Introducción a Jess con Java: Índice

- Jess incrustado en Java (empotrado): objeto RETE
- Uso de Jess desde Java
- Otras funciones útiles
- Instalación de Jess con Java
- Ejemplo sencillo de inferencia en Jess con E/S desde Java
  - ESJessJava
- Ejemplo más complejo
  - Pricing_engine
- Documentación útil

Formas de uso de Jess

- Jess puede ser utilizado de varias formas incluyendo aplicaciones de línea de comandos, aplicaciones con interfaz gráfica, servlets y applets.

- Se pueden desarrollar distintos tipos de aplicaciones que dependen de dónde se quiera escribir el código.
  - Es decir, hay que decidir una arquitectura en la que se escribe principalmente código Java o código Jess.
  - Aquí nos vamos a centrar en una arquitectura Java que hace uso de Jess para gestionar el conocimiento representado en forma de reglas.

Jess empotrado en Java

- Una forma de utilizar Jess en la que el control estará principalmente en el código Java consiste en manejar **objetos de la clase RETE** de la biblioteca.

- Cada objeto `rete` representa un motor de razonamiento independiente, de forma que un mismo programa puede incluir varios motores independientes (cada uno con su memoria de trabajo, base de reglas...).

- Para utilizar Jess de forma incrustada en una aplicación Java simplemente será necesario crear uno (o más) objetos `rete` y manipularlo a través de los métodos adecuados.

Ejemplo de uso de Jess mediante un objeto RETE

Cómo ejecutar un programa Jess desde Java

- public Value **eval**(java.lang.String cmd) throws JessException
  - Evalua una expresión Jess
  - Devuelve el resultado de evaluar la expresión como un jess.Value
    - Se pueden usar los métodos de la clase para extraer datos
    - Ejemplo: define "square" y luego lo ejecuta (evalúa)

```
Rete r = new Rete();
r.eval("(deffunction square (?n) (return (* ?n ?n)))");
Value v = r.eval("(square 3)");
int nine = v.intValue(r.getGlobalContext()));
```

  - SOLO se puede pasar a **eval()** una llamada a función o constructor cada vez
  - Parametros: cmd - una string que contiene una expresión Jess

Ejemplo de uso de Jess mediante un objeto RETE

- La clase Value representa los valores tipados de Jess.
- El método type() devuelve la constante de tipo que representa el valor.
- clase jess.RU (Rete Utilities)
  - No tiene constructor
  - Sus métodos y atributos son estáticos
  - Tiene los tipos de datos de Jess que se pueden usar en Java
  - Ejemplo: v1 = new Value(“7372”, RU.STRING)

Ejemplo de uso de Jess mediante un objeto RETE

Tipos de valores definidos en la clase jess.RU

Name	Method	Meaning
ATOM	atomValue()	Jess symbol
STRING	stringValue()	Jess string
INTEGER	intValue()	Jess integer
VARIABLE	variableValue()	Jess variable
MULTIVARIABLE	variableValue()	Jess multifield
FACT	factValue()	Jess fact
FLOAT	floatValue()	Jess float
FUNCALL	funcallValue()	Jess function call
LIST	listValue()	Jess list
EXTERNAL_ADDRESS	externalAddressValue()	Java object
LONG	longValue()	Java long

Intercambio de valores entre Java y Jess

- Existen mecanismos de intercambio de información estructurada en forma de objetos java que se asertan como objetos o hechos CLIPS.
  - Ver pricing_engine
    - add, addAll
    - getObjects
- Estas características se pueden consultar en la documentación pero no son necesarias para la práctica

Otras funciones que pueden ser útiles:

- **Para añadir y eliminar hechos**
  - public Fact **assertFact**(Fact f) throws JessException
  - public Fact **retract**(Fact f) throws JessException

- Clase Fact

- La clase jess.Fact (subclase de ValueVector) se usa para representar los hechos.

- Cada hecho se guarda como una lista en la que las entradas se corresponden con los slots.

- El nombre del hecho se guarda en una variable separada (que se puede obtener a través del método getName()).

- Una vez que un objeto jess.Fact se aserta pasa a formar parte de las estructuras de datos internas del objeto Rete (por lo que se deja de tener el control sobre él y no se deben hacer cambios sobre sus slots ya que no se reflejan en el hecho de la parte jess).

- Al hacer retract del hecho, se devuelve el control sobre el objeto Fact.

Ejemplo : construcción de hechos no ordenados desde Java

- Creación de un template y aserto de un hecho no ordenado con esa plantilla.

```
import jess.*;
public class ExPoint
{
  public static void main(String[] unused) throws JessException
  { Rete r = new Rete();
    r.eval("(deftemplate point \"A 2D point\" (slot x) (slot y))");
    Fact f = new Fact("point", r);
    f.setSlotValue("x", new Value(37, RU.INTEGER));
    f.setSlotValue("y", new Value(49, RU.INTEGER));
    r.assertFact(f);
    r.eval("(facts)");
  } }
C:\> java ExPoint
  f-0    (MAIN::point (x 37) (y 49))
  For a total of 1 facts.
```

Ejemplo: en el que el template tiene un multislot

- En Java, un multislot se representa mediante un objeto Value de tipo RU.LIST;
  - el objeto Value contiene un ValueVector con los campos del multislot.

```
import jess.*;
public class ExMulti   {
  public static void main(String[] unused) throws JessException    {
    Rete r = new Rete();
    r.eval("(deftemplate vector \"A named vector\"" + " (slot name) (multislot list))");
    Fact f = new Fact("vector", r);
    f.setSlotValue("name", new Value("Groceries", RU.SYMBOL));
    ValueVector vv = new ValueVector();
    vv.add(new Value("String Beans", RU.STRING));
    vv.add(new Value("Milk", RU.STRING));
    vv.add(new Value("Bread", RU.STRING));
    f.setSlotValue("list", new Value(vv, RU.LIST));
    r.assertFact(f);
    r.eval("(facts)");
  } }
C:\> java ExMulti
  f-0   (MAIN::vector (name Groceries) (list "String Beans" "Milk" "Bread"))
  For a total of 1 facts.
```

Ejemplo: creación de un hecho ordenado desde Java

- Un hecho ordenado se representa como
  - un hecho no ordenado con un único multislot llamado **__data**
- NO es necesario crear un template para un hecho ordenado:
  - se creará automáticamente si no existe.

```
import jess.*;
public class ExOrdered  {
  public static void main(String[] unused) throws JessException    {
    Rete r = new Rete();
    Fact f = new Fact("letters", r);
    ValueVector vv = new ValueVector();
    vv.add(new Value("a", RU.SYMBOL));
    vv.add(new Value("b", RU.SYMBOL));
    vv.add(new Value("c", RU.SYMBOL));
    f.setSlotValue("__data", new Value(vv, RU.LIST));
    r.assertFact(f);
    r.eval("(facts)");
  }
}
C:\> java ExOrdered
  f-0   (MAIN::letters a b c)
  For a total of 1 facts.
```

Otras funciones que pueden ser útiles:

- Los siguientes métodos de la clase Rete permiten añadir elementos a un motor de reglas (un objeto Rete):
  - public void addDeffacts(Deffacts)
  - public void addDefglobal(Defglobal)
  - public void addDefrule(Defrule)
  - public void addDeftemplate(Deftemplate)
  - public void addUserfunction(Userfunction)
  - public void addUserpackage(Userpackage)

- Los siguientes métodos de la clase Rete devuelven (dado un nombre) los elementos existentes en un motor de reglas.
  - public Defglobal findDefglobal(String)
  - public Defrule findDefrule(String)
  - public Deftemplate findDeftemplate(String)
  - public Userfunction findUserfunction(String)

Otras funciones que pueden ser útiles:

- Los siguientes métodos de la clase Rete devuelven elementos java.util.Iterators para las distintas estructuras de datos de un motor de reglas:
  - public Iterator listActivations()
  - public Iterator listDeffacts()
  - public Iterator listDefglobals()
  - public Iterator listDefrules()
  - public Iterator listDeftemplates()
  - public Iterator listFacts()
  - public Iterator listFunctions()

Creación proyecto Java con Jess en eclipse

- Proyectos disponibles en Campus Virtual
  - ESJessJava y PreciosJess (basta importarlos desde Eclipse como proyectos)

- Creación de un proyecto nuevo Java + Jess en Eclipse
  - Crear un proyecto java nuevo (*File* + *New Project*)
  - Crear fichero `.java` dentro de la carpeta *src*
  - Crear fichero jess (con sufijo .clp) dentro del proyecto pero fuera de *src*
  - Incluir las librerías de Jess: *jess.jar* y *jsr94.jar*
    - Ratón encima del proyecto + botón derecho + *Build Path* + *Configure Build Path*
      - también en la barra de menús “Project” opción “Properties” + *Java Build Path*
    - Pestaña “Libraries” + columna derecha “add external JARs” :
    - Buscar la distribución de Jess \Jess71p2\lib
      - marcar las dos *jess.jar* y *jsr94.jar*

Ejemplo sencillo de inferencia en Jess con E/S desde Java

```
public class ESJessJava {


    public static void main(String[] args) throws JessException {


                Rete miRete;
                miRete = new Rete();

                String ficheroReglas = "diet.clp";
                System.out.println("Cargando las reglas");
                try
                {
                        Value v = miRete.batch(ficheroReglas);
                        System.out.println("Value " + v);
                } catch (JessException je0) {
                        System.out.println("Error de lectura en " + ficheroReglas);
                        je0.printStackTrace();
                }
```

Ejemplo sencillo de inferencia en Jess con E/S desde Java

```
/*Crea un deffacts y añade un hecho siguiendo la template usuario
    definida en diet.clp. En una aplicación real, los datos de usuario se
    leerían en java y se añadirían así al programa jess*/
Deffacts deffacts = new Deffacts("DatosJava", null, miRete);
Fact f = new Fact("usuario", miRete);
f.setSlotValue("edad", new Value(30, RU.INTEGER));
f.setSlotValue("altura", new Value(150, RU.INTEGER));
f.setSlotValue("peso", new Value(65, RU.INTEGER));
f.setSlotValue("sexo", new Value("h", RU.SYMBOL));
f.setSlotValue("actividadFisica", new Value(1.2, RU.FLOAT));
deffacts.addFact(f);
miRete.addDeffacts(deffacts);
miRete.reset();
```

Ejemplo sencillo de inferencia en Jess con E/S desde Java

```
// A continuación se ejecuta el motor jess
// Si el programa jess no tiene módulos basta con hacer run()

                miRete.run();

/* Si el programa jess tiene módulos hay que poner el foco sucesivamente en
cada uno de ellos, en el orden adecuado, y a continuación un run() por cada
módulo
                miRete.setFocus("módulo1");
                miRete.run();
                miRete.setFocus("módulo2");
                miRete.run();
                miRete.setFocus("módulo3");
                miRete.run();
                miRete.setFocus("módulo4");
                miRete.run();
```

Ejemplo sencillo de inferencia en Jess con E/S desde Java

```
// Se listan los hechos que hay en la memoria de trabajo

                listaHechos(miRete);
                extraeHechos(miRete);

                // Para parar el motor de reglas
                miRete.halt();

    }
```

Ejemplo sencillo de inferencia en Jess con E/S desde Java

```
// Obtiene e imprime la lista de hechos
public static void listaHechos(Rete miRete) {
        Iterator<Fact> iterador;
        iterador = miRete.listFacts();

        System.out.println("Lista de hechos:");
        while (iterador.hasNext()) {
                System.out.println(iterador.next());
        }
}
```

Ejemplo sencillo de inferencia en Jess con E/S desde Java

```
public static void extraeHechos(Rete miRete) {
        // Ejemplo de cómo podemos seleccionar sólo los hechos de la template usuario
        // Y de esos hechos quedarnos sólo con el slot edad e imprimir su valor
        Iterator<Fact> iterador;
        iterador = miRete.listFacts();
        Fact f;
        Value ed;
        while (iterador.hasNext()) {
                f = iterador.next();
                if (f.getName().equals("MAIN::usuario"))
                {
                        try {
                                ed = f.getSlotValue("edad");
                                System.out.println("edad " + ed);
                        } catch (JessException e) {
                                // TODO Auto-generated catch block
                                e.printStackTrace();
                        } }
        }
}
```

Proyecto *pricing_engine*

- Este ejemplo está explicado con detalle en el capitulo 11 del manual, "Embedding Jess in a Java Application" .
  - /Jess71p2/docs/embedding.html#

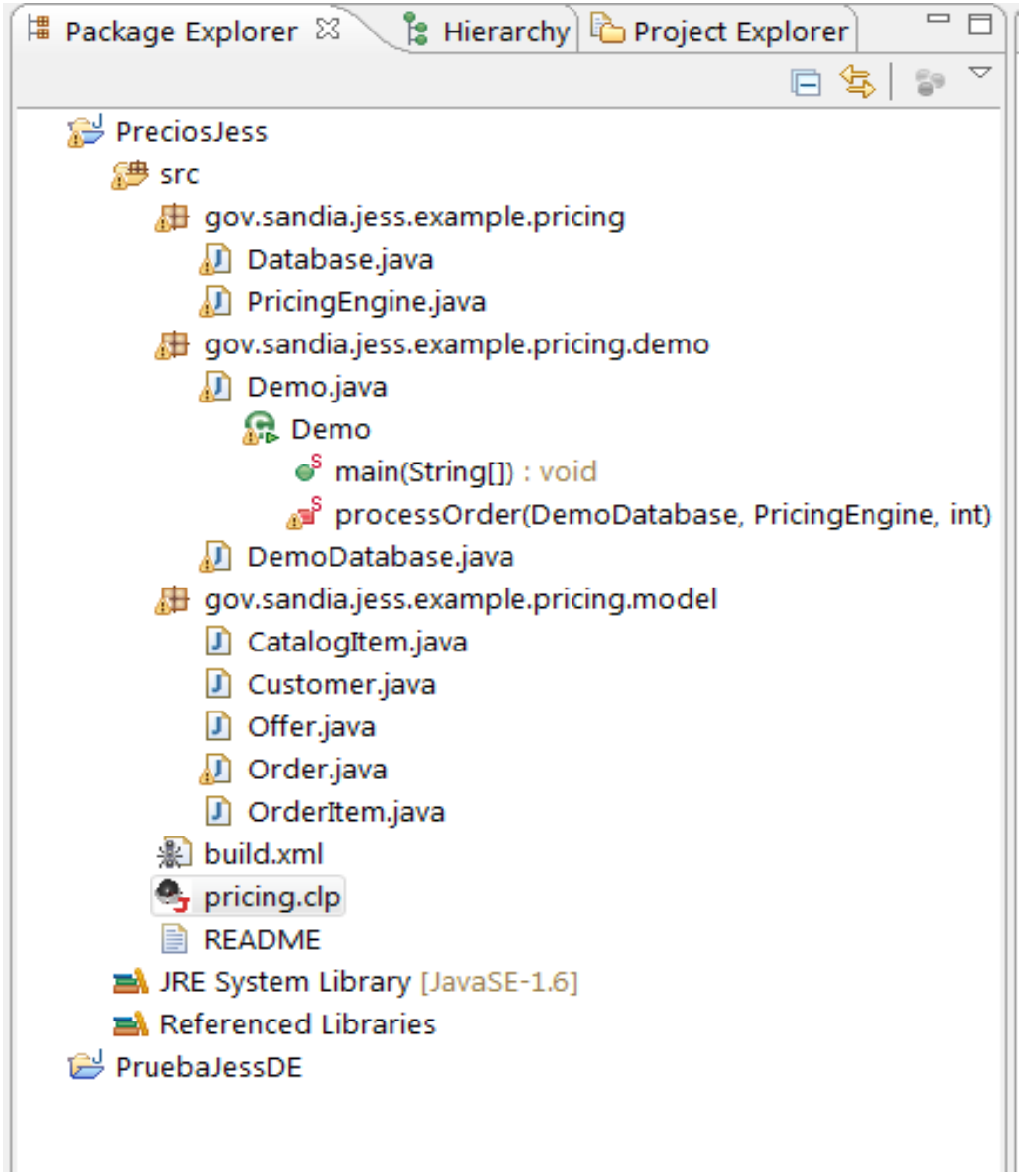
- Hay tres paquetes Java:
  - `gov.sandia.jess.example.pricing`, `gov.sandia.jess.example.pricing.model`, `gov.sandia.jess.example.pricing.demo`

- `gov.sandia.jess.example.pricing`:
  - La clase `gov.sandia.jess.example.pricing.PricingEngine`
    - Muestra un modo de usar la API de Jess para Java.
      - Crea el motor con Rete, carga las reglas del fichero "pricing.clp"
      - Carga la base de datos en memoria de trabajo
      - Añade cada orden, cliente e items a la memoria de trabajo del motor
  - La clase `gov.sandia.jess.example.pricing.Database`
    - Interface de la base de datos simulada:
    - implementada con datos en paquete demo con la clase `DemoDatabase`

Proyecto *pricing_engine*

- `gov.sandia.jess.example.pricing.model`
  - Tiene las clases del modelo para una aplicación simple de e-commerce
- `gov.sandia.jess.example.pricing.demo`
  - Tiene la clase driver para ejecutar todo y algunos datos simulando una base de datos
- Las reglas están en el fichero *pricing.clp*
- También hay un fichero *ant* para compilar todo: *build.xml* (desde eclipse es automático)
- Para ejecutar: compilar todo y ejecutar `gov.sandia.jess.example.pricing.demo.Demo`

pricing_engine: clases



Fichero de reglas

```
;; First define templates for the model classes so we can use them
;; in our pricing rules. This doesn't create any model objects --
;; it just tells Jess to examine the classes and set up templates
;; using their properties

(import gov.sandia.jess.example.pricing.model.*)
(deftemplate Order       (declare (from-class Order)))
(deftemplate OrderItem   (declare (from-class OrderItem)))
(deftemplate CatalogItem (declare (from-class CatalogItem)))
(deftemplate Customer    (declare (from-class Customer)))
```

Fichero de reglas

```
;; Now define the pricing rules themselves. Each rule matches a set
;; of conditions and then creates an Offer object to represent a
;; bonus of some kind given to a customer. The rules assume that
;; there will be just one Order, its OrderItems, and its Customer in
;; working memory, along with all the CatalogItems.

(defrule 10%-volume-discount
   "Give a 10% discount to everybody who spends more than $100."
   ?o <- (Order {total > 100})
   =>
   (add (new Offer "10% volume discount" (/ ?o.total 10))))

(defrule 25%-multi-item-discount
   "Give a 25% discount on items the customer buys three or more of."
   (OrderItem {quantity >= 3} (price ?price))
   =>
   (add (new Offer "25% multi-item discount" (/ ?price 4))))
```

Fichero de reglas

```
(defrule free-cd-rw-disks
   "If somebody buys a CD writer, send them a free sample of CD-RW
   disks, catalog number 782321; but only if they're a repeat customer.
   We use a regular expression to match the CD writer's description."
   (CatalogItem (partNumber ?partNumber) (description /CD Writer/))
   (CatalogItem (partNumber 782321) (price ?price))
   (OrderItem (partNumber ?partNumber))
   (Customer {orderCount > 1})
   =>
   (add (new Offer "Free CD-RW disks" ?price)))
```

Documentación útil

- Documentación dentro de la distribución
  - /Jess71p2/docs/index.html
  - /Jess71p2/docs/intro.html
  - /Jess71p2/docs/rules.html
  - /Jess71p2/docs/api.html
    - Jess - the Rule Engine for the Java Platform v70p2
      - The Jess Library
      - The JSR94 API
        - » **javax.rules, javax.rules.admin**
  - Ejemplos
    - \Jess71p2\examples
- **Documentación de la clase RETE:** http://herzberg.ca.sandia.gov/docs/70/api/jess/Rete.html
- **Manual de Jess:** http://www.jessrules.com/jess/docs/71/library.html http://www.jessrules.com/jess/docs/71/embedding.html