Creating and Deploying Custom Jackpot Queries and Transformers

Tom Ball

Senior Staff Engineer
Sun Microsystems, Inc.
http://jackpot.netbeans.org

TS-1278

Goal of This Talk

Learn how to create custom Jackpot queries and transformers, and how to share them with other developers

Agenda

Jackpot Overview

The Jackpot Developer Module

Writing Rule Files

Creating Query Classes

Custom Transformer Classes

Sharing Your Jackpot Commands

Agenda

Jackpot Overview

The Jackpot Developer Module

Writing Rule Files

Creating Query Classes

Custom Transformer Classes

Sharing Your Jackpot Commands

Agenda

Jackpot Overview

The Jackpot Developer Module

Writing Rule Files

Creating Query Classes

Custom Transformer Classes

Sharing Your Jackpot Commands

Agenda

Jackpot Overview

The Jackpot Developer Module

Writing Rule Files

Creating Query Classes

Custom Transformer Classes

Sharing Your Jackpot Commands

Agenda

Jackpot Overview

The Jackpot Developer Module

Writing Rule Files

Creating Query Classes

Custom Transformer Classes

Sharing Your Jackpot Commands

Agenda

Jackpot Overview

The Jackpot Developer Module

Writing Rule Files

Creating Query Classes

Custom Transformer Classes

Sharing Your Jackpot Commands

Jackpot Overview

Jackpot Is a Technology For:

- Searching Java™ source code
- Safely and correctly transforming patterns in code
- Writing minimal changes back to source

Jackpot Overview

- Jackpot is a technology for:
  - Searching Java source code
  - Safely and correctly transforming patterns in code
  - Writing minimal changes back to source
- Jackpot relies heavily on the new **`javac`** API:
  - JSR 198: A Standard Extension API for Integrated Development Environments
  - JSR 269: Pluggable Annotation Processing API
  - Tree API: **`com.sun.source.tree`**, **`com.sun.source.util`**
  - Runs on Java Platform, Standard Edition 5 or later, but supports all sources

Jackpot Overview

- Jackpot is a technology for:
  - Searching Java source code
  - Safely and correctly transforming patterns in code
  - Writing minimal changes back to source
- Jackpot relies heavily on the new **`javac`** API:
  - JSR 198: A Standard Extension API for Integrated Development Environments
  - JSR 269: Pluggable Annotation Processing API
  - Tree API: **`com.sun.source.tree`**, **`com.sun.source.util`**
  - Runs on Java SE 5 or later, but supports all sources
- Tightly integrated with NetBeansTM IDE

Is Jackpot Another Refactoring Tool?

Is Jackpot Another Refactoring Tool?

- **No:** Refactoring tools usually focus on individual changes
    - Jackpot focuses on project-wide transformations

Is Jackpot Another Refactoring Tool?

- No: Refactoring tools usually focus on individual changes
    - Jackpot focuses on project-wide transformations
- **No:** Refactoring tools can be difficult to extend
    - Jackpot has a pattern language and API for creating custom transformations

Is Jackpot Another Refactoring Tool?

- No: Refactoring tools usually focus on individual changes
  - Jackpot focuses on project-wide transformations
- No: Refactoring tools can be difficult to extend
  - Jackpot has a pattern language and API for creating custom transformations
- **Yes:** Refactorings can use the Jackpot engine
  - Future NetBeansTM based refactorings will use Jackpot

The Jackpot Developer Module

- Adds Jackpot extension support to NetBeans IDE

The Jackpot Developer Module

- Adds Jackpot extension support to NetBeans IDE
- Provides:
  - Rule, Query and Transformer templates
  - Rule file editor with syntax coloring
  - Test bench for rule files
  - Class Libraries with Javadoc™ documentation
    - Jackpot API
    - Tree API (**`com.sun.source.*, javax.lang.model.*`**)

The Jackpot Developer Module

- Adds Jackpot extension support to NetBeans IDE
- Provides:
  - Rule, Query and Transformer templates
  - Rule file editor with syntax coloring
  - Test bench for rule files
  - Class Libraries with Javadoc™ documentation
    - Jackpot API
    - Tree API (**`com.sun.source.*, javax.lang.model.*`**)
- Module not required for Jackpot use

Jackpot Concepts

- Rule
  - A search and replace expression with optional tests
  - Uses Java based statements and expressions
  - Rule file: a text file with one or more rules

Jackpot Concepts

- Rule
  - A search and replace expression with optional tests
  - Uses Java based statements and expressions
  - Rule file: a text file with one or more rules
- Query
  - A Java class used to search for patterns, usages

Jackpot Concepts

- Rule
  - A search and replace expression with optional tests
  - Uses Java based statements and expressions
  - Rule file: a text file with one or more rules
- Query
  - A Java class used to search for patterns, usages
- Transformation
  - A query class which replaces any patterns found

Jackpot Concepts

- Rule
  - A search and replace expression with optional tests
  - Uses Java based statements and expressions
  - Rule file: a text file with one or more rules
- Query
  - A Java class used to search for patterns, usages
- Transformation
  - A query class which replaces any patterns found
- Query Set
  - A group of queries and transformations run together

Rule Language

- Match using code fragment
  - Define meta-variables using “**$***identifier*”
  - A meta-list is defined using “**$***identifier***$**”
  - *Implies* token (→) defines action to take on match

```
$a ? $b : $c => note(“conditional found”);

if ($v == null)
    throw new
java.lang.NullPointerException();
else
    $v.$m($args$)
  => $v.$m($args$);
```

Guard Expressions

- Restricts match as necessary
  - Use *SuchThat* token (`::`) with conditional expression
  - Built-in expressions
    - sideEffectFree(), assignedIn(), couldThrow(), etc.

```
new java.lang.String($s)=>$s ::
    $s instanceof java.lang.String;

$a ^ $a => 0 :: sideEffectFree($a);
```

Creating a Rule File

- Create rule file
  - Click New..., select Jackpot→Rule File

    or
  - Open Tools→Refactoring Manager, click New Query...

Creating a Rule File

- Create rule file
    - Click New..., select Jackpot→Rule File

or

    - Open Tools→Refactoring Manager, click New Query...
- Add sample test code

Creating a Rule File

- Create rule file
  - Click New..., select Jackpot→Rule File

    or
  - Open Tools→Refactoring Manager, click New Query...
- Add sample test code
- Edit rule
  - Start by copying Java code fragment(s) from test code

Creating a Rule File

- Create rule file
    - Click New..., select Jackpot→Rule File

      or
    - Open Tools→Refactoring Manager, click New Query...
- Add sample test code
- Edit rule
    - Start by copying Java code fragment(s) from test code
- Review results

Creating a Rule File

- Create rule file
  - Click New..., select Jackpot→Rule File

or

  - Open Tools→Refactoring Manager, click New Query...
- Add sample test code
- Edit rule
  - Start by copying Java code fragment(s) from test code
- Review results
- Rinse and repeat

DEMO

Creating a Rule File

Creating Query Classes

- Create or open a project to hold class
  - Use Module Project if distributing as module
  - Set Source Level to 1.5

Creating Query Classes

- Create or open a project to hold class
  - Use Module Project if distributing as module
  - Set Source Level to 1.5
- Select New→Jackpot→Query Class

Creating Query Classes

- Create or open a project to hold class
  - Use Module Project if distributing as module
  - Set Source Level to 1.5
- Select New→Jackpot→Query Class
- Open Library Manager
  - Add Jackpot Engine
  - Add Javac API, Utilities API

Creating Query Classes

- Create or open a project to hold class
  - Use Module Project if distributing as module
  - Set Source Level to 1.5
- Select New→Jackpot→Query Class
- Open Library Manager
  - Add Jackpot Engine
  - Add Javac API, Utilities API
- Override visitor methods to inspect nodes

Creating Query Classes

- Create or open a project to hold class
    - Use Module Project if distributing as module
    - Set Source Level to 1.5
- Select New→Jackpot→Query Class
- Open Library Manager
    - Add Jackpot Engine
    - Add Javac API, Utilities API
- Override visitor methods to inspect nodes
- Use `addResult()` to record query results

Creating Transformer Classes

- Same steps as for Inspector classes
  - Use New→Jackpot→Transformer Class

Creating Transformer Classes

- Same steps as for Inspector classes
  - Use New→Jackpot→Transformer Class
- Model is immutable, so:
  - Use **`org.netbeans.jackpot.tree.TreeMaker`**
    - New trees using factory methods
    - Copy existing tree with **`createMutableTree()`**
  - Add changes to transformer's **`ChangeSet`**
    - Model is rewritten on commit
  - Undo restores original model

Add a Query Command

- Open Tools→Refactoring Manager
- Click Import...
- Give a descriptive name
- Specify rule file
  - File is not imported into NetBeans file system
    - Changes are automatically included

DEMO

Create an Inspector and Transformer class

Deploying Rule Files

- Easiest: distribute as text
  - Rules can be shared as email, web pages, blogs, etc.
  - Store as ordinary text in version control systems

Deploying Rule Files

- Easiest: distribute as text
  - Rules can be shared as email, web pages, blogs, etc.
  - Store as ordinary text in version control systems
- Issues:
  - Users must install command
  - Hard to update

Deploying Rule Files

- Easiest: distribute as text
  - Rules can be shared as email, web pages, blogs, etc.
  - Store as ordinary text in version control systems
- Issues:
  - Users must install command
  - Hard to update
- Alternative: Distribute NetBeans module
  - Full control over files, command names, installation
  - Easy maintenance via Update Center

Deploy With NetBeans Module

- Create module
  - Create new Module Project
  - Copy rule files and/or query classes into it
  - Copy relevant XML layer entries from your *userdir*
  - Register query classes as services
  - Optional: set project's API Versioning properties
  - Build using Create NBM target

Deploy With NetBeans Module

- Create module
  - Create new Module Project
  - Copy rule files and/or query classes into it
  - Copy relevant XML layer entries from your *userdir*
  - Register query classes as services
  - Optional: set project's API Versioning properties
  - Build using Create NBM target
- Install module
  - Open in NBM in NetBeans software, follow Update Center steps

Deploy With NetBeans Module

- Create module
  - Create new Module Project
  - Copy rule files and/or query classes into it
  - Copy relevant XML layer entries from your *userdir*
  - Register query classes as services
  - Optional: set project's API Versioning properties
  - Build using Create NBM target
- Install module
  - Open in NBM in NetBeans software, follow Update Center steps
- Optional: Publish Update Center file

Summary

- Jackpot is extensible
  - Queries report facts about your Java based projects
  - Transformers make global changes to them
  - Write new ones using the rule language or Java code

Summary

- Jackpot is extensible
  - Queries report facts about your Java based projects
  - Transformers make global changes to them
  - Write new ones using the rule language or Java code
- New queries are easily added to NetBeans IDE software

Summary

- Jackpot is extensible
    - Queries report facts about your Java based projects
    - Transformers make global changes to them
    - Write new ones using the rule language or Java code
- New queries are easily added to NetBeans IDE software
- Jackpot extensions can be shared and deployed
    - Helps projects maintain standards

For More Information

- Related Sessions
  - TS-1387: Twelve Reasons to Use NetBeans™ Software
  - TS-1512: Effective Java™ Reloaded
  - TS-1188: The Continuing Adventures of Java™ Puzzlers: Tiger Traps
- URLs
  - Jackpot: http://jackpot.netbeans.org/
  - NetBeans: http://www.netbeans.org/

Q&A

Creating and Deploying Custom Jackpot Queries and Transformers

Tom Ball

Senior Staff Engineer
Sun Microsystems, Inc.
http://jackpot.netbeans.org

TS-1278