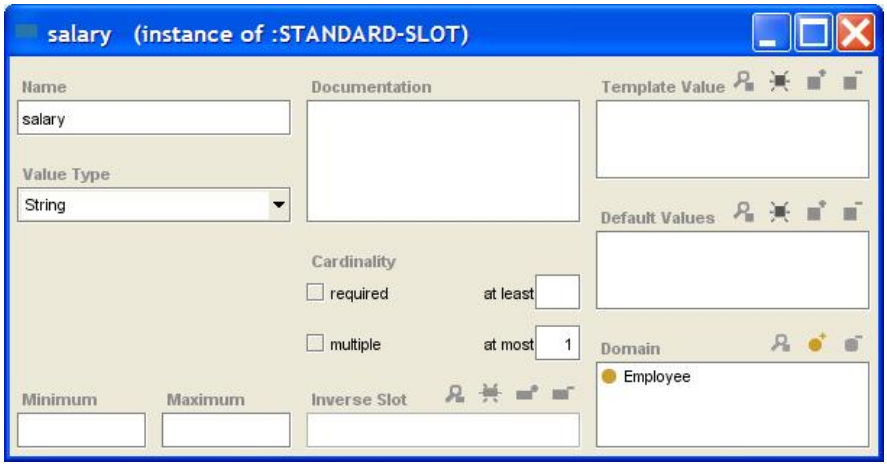

Рисунок 31. Редактирование слота salary.

4. В открывшейся форме редактирования слота, выберите Float из списка выбора типа значения (Value Type). Теперь при создании экземпляров, можно будет вводить для этого слота только правильные значения в формате с плавающей запятой.

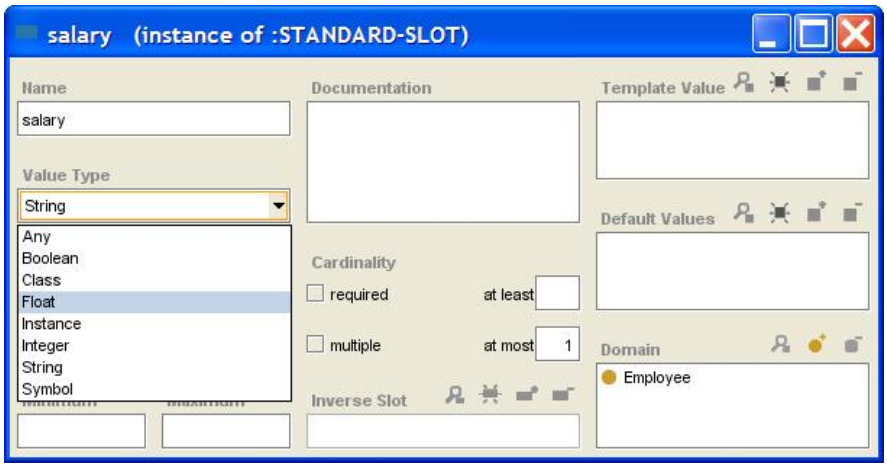

Рисунок 32. Выбор типа значения.

5. Введите 0 (ноль) в поле Minimum (минимальное значение). Таким образом , мы можем быть уверены, что теперь любое значение для поля “зарплата” будет не отрицательным.

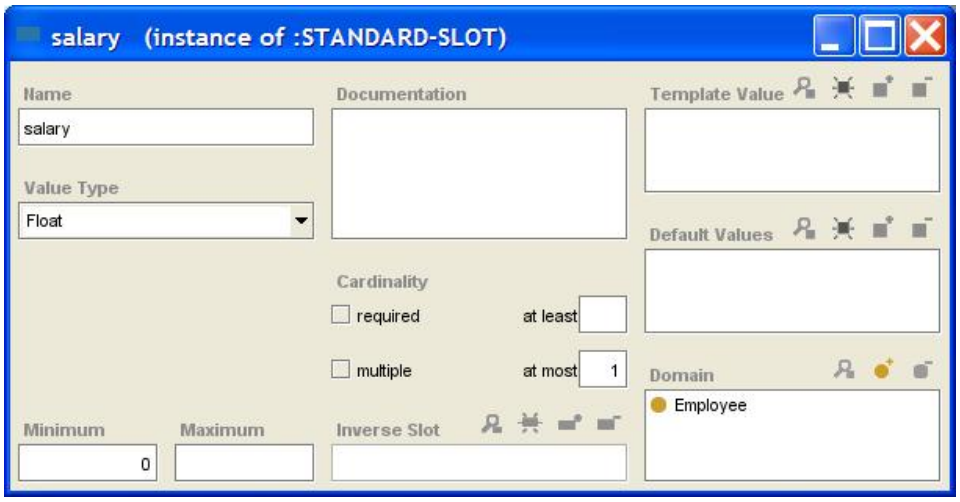

Рисунок 33. Ввод минимального значения.

6. Закройте, диалог редактирования слота, и Вы сможете увидеть, что описание слота в панели шаблонов слотов изменилось. В колонке тип теперь указан Float а минимальное значение = 0 появилось в колонке Other facets (другие аспекты).

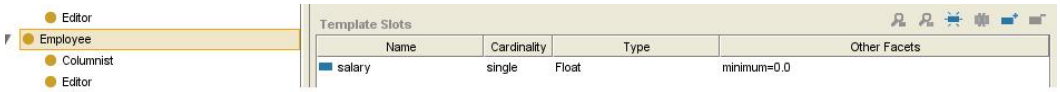

Рисунок 34. Обновленное описание слота.

Создание отношения между классами

Система Protégé также позволяет вам создавать слоты, которые могут быть использованы для описания отношений между классами, которые не определены в иерархии классов. Для этого существуют слоты Instance (экземпляр) или Class (класс). К примеру, “Редактор” (Editor) может быть ответственным за одного или более работников. Мы можем создать новый слот, который бы описывал связь между “Редактором” и “Работником”:

1. Выберите класс “Редактор” (Editor) в навигаторе классов.

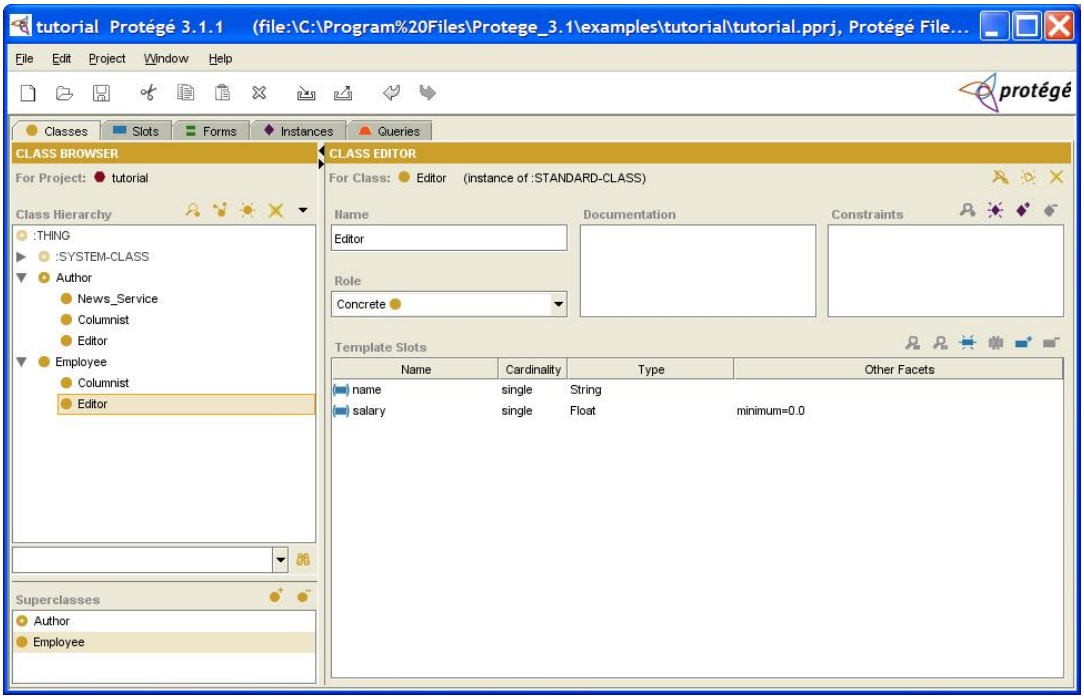

Рисунок 35. Редактирование класса Editor.

2. Нажмите кнопку **Create Slot** для того чтобы создать и связать новый слот с классом “Редактор” (Editor).

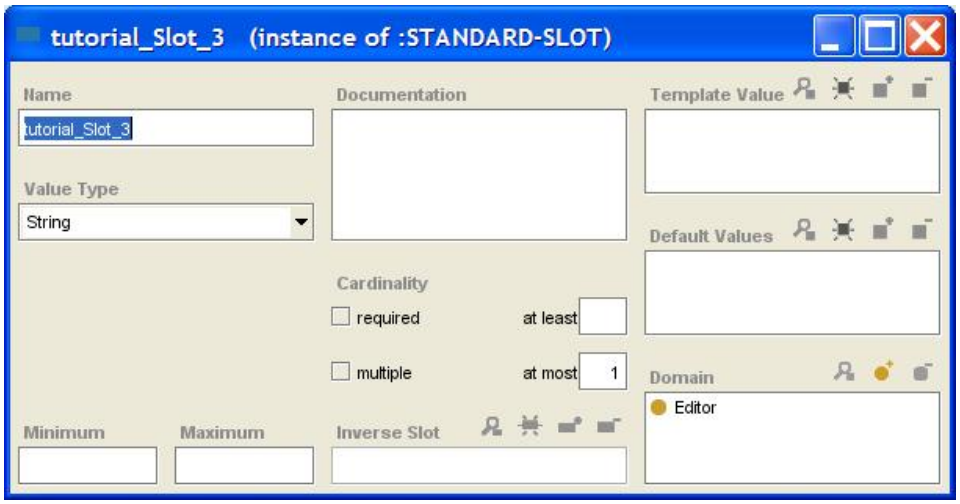

Рисунок 36. Создание нового слота.

3. В открывшейся форме редактирования, наберите в поле имя (Name) **responsible_for** (ответственный за).

Рисунок 37. Поле для ввода имени слота.

4. Выберите Instance (экземпляр) из списка типов значений (Value Type).

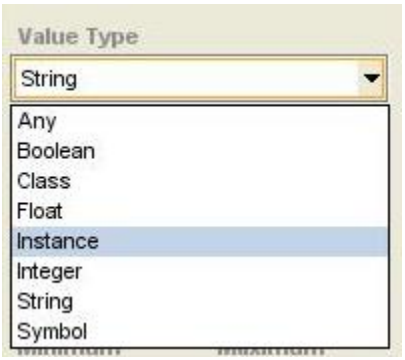

Рисунок 38. Список типов значений слота.

Новое поле, доступные классы (allowed classes), будет отображено под меню тип значения (Value Type).

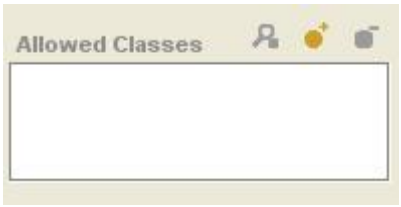

Рисунок 39. Панель Allowed classes.

5. Нажмите кнопку **Add Class** (справа сверху на панели Allowed classes) . Появится окно выбора классов, где будут показаны все классы проекта. Выберите класс “Работник” (Employee) и нажмите **OK**.

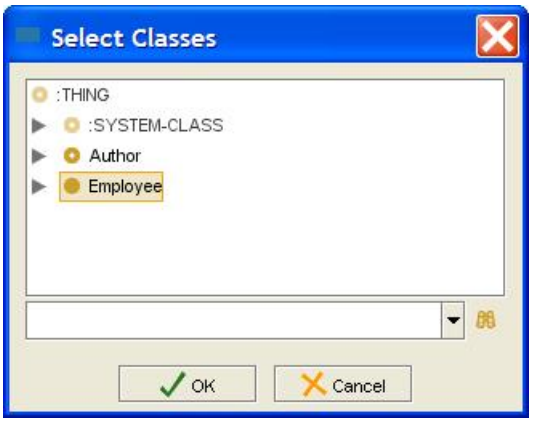

Рисунок 40. Выбор классов из списка.

6. Чтобы разрешить редактору, быть ответственным более чем за одного сотрудника, поставьте галочку в пункте multiple, в панели мощности (cardinality).

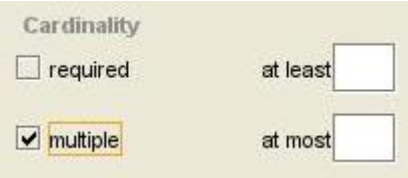

Рисунок 41. Панель cardinality.

После завершения шагов 1..6, слот форма для **responsible_for** будет выглядеть следующим образом:

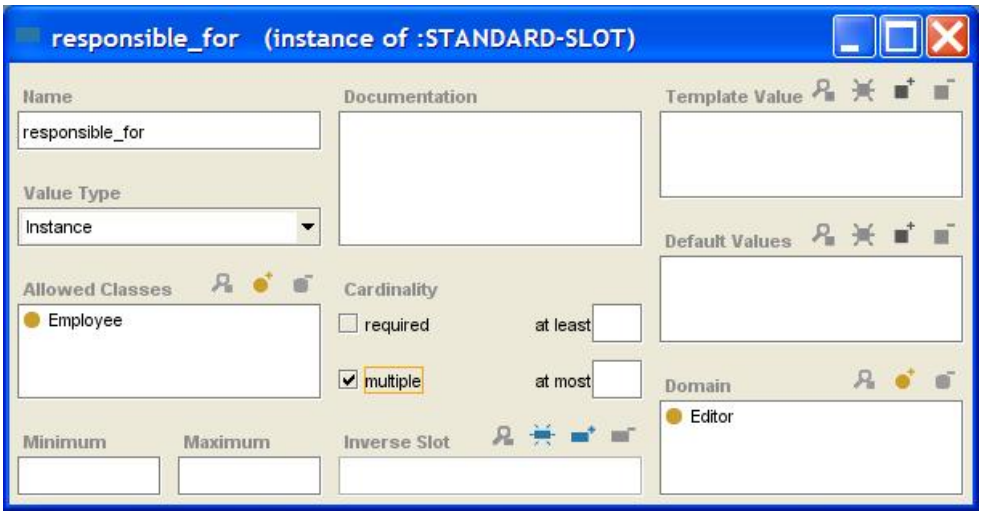

Рисунок 42. Окончательный вид редактора слота responsible_for.

Что же мы создали? Мы создали слот, который может содержать один или более экземпляров класса “работник” в качестве значения. Позднее, когда мы будем создавать экземпляры класса “Редактор” и захотим указать, за каких работников он несет ответственность, мы сможем выбрать один или более экземпляров класса “работник”, чтобы заполнить **responsible_for** слот.

Создание экземпляров классов

Экземпляры классов – это и есть собственно данные вашей базы знаний. Вообще, хорошим правилом, перед вводом конечных данных, является окончательная проверка структуры проекта, потому что когда данные будут введены, необходимость изменения структур проекта может повлечь за собой потерю уже введенной информации. Кроме того, при добавлении новых слотов, необходимо заполнять их значения для старых экземпляров классов.

В этой секции, мы создадим два экземпляра класса редактор:

1. Перейдите на закладку экземпляров (instances). Закладка имеет три панели. Первая, слева, отображает иерархию классов. Средняя панель,

которая сейчас пуста, показывает список экземпляров, созданных для конкретного класса. Третья панель показывает редактор экземпляра класса, где вы можете ввести значения слотов текущего выбранного класса.

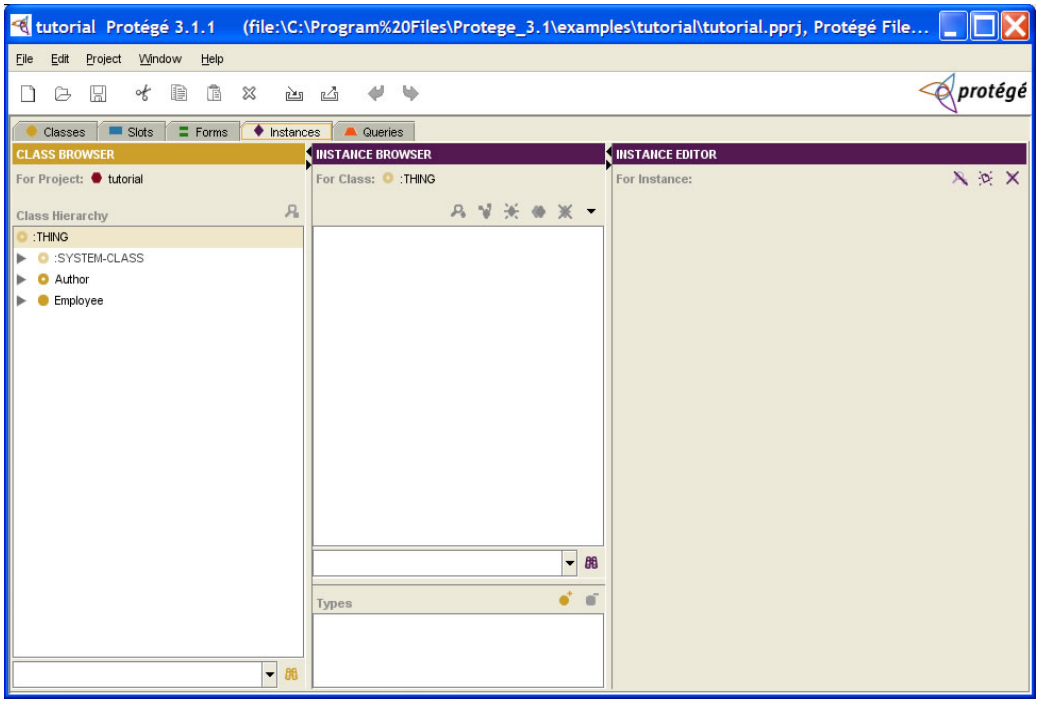

Рисунок 43. Закладка экземпляров классов (instances).

2. Раскройте список подклассов класса “работник” (Employee).
3. Выберите класс редактор (Editor). Кнопка **Create Instance** станет активной, означая, что можно создать экземпляр класса.

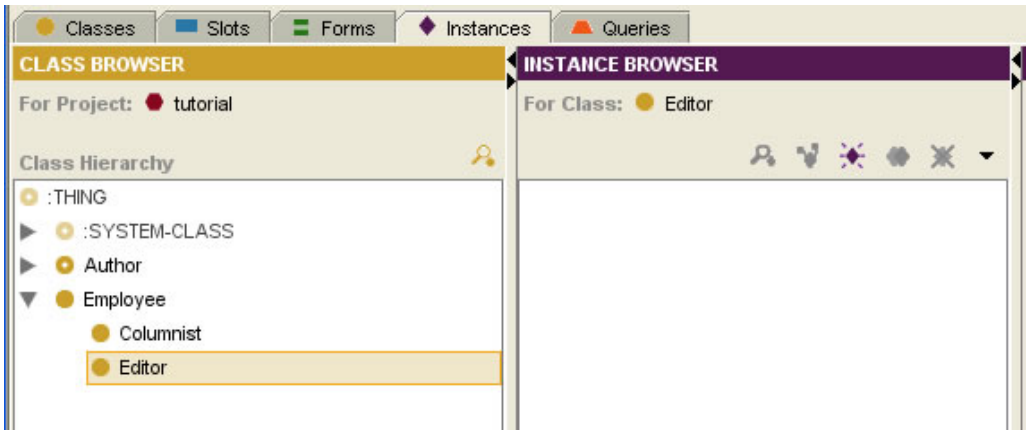

Рисунок 44. Подготовка к созданию экземпляра.

4. Нажмите кнопку **Create Instance**. Экземпляр создан и появилась форма редактора экземпляра. Видно, что на ней много полей, по одному полю для каждого созданного слота. Используйте эти поля, для того чтобы заполнить слоты значениями. Заметьте, что отображение для класса Редактор (Editor) в панели иерархии классов (Class

Hierarchy) изменилось после того, как был создан новый экземпляр класса. Единица в скобках означает, что этот класс имеет один экземпляр.

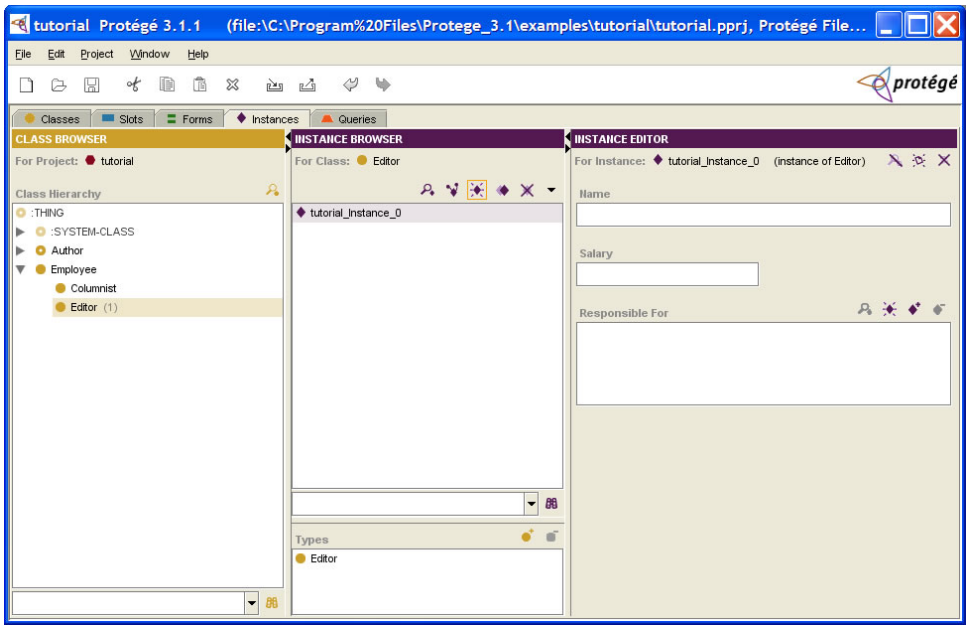

Рисунок 45. Вид редактора класса с экземпляром.

5. Введите *Chief Honcho* в поле Имя (Name).

Рисунок 46. Имя экземпляра.

6. Введите 15000 в поле зарплата (salary). Заметьте, что символы в этом поле будут подсвечены красным цветом, если что-то другое, нежели число в формате с плавающей запятой будет введено. В системе Protégé, при попытке ввода значений, которые не удовлетворяют ограничениям слота, значения подсвечиваются красным цветом.

Рисунок 47. Значение слота salary.

Теперь закладка экземпляров выглядит следующим образом (заметим, что экземпляр в навигаторе экземпляров (Instance Browser) все еще имеет стандартное имя, такое как “tutorial_instance_0”). Как изменить имя будет показано в следующем разделе.

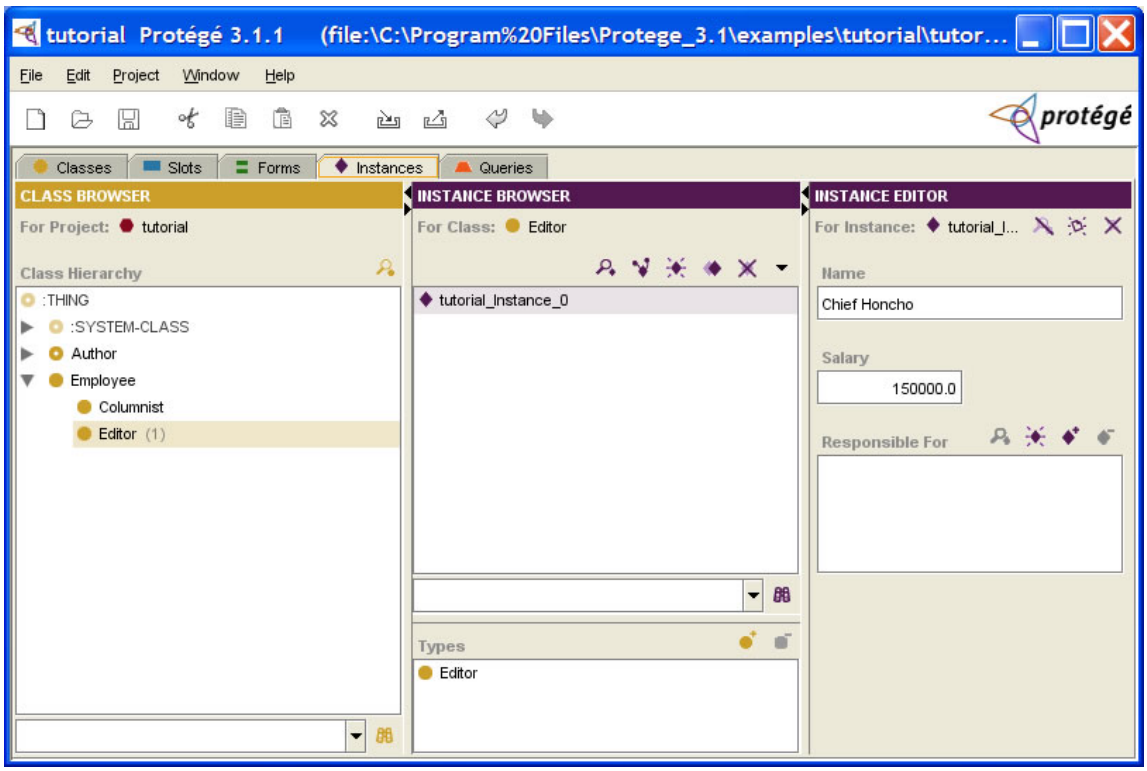

Рисунок 48. Навигатор экземпляров.

Создадим еще один экземпляр класса Редактор (Editor):

1. Нажмите кнопку **Create Instance** в навигаторе экземпляров (Instance Browser).

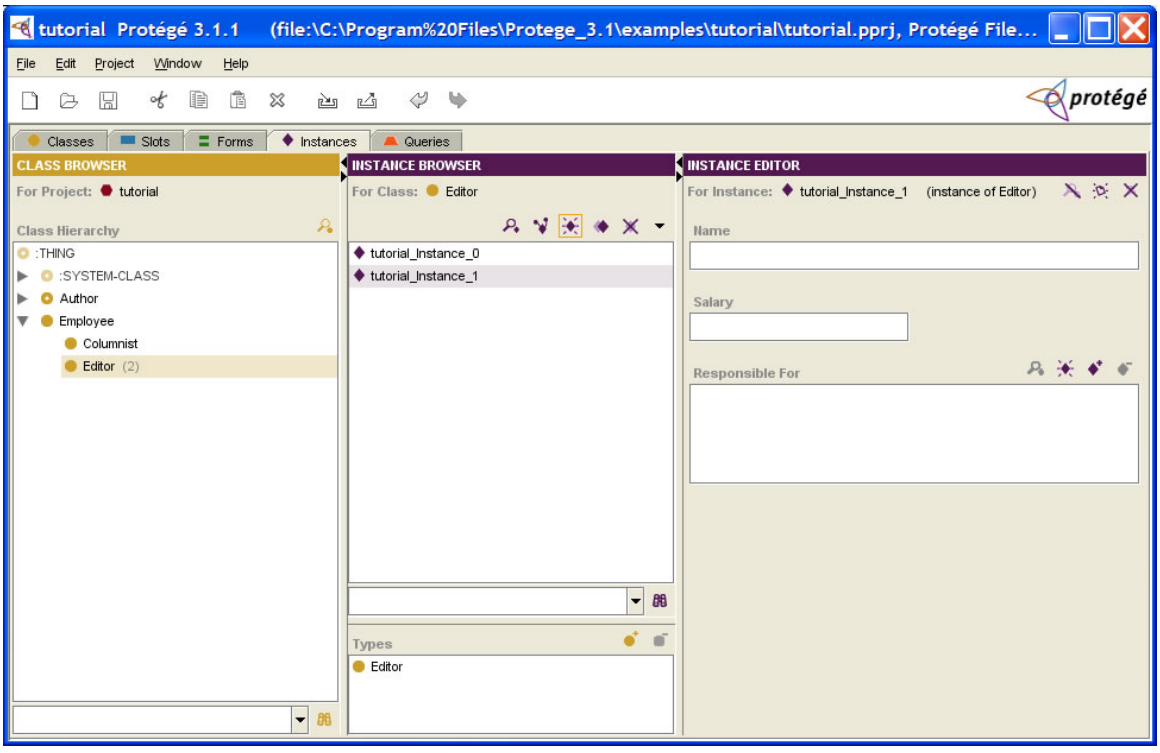

Рисунок 49. Добавление второго экземпляра.

2. Наберите *Mr. Science* в поле имя (name).

Рисунок 50. Имя второго экземпляра.

3. Введите *60000* в поле зарплата (salary).

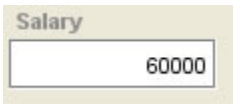

Рисунок 51. Значение слота salary для второго экземпляра.

Теперь, так как вы создали более чем один экземпляр класса, вы можете определить отношения (связи) между ними, к примеру, вы могли бы сказать, что “Chief Honcho” будет ответственным за работу “Mr. Science”. Перед тем, как это сделать, для того чтобы работа с экземплярами была легче, необходимо указать слот отображения для класса “Редактор” (Editor). Система Protégé будет показывать значение слота отображения, каждый раз при выводе на экран экземпляра класса. О том, как это сделать, будет рассказано в следующем разделе.

Установка слота отображения

Для каждого класса в вашей онтологии, вы можете указать, что один из его слотов будет слотом отображения. Система Protégé будет показывать значение этого слота, при каждом выводе экземпляра класса на экран. Если слот отображения не будет указан, то будет выведено стандартное имя, сгенерированное системой (например, “tutorial_Instance_0”). Обычно очень полезно устанавливать слот отображения для классов, которые будут иметь экземпляры. На самом деле, вы можете выбрать слот отображения даже до того, как будут созданы экземпляры класса.

Для того чтобы указать слот отображения для класса “Редактор” (Editor).

1. Выберите закладку экземпляров (Instances).
2. Выберите класс “Редактор” в панели иерархии классов.
3. Нажмите кнопку, меню экземпляров (стрелочка вниз), в верхней правой части навигатора экземпляров.

Рисунок 52. Кнопки панели иерархии экземпляров.

4. Выберите пункт задать слот отображения (set display slot).

Рисунок 53. Выбор слота отображения.

5. Выберите поле имя (name) из списка.
6. Вид списка экземпляров, в навигаторе экземпляров, поменяется, чтобы показать новые значения слота отображения. Экземпляры класса “Редактор” (Editor) теперь будут перечислены, как значения слота имя (name). Начиная с этого момента, вы можете перебирать экземпляры класса “редактор’ по его имени везде, где будет появляться список экземпляров классов.

Создание отношений (связей) между экземплярами классов

В этом разделе, мы модифицируем экземпляр *Chief Honcho* и сделаем так, чтобы он стал ответственным за экземпляр *Mr. Science*:

1. Перейдите на закладку экземпляров (instances), разверните класс “работник” (Employee) в панели иерархии классов (Class Hierarchy) и выберите класс “редактор” (Editor). Экземпляры редактора теперь показаны в навигаторе экземпляров (Instance Browser).

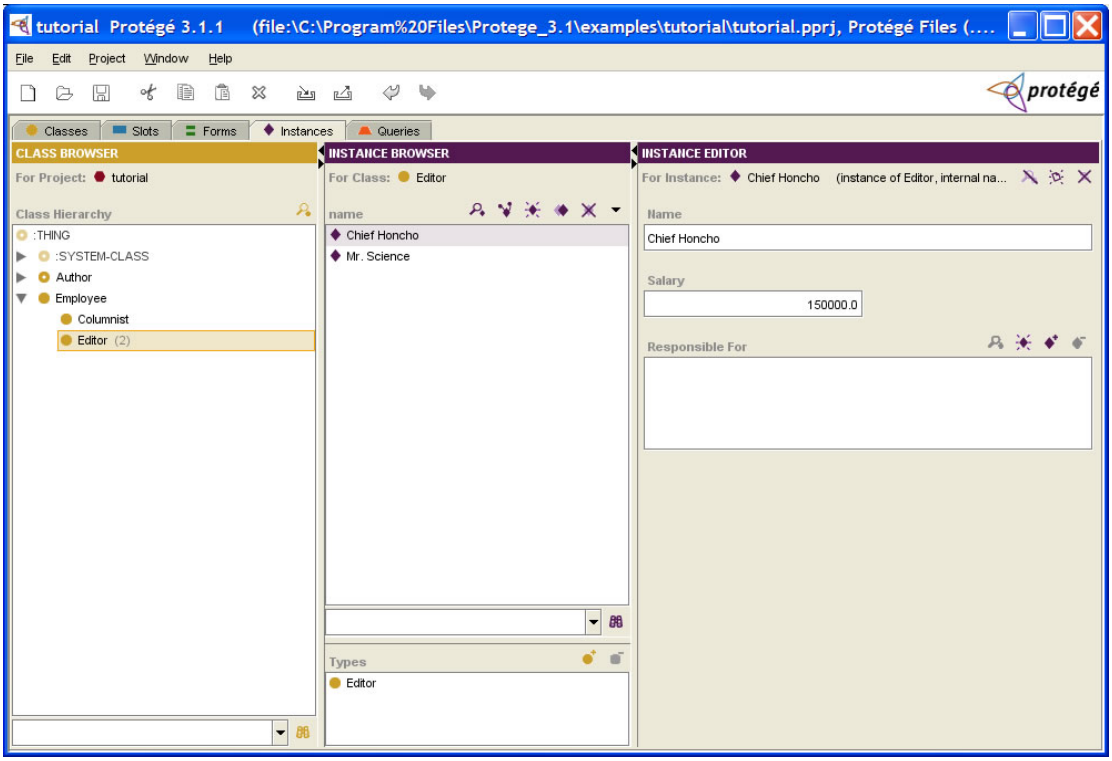

Рисунок 54. Экземпляры класса Editor.

2. Выберите *Chief Honcho* в навигаторе экземпляров. Слоты для *Chief Honcho* будут показаны в редакторе экземпляров, включая слот responsible_for (ответственный за). Заметьте, что система Protégé использует имена слотов в форме редактора, но автоматически заменяет подчеркивания в пробелы и переводит в верхний регистр первую букву каждого слова.
3. Нажмите кнопку **Add Instance** , справа сверху рядом с полем Responsible For.

Рисунок 55. Кнопки поля Responsible For.

4. Откроется окно диалога с двумя панелями. Слева будет показана иерархия доступных классов для слота responsible_for.

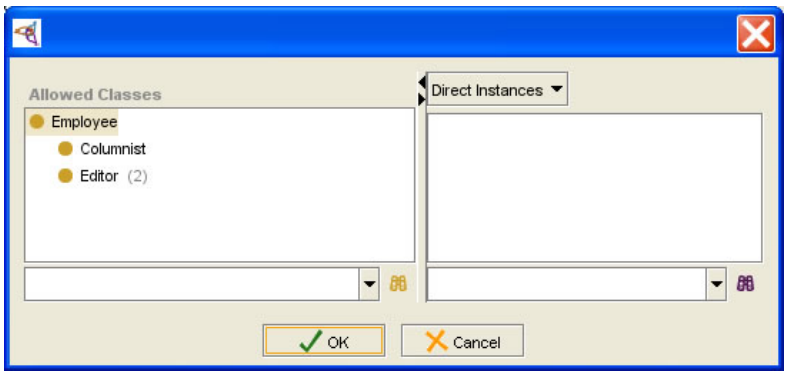

Рисунок 56. Окно установки отношений.

5. Выберите класс Editor (редактор). Справа будут показаны все экземпляры класса. Выберите *Mr. Science* и нажмите **ОК**.

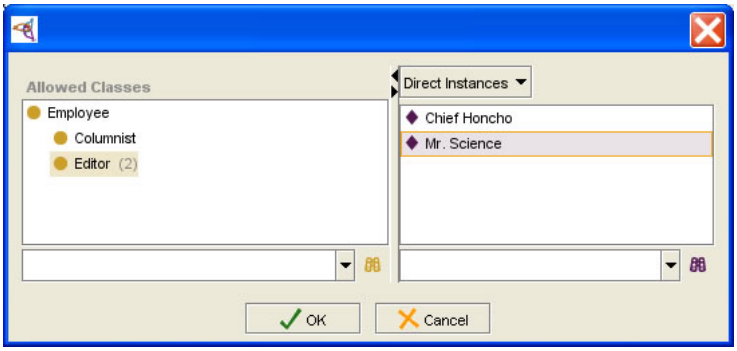

Рисунок 57. Отношение класса Editor.

6. Вы только что создали отношение (связь) в своей онтологии, которая гласит, что работник Chief Honcho является ответственным за работника Mr. Science.

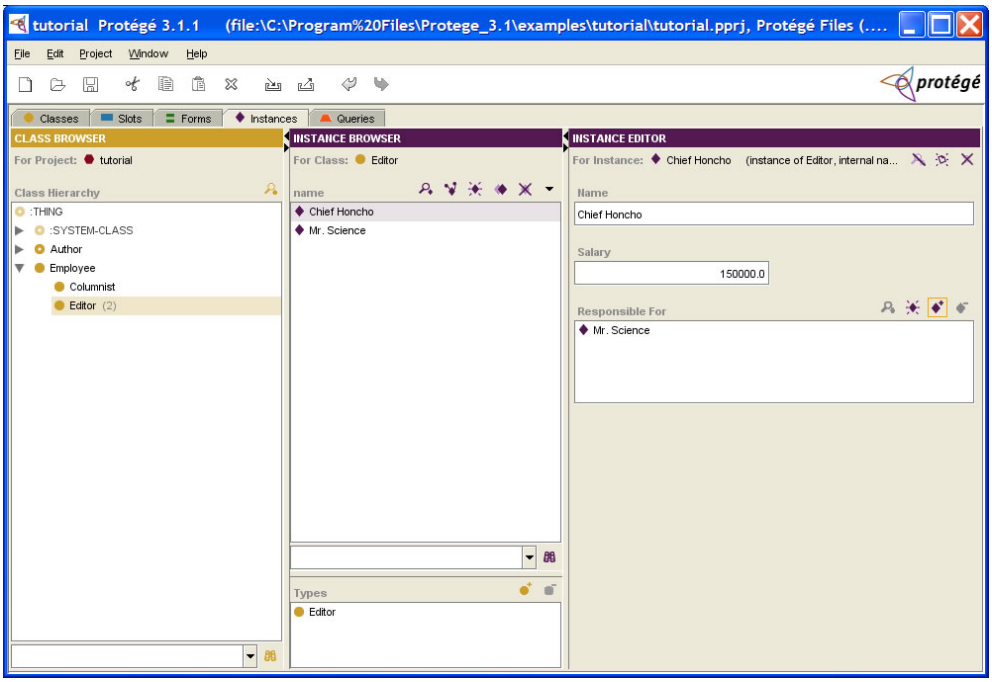

Рисунок 58. Вид редактора классов после добавления отношения.

Настройка формы ввода

Для каждого класса в вашей онтологии, Protégé генерирует форму по умолчанию, которую вы можете использовать для ввода данных экземпляра. Формы содержат поля ввода данных, или “виджеты” для каждого слота связанного с классом. Для разных типов данных слотов существуют разные типы “виджетов”, например, Protégé использует текстовый “виджет”

(TextFieldWidget) для слотов с типом данных строка, целочисленный “виджет” (IntegerFieldWidget) для полей, у которых значение представлено как целое число, “виджет” список экземпляров (InstanceListWidget) для слотов, у которых в качестве типа установлен экземпляр класса и при этом мощность (количество элементов) больше одного и т.д.

Если вам не подходит стандартная форма, созданная Protégé, вы можете изменить ее с помощью закладки форм (Forms). Среди других возможностей, вы можете изменить размер “виджетов”, перемещать их по форме, скрывать и даже менять тип “виджета”.

Для того чтобы понаблюдать, как изменения, которые были сделаны на закладке форм, отображаются в редакторе экземпляров, перейдите на закладку экземпляров, и два раза щелкните по Chief Honcho в навигаторе экземпляров, чтобы появилось отдельное окно редактора. Заметьте, что если вы создавали слоты для класса Editor в другом порядке, чем было описано в данном руководстве, ваша форма может выглядеть отлично от картинок в следующих секциях.

Изменение размера “виджета”

Вы можете изменить размер выбранного “виджета”, растянув его за угол или границу, для этого:

1. Перейдите на закладку Формы.
2. Удостоверьтесь, что именно класс “редактор” (Editor) выбран в навигаторе форм (Form Browser) слева. Затем, выберите FloatFieldWidget для слота salary (зарплата), щелкнув на нем в редакторе формы справа. В этом случае он будет подсвечен зеленым. Заметьте, что выбранный тип виджета в списке “Selected Widget Type” справа сверху указывает на то, что это “виджет” используется для ввода элементов с плавающей запятой (FloatFieldWidget).

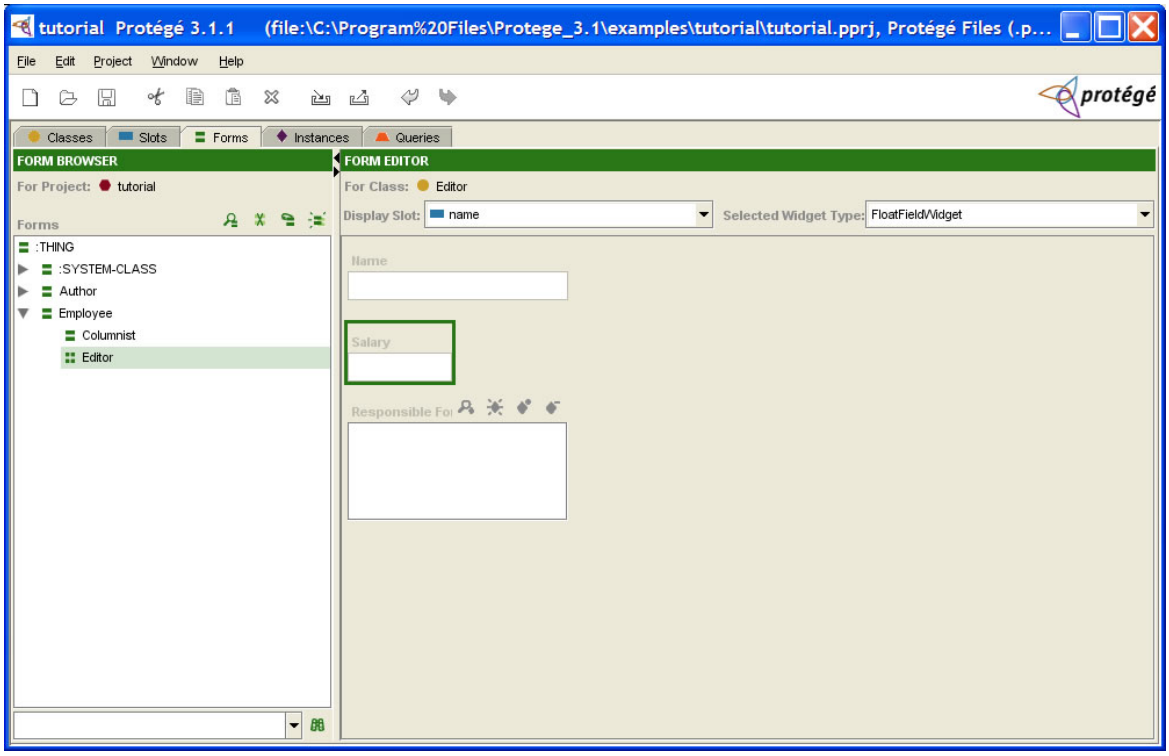

Рисунок 59. Редактор формы.

3. Щелкните по правой границе “виджета”, и удерживая кнопку мыши нажатой, перетащите ее, чтобы изменить размер “виджета”. Попытайтесь выровнять правую границу “виджета”, так чтобы она совпадала с правой границей “виджета” имя (name).

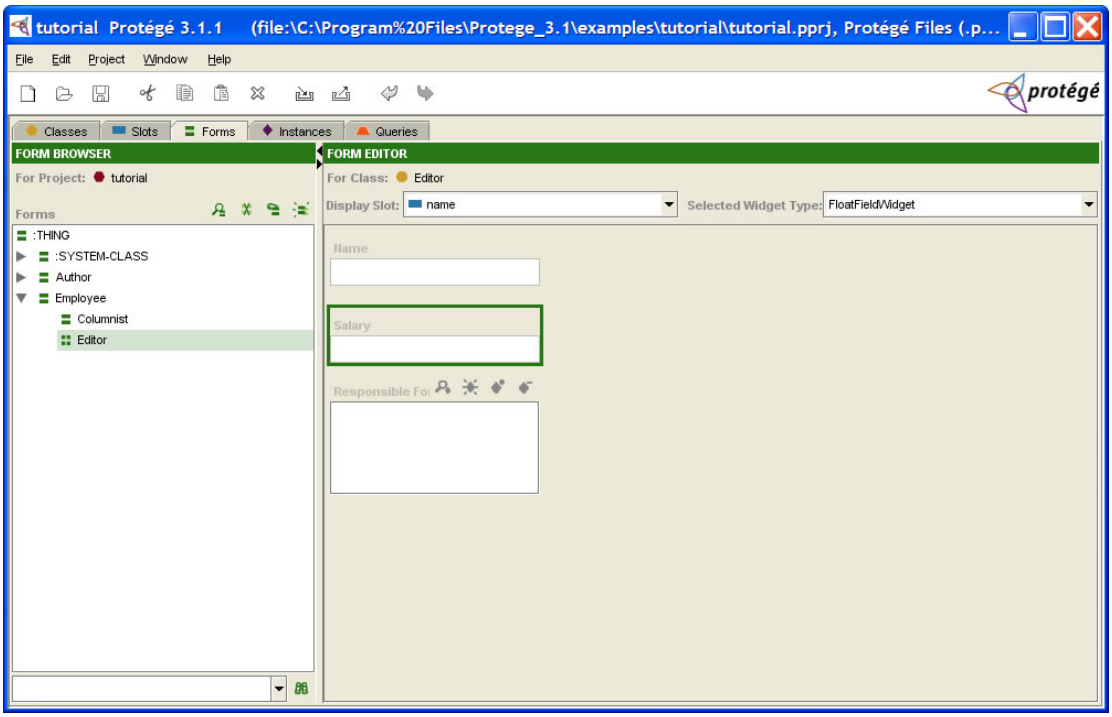

Рисунок 60. Измененная форма “виджета”.

4. Заметьте, что иконка перед формой класса Editor (редактор) в навигаторе форм изменилась. Новая иконка указывает, форма этого класса была изменена, и больше не является стандартной.

Перемещение “виджета”

Вы можете изменить положение “виджета” на форме, также при помощи перетаскивания:

1. Выберите InstanceListWidget (“виджет” для редактора экземпляров) для слота **responsible_for (ответственный за).**
2. Перетащите его на правый верхний угол формы, так чтобы верхняя граница “виджета” совпадала с верхней границей “виджета” слота “имя” (name).

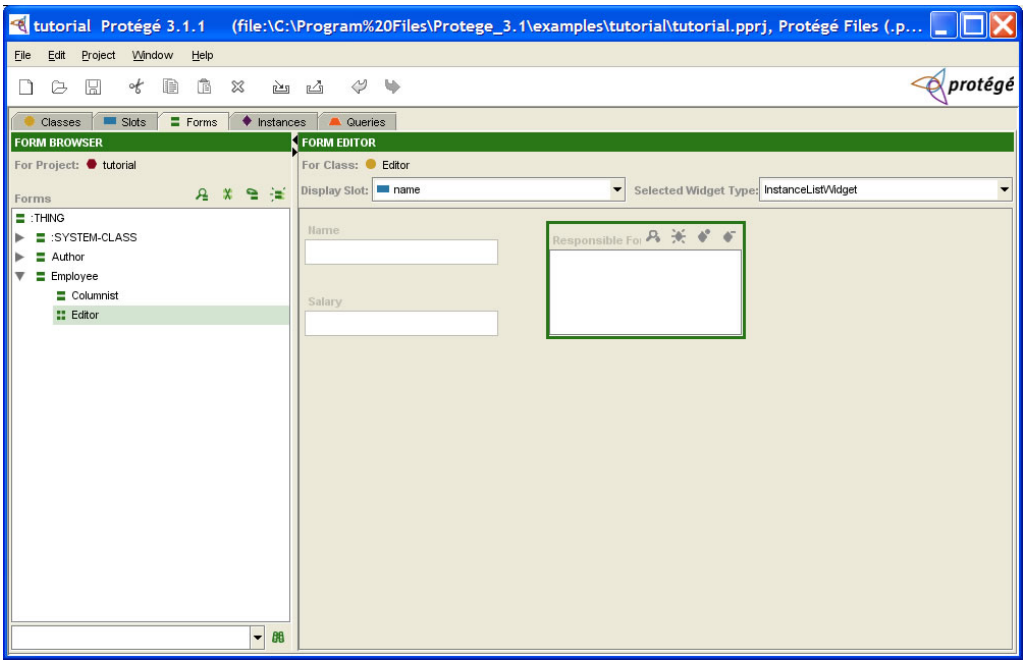

Рисунок 61. Вид формы “виджета” после перемещения.

Изменение кнопок “виджета”

Вы можете настроить “виджет” так, чтобы он показывал другую надпись или другой список кнопок, нежели по умолчанию. К примеру, вы хотите удалять экземпляр из вашего проекта, просто нажав кнопку в

InstanceListWidget (“виджете” экземпляров) для слота responsible_for. Для того чтобы показать кнопку удаления сделайте следующее:

1. Два раза щелкните на InstanceListWidget (“виджете” экземпляров) с именем “Responsible For” в редакторе форм.

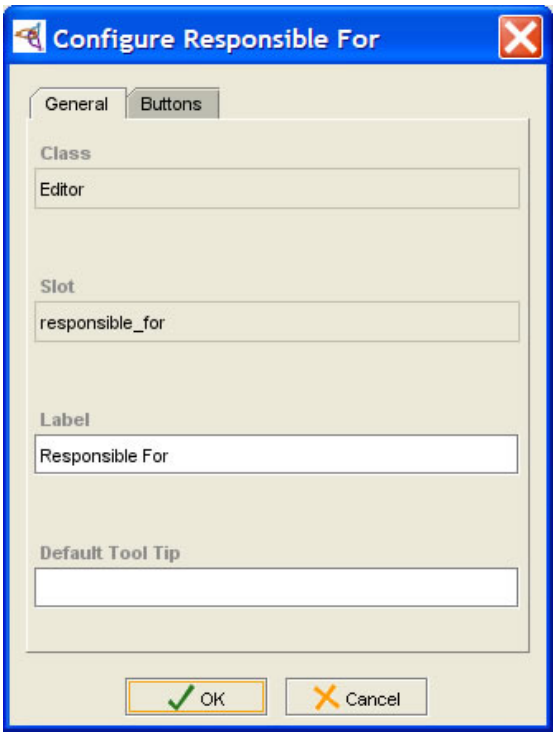

Рисунок 62. Окно конфигурации “виджета”.

2. Перейдите на закладку кнопки.

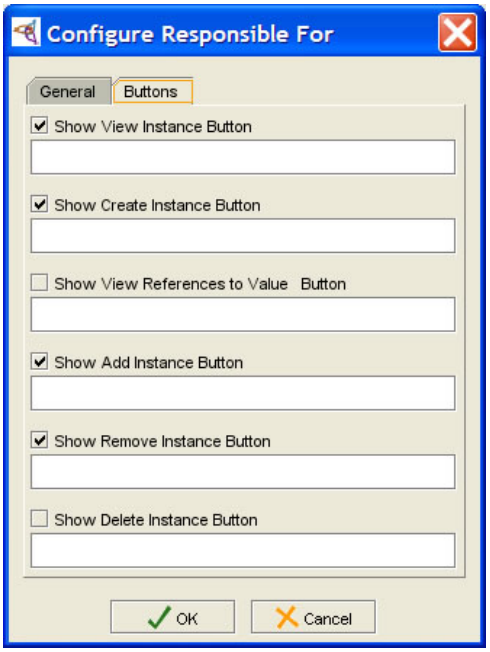

Рисунок 63. Редактор кнопок “виджета”.

3. Установите галочку для пункта Show Delete Instance Button (показывать кнопку удаления).

Рисунок 64. Добавление кнопки Delete Instance.

4. Нажмите **ОК**. “Виджет” для слота **responsible_for** (ответственный за) теперь имеет пять кнопок.

Рисунок 65. Вид “виджета”с кнопкой Delete Instance.

Скрытие “виджета”

Вы можете скрыть “виджет” так, чтобы он не был виден на форме и в редакторе экземпляров (при этом информация из онтологии не удаляется). Например, вы хотите скрыть “виджет” для слота salary (зарплата). Для этого:

1. Выберите FloatFieldWidget (виджет для поля с плавающей запятой) под названием **Salary** в редакторе формы.

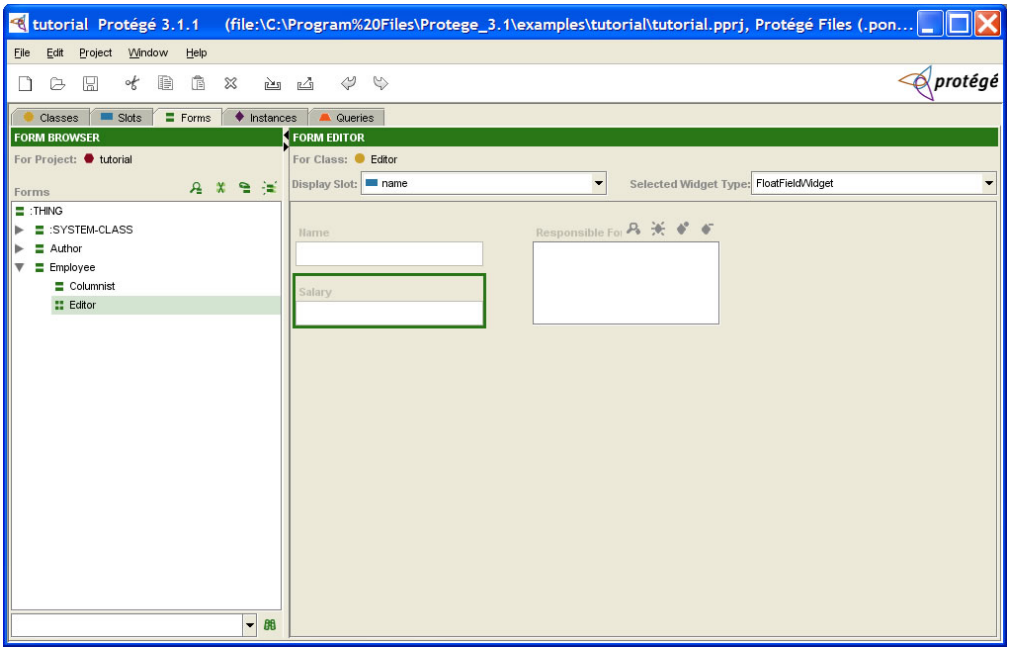

Рисунок 66. Редактирование FloatFieldWidget.

2. Выберите "<none>" из списка выбора типа “виджета” (Selected Widget Type).

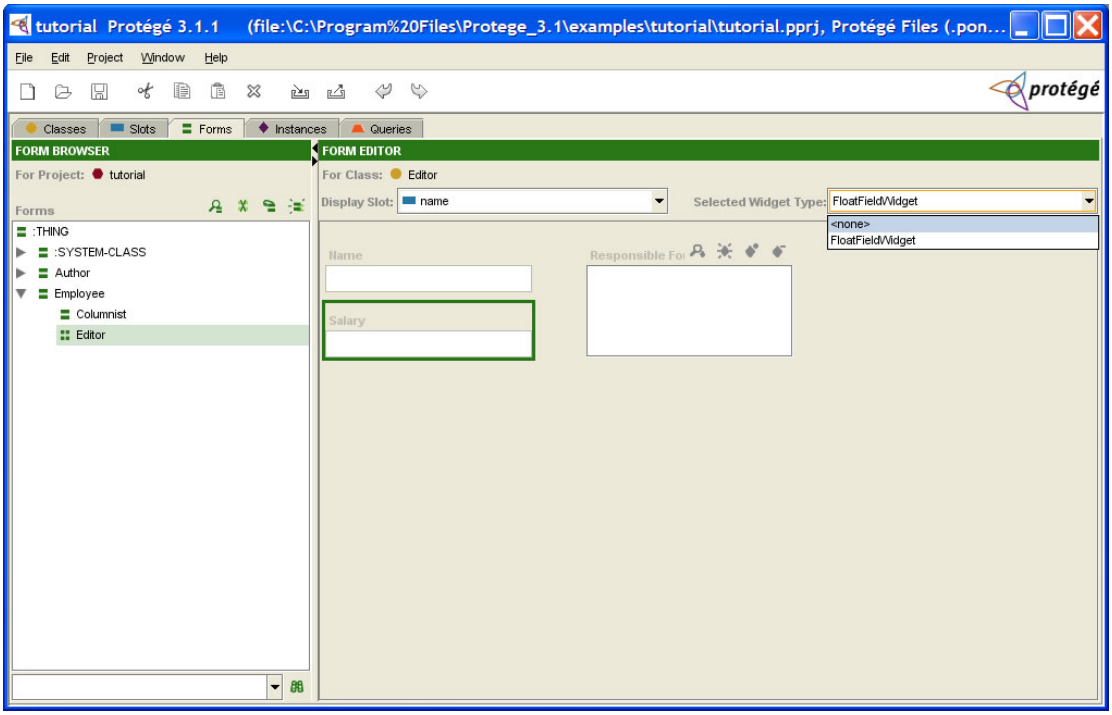

Рисунок 67. Список выбора типа “виджета”.

3. “Виджет” для слота salary (зарплата) больше не виден в редакторе формы.

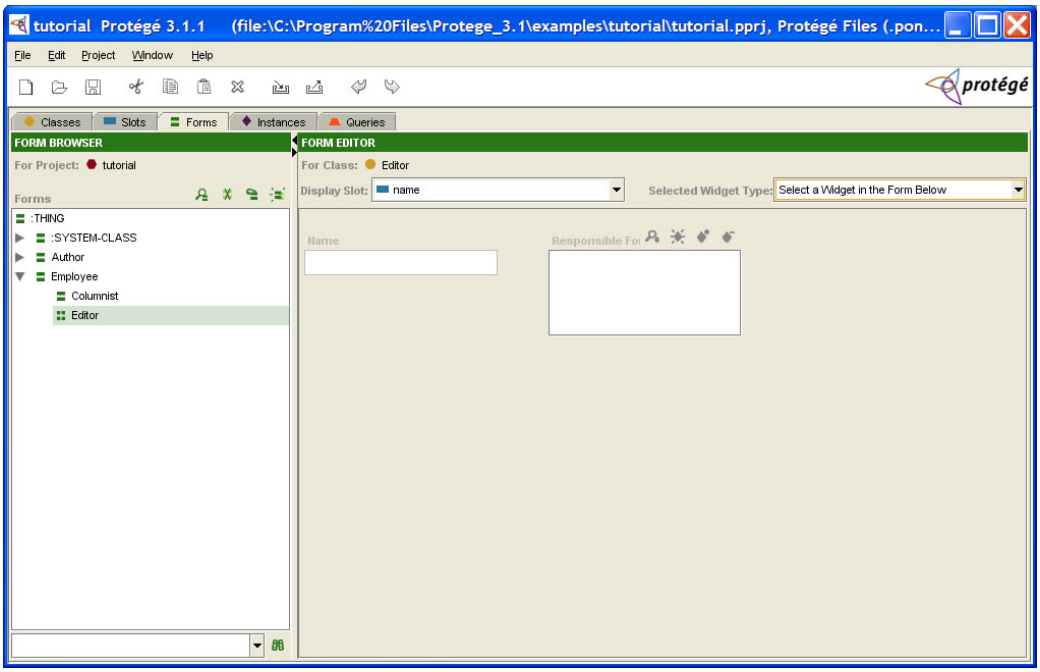

Рисунок 68. Форма с скрытым “виджетом”.

Отображение скрытого “виджета”

Для того чтобы показать виджет, который был скрыт на предыдущем шаге, сделайте следующее:

1. В навигаторе форм, нажмите кнопку **View Form Customizations** (посмотреть изменения формы) , сверху справа.

Рисунок 69. Навигатор форм.

2. В диалоге конфигурации формы, вы сможете увидеть список слотов и соответствующих им “виджетов”.

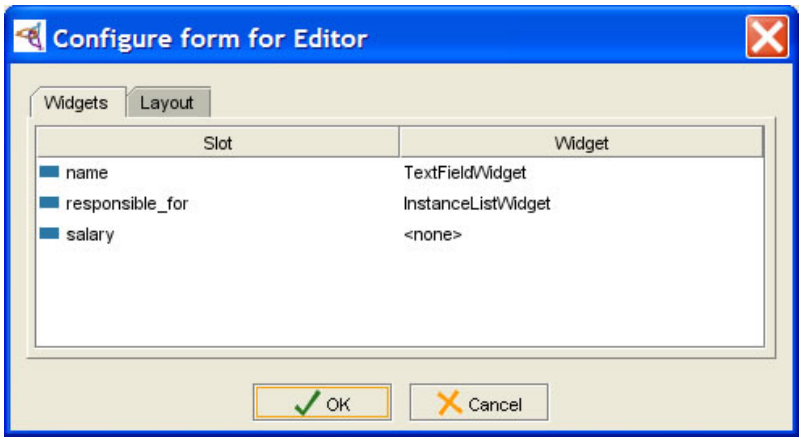

Рисунок 70. Диалог конфигурации формы.

3. Щелкните по "<none>" справа от поля salary (зарплата), а затем выберите FloatFieldWidget (“виджет” для полей с плавающей точкой).

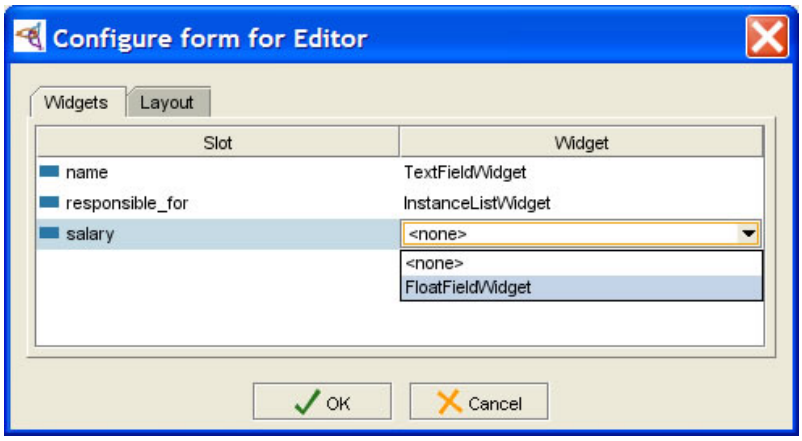

Рисунок 71. Выбор виджета для слота salary.

4. Нажмите **OK**.

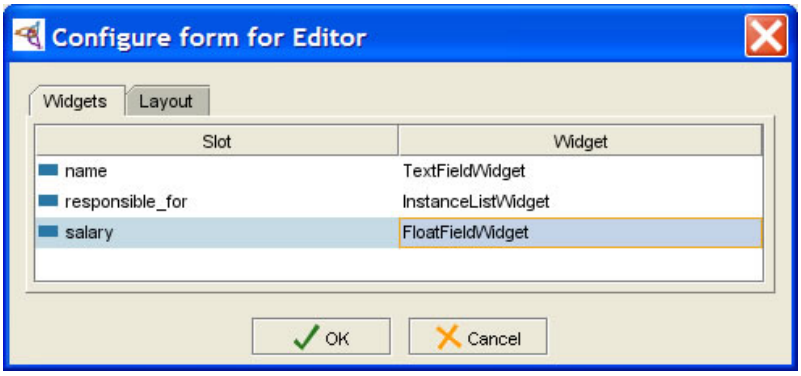

Рисунок 72. Окончательный вид конфигурации формы.

5. “Виджет” будет снова виден в редакторе формы.

Использование расположения по умолчанию

Если вы хотите отменить все свои изменения, то система Protégé позволяет вернуть все «виджеты» на форме в стандартное положение, в соответствии с их текущим размером.

Для того чтобы расположить “виджеты” стандартным образом:

1. Выберите форму класса редактор (Editor) в навигаторе форм.
2. Нажмите кнопку **Remove Form Customizations (**удалить изменения формы) в верхнем правом углу навигатора форм.
3. Форма будет выровнена по стандартам Protégé. Все скрытые виджеты будут показаны снова. Иконка, информирующая о том, что форма была изменена, будет заменена на .

Создание и сохранение запросов

Закладка запросов позволяет вам писать получать сведения из вашего проекта по всем экземплярам классов, которые удовлетворяют интересующим критериям. Для того чтобы создать запрос, вы должны выбрать один или более классов и один или более слотов в классе. Вы можете также сохранить запросы в библиотеку для последующего использования.

Создание запроса

Предположим, что вам интересно найти всех работников, которые имеют зарплату больше чем 75000$ в год. Для этого создадим запрос:

1. Щелкните на закладке запросов.

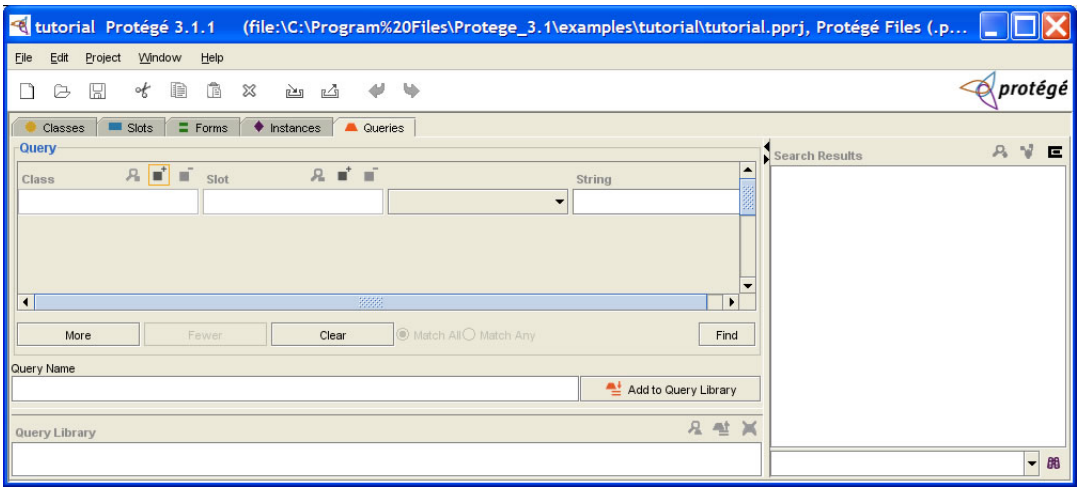

Рисунок 73. Редактор запросов.

2. Щелкните на кнопке **Select Classes** (выбрать классы) чуть повыше текстового поля Class (класс) в панели запросов.

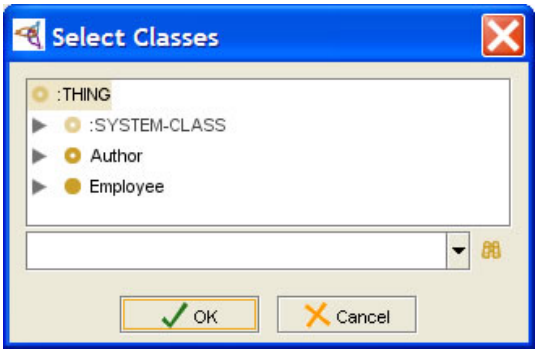

Рисунок 74. выбор класса.

3. Выберите класс “работник” (Employee) из панели выбора классов, затем нажмите ОК.
4. Теперь класс Employee (работник) отображается в текстовом окне Class (класс).

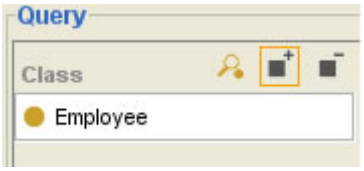

Рисунок 75. Класс запроса.

5. Нажмите кнопку выбора слота (Select Slot) чуть выше текстового поля Slot (слот).
6. Выберите salary (зарплата) в диалоговом окне выбора слота и нажмите **ОК**.

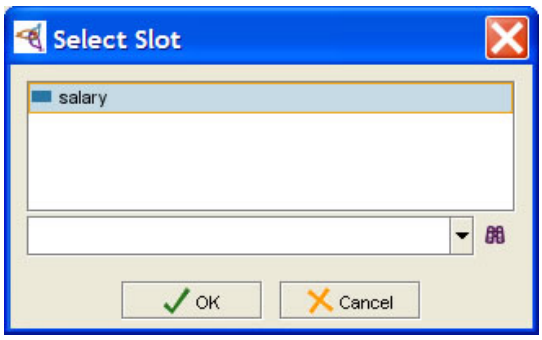

Рисунок 76. Выбор слота.

7. Меню справа от поля Slot (слот) теперь активно, и текстовое поле в правом дальнем углу напоминает, что тип значение выбранного слота число с плавающей запятой.

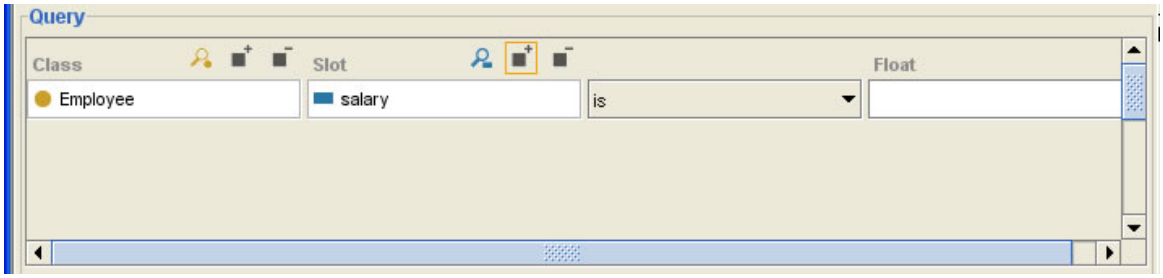

Рисунок 77. Меню запроса.

8. Выберите “is greater than” (больше чем) в выпадающем списке. Затем, введите “75000” в поля ввода под надписью Float.

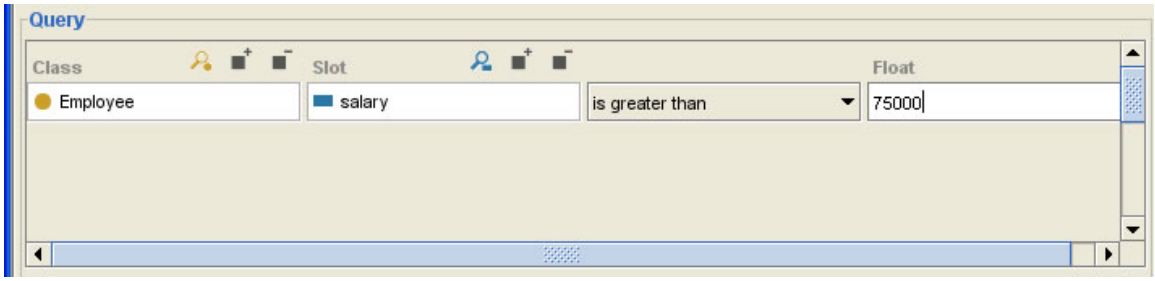

Рисунок 78. Выбор типа запроса “is greater than”.

Запуск запроса

Теперь, когда вы создали запрос, вы можете запустить его и посмотреть результаты.

1. Для запуска, нажмите кнопку поиска (Find) в нижнем правом углу панели запросов.

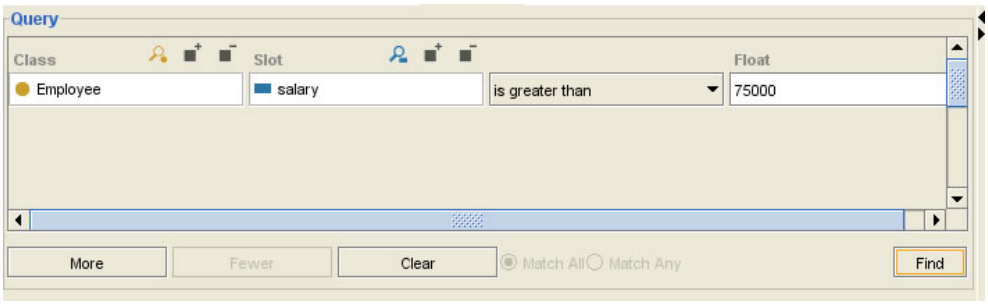

Рисунок 79. Панель запросов.

2. Результаты будут показаны в панели отображения результатов поиска справа. Если вы не можете видеть результаты полностью, то можно расширить окно или подвинуть разделитель полей.

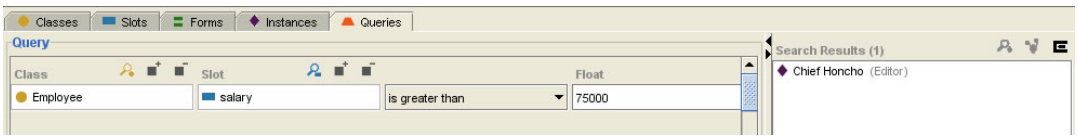

Рисунок 80. Результаты запроса.

Сохранение запроса

Вы можете сохранить запрос перед или после того как он будет запущен. Для того чтобы сохранить запрос в библиотеку запросов, сделайте следующее:

1. Нажмите кнопку добавить запрос в библиотеку (**Add to Query Library**) справа от поля query name (имя запроса).

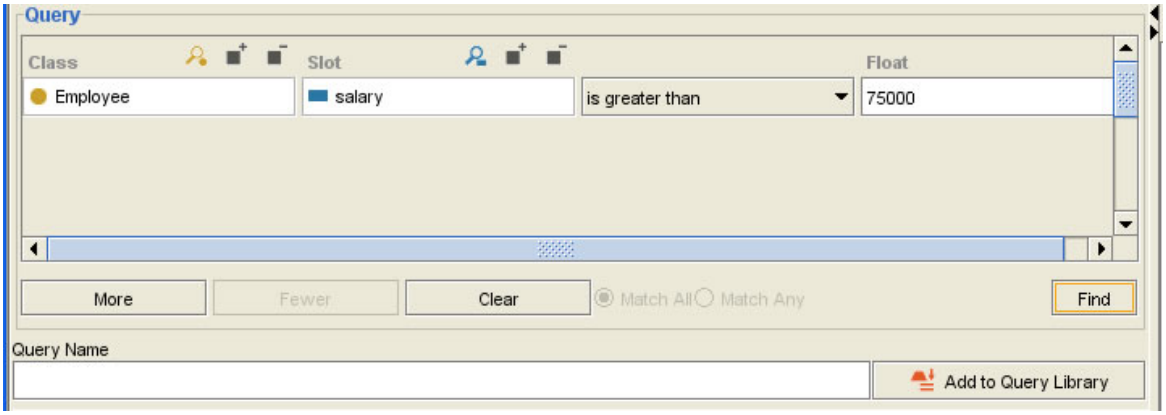

Рисунок 81. Кнопка Add to Query Library.

2. Наберите "sample_query" в окне ввода имени запроса.

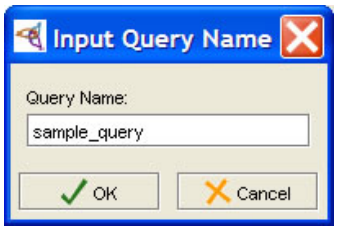

Рисунок 82. Имя запроса.

3. Нажмите **ОК**. Имя будет показано в поле query name (имя запроса) и запрос будет отображен в панели библиотеки запросов (Query Library).

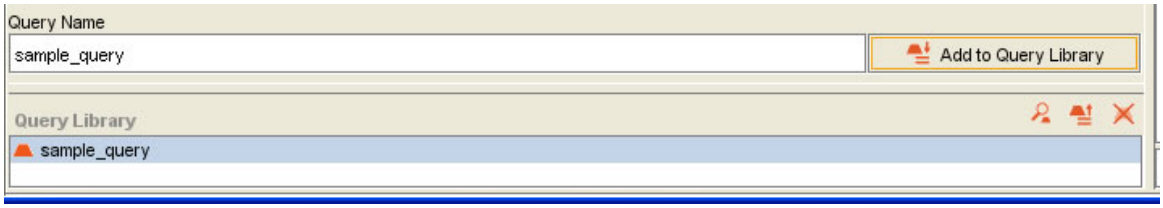

Рисунок 83. Панель библиотеки запросов.

Загрузка запроса

Для того чтобы загрузить сохраненный запрос, вы должны выбрать его в библиотеке запросов.

1. Для начала, так как мы запускаем тот же запрос снова, нажмите кнопку очистки панели результатов **Clear**. Иначе не будет заметно изменений.

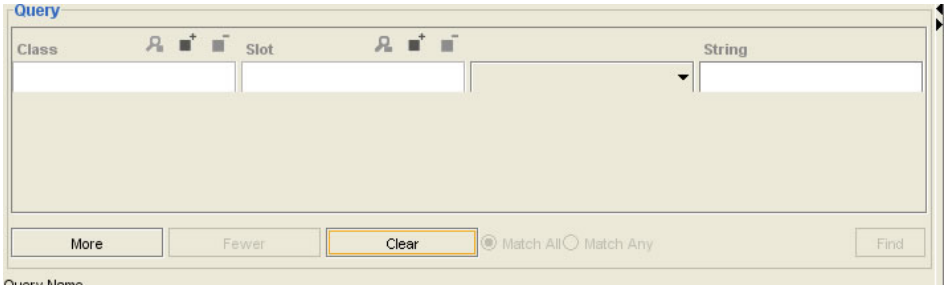

Рисунок 84. Кнопка Clear.

2. Выберите *sample_query* в библиотеке запросов внизу экрана.

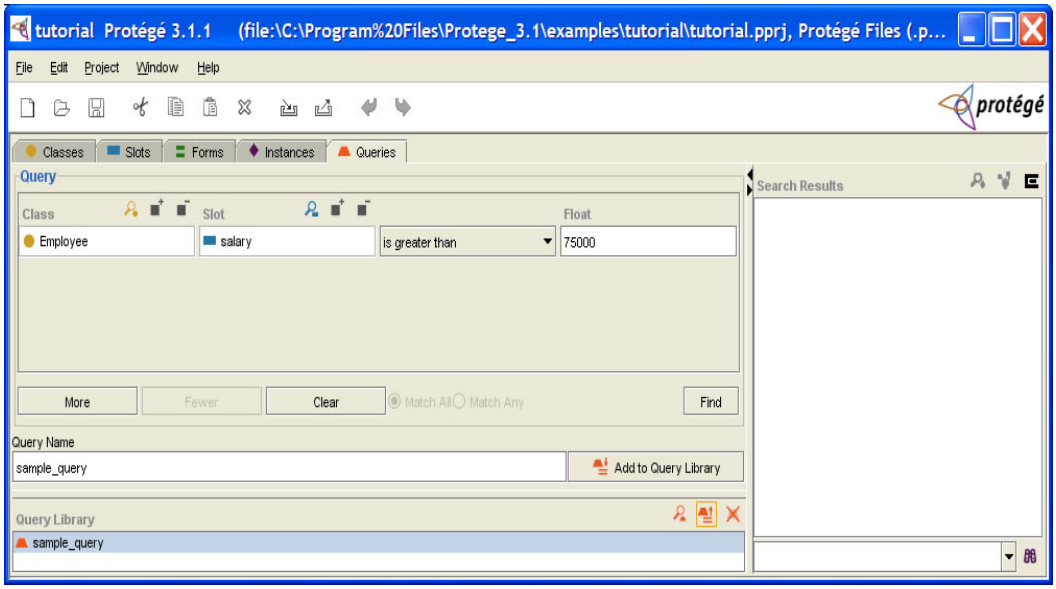

Рисунок 85. Окно библиотеки запросов.

3. Нажмите кнопку **retrieve query** .
4. Сохраненный запрос теперь отображается в верхней части окна (при желании можно изменить его). Вы также можете объединить запросы, нажав кнопку **More**.

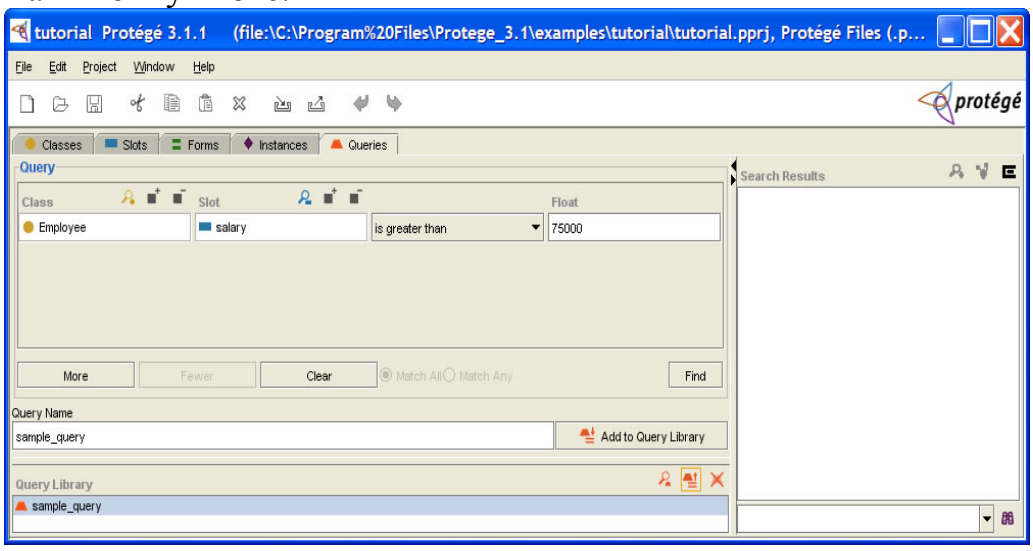

Рисунок 86. Объединение запросов.

5. Для запуска запроса нажмите кнопку **Find** (поиск).

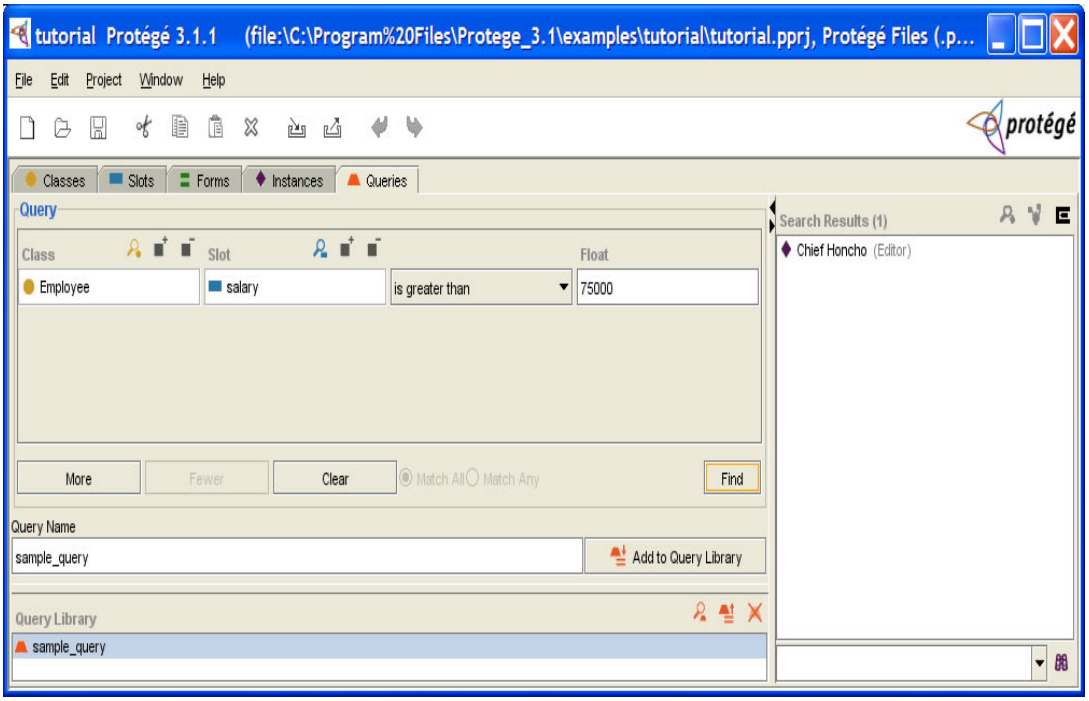

Рисунок 87. Запуск запроса.