Creating performance test data with the benerator

Volker Bergmann
Software Architect
databene
Volker Bergmann

- Passionate Software Architect & Troubleshooter
- Specialized in Performance Assessment and Tuning
- 10 years of experience with enterprise application performance
- Creator of benerator
Presentation Goal

Learn what benerator is good for and how to get started populating databases
Have a test setup that resembles production in:

- Hard- and Software setup
- Client load (LoadRunner, JMeter)
- Database content (production data, generator)

Database content is essential for confident test results!
Perftest Essentials

- Have a test setup that resembles production in
  - Hard- and Software setup
  - Client load (LoadRunner, JMeter)
  - Database content (production data, benerator)

- Database content is essential for confident test results!
Have a test setup that resembles production in:

- Hard- and Software setup
- Client load (LoadRunner, JMeter)
- Database content (production data, generator)

Database content is essential for confident test results!
Perftest Essentials

- Have a test setup that resembles production in:
  - Hard- and Software setup
  - Client load (LoadRunner, JMeter)
  - Database content (production data, benerator)

- Database content is essential for confident test results!
benerator is the leading tool to address an essential (and often neglected) issue of J2EE performance testing: creating production-like data!
Getting the data
Getting the data

- performance testers like testing with production data
Getting the data

- performance testers like testing with production data
  
  ...but often use of production data is not viable

- outsourcing
- data quality concerns / outdated data
- secrecy concerns
Getting the data

- performance testers like testing with production data
  - ...but often use of production data is not viable
    - outsourcing
    - data quality concerns / outdated data
    - secrecy concerns
- use of production data
  - does not automatically assure realistic performance tests results
  - provides the most realistic data...
Getting the data

- performance testers like testing with production data
  
  ...but often use of production data is not viable

- outsourcing

- data quality concerns / outdated data

- secrecy concerns

- use of production data
  
  does not automatically assure realistic performance tests results

  provides the most realistic data...

  ...for the past!
Future of performance testing
In most cases we would like to predict future performance:
Future of performance testing

- In most cases we would like to predict future performance:
  - scenario testing (supporting decision-making)
In most cases we would like to predict future performance:

- scenario testing (supporting decision-making)
- complementing functional testing with equivalence class performance testing
In most cases we would like to predict future performance:

- scenario testing (supporting decision-making)
- complementing functional testing with equivalence class performance testing
- functional Denial-of-Service Testing
Data generator requirements
Data generator requirements

- **Must-haves**
  - passing complex validation
  - customizability / extendability
  - performance
Data generator requirements

- **Must-haves**
  - passing complex validation
  - customizability / extendability
  - performance

- **Desirable**
  - production data export + anonymization
  - early applicability in projects (little initial effort)
  - little maintenance effort with ongoing implementation
**Data generator requirements**

- **Must-haves**
  - passing complex validation
  - customizability / extendability
  - performance

- **Desirable**
  - production data export + anonymization
  - early applicability in projects (little initial effort)
  - little maintenance effort with ongoing implementation

- **Nice to have**
  - platform independence
  - reusability among companies and technologies
databene benerator

- started in June 2006 by Volker Bergmann
- http://databene.org/benerator
- leading open source test data generator
- pioneer work, continuously evolving with experience: 'Eat your own dog food'
- version 0.5.6 (as of December 2008), 14th release
- version 1.0 planned for 2009
- well documented: free PDF reference book (130 pages)
- Configuration by exception
- reduces data definition effort from weeks to days
What it can do

- Generate data in a platform-independent manner
- Populate databases
- Create (batch) data files
- Automatically generate XML files from XML Schema file
- Be easily extended for other system types / file formats
abstract data generation

benerator
reads metadata...

Database
TABLE: CUSTOMER
abstract data generation

benerator
reads metadata...

```
Database
TABLE : CUSTOMER

descriptor
first n me
last n me
```
abstract data generation

benerator  
reads metadata...

...generates  
abstract entities...

Database
TABLE: CUSTOMER

descriptor
first_name: Volker
last_name: Bergmann

customer
first_name: Volker
last_name: Bergmann

metadata
abstract data generation

benerator reads metadata...
...generates abstract entities...
...and Consumers map them to concrete entities

<Customer
  first_name="Volker"
  last_name="Bergmann"
/>

CUSTOMER
<table>
<thead>
<tr>
<th>FIRST_NAME</th>
<th>LAST_NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>Volker</td>
<td>Bergmann</td>
</tr>
</tbody>
</table>

customers.csv

first_name, last_name
Volker, Bergmann
architecture

• platform-independent meta data model (types, constraints, generation setup)
• pluggable metadata import
• rich set of predefined generators
architecture

- platform-independent meta data model (types, constraints, generation setup)
- pluggable metadata import
- rich set of predefined generators
- rich set of plugin interfaces, e.g. for probability distributions, randomizers, meta data import, concrete data import/export, data conversion, custom generators
- pluggable data mapping
- predefined domain packages
- extendable with 3rd party domain packages
- localization, nested regions
- platform-independent meta data model (types, constraints, generation setup)
- pluggable metadata import
- rich set of predefined generators
- rich set of plugin interfaces, e.g. for probability distributions, randomizers, meta data import, concrete data import/export, data conversion, custom generators
- pluggable data mapping
building blocks

Applications/Systems
Oracle DB
IBM DB2
Microsoft SQL Server
MySQL
PostgreSQL
HSQL
Derby
[Web Service]
[EJB Service]
building blocks

Applications/Systems
- Oracle DB
- IBM DB2
- Microsoft SQL Server
- MySQL
- PostgreSQL
- HSQL
- Derby
- [Web Service]
- [EJB Service]

benerator

XML Schema [XMI]

descriptor file
building blocks

Applications/Systems
Oracle DB
IBM DB2
Microsoft SQL Server
MySQL
PostgreSQL
HSQL
Derby
[Web Service]
[EJB Service]

Data Files
CSV
XML
Flat Files
DbUnit
Excel
SQL
scripted
[YAML]

benerator
descriptor file
XML Schema
[XMI]

query
read
write

metadata

read
write

query
metadata
building blocks

Applications/Systems
Oracle DB
IBM DB2
Microsoft SQL Server
MySQL
PostgreSQL
HSQL
Derby
[Web Service]
[EJB Service]

Data Files
CSV
XML
Flat Files
DbUnit
Excel
SQL
scripted
[YAML]

Distribution
Converter
Validator

Descriptor file
XML Schema
[XMI]

Generator

Extend

Read
Write
Query
Metadata
building blocks

- SQL
- Shell
- JavaScript
- JRuby
- Jython
- Groovy

- Applications/Systems
  - Oracle DB
  - IBM DB2
  - Microsoft SQL Server
  - MySQL
  - PostgreSQL
  - HSQL
  - Derby
  - [Web Service]
  - [EJB Service]

- benerator
  - scripts
  - descriptor file
  - XML Schema [XMI]

- read
- write
- query
- metadata

- extend
- generator
- converter
- distribution
- validator

- Data Files
  - CSV
  - XML
  - Flat Files
  - DbUnit
  - Excel
  - SQL
  - scripted
  - [YAML]

- read
- write
- data files

- invoke
- run
- configure

- invoke
- run
shop schema
benerator in action
Custom Generators

For defining a simple custom generator, inherit LightweightGenerator:

```java
public class AlternatingGenerator extends LightweightGenerator<Integer> {
    private int value = 1;

    public Integer generate() {
        value = (value + 1) % 2;
        return value;
    }
}
```
Calling scripts

You can use your favorite scripting language: JavaScript, Python, Scala, Ruby, Groovy:

```execute type="js">
importPackage(org.databene.model.data);
print('DB-URL' + db.getUrl());

// create user Alice
var alice = new Entity('db_user');
alice.set('id', 1);
alice.set('name', 'Alice');
db.store(alice);

// create user Bob
var bob = new Entity('db_user', 'id', '2', 'name', 'Bob');
db.store(bob);

// persist everything
db.flush();
</execute>
users

- 4,000 downloads
- Mainly
  - Outsourcers
  - Finance
  - Consultancies

LOYALTY PARTNER SOLUTIONS

C1 SetCon

cellent.
Conclusion

benerator is the platform of choice for generating useable performance test data!
future work

- extending benerator
  - establishing an ecosystem of domain plugins
  - generating JavaBeans, calling EJB services
  - integrating Spring -> Spring scripting tool
  - adding load generator features
- going professional with consulting and add-ons
- finding cooperators, research partners & employees
Q&A

... do you have questions?
Thanks for your attention!

http://databene.org
volker@databene.org