

CS 580 Client-Server Programming
Fall Semester, 2002
Doc 24 VW Database Connection
Contents

VW Database Interface 2
 Sample Connection 4
 Connection 5
 Exceptions from Database 7
 Simple Example 8
 Variables in Queries 11
 Select Queries 16
 Adding SQL to Objects 21

References

VisualWorks Database Application Developer's Guide,
Chapter 7 EXDI Database Interface

VW Database Interface

Load the PostgreSQLEXDI parcel

See the Database section of the Parcel manager

Converting between Types

Smalltalk Type	PostgreSQL type
ByteString	text,
ByteString	varchar(4),
ByteString	char(4),
SmallInteger	integer,
SmallInteger	int2,
SmallInteger	int8,
SmallInteger	oid,
FixedPoint	numeric(6,2),
Double	float,
Double	float4,
Date	date,
Time	time,
String	timestamp,
SmallInteger	interval,
True	bool,
Point	point,
LineSegment	lseg,
Polyline	path,
Rectangle	box,
Circle	circle,
Polyline	polygon,
ByteString	inet,
ByteString	cidr,
ByteString	macaddr

Sample Connection

“Prints all rows of table pg_tables”

```
| connection session answer columns |
connection := PostgreSQLDConnection new.
connection
  username: 'whitney';
  password: 'notsoEasy';
  environment: 'rugby.sdsu.edu_test';
  connect.
(session := connection getSession)
  prepare: 'SELECT * FROM pg_tables';
  execute.
answer := session answer.
columns := answer columnDescriptions.
Transcript clear.
columns do: [:each | Transcript show: each name]
  separatedBy: [Transcript tab].
Transcript cr.
[answer atEnd] whileFalse:
  [| row |
  row := answer next.
  row do: [:each | Transcript print: each] separatedBy: [Transcript tab].
  Transcript cr].
connection disconnect.
Transcript flush
```

Connection

Environment string format: host:port_databaseName

Host

- DNS name or
- IP address
- Required

port is optional

- Defaults to PostgreSQL port standard port 5432

Isolating Database Vendor Information

ExternalDatabaseConnection isolates the selection of

- Driver
- Database location

Setting the Concrete driver

```
ExternalDatabaseConnection defaultConnection:  
    #PostgreSQLExDIDConnection.
```

Set the database

```
ExternalDatabaseConnection defaultEnvironment: 'rugby.sdsu.edu_test'.
```

Using the defaults

After setting the defaults

ExternalDatabaseConnection new returns the default Driver class

Driver class uses the default environment

```
connection := ExternalDatabaseConnection new.  
connection  
    username: 'whitney';  
    password: 'idontthnkso';  
    connect.  
"some work done here"  
connection disconnect
```

Exceptions from Database

Errors in the SQL results in exceptions in Smalltalk

```
[(session := connection getSession)
 prepare: 'put your sql here';
 execute]
  on: connection class externalDatabaseErrorSignal
  do: [:exception | Put your error handler code here].
```

There are several subexceptions that are possible

See the documentation for details

Simple Example

Creation The Table

title	starttime	abstract

Example assumes set defaults in
ExternalDatabaseConnection

```

connection := ExternalDatabaseConnection new.
connection
  username: 'whitney';
  password: 'idontthnkso';
  connect.
session := connection getSession.
createSQL :=
  'CREATE TABLE events(title text, starttime timestamp, abstract text)'.
[session
  prepare: createSQL
  execute]
  on: connection class externalDatabaseErrorSignal
  do: [:exception | Dialog warn: exception parameter first
dbmsErrorString].
connection disconnect.

```


Insertion

```
acm := 'INSERT INTO events(title, starttime, abstract) VALUES ("BREW",  
"11-21-2002 18:30:00", "Qualcomm ...") '.
```

```
session := connection getSession.
```

```
[session
```

```
  prepare: acm;
```

```
  execute]
```

```
  on: connection class externalDatabaseErrorSignal
```

```
  do: [:exception | Dialog warn: exception parameter first  
dbmsErrorString].
```

```
  connection disconnect.
```

Connection is obtained as previous slide

The values are in two single quotes

The Count of Rows Affected

PostgreSQL does not implement the rowCount method

PostgreSQLAdditions package on course STORE adds the method

Returns the number of rows add/modified

```
acm := 'INSERT INTO events(title, starttime, abstract) VALUES ("BREW",  
"11-21-2002 18:30:00", "Qualcomm ...") '.
```

```
session := connection getSession.
```

```
[session
```

```
  prepare: acm;
```

```
  execute]
```

```
    on: connection class externalDatabaseErrorSignal
```

```
    do: [:exception | Dialog warn: exception parameter first
```

```
dbmsErrorString].
```

```
  connection disconnect.
```

```
^session rowCount
```

Variables in Queries

There are three ways to indicate variables

- ?
- :index
- :methodName

? as Variable

```
acm := 'INSERT INTO events(title, starttime, abstract) VALUES (?, ?, ?)'
```

```
[session := connection getSession.
```

```
session prepare: acm.
```

```
session
```

```
  bindInput: #('BREW' '11-21-2002 18:30:00' 'Qualcomm ...');
```

```
  execute]
```

```
  on: connection class externalDatabaseErrorSignal
```

```
  do: [:exception | Dialog warn: exception parameter first
```

```
dbmsErrorString].
```

```
connection disconnect.
```

:index as Variable

:n represents the n'th location in the input

```
acm := 'INSERT INTO events(title, starttime, abstract) VALUES (:1, :2, :3)';
```

```
[session := connection getSession.
```

```
session prepare: acm.
```

```
session
```

```
  bindInput: #('BREW' '11-21-2002 18:30:00' 'Qualcomm ...');
```

```
  execute]
```

```
    on: connection class externalDatabaseErrorSignal
```

```
    do: [:exception | Dialog warn: exception parameter first
```

```
dbmsErrorString].
```

```
connection disconnect.
```

```
acm := 'INSERT INTO events(title, starttime, abstract) VALUES (:1, :3, :2)';
```

```
[session := connection getSession.
```

```
session prepare: acm.
```

```
session
```

```
  bindInput: #('BREW' 'Qualcomm ...' '11-21-2002 18:30:00');
```

```
  execute]
```

```
    on: connection class externalDatabaseErrorSignal
```

```
    do: [:exception | Dialog warn: exception parameter first
```

```
dbmsErrorString].
```

```
connection disconnect.
```

Objects and Variables in Queues Class used in Examples

```
Smalltalk defineClass: #Event
  superclass: #{Core.Object}
  instanceVariableNames: 'title startTime abstract '
  category: 'DatabseExamples'
```

Event class methodsFor: 'instance creation'

```
title: aString time: aTimeStamp
  ^self new
    title: aString;
    time: aTimeStamp
```

Event methodsFor: 'accessing'

```
abstract
  ^abstract

abstract: aString
  abstract := aString

time
  ^startTime

time: aString
  startTime := aString

title
  ^title

title: aString
  title := aString
```

Using an Object for input

:n indicates which instance variable in the object to select

```
acm := 'INSERT INTO events(title, starttime, abstract) VALUES (:1, :2, :3)';
```

```
data := Event title: 'brew' time: '11-21-2002 17:30:00'.
```

```
data abstract: 'test'.
```

```
[session := connection getSession.
```

```
session prepare: acm.
```

```
session
```

```
    bindInput: data;
```

```
    execute]
```

```
    on: connection class externalDatabaseErrorSignal
```

```
        do: [:exception | Dialog warn: exception parameter first
```

```
dbmsErrorString].
```

```
connection disconnect.
```

Name Input Binding

Positional binding can be error prone

Can indicate a method to get values for variables

Class just needs getter methods

```
acm := 'INSERT INTO events(title, time, abstract)
      VALUES (:title, :time, :abstract)'.
data := Event title: 'brew' time: '11-21-2002 17:30:00'.
data abstract: 'test'.
```

```
[session := connection getSession.
session prepare: acm.
session
  bindInput: data;
  execute]
  on: connection class externalDatabaseErrorSignal
  do: [:exception | Dialog warn: exception parameter first
dbmsErrorString].
connection disconnect.
```

Select Queries

```
recent := 'SELECT * FROM events WHERE starttime > "11-21-2002  
12:30:00".'
```

```
[session := connection getSession.
```

```
session
```

```
  prepare: recent;
```

```
  execute]
```

```
    on: connection class externalDatabaseErrorSignal
```

```
    do: [:exception | Dialog warn: exception parameter first
```

```
dbmsErrorString].
```

```
answer :=session answer.
```

```
allRows := answer upToEnd.
```

```
connection disconnect.
```

```
^allRows
```


session answer

Returns

- #noAnswerStream

If query is INSERT, UPDATE, CREATE etc that does not return result set

- #ExternalDatabaseAnswerStream

If query is SELECT

- #noMoreAnswers

Some databases permit returning multiple result sets

If query returns multiple result sets

keep sending answer until you get this returned

ExternalDatabaseAnswerStream

Some useful methods

next

Returns the next row

By default a row is an array of columns

upToEnd

Returns an array of all the rows from current to end

atEnd

Returns true if we are at the end of the result set

```
[session := connection getSession.
```

```
session
```

```
  prepare: recent;
```

```
  execute]
```

```
  on: connection class externalDatabaseErrorSignal
```

```
  do: [:exception | put handler here].
```

```
answer :=session answer.
```

```
rowArray := answer next.
```

Getting Objects Back

```
recent := 'SELECT * FROM events WHERE starttime > "11-21-2002  
12:30:00"'
```

```
[session := connection getSession.
```

```
session
```

```
  prepare: recent;
```

```
  bindOutput: Event new;
```

```
  execute]
```

```
    on: connection class externalDatabaseErrorSignal
```

```
    do: [:exception | Dialog warn: exception parameter first dbmsErrorString].
```

```
answer :=session answer.
```

```
anEventObject := answer next.
```

```
connection disconnect.
```

```
^anEventObject
```

Columns are mapped to instance variables

First column is mapped to first instance variable in definition

If instance variable containing an instance of Object are skipped

Using Column Names as Setters

```
[session := connection getSession.  
session  
  prepare: recent;  
  bindOutputNamed: Event new;  
  execute]  
  on: connection class externalDatabaseErrorSignal  
  do: [:exception | Dialog warn: exception parameter first dbmsErrorString].  
answer :=session answer.  
anEventObject := answer next.  
connection disconnect.  
^anEventObject
```

When using bindOutputNamed: Event new;

The columns names of the table are used to form names of setter methods to set values in the object

Adding SQL to Objects

Object should know how to

- Save self to database
- Update self to database
- Retrieve from database

Modified Events Table

```
CREATE TABLE events(title text, starttime timestamp, abstract text, id
serial)
```

New Events Class

```
Smalltalk defineClass: #Event
  superclass: #{Core.Object}
  indexedType: #none
  private: false
  instanceVariableNames: 'title startTime abstract id '
  classInstanceVariableNames: ''
  imports: ''
  category: 'DatabseExamples'
```

Event class methodsFor: 'instance creation'

```
title: aString time: aTimeStamp
  ^self new
    title: aString;
    startTime: aTimeStamp
```

Event class methodsFor: 'database connection'

```
connection
  ^DatabaseProxy connection "DatabaseProxy not shown"
```

Event class methodsFor: 'sql'

save: anEvent

| connection saveSQL session |

connection := self connection.

saveSQL := 'INSERT INTO events(title, starttime, abstract) VALUES (?, ?, ?)'.
?

[session := connection getSession.

session prepare: saveSQL.

session

bindInput: anEvent;

execute]

on: connection class externalDatabaseErrorSignal

do: [:exception | Dialog warn: exception parameter first
dbmsErrorString].

connection disconnect.

^session rowCount

Event methodsFor: 'accessing'

abstract

^abstract

abstract: aString

abstract := aString

save

self class save: self

startTime: aTimestamp

startTime := aTimestamp

time

^startTime

title

^title

title: aString

title := aString