# Java

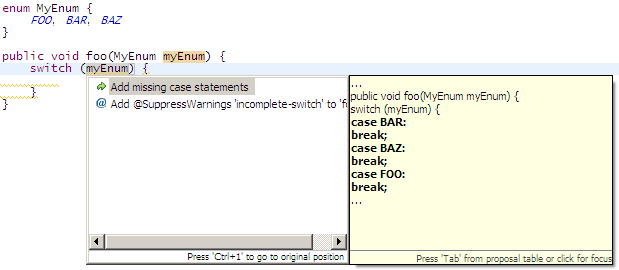
## Switch my\_enum

It has been in Eclipse for ages. It's admittedly only a bit hard to find. First start with

switch (myEnum) {

}

At that point, your cursor would usually be inside the statement block {}. You need to put your cusror back to the line with the switch keyword and press Ctrl+1 and choose Add missing case statements. This way it will insert any possible case.



You'd intuitively expect this option to be available inside the statement block {} as well, but no.

## Java Regex : Validate Credit Card Numbers

[NOVEMBER 18, 2014](http://howtodoinjava.com/2014/11/18/java-regex-validate-credit-card-numbers/) [LOKESH](http://howtodoinjava.com/author/lokeshgupta1981/)[LEAVE A COMMENT](http://howtodoinjava.com/2014/11/18/java-regex-validate-credit-card-numbers/#respond)

 In this java regex tutorial, we will Learn to use regular expressions to validate Credit Card Numbers. In this tutorial, we will learn about format and validations of Credit Card Numbers from multiple providers such as VISA, Mastercard, Amex and Diners etc.

## Valid Credit Card Numbers Format

On an actual credit card, the digits of the embossed card number are usually placed into groups of four. That makes the card number easier for humans to read. Each of the credit card companies uses this number format. We’ll exploit that difference of formats between each company to allow users to enter a number without specifying a company. The company can be determined from the number. The format for each company is:

* **Visa** : 13 or 16 digits, starting with 4.
* **MasterCard** : 16 digits, starting with 51 through 55.
* **Discover** : 16 digits, starting with 6011 or 65.
* **American Express** : 15 digits, starting with 34 or 37.
* **Diners Club** : 14 digits, starting with 300 through 305, 36, or 38.
* **JCB** : 15 digits, starting with 2131 or 1800, or 16 digits starting with 35.

Below given regex assumes that before performing the check for a valid number, you will search-and-replace all spaces and hyphens explicitly.

With spaces and hyphens stripped from the input, the next regular expression checks if the credit card number uses the format of any of the six major credit card companies. It uses named capture to detect which brand of credit card the customer has. If you don’t need to determine which type the card is, you can remove the six capturing groups that surround the pattern for each card type, as they don’t serve any other purpose.

If you accept only certain brands of credit cards, you can delete the cards that you don’t accept from the regular expression. For example, when deleting JCB, make sure to delete the last remaining “|” in the regular expression as well. If you end up with “|” in your regular expression, it will accept the empty string as a valid card number as well.

Regex : ^(?:(?4[0-9]{12}(?:[0-9]{3})?)|

(?5[1-5][0-9]{14})|

(?6(?:011|5[0-9]{2})[0-9]{12})|

(?3[47][0-9]{13})|

(?3(?:0[0-5]|[68][0-9])?[0-9]{11})|

(?(?:2131|1800|35[0-9]{3})[0-9]{11}))$

Read more about credit card number formats in this [**wiki page**](http://en.wikipedia.org/wiki/Bank_card_number).

## Example Code to Validate Credit Card Numbers

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34 | public static void main(String[] args)  {  List<String> cards = new ArrayList<String>();    //Valid Credit Cards  cards.add("xxxx-xxxx-xxxx-xxxx");  //Masked to avoid any inconvenience unknowingly    //Invalid Credit Card  cards.add("xxxx-xxxx-xxxx-xxxx"); //Masked to avoid any inconvenience unknowingly    String regex = "^(?:(?<visa>4[0-9]{12}(?:[0-9]{3})?)|" +          "(?<mastercard>5[1-5][0-9]{14})|" +          "(?<discover>6(?:011|5[0-9]{2})[0-9]{12})|" +          "(?<amex>3[47][0-9]{13})|" +          "(?<diners>3(?:0[0-5]|[68][0-9])?[0-9]{11})|" +          "(?<jcb>(?:2131|1800|35[0-9]{3})[0-9]{11}))$";    Pattern pattern = Pattern.compile(regex);    for (String card : cards)  {      //Strip all hyphens      card = card.replaceAll("-", "");        //Match the card      Matcher matcher = pattern.matcher(card);        System.out.println(matcher.matches());        if(matcher.matches()) {          //If card is valid then verify which group it belong          System.out.println(matcher.group("mastercard"));      }  } |

## Checksum Validation with the Luhn Algorithm

There is an extra validation check that you can do on the credit card number before processing the order. The last digit in the credit card number is a checksum calculated according to the [**Luhn algorithm**](http://en.wikipedia.org/wiki/Luhn_algorithm). Since this algorithm requires basic arithmetic, you cannot implement it with a regular expression.

Below is the method which you can use to run checksum validation using Luhn Algorithm. This function takes a string with the credit card number as a parameter. The card number should consist only of digits.

The actual algorithm runs on the array of digits, calculating a checksum. If the sum modulus 10 is zero, then the card number is valid. If not, the number is invalid.

I have taken the reference implementation of Luhn Algo from [**Google Code**](https://code.google.com/p/gnuc-credit-card-checker/source/browse/trunk/CCCheckerPro/src/com/gnuc/java/ccc/Luhn.java).

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23 | public class Luhn  {      public static boolean Check(String ccNumber)      {          int sum = 0;          boolean alternate = false;          for (int i = ccNumber.length() - 1; i >= 0; i--)          {              int n = Integer.parseInt(ccNumber.substring(i, i + 1));              if (alternate)              {                  n \*= 2;                  if (n > 9)                  {                      n = (n % 10) + 1;                  }              }              sum += n;              alternate = !alternate;          }          return (sum % 10 == 0);      }  } |

Feel free to modify above code samples to match other validation rules in above regex, if you need it.

**Happy Learning !!**

### Related Posts:

1. [Java Regex : Validate Social Security Numbers (SSN)](http://howtodoinjava.com/2014/11/14/java-regex-validate-social-security-numbers-ssn/)
2. [Java Regex : Validate International Phone Numbers](http://howtodoinjava.com/2014/11/12/java-regex-validate-international-phone-numbers/)
3. [Java Regex : Validate and Format North American Phone Numbers](http://howtodoinjava.com/2014/11/12/java-regex-validate-and-format-north-american-phone-numbers/)
4. [Java Regex : Validate International Standard Book Number (ISBNs)](http://howtodoinjava.com/2014/11/17/java-regex-validate-international-standard-book-number-isbns/)
5. [Java Regex : Validate U.K. Postal Codes (Postcodes)](http://howtodoinjava.com/2014/11/17/java-regex-validate-u-k-postal-codes-postcodes/)
6. [Java Regex : Validate Canadian Postal Zip Codes](http://howtodoinjava.com/2014/11/17/java-regex-validate-canadian-postal-zip-codes/)

Use proper HTTP status codes   
For a **DELETE** request: **HTTP 200** or **HTTP 204** should imply "resource deleted successfully". **HTTP 202**can also be returned which would imply that the instruction was accepted by the server and the "resource was marked for deletion".

<http://tools.ietf.org/html/rfc2616#section-10.2.3>

(https://bourgeois.me/rest/)

Always use proper HTTP status codes when returning content (for both successful and error requests). Here a quick list of non common codes that you may want to use in your application.

### Success codes

* 201 Created should be used when creating content (INSERT),
* 202 Accepted should be used when a request is queued for background processing (async tasks),
* 204 No Content should be used when the request was properly executed but no content was returned (a good example would be when you delete something).

### Client error codes

* 400 Bad Request should be used when there was an error while processing the request payload (malformed JSON, for instance).
* 401 Unauthorized should be used when a request is not authenticiated (wrong access token, or username or password).
* 403 Forbidden should be used when the request is successfully authenticiated (see 401), but the action was forbidden.
* 406 Not Acceptable should be used when the requested format is not available (for instance, when requesting an XML resource from a JSON only server).
* 410 Gone Should be returned when the requested resource is permenantely deleted and will never be available again.
* 422 Unprocesable entity Could be used when there was a validation error while creating an object.

A more complete list of status codes can be found in [RFC2616](http://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html).

## Always return a consistent error payload

When an exception is raised, you should always return a consistent payload describing the error. This way, it will be easier for other to parse the error message (the structure will always be the same, whatever the error is).

Here one I often use in my web applications. It is clear, simple and self descriptive.

HTTP/1.1 401 Unauthorized

{

"status": "Unauthorized",

"message": "No access token provided.",

"request\_id": "594600f4-7eec-47ca-8012-02e7b89859ce"

}

# Java EE

<http://www.dataart.ru/blog/2013/03/web-application-with-spring-and-hibernate-from-scratch/>

Авторы Bruce A. Tate и Justin Gehtland

Доклады по паттернам в confluence

http://10.233.106.211:8090/display/MDKB/Patterns+and+more#Patternsandmore-Ссылкинаматериалы

# JavaScript

<http://citforum.ru/internet/js_tut/index.shtml>

[**http://learn.javascript.ru/first-steps**](http://learn.javascript.ru/first-steps)

# Spring

http://docs.spring.io/spring/docs/3.1.x/spring-framework-reference/html/mvc.html#mvc-ann-exceptionhandler

## Spring Exceptions

Answer about errors in json format

<http://springinpractice.com/2013/10/09/generating-json-error-object-responses-with-spring-web-mvc>

### Generating JSON Error Object Responses With Spring Web MVC

OCT 9TH, 2013 | [COMMENTS](http://springinpractice.com/2013/10/09/generating-json-error-object-responses-with-spring-web-mvc#disqus_thread)

The other day I wrote a post called [Handling JSON error object responses with Spring’s RestTemplate](http://springinpractice.com/2013/10/07/handling-json-error-object-responses-with-springs-resttemplate/). Judging by the Twitter activity, people found it useful, so this time around I’m going to write about the other side of the equation, which is generating the JSON error objects using Spring Web MVC. Something like this:

{

"code": "InvalidRequest",

"message": "Invalid doodad",

"fieldErrors": [

{

"resource": "doodadResource",

"field": "key",

"code": "NotNull",

"message": "may not be null"

},

{

"resource": "doodadResource",

"field": "name",

"code": "NotNull",

"message": "may not be null"

}

]

}

There are various ways to do this, but Spring 3.2 introduces a pretty elegant approach via the @ControllerAdviceannotation. The basic concept here is that we can define AOP-like “advice” around Spring Web MVC controllers. This advice captures exceptions and then maps them to JSON objects, which the advice sends in the response body. Of course we can also send the appropriate HTTP status code in the headers too.

(You can find out more about @ControllerAdvice and @ExceptionHandler in the post [Error Handling for REST with Spring 3](http://www.baeldung.com/2013/01/31/exception-handling-for-rest-with-spring-3-2/) by [Eugen Paraschiv](https://twitter.com/baeldung).)

Note that the error-triggering event doesn’t really have to be an exception per se. For example, we might want bean validation errors or authorization errors—neither of which manifests itself as an exception—to map to JSON error objects. The key is to have these triggers generate exceptions that we can capture using the @ControllerAdvice component.

Let’s look at an example involving bean validation. Here we have a controller. If the incoming resource is invalid, we want to generate a JSON error object. So first we do this:

package myapp.web.controller;

... various imports ...

@Controller

@RequestMapping("/doodads")

public class DoodadController {

@Inject private DoodadService doodadService;

@RequestMapping(

value = "/{id}",

method = RequestMethod.PUT,

consumes = MediaType.APPLICATION\_JSON\_VALUE)

public void updateDoodad(

@PathVariable Long id,

@RequestBody @Valid DoodadResource doodad,

BindingResult bindingResult) {

if (bindingResult.hasErrors()) {

throw new InvalidRequestException("Invalid doodad", bindingResult);

}

doodadService.updateDoodad(doodad);

}

}

Note that InvalidRequestException is just a custom exception I created that takes an Errors object as an argument. (BindingResult implements Errors.) Just for completeness, here’s InvalidRequestException:

package myapp.exception;

import org.springframework.validation.Errors;

@SuppressWarnings("serial")

public class InvalidRequestException extends RuntimeException {

private Errors errors;

public InvalidRequestException(String message, Errors errors) {

super(message);

this.errors = errors;

}

public Errors getErrors() { return errors; }

}

So far so good. But now we need that @ControllerAdvice to capture the InvalidRequestException and generate the JSON error object:

package myapp.web.controller;

import java.util.ArrayList;

import java.util.List;

import org.springframework.http.HttpHeaders;

import org.springframework.http.HttpStatus;

import org.springframework.http.MediaType;

import org.springframework.http.ResponseEntity;

import org.springframework.validation.FieldError;

import org.springframework.web.bind.annotation.ControllerAdvice;

import org.springframework.web.bind.annotation.ExceptionHandler;

import org.springframework.web.context.request.WebRequest;

import org.springframework.web.servlet.mvc.method.annotation.ResponseEntityExceptionHandler;

import myapp.binding.ErrorResource;

import myapp.binding.FieldErrorResource;

import myapp.exception.InvalidRequestException;

@ControllerAdvice

public class MyExceptionHandler extends ResponseEntityExceptionHandler {

@ExceptionHandler({ InvalidRequestException.class })

protected ResponseEntity<Object> handleInvalidRequest(RuntimeException e, WebRequest request) {

InvalidRequestException ire = (InvalidRequestException) e;

List<FieldErrorResource> fieldErrorResources = new ArrayList<>();

List<FieldError> fieldErrors = ire.getErrors().getFieldErrors();

for (FieldError fieldError : fieldErrors) {

FieldErrorResource fieldErrorResource = new FieldErrorResource();

fieldErrorResource.setResource(fieldError.getObjectName());

fieldErrorResource.setField(fieldError.getField());

fieldErrorResource.setCode(fieldError.getCode());

fieldErrorResource.setMessage(fieldError.getDefaultMessage());

fieldErrorResources.add(fieldErrorResource);

}

ErrorResource error = new ErrorResource("InvalidRequest", ire.getMessage());

error.setFieldErrors(fieldErrorResources);

HttpHeaders headers = new HttpHeaders();

headers.setContentType(MediaType.APPLICATION\_JSON);

return handleExceptionInternal(e, error, headers, HttpStatus.UNPROCESSABLE\_ENTITY, request);

}

... other handlers for other exceptions ...

}

The important pieces here are @ControllerAdvice (which derives from @Controller, so we can component scan it), ResponseEntityExceptionHandler (provides the handleExceptionInternal() method), and @ExceptionHandlerannotation. @ExceptionHandler accepts an array of match exceptions, and then its implementation builds the JSON error object, which here involves custom ErrorResource and FieldErrorResource beans that can be whatever we want to display to the client. Finally we pass response-related information to handleExceptionInternal(), where the error object ends up as the response body. Here we’re using “Unprocessable Entity” (HTTP 422), a WebDAV extension to HTTP, since “Bad Request” (HTTP 400) is for syntactic rather than semantic errors. (See [HTTP Status Codes For Invalid Data: 400 vs. 422](http://www.bennadel.com/blog/2434-HTTP-Status-Codes-For-Invalid-Data-400-vs-422.htm) by Ben Nadel for more information.)

Again in the interest of completeness, here are the error objects I’m using. These are just examples of what’s possible; choose error representations that fit your needs. First, the top-level error object:

package myapp.binding;

import java.util.List;

import com.fasterxml.jackson.annotation.JsonIgnoreProperties;

@JsonIgnoreProperties(ignoreUnknown = true)

public class ErrorResource {

private String code;

private String message;

private List<FieldErrorResource> fieldErrors;

public ErrorResource() { }

public ErrorResource(String code, String message) {

this.code = code;

this.message = message;

}

public String getCode() { return code; }

public void setCode(String code) { this.code = code; }

public String getMessage() { return message; }

public void setMessage(String message) { this.message = message; }

public List<FieldErrorResource> getFieldErrors() { return fieldErrors; }

public void setFieldErrors(List<FieldErrorResource> fieldErrors) {

this.fieldErrors = fieldErrors;

}

}

And finally here’s a class for field errors:

package myapp.binding;

import com.fasterxml.jackson.annotation.JsonIgnoreProperties;

@JsonIgnoreProperties(ignoreUnknown = true)

public class FieldErrorResource {

private String resource;

private String field;

private String code;

private String message;

public String getResource() { return resource; }

public void setResource(String resource) { this.resource = resource; }

public String getField() { return field; }

public void setField(String field) { this.field = field; }

public String getCode() { return code; }

public void setCode(String code) { this.code = code; }

public String getMessage() { return message; }

public void setMessage(String message) { this.message = message; }

}

Have fun!

## Writing Unit Tests for a REST API

Resorce:

http://www.petrikainulainen.net/programming/spring-framework/unit-testing-of-spring-mvc-controllers-rest-api/

Add to the build.gradle

testCompile **group**:'org.hamcrest', **name**:'hamcrest-all', **version**:'1.3'

testCompile **group**: 'org.springframework', **name**: 'spring-test', **version**:'3.2.3.RELEASE'

testCompile **group**: 'org.mockito', **name**: 'mockito-core', **version**:'1.9.5'

testCompile **group**: 'junit', **name**: 'junit', **version**:'4.11'

testCompile **group**: 'com.jayway.jsonpath', **name**: 'json-path', **version**:'0.9.1'

testCompile **group**: 'com.jayway.jsonpath', **name**: 'json-path-assert', **version**:'0.9.1'

## Использование Spring в OSGi-контейнере

[JAVA](http://habrahabr.ru/hub/java/)\*

  
Вряд ли найдётся Java разработчик, который не знает что такое Spring Framework. Одними из базовых технологий данного фреймворка являются IoC контейнер и поддержка AOP. Эти технологии позволяют успешно разбивать архитектуру приложения на обособленные слои, как на уровне классов, так и на уровне компоновки объектов во время выполнения. Казалось бы, что приложение отлично структурировано на элементы/слои, но по своей сути оно остаётся монолитным. Монолитным во время выполнения (runtime)! Только в рамках данного фреймворка не существует универсальных решений этой проблемы. Чуть меньшее количество Java разработчиков слышало об OSGi. Это спецификация модульных систем для Java платформы. Использование конкретной реализации данной спецификации в качестве основы приложения позволяет сделать его модульным, как во время выполнения, так и на физическом уровне (уровне файлов). О синергии этих технологий и пойдёт речь в этой статье.

#### Spring Framework

Я рассчитываю, что читатель знаком с данным фреймворком хотя бы на уровне прочтения первого десятка глав из официального руководства. Не представляется возможным описать всё это в одной статье. Поэтому отсылаю всех, кому это нужно, к официальному руководству. Ниже приведу лишь краткую справку по основным компонентам Spring Framework.

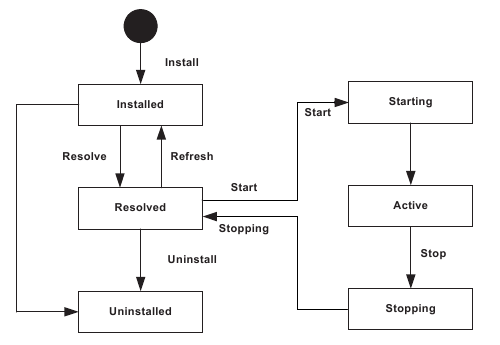
* IoC контейнер с реализацией паттерна DI — механизм управляющий созданием, конфигурированием и связыванием объектов во время выполнения, базовая неотъемлемая часть фреймвока.
* AOP — поддержка механизмов аспекнто-ориентированного программирования. Базовый механизм, который используют множество подсистем.
* Интеграция с источниками данных, поддержка ORM и поддержка транзакций — механизмы обеспечивающие взаимодействие с различными источниками данных от XML до СУБД, интеграцию со сторонними фреймворками в данных областях, поддержка локальных и глобальных транзакций, трансляция специфичных иерархий исключений доступа к источникам данных в собственную иерархию и прочее.
* Web компоненты — обеспечивают различные механизмы от собственной реализации паттерна MVC до интеграции с различными фреймворками связанными с Web (JSF, Struts, WS и прочее).

Множество различных субпроектов из портфолио SpringSource.

#### Модульность

В идеале, действительно модульная программа должна состоять из элементов, которые обладают следующими ярко выраженными признаками: слабая связанность (взаимодействие через чётко выраженный интерфейс, лёгкая заменяемость, многократное использование), инкапсуляция (модуль рассматривается снаружи как чёрный ящик с определенным интерфейсом взаимодействия), динамичность (возможность изменять множество модулей во время выполнения). Java позволяет реализовать слабую связанность и инкапсуляцию, как на уровне классов, так и на уровне пакетов. Конечно есть способы нарушить инкапсуляцию на уровне модификаторов доступа (private, protected) с помощью рефлексии (Reflection API) и тем самым связать свой код с внутренними механизмами стороннего кода, либо использовать механизмы изменения кода классов (javassist, cglib, изменения кода классов вручную и т.д.), но это скорее хаки. А вот что на счёт динамичности, то есть возможности заменять модули во время выполнения. В данной области дела обстоят хуже. Можно конечно реализовать поддержку динамичности переопределяя загрузчики классов (ClassLoader) своими реализациями, но это очень большой объём работы, который за нас уже проделали разработчики реализаций стандарта OSGi. Любой контейнер OSGi уже поддерживает модульность на уровне базовой архитектуры и API.

#### OSGi

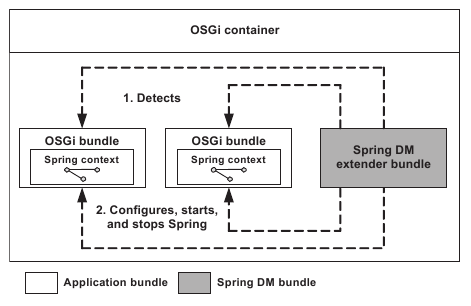
Open Services Gateway initiative — спецификация имеющая множество реализаций. Основные Open Source реализации это Apache Felix, Equinox и Knopflerfish. Данная спецификация описывает модульную систему, которая динамически может связывать различные модули (bundles). Состав модулей может изменяться во время выполнения. Взаимодействие между модулями осуществляется с помощью сервисов, которые зарегистрированы в регистре сервисов (Service Register). Модули обладают жизненным циклом, который состоит из нескольких состояний (INSTALLED, RESOLVED, STARTING, ACTIVE, STOPPING, UNINSTALLED). Жизненным циклом модуля управляет OSGi-контейнер.  
  
  
Диаграмма состояний модуля  
  
OSGi-модуль (bundle) должен иметь дополнительные метаданные в файле META-INF/MANIFEST.MF. Некоторые из основных представлены ниже:

* Bundle-Name — удобочитаемое имя модуля;
* Bundle-Version — версия модуля в формате число[.число[.число[.строка]]], по-умолчанию версия 0.0.0;  
  Bundle-SymbolicName — символьное имя модуля, совместно с версией служат уникальным идентификаторов модуля;
* Export-Package — список экспортируемых Java-пакетов модуля с возможными дополнительными директивами;
* Import-Package — список импортируемых Java-пакетов используемых в модуле с возможными дополнительными директивами.

#### Как появилась возможность использовать Spring вместе с OSGi

OSGi реализации позволяют создавать действительно модульные программы на Java. Spring Framework даёт возможность избавиться от ручной поддержки связей между объектами благодаря использованию IoC контейнера, расширить функционал существующих классов с помощью AOP, и использовать любые сопутствующие технологии из портфолио SpringSource. Объединение этих технологий должно приносить немалую пользу при разработке программного обеспечения. Неудивительно, что эта мысль пришла в голову нескольким людям из Interface21 (позднее SpringSource) в 2006 году и они смогли подключить к разработке людей из OSGi Alliance, BEA, Oracle, IBM. В результате совместных усилий появился продукт Spring dm под крылом SpringSource, позже этот продукт дал жизнь спецификации Blueprint в OSGi Service Platform Service Compendium. Его кодовая база перешла под опеку Eclipse Foundation и получила название Gemini Blueprint. SpringSource также выпускала dm Server, теперь его кодовая база стала проектом Virgo в Eclipse Foundation.

#### Архитектура Gemini Blueprint (Spring DM)

Каким же образом Gemini Blueprint позволяет использовать Spring Framework в OSGi-контейнере? Для ответа на этот вопрос необходимо рассмотреть основные моменты архитектуры Gemini Blueprint.  
  
Базовым элементом архитектуры является extender (реализация extender паттерна OSGi). Extender отслеживает события запуска новых модулей (bundle) в сервисной платформе OSGi и проверяет поддерживает ли данный модуль спецификацию blueprint. В случае поддержки модулем данной спецификации, extender создает Spring-контекст модуля и инициализирует его. Таким образом каждый Spring-модуль имеет собственный Spring-контекст. В добавок, из extender вызывается код, который ответственен за публикацию Spring-bean в качестве сервисов OSGi и получения ссылки на сервисы OSGi, и прочее.  
  
  
Схема функционирования extender  
  
Всю информацию на базе которой extender создаёт контекст берётся из конфигурации в виде xml-файлов. Здесь необходимо подчеркнуть существование двух возможных вариантов конфигурации spring и blueprint. Оба варианта во многом схожи, но если первый соответствует родной Spring конфигурации с дополнительными именованными пространствами для конфигурирования сервисов, то второй чётко определён спецификацией Blueprint OSGi Service Platform Service Compendium. На самом деле есть ещё и третий вариант, обратно совместимый с Spring dm.  
Для более подробного изучения архитектуры смотрите ссылки в конце статьи.

#### Пример использования Gemini Blueprint и OSGi

В качестве примера использования, напишем GUI приложение с двумя реализациями сервисов и одним клиентом. Этот пример продемонстрирует насколько использование Gemini Blueprint облегчает разработку OSGi-программ. Во-первых, нам не понадобится использование OSGi API. Во-вторых, вопрос динамичности сервисов решается самостоятельно Gemini Blueprint, нам лишь останется позаботиться только о логике приложения. В-третьих, мы будем использовать Spring Framework для компоновки приложения.  
  
Сборка приложения осуществляется с помощью maven 3. Файловая структура проекта выглядит следующим образом:

blueprint-example

├── consumer

│   ├── pom.xml

│   └── src

│   ├── main

│   │   ├── java

│   │   │   └── blueprint

│   │   │   └── example

│   │   │   └── consumer

│   │   │   ├── ConsumerFrame.java

│   │   │   ├── Consumer.java

│   │   │   └── RefreshListener.java

│   │   └── resources

│   │   └── META-INF

│   │   └── spring

│   │   ├── osgi-context.xml

│   │   └── spring-context.xml

│   └── test

│   ├── java

│   └── resources

├── date-producer

│   ├── pom.xml

│   └── src

│   ├── main

│   │   ├── java

│   │   │   └── blueprint

│   │   │   └── example

│   │   │   └── dateproducer

│   │   │   └── ProducerImpl.java

│   │   └── resources

│   │   └── META-INF

│   │   └── spring

│   │   ├── osgi-context.xml

│   │   └── spring-context.xml

│   └── test

│   ├── java

│   └── resources

├── int-producer

│   ├── pom.xml

│   └── src

│   ├── main

│   │   ├── java

│   │   │   └── blueprint

│   │   │   └── example

│   │   │   └── intproducer

│   │   │   └── ProducerImpl.java

│   │   └── resources

│   │   └── META-INF

│   │   └── spring

│   │   ├── osgi-context.xml

│   │   └── spring-context.xml

│   └── test

│   ├── java

│   └── resources

├── pom.xml

└── producer-api

├── pom.xml

└── src

├── main

│   ├── java

│   │   └── blueprint

│   │   └── example

│   │   └── producer

│   │   └── api

│   │   └── Producer.java

│   └── resources

│   └── META-INF

└── test

├── java

└── resources

В директориях consumer, producer-api, date-producer и int-producer расположены соответственно проекты клиента, API сервиса, реализации сервисов на основе даты и целого числа. Звучит запутанно? Не волнуйтесь, сейчас разберём всё по порядку.  
  
Каждый из перечисленных проектов представляет собой обособленный модуль (bundle) для OSGi-контейнера. Проект producer-api содержит всего один файл с интерфейсом Producer. Это типичный пример выделения API OSGi-приложения в отдельный модуль, который будут импортировать все нуждающиеся в нём модули. Этот модуль не является blueprint-модулем, потому что его цель только в обеспечении необходимыми классами.  
  
Для сборки всех модулей используется maven-bundle-plugin. Это плагин от разработчиков Apache Felix, который использует внутри код (знаменитой в OSGi кругах) утилиты [Bnd](http://www.aqute.biz/Bnd/Bnd). Этот плагин конфигурируется следующим образом:

<plugin>

<groupId>org.apache.felix</groupId>

<artifactId>maven-bundle-plugin</artifactId>

<version>2.3.5</version>

<extensions>true</extensions>

<configuration>

<instructions>

<Bundle-Name>

${name} ${version}

</Bundle-Name>

<Bundle-SymbolicName>

${groupId}.${artifactId}

</Bundle-SymbolicName>

<Export-Package>

blueprint.example.producer.api

</Export-Package>

<Import-Package>\*</Import-Package>

</instructions>

</configuration>

</plugin>

Внимательный читатель должен заметить, что наименование записей в секции instructions совпадают с наименованиями специфических заголовков для метаданных OSGi-модуля в META-INF/MANIFEST.MF. Это они и есть, только не в своём натуральном виде, а в виде инструкций bnd. Эта утилита достаточно интеллектуальна и умеет вычислять зависимости по коду и подставлять их в секцию Import-Package, а также добавлять различные метаданные. На основе своей конфигурации и вычислений она генерирует MANIFEST.MF. В данном случае, мы определяем только основные заголовки.  
  
Для того, чтобы extender распознал модуль как blueprint-модуль, по-умолчанию, используется следующее правило: модуль должен иметь xml файлы конфигурации в директории META-INF/spring, либо должен присутствовать заголовок Spring-Context указывающий на месторасположение конфигурации. В проектах перечисленных выше используются по два файла конфигурации в директории META-INF/spring. Первый с префиксом «osgi-» содержит конфигурацию специфичную для blueprint. Второй с префиксом «spring-» содержит обычную конфигурацию для spring-приложения.  
Рассмотрим реализацию сервисов на основе даты и на основе произвольного целого числа:

@Component

public class ProducerImpl implements Producer {

@Override

public String produceString() {

return new Date().toString();

}

}

@Component

public class ProducerImpl implements Producer {

private Random random = new Random(new Date().getTime());

@Override

public String produceString() {

return String.valueOf(random.nextInt());

}

}

Сервис в нашем учебном примере это всего лишь класс с одним методом String produceString(), который возвращает произвольную строку. В первом случае локализованную дату, во втором строку с произвольным целым числом. В spring-конфигурации данных модулей задаётся процесс сканирования аннотированных классов. В osgi-конфигурации регистрируется сервис.

<?xml version="1.0" encoding="UTF-8"?>

<beans xmlns="http://www.springframework.org/schema/beans"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xmlns:context="http://www.springframework.org/schema/context"

xsi:schemaLocation="http://www.springframework.org/schema/beans

http://www.springframework.org/schema/beans/spring-beans-3.0.xsd

http://www.springframework.org/schema/context

http://www.springframework.org/schema/context/spring-context-3.0.xsd

">

<context:annotation-config/>

<context:component-scan base-package="blueprint.example"/>

</beans>

<?xml version="1.0" encoding="UTF-8"?>

<beans:beans xmlns="http://www.eclipse.org/gemini/blueprint/schema/blueprint"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xmlns:beans="http://www.springframework.org/schema/beans"

xsi:schemaLocation="

http://www.eclipse.org/gemini/blueprint/schema/blueprint

http://www.eclipse.org/gemini/blueprint/schema/blueprint/gemini-blueprint.xsd

http://www.springframework.org/schema/beans

http://www.springframework.org/schema/beans/spring-beans-3.0.xsd

">

<service id="producerService" ref="producerImpl"

interface="blueprint.example.producer.api.Producer"/>

</beans:beans>

Вот и обещанный пример упрощения жизни разработчика. Нет необходимости использовать OSGi API для регистрации сервиса, достаточно всего лишь заполнить xml-конфигурацию.  
  
Настало время рассмотреть последний модуль. Клиент наших сервисов выводит на экран GUI форму с текстовым полем отображения и кнопкой обновления значения. Давайте взглянем на код клиента.

@Component

public class Consumer {

@Autowired

private ConsumerFrame consumerFrame;

@Autowired

private Producer producer;

@PostConstruct

public void start() {

consumerFrame.setRefreshListener(new RefreshListener() {

@Override

public String refresh() {

return producer.produceString();

}

});

SwingUtilities.invokeLater(

new Runnable() {

@Override

public void run() {

consumerFrame.setVisible(true);

}

}

);

}

}

Класс Consumer очень простой. В нём находятся два атрибута, которые инжектируются контейнером. Это объект формы и ссылка на наш сервис. Далее в методе, который вызывается после конструирования объект, регистрируется слушатель события обновления с кодом обращения к сервису и код отображения формы на экран. Spring-конфигурация данного проекта выглядит идентично представленной выше, а osgi-конфигурация выглядит так:

<?xml version="1.0" encoding="UTF-8"?>

<beans:beans xmlns="http://www.eclipse.org/gemini/blueprint/schema/blueprint"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xmlns:beans="http://www.springframework.org/schema/beans"

xsi:schemaLocation="

http://www.eclipse.org/gemini/blueprint/schema/blueprint

http://www.eclipse.org/gemini/blueprint/schema/blueprint/gemini-blueprint.xsd

http://www.springframework.org/schema/beans

http://www.springframework.org/schema/beans/spring-beans-3.0.xsd

">

<reference id="producer"

interface="blueprint.example.producer.api.Producer"/>

</beans:beans>

В данной конфигурации регистрируется ссылка на сервис. По-умолчанию, данный сервис является обязательным, и все вызовы к нему, при его отсутствии, будут блокируемыми на определенный интервал, по истечении которого будет возникать исключение. Это поведение полностью поддаётся конфигурированию.  
  
У программистов знакомых со Spring Framework наверняка возник вопрос. Каким образом ссылка на сервис обеспечивает динамичность этого сервиса? Ведь все объекты конфигурируются на этапе загрузки (eager), либо на этапе обращения (lazy). Что же будет если ссылка producer утратит свою актуальность по причине исчезновения сервиса. Ответ: blueprint заботится об этом. На самом деле в качестве ссылки на сервис инжектируется прокси-объект, который и обслуживает динамичность. Этот объект обеспечивает поведение блокирования вызова, а также генерирование исключения.  
  
В архиве с исходным кодом, помимо него присутствует директория с настроенным контейнером equinox и всеми необходимыми библиотеками (equinox-for-example). Думаю это облегчит быстрый старт. Для запуска контейнера необходимо зайти в директорию с ним и выполнить команду:

java -jar org.eclipse.osgi\_3.6.2.R36x\_v20110210.jar -console

После неё на экране появятся сообщения от инициализирующихся модулей. И в итоге приглашение командной строки equinox. После ввода команды ss вывод должен быть идентичным представленному ниже:

osgi> ss

Framework is launched.

id State Bundle

0 ACTIVE org.eclipse.osgi\_3.6.2.R36x\_v20110210

1 ACTIVE com.springsource.net.sf.cglib\_2.2.0

2 ACTIVE com.springsource.org.aopalliance\_1.0.0

3 ACTIVE com.springsource.org.apache.log4j\_1.2.16

4 ACTIVE com.springsource.slf4j.api\_1.6.1

Fragments=5

5 RESOLVED com.springsource.slf4j.log4j\_1.6.1

Master=4

6 ACTIVE com.springsource.slf4j.org.apache.commons.logging\_1.6.1

7 ACTIVE org.eclipse.gemini.blueprint.core\_1.0.0.RELEASE

8 ACTIVE org.eclipse.gemini.blueprint.extender\_1.0.0.RELEASE

9 ACTIVE org.eclipse.gemini.blueprint.io\_1.0.0.RELEASE

10 ACTIVE org.springframework.aop\_3.0.6.RELEASE

11 ACTIVE org.springframework.asm\_3.0.6.RELEASE

12 ACTIVE org.springframework.aspects\_3.0.6.RELEASE

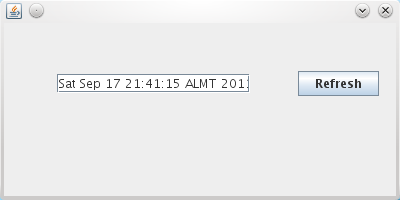
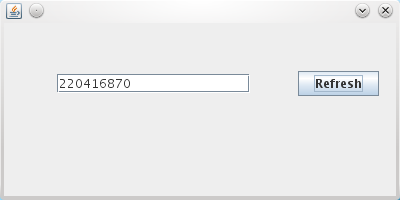
13 ACTIVE org.springframework.beans\_3.0.6.RELEASE

14 ACTIVE org.springframework.context\_3.0.6.RELEASE

15 ACTIVE org.springframework.context.support\_3.0.6.RELEASE

16 ACTIVE org.springframework.core\_3.0.6.RELEASE

17 ACTIVE org.springframework.expression\_3.0.6.RELEASE

Это список всех модулей установленных на текущий момент. Теперь необходимо установить тестовое приложение. Для этого вызовите команду install file:///путь\_к\_jar\_файлу\_модуля для каждого из 4-х модулей (собрать модули можно maven-командой package). После этого запустите командой start id на выполнения модули с API, один из сервисов и клиента. На экране должно появиться окно тестового приложения.  
  
  
Использование date-producer  
  
Затем остановите командой stop id модуль сервиса и запустите другой сервис. Обновите содержимое текстового поля окна, нажав кнопку refresh.  
  
  
Использование int-producer  
  
На этом описание тестового примера закончено. Хочу подчеркнуть, что это только малая часть возможностей Gemini Blueprint.

#### Область применения

OSGi давно стал нечто большим нежели стандарт модульных систем для встраиваемой техники. В подтверждение вышесказанного достаточно привести пример таких широко распространённых проектов основанных на OSGi, как Eclipse и Glassfish. Spring Framework имеет уверенное положение на корпоративном рынке. Для чего же может быть использована синергия этих технологий? Основной областью применения этой связки, по задумке создателей, должна была стать область корпоративных приложений. Собственно это успешная попытка привнесения OSGi в корпоративный мир. И теперь мы имеем возможность использовать OSGi в любых корпоративных и прикладных приложениях совместно со Spring Framework.

#### Недостатки

Не смотря на большие преимущества, которыми сулит динамическая модульная архитектура, у связки Spring и OSGi есть свои недостатки. Наиболее существенным недостатком я считаю сравнительно малую распространённость данной связки. Конечно по отдельности эти технологии не вызывают сомнений в своей живучести, но количество пользователей использующих их вместе не так велико. Субъективно, процесс передачи проектов от SpringSource под крыло Eclipse Foundation замедлил их развитие. Вторым недостатком можно назвать дополнительную сложность, которую добавляют к проекту использование данной связки. И это ни столько сложность использования данных технологий, сколько сложность добавляемая проблемами совместимости со сторонними библиотеками, вернее проблемами использования их в OSGi контейнере.

#### Проекты связанные с применением Spring в OSGi контейнере

* Spring DM — проект предназначенный для облегчения использования Spring-приложений в OSGi-контейнере. В данный момент не развивается (см. Eclipse Gemini).
* Spring DM Server — полностью модульный сервер приложений Java, разработанный для запуска корпоративных приложений и Spring-приложений. Обладет высокой степенью гибкости и надёжности. В данный момент не развивается (см. Eclipse Virgo).
* Gemini Blueprint — наследник Spring DM. Проект позволяет запускать Spring-приложения в OSGi-контейнере.
* Gemini Web — базируется на референсной реализации спецификации Web Applications от OSGi Alliance. Проект для запуска Spring Web-модулей в OSGi-контейнере.
* Gemini JPA — модульная реализация Java Persistence API для OSGi-контейнера. На данный момент обеспечивает интеграцию c JPA провайдером EclipseLink.
* Gemini DBAccess — обеспечивает распространение JDBC-драйверов подходящих для запуска в OSGi-контейнере и механизмы для их использвоания.
* Gemini Management — проект для удалённого управления модульной системой с использованием средств JMX.
* Gemini Naming — поддержка JNDI в OSGi-контейнере.
* Eclipse Virgo — OSGi сервер приложений, наследник Spring DM Server. Поддерживает следующие виды форматов развёртывания (deployment formats): OSGi модули (bundle), Java EE WAR, Web-модули (Web-bundles), PAR (подобие EAR для Java EE, содержит архивы компонентов), Plan (конфигурация нескольких модулей в общее приложение), Configuration (механизм динамического обновления конфигурации приложения). Содержит механизм горячего развёртывания приложений, панель администрирования, дополнительные библиотеки. Может поставляться с двумя servlet-контейнерами Tomcat и Jetty. А также множество дополнительного функционала.

#### Заключение

Я задумывал статью как вводное знакомство с технологией совместного использования Spring и OSGi. Для того чтобы полностью описать всю информацию по вводной части мне потребовался бы гораздо больший объём, поэтому предлагаю заинтересованному читателю самостоятельно изучить дополнительные источники. К тому же, я буду рад ответить на вопросы по этой замечательной технологии (в границах своей компетентности).

#### Дополнительные источники

[Архив с примером](https://sites.google.com/site/programmersnook/files/blueprint-example.tar.bz2)  
[Eclipse Gemini](http://eclipse.org/gemini/)  
[Bnd](http://www.aqute.biz/Bnd/Bnd)  
[OSGi Specifications](http://www.osgi.org/Specifications/HomePage)  
[Spring Framework](http://www.springsource.org/about)  
[Spring Dynamic Modules](http://www.springsource.org/osgi)  
[OSGi SpringSource Blog](http://blog.springsource.com/category/osgi/)  
[Spring Dynamic Modules in Action](http://www.amazon.com/Spring-Dynamic-Modules-Action-Cogoluegnes/dp/1935182307/ref=sr_1_1?s=books&ie=UTF8&qid=1316275180&sr=1-1)  
[Eclipse Virgo](http://www.eclipse.org/virgo/)

* [Blueprint](http://habrahabr.ru/search/?q=%5BBlueprint%5D&target_type=posts)
* , [Java](http://habrahabr.ru/search/?q=%5BJava%5D&target_type=posts)
* , [OSGi](http://habrahabr.ru/search/?q=%5BOSGi%5D&target_type=posts)
* , [Spring Framework](http://habrahabr.ru/search/?q=%5BSpring%20Framework%5D&target_type=posts)

**+49**

9994

109

[**ipogudin**](http://habrahabr.ru/users/ipogudin/)**16,0**[**G+**](https://plus.google.com/114907486554440803357/?rel=author)

**[Private cloud.](http://ad.adriver.ru/cgi-bin/click.cgi?sid=1&ad=494925&bt=21&pid=1796016&bid=3623906&bn=3623906&rnd=1070031622" \t "_blank)**[От планирования  
конфигурации ИТ-системы до запуска](http://ad.adriver.ru/cgi-bin/click.cgi?sid=1&ad=494925&bt=21&pid=1796016&bid=3623906&bn=3623906&rnd=1070031622" \t "_blank)

Похожие публикации

[Java 8, Spring, Hibernate, SSP — начинаем играться](http://habrahabr.ru/post/222077/) 7 мая в 12:07

[Простая и масштабируемая подписка на события с WebSockets, STOMP, SockJS и Spring Framework 4.0](http://habrahabr.ru/post/187822/) 25 июля 2013 в 21:56

[SOAP-сервер на Java при участии Apache CXF и Spring](http://habrahabr.ru/post/137543/) 3 февраля 2012 в 13:13

[Hibernate-Extender или Hibernate, Spring и OSGi](http://habrahabr.ru/post/135651/) 3 января 2012 в 12:56

[Вышел Spring Framework 3.1 GA](http://habrahabr.ru/post/134546/) 13 декабря 2011 в 20:57

[@Autowired для сервлетов в OSGi-контейнере](http://habrahabr.ru/post/132569/) 15 ноября 2011 в 19:00

[Выполнение кода по расписанию в Spring Framework](http://habrahabr.ru/post/130306/) 13 октября 2011 в 03:03

[Внедрение Spring Security в связку ZK+Spring Framework+Hibernate: часть первая](http://habrahabr.ru/post/129392/) 29 сентября 2011 в 10:06

[Open Source RIA на Spring Framework'e](http://habrahabr.ru/post/128968/) 22 сентября 2011 в 18:46

[Кеширование в Spring Framework 3.1](http://habrahabr.ru/post/113945/) 17 февраля 2011 в 00:57

## Комментарии (16)

**0**

[http://habrastorage.org/getpro/habr/avatars/8c7/3c5/f6f/8c73c5f6fd126f39b98ca82a288ea086_small.jpg**kreon**](http://habrahabr.ru/users/kreon/)18 сентября 2011 в 12:44[#](http://habrahabr.ru/post/128653/#comment_4255635)

Спасибо, познавательно.

**+1**

[http://habrastorage.org/getpro/habr/avatars/166/744/a13/166744a13521f259bcbab3f426f8fc09_small.png**Dair\_Targ**](http://habrahabr.ru/users/Dair_Targ/)18 сентября 2011 в 15:39[#](http://habrahabr.ru/post/128653/#comment_4255885)

Спасибо. Жаль, что не довелось ещё поиспользовать — выглядит красиво и, главное, заставляет делать проекты модульными.

**0**

[http://habrastorage.org/getpro/habr/avatars/8ba/b39/8f7/8bab398f7dd042ef209af244fda1a131_small.png**ipogudin**](http://habrahabr.ru/users/ipogudin/)18 сентября 2011 в 16:16[#](http://habrahabr.ru/post/128653/#comment_4255945)

↵

[↑](http://habrahabr.ru/post/128653/#comment_4255885)

А вы попробуйте. Для начала на тестовом проекте, а потом, возможно, и в production. Чем более популярна будет технология, тем активней будет развиваться.

**0**

[http://habrastorage.org/getpro/habr/avatars/166/744/a13/166744a13521f259bcbab3f426f8fc09_small.png**Dair\_Targ**](http://habrahabr.ru/users/Dair_Targ/)18 сентября 2011 в 18:46[#](http://habrahabr.ru/post/128653/#comment_4256194)

↵

[↑](http://habrahabr.ru/post/128653/#comment_4255945)

Я б с радостью. Да только в ближайшие пару лет совершенно в другой области работать буду.

**0**

[http://habrahabr.ru/i/avatars/stub-user-small.gif**CHEM\_Eugene**](http://habrahabr.ru/users/CHEM_Eugene/)18 сентября 2011 в 17:09[#](http://habrahabr.ru/post/128653/#comment_4256050)

Неплохая связка, мы используем в своем проекте. Интересно было бы узнать кто её применяет и в каких проектах.

**0**

[http://habrastorage.org/getpro/habr/avatars/8ba/b39/8f7/8bab398f7dd042ef209af244fda1a131_small.png**ipogudin**](http://habrahabr.ru/users/ipogudin/)18 сентября 2011 в 18:10[#](http://habrahabr.ru/post/128653/#comment_4256146)

↵

[↑](http://habrahabr.ru/post/128653/#comment_4256050)

На данный момент я разрабатываю свой хобби-проект на базе Eclipse Virgo. Это будет набор приложений для совместной работы над проектами (преимущественно разработкой ПО). Выбор был сделан в пользу данной связки, дабы не изобретать свой плагинный движок. А в каком проекте вы используете Spring и OSGi?

**0**

[http://habrahabr.ru/i/avatars/stub-user-small.gif**CHEM\_Eugene**](http://habrahabr.ru/users/CHEM_Eugene/)22 сентября 2011 в 12:42[#](http://habrahabr.ru/post/128653/#comment_4266126)

↵

[↑](http://habrahabr.ru/post/128653/#comment_4256146)

production, довольно крупный проект

**0**

[http://habrastorage.org/getpro/habr/avatars/f50/b74/462/f50b744625ff1e41ec0bdbb754b62824_small.jpg**philpirj**](http://habrahabr.ru/users/philpirj/)22 сентября 2011 в 22:46[#](http://habrahabr.ru/post/128653/#comment_4267445)

↵

[↑](http://habrahabr.ru/post/128653/#comment_4266126)

Не расскажете поподробнее, например, нету ли проблем с разрешением зависимостей?

**0**

[http://habrastorage.org/getpro/habr/avatars/3f1/159/17a/3f115917a414a73233d66d1c904270a0_small.jpg**r0zh0k**](http://habrahabr.ru/users/r0zh0k/)18 сентября 2011 в 18:27[#](http://habrahabr.ru/post/128653/#comment_4256166)

Можете привести реальный пример использования? Из своего опыта.

**+1**

[http://habrastorage.org/getpro/habr/avatars/8ba/b39/8f7/8bab398f7dd042ef209af244fda1a131_small.png**ipogudin**](http://habrahabr.ru/users/ipogudin/)18 сентября 2011 в 18:30[#](http://habrahabr.ru/post/128653/#comment_4256172)

↵

[↑](http://habrahabr.ru/post/128653/#comment_4256166)

Веб-приложение с плагинной системой см. выше.

**0**

[http://habrastorage.org/getpro/habr/avatars/3f1/159/17a/3f115917a414a73233d66d1c904270a0_small.jpg**r0zh0k**](http://habrahabr.ru/users/r0zh0k/)18 сентября 2011 в 19:08[#](http://habrahabr.ru/post/128653/#comment_4256225)

↵

[↑](http://habrahabr.ru/post/128653/#comment_4256172)

Если не сложно, опишите конкретные задачи, которые вы предполагаете решать с помощью OSGi и приемущества перед стандартным подходом.

**+1**

[http://habrastorage.org/getpro/habr/avatars/8ba/b39/8f7/8bab398f7dd042ef209af244fda1a131_small.png**ipogudin**](http://habrahabr.ru/users/ipogudin/)18 сентября 2011 в 19:46[#](http://habrahabr.ru/post/128653/#comment_4256307)

↵

[↑](http://habrahabr.ru/post/128653/#comment_4256225)

Приложение формирует пользовательское окружение на основе сервисов, которые зарегистрированы в контейнере. Например: в некую абстракция рабочей области встраивается возможность доступа к wiki-движку, и системе управления взаимодействием. Каждая из подсистем представляется в виде сервиса.

**0**

[http://habrahabr.ru/i/avatars/stub-user-small.gif**Throwable**](http://habrahabr.ru/users/Throwable/)18 сентября 2011 в 19:18[#](http://habrahabr.ru/post/128653/#comment_4256241)

Спасибо за обзор. Думаю использовать Spring DM в следующем проекте, однако в специфику пока не вникал. Посему есть несколько неясностей. Буду признателен, если проясните. Пусть модуль P предоставляет некий сервис с интерфейсом I и несколько связанных с ним структур данных, а модуль C его собирается использовать.  
  
1. Как передаются параметры при вызове от модуля к модулю: по значению (сериализация-десериализация), или по ссылке?  
2. Как происходит резольвинг классов между модулями? Насколько я понимаю, каждый модуль имеет свой ClassLoader. И насколько я понимаю, delegation между ними не используется. Иначе когда модуль P недоступен, С не смог бы сделать резольвинг интерфейса I. То есть модуль C использует непосредственно классы из P или имеет собственную копию exported-пакетов?  
3. А что будет если мы внезапно запустили новую версию сервиса P, которая поменяла интерфейс I и несколько экспортнутых классов? Если передача параметров идет по значению, то проблем не должно быть если классы остались serializable-совместимыми. Если же по ссылке, то возникнет ClassCastException, т.к. модуль C уже отрезольвил старую версию классов.  
4. Я слышал, что можно одновременно иметь несколько версий одного и того же сервиса. И что consumer будет использовать именно последнюю версию сервиса. Можно ли как-нибудь динамически контролировать какую версию сервиса использовать?  
5. Чтобы сделать unit-тесты для сервисов без испльзования OSGi, как я понимаю, очень несложно. Для этого нужно прописать соответствующие сервисы в тестовом spring context. А как делать автоматические integration-тесты в OSGi-контейнере? Есть какая-нибудь тулза?

**0**

[http://habrastorage.org/getpro/habr/avatars/8ba/b39/8f7/8bab398f7dd042ef209af244fda1a131_small.png**ipogudin**](http://habrahabr.ru/users/ipogudin/)18 сентября 2011 в 19:40[#](http://habrahabr.ru/post/128653/#comment_4256293)

↵

[↑](http://habrahabr.ru/post/128653/#comment_4256241)

1. Параметры передаются также, как и в обычной Java программе.  
2. Резольвинг происходит согласно правилам import и export. Т.е. ClassLoader модуля связан с фреймворком и знает какие классы доступны через export у других модулей, и знает какие сделать доступными текущиму модулю из import.  
3. После изменения кода интерфейсов и классов необходимо выполнить операцию refresh на всех модулях, которые использовали их.  
4. Возможно указывать правила фильтров, какие версии сервисов использовать. Можно получать целую коллекцию сервисов с заданными интерфейсами и выбирать необходимы сервис. При это коллекция будет при каждом обращении соответствовать актуальному набору сервисов, которые присутствуют в текущий момент.  
5. Spring dm и Gemini Blueprint поддерживают модульное тестирования и имеют специальные классы для поднятия OSGi окружения при тестировании в jUnit. Смотрите [Testing OSGi based Applications](http://www.eclipse.org/gemini/blueprint/documentation/reference/1.0.0.RELEASE/html/testing.html).

**0**

[http://habrahabr.ru/i/avatars/stub-user-small.gif**dpishchukhin**](http://habrahabr.ru/users/dpishchukhin/)19 сентября 2011 в 10:39[#](http://habrahabr.ru/post/128653/#comment_4257435)

↵

[↑](http://habrahabr.ru/post/128653/#comment_4256241)

Думаю, что на почти на все вопросы ответит OSGi спецификация [OSGi Spec 4.2](http://www.osgi.org/Download/Release4V42) Чтобы не копировать — вот ссылки:  
2. OSGi Spec 4.2 Core: 3.4. Class Loading Architecture, 3.7. Resolving Process, 3.8. Runtime Class Loading  
3&4. OSGi Spec 4.2 Core: 5. Service Layer  
5. [Pax-Exam — OSGi Unit tests framework](http://team.ops4j.org/wiki/display/paxexam/Pax+Exam)

**0**

[http://habrastorage.org/getpro/habr/avatars/a47/ea4/3cc/a47ea43cc77a140fbea48cca56fb5be5_small.jpg**garbuz**](http://habrahabr.ru/users/garbuz/)19 сентября 2011 в 13:21[#](http://habrahabr.ru/post/128653/#comment_4257877)

Спасибо, статью в закладки.  
В одном проекте использовали связку Spring и OSGI (Equinox), все довольно удачно получилось, но столкнулись с проблемой, что со спрингом был лишь один модуль, так как больше одного спрингового контекста было почему-то не поднять. У вас как я вижу такая проблема отсутствует. Еще раз спасибо за статью, возьму на заметку.

Только зарегистрированные пользователи могут оставлять комментарии. [Войдите](https://habrahabr.ru/auth/login/), пожалуйста.

# DataBase

## Validating the Integrity of Base Tables In a Referential Constraint Relationship

If you decide to specify a non‑enforced Referential Constraint for a table, the responsibility for ensuring or validating the referential integrity of that table is yours alone. The system does not enforce referential integrity when you specify Referential Constraint relationships between tables, because no declarative constraints are declared in the appropriate column definitions.

From the aspect of integrity assurance, the best way to guarantee the referential integrity of a table without taking advantage of a declarative standard or batch referential constraint is to use a procedural constraint such as a set of triggers to handle inserts, updates, and deletions to the tables in the relationship.

For example, you might want to create DELETE/UPDATE triggers on parent tables, and INSERT/UPDATE triggers on child tables to enforce referential integrity. The reasons for preferring declarative constraints over procedural constraints are described briefly in *Database Design*.

There is the additional likelihood that actively firing triggers will have a greater negative effect on system performance than the simple declarative constraint they are intended to replace.

If you decide not to enforce any form of referential integrity constraint, then you are strongly advised to enforce a set of validation procedures that can detect when and where referential integrity violations occur.

The following scenario uses a basic SQL query to interrogate a table set for violations of the referential integrity rule.

Suppose you create the tables *pk\_tbl1* and *softri\_tbl1*, with column *b2* of *softri\_tbl1* having a Referential Constraint relationship with column *a1* of *pk\_tbl1*. In this relationship, table *pk\_tbl1* is the parent and table *softri\_tbl1* is the child. The DDL for defining these tables might look like the following CREATE TABLE requests.

     CREATE TABLE pk\_tbl1 (

       a1 INTEGER NOT NULL PRIMARY KEY,

       a2 INTEGER);

     CREATE TABLE softri\_tbl1 (

       b1 INTEGER,

       b2 INTEGER CONSTRAINT softri\_1

          REFERENCES WITH NO CHECK OPTION pk\_tbl1(a1));

Column *softri\_tbl1.b2* is an implicit foreign key referencing the primary key column   
*pk\_tbl1.a1*.

Now populate the tables with data as follows.

     INSERT INTO pk\_tbl1 (a1, a2) VALUES (11, 111);

     INSERT INTO pk\_tbl1 (a1, a2) VALUES (22, 222);

     INSERT INTO softri\_tbl1 (b1, b2) VALUES (100, 11);

     INSERT INTO softri\_tbl1 (b1, b2) VALUES (200, 22);

     INSERT INTO softri\_tbl1 (b1, b2) VALUES (300, 33);

     INSERT INTO softri\_tbl1 (b1, b2) VALUES (400, 44);

The third and fourth inserts into table *softri\_tbl1* violate the implicit referential integrity relationship between the tables because the set of distinct values for *softri\_tbl1.b2*should be identical to the set of values for *pk\_tbl1* (which, as the primary key value set, constitute a distinct set by default). In this particular scenario, the *softri\_tbl1.b2*values 33 and 44 identify the rows that violate the implied referential integrity relationship.

The following SELECT request is a generic query to test for this type of corruption. The exclusion of foreign key nulls is included in the query because it is not possible to determine what values they represent.

     SELECT DISTINCT childtable.\*

     FROM childtable, parenttable

     WHERE childtable.fk NOT IN (SELECT pk

                                 FROM parenttable)

     AND   childtable.fk IS NOT NULL;

The scenario defines the following set of correspondences.

|  |  |
| --- | --- |
| **Generic Query Element** | **Specific Query Element** |
| childtable | softri\_tbl1 |
| parentable | pk\_tbl1 |
| childtable.fk | softri\_tbl1.b2 |

From these correspondences, the specify query to test for corrupt rows in this scenario would look like the following SELECT request.

     SELECT DISTINCT softri\_tbl1.\*

     FROM softri\_tbl1, pk\_tbl1

     WHERE softri\_tbl1.b2 NOT IN (SELECT a1

                                  FROM pk\_tbl1)

     AND   softri\_tbl1.b2 IS NOT NULL;

      \*\*\* Query completed. 2 rows found. 2 columns returned.

      \*\*\* Total elapsed time was 1 second.

      b1 b2

     ----------- -----------

      300 33

      400 44

The report generated by this query returns the rows with the implicit foreign key values 33 and 44, which is what was expected: the set of rows from the child table whose foreign key values do not match any element of the primary key value set from the parent table. This set of non‑matching rows must be deleted from the child table to maintain the referential integrity of the database.

You should perform this validation query regularly, particularly after any substantial updates are made to the tables in the implied relationship.

Keep in mind that your data can be corrupt for the entire interval between the times you run this query, and you still have the burden of repairing any problems that it detects. The query does *not* solve the problem of failing to enforce referential integrity in your database: all it does is allow you to detect whether the referential integrity defined by *1* primary or alternate key in a parent table and *1* foreign key in a child table has been violated. Each such unenforced referential relationship must be probed regularly to detect violations.

Copyright © 1998-2013 by Teradata Corporation. All Rights Reserved.

## Data integrity From Wikipedia, the free encyclopedia

**Data integrity** refers to maintaining and assuring the accuracy and consistency of [data](http://en.wikipedia.org/wiki/Data) over its entire [life-cycle](http://en.wikipedia.org/wiki/Information_Lifecycle_Management),[[1]](http://en.wikipedia.org/wiki/Data_integrity#cite_note-1) and is a critical aspect to the design, implementation and usage of any system which stores, processes, or retrieves data. The term **data integrity** is broad in scope and may have widely different meanings depending on the specific context – even under the same general umbrella of [computing](http://en.wikipedia.org/wiki/Computing). This article provides only a broad overview of some of the different types and concerns of data integrity.

Data integrity is the opposite of [data corruption](http://en.wikipedia.org/wiki/Data_corruption), which is a form of [data loss](http://en.wikipedia.org/wiki/Data_loss). The overall intent of any data integrity technique is the same: ensure data is recorded exactly as intended (such as a database correctly rejecting mutually exclusive possibilities,) and upon later retrieval, ensure the data is the same as it was when it was originally recorded. In short, data integrity aims to prevent unintentional changes to information. Data integrity is not to be confused with [data security](http://en.wikipedia.org/wiki/Data_security), the discipline of protecting data from unauthorized parties.

Any unintended changes to data as the result of a storage, retrieval or processing operation, including malicious intent, unexpected hardware failure, and human error, is failure of data integrity. If the changes are the result of unauthorized access, it may also be a failure of [data security](http://en.wikipedia.org/wiki/Data_security). Depending on the data involved this could manifest itself as benign as a single pixel in an image appearing a different color than was originally recorded, to the loss of vacation pictures or a business-critical database, to even catastrophic loss of human life in a [life-critical system](http://en.wikipedia.org/wiki/Life-critical_system).

**Contents**

  [[hide](http://en.wikipedia.org/wiki/Data_integrity)]

* [1 Physical vs. logical integrity](http://en.wikipedia.org/wiki/Data_integrity#Physical_vs._logical_integrity)
* [2 Databases](http://en.wikipedia.org/wiki/Data_integrity#Databases)
  + [2.1 Types of integrity constraints](http://en.wikipedia.org/wiki/Data_integrity#Types_of_integrity_constraints)
  + [2.2 Examples](http://en.wikipedia.org/wiki/Data_integrity#Examples)
* [3 File systems](http://en.wikipedia.org/wiki/Data_integrity#File_systems)
* [4 Data storage](http://en.wikipedia.org/wiki/Data_integrity#Data_storage)
* [5 See also](http://en.wikipedia.org/wiki/Data_integrity#See_also)
* [6 References](http://en.wikipedia.org/wiki/Data_integrity#References)
* [7 Further reading](http://en.wikipedia.org/wiki/Data_integrity#Further_reading)

Physical vs. logical integrity[[edit](http://en.wikipedia.org/w/index.php?title=Data_integrity&action=edit&section=1)]

Data integrity can be roughly divided into two overlapping categories:

**Physical integrity** - deals with challenges associated with correctly storing and fetching the data itself. Challenges with physical integrity may include electromechanical faults, design flaws, material fatigue, corrosion, power outages, natural disasters, acts of war and terrorism, and other special environmental hazards such as ionizing radiation, extreme temperatures, pressures and g-forces. Ensuring physical integrity includes methods such as redundant hardware, an[uninterruptible power supply](http://en.wikipedia.org/wiki/Uninterruptible_power_supply), certain types of [RAID](http://en.wikipedia.org/wiki/RAID) arrays, [radiation hardened](http://en.wikipedia.org/wiki/Radiation_hardened) chips, [ECC memory](http://en.wikipedia.org/wiki/ECC_memory), use of a [clustered file system](http://en.wikipedia.org/wiki/Clustered_file_system), using file systems that employ block level checksums such as [ZFS](http://en.wikipedia.org/wiki/ZFS), storage arrays that compute parity calculations such as [Exclusive or](http://en.wikipedia.org/wiki/Exclusive_or) or use a [Cryptographic hash function](http://en.wikipedia.org/wiki/Cryptographic_hash_function) and even having a[watchdog timer](http://en.wikipedia.org/wiki/Watchdog_timer) on critical subsystems.

Physical integrity often makes extensive use of error detecting algorithms known as [error-correcting codes](http://en.wikipedia.org/wiki/Error-correcting_codes). Human induced data integrity errors are often detected through the use of simpler check digits and algorithms used to detect them such as the [Damm algorithm](http://en.wikipedia.org/wiki/Damm_algorithm) or [Luhn algorithm](http://en.wikipedia.org/wiki/Luhn_algorithm). These are used to maintain data integrity after manual transcription from one computer system to another by a human intermediary. Examples include credit card and bank routing numbers. Computer induced transcription errors can be detected through [hash functions](http://en.wikipedia.org/wiki/Hash_functions).

In production systems these techniques are used in combination to ensure various degrees of data integrity. For example a computer [file system](http://en.wikipedia.org/wiki/File_system) may configured on a fault tolerant [RAID](http://en.wikipedia.org/wiki/RAID) array, but might not provide block level checksums to detect and prevent [silent data corruption](http://en.wikipedia.org/wiki/Silent_data_corruption). A database management system might be[ACID](http://en.wikipedia.org/wiki/ACID) compliant, but the raid controller or hard drive's internal write-cache might not be.

**Logical integrity** - concerned with the correctness or rationality of a piece of data, given a particular context. This includes topics such as [referential integrity](http://en.wikipedia.org/wiki/Referential_integrity) and[entity integrity](http://en.wikipedia.org/wiki/Entity_integrity) in a [relational database](http://en.wikipedia.org/wiki/Relational_database) or correctly ignoring impossible sensor data in robotic systems. These concerns involve making certain the data "makes sense" given its environment. Challenges include [software bugs](http://en.wikipedia.org/wiki/Software_bugs), design flaws, human error. Common methods of ensuring logical integrity include things such as a[Check constraint](http://en.wikipedia.org/wiki/Check_constraint), [foreign key constraint](http://en.wikipedia.org/wiki/Foreign_key_constraint), program [assertion (computing)](http://en.wikipedia.org/wiki/Assertion_(computing)) and other runtime sanity checks.

Both physical and logical integrity often share many common challenges such as human error, design flaws and both must appropriately deal with concurrent requests to record and retrieve data, the later of which is its own subject entirely. See [mutex](http://en.wikipedia.org/wiki/Mutex) and [Copy-on-write](http://en.wikipedia.org/wiki/Copy-on-write).

Databases[[edit](http://en.wikipedia.org/w/index.php?title=Data_integrity&action=edit&section=2)]

Data integrity contains guidelines for [data retention](http://en.wikipedia.org/wiki/Data_retention), specifying or guaranteeing the length of time data can be retained in a particular database. It specifies what can be done with data values when their validity or usefulness expires. In order to achieve data integrity, these rules are consistently and routinely applied to all data entering the system, and any relaxation of enforcement could cause errors in the data. Implementing checks on the data as close as possible to the source of input (such as human data entry), causes less erroneous data to enter the system. Strict enforcement of data integrity rules causes the error rates to be lower, resulting in time saved troubleshooting and tracing erroneous data and the errors it causes algorithms.

Data integrity also includes rules defining the relations a piece of data can have, to other pieces of data, such as a *Customer* record being allowed to link to purchased *Products*, but not to unrelated data such as *Corporate Assets*. Data integrity often includes checks and correction for invalid data, based on a fixed[schema](http://en.wikipedia.org/wiki/Database_schema) or a predefined set of rules. An example being textual data entered where a date-time value is required. Rules for data derivation are also applicable, specifying how a data value is derived based on algorithm, contributors and conditions. It also specifies the conditions on how the data value could be re-derived.

**Types of integrity constraints**[[edit](http://en.wikipedia.org/w/index.php?title=Data_integrity&action=edit&section=3)]

Data integrity is normally enforced in a [database system](http://en.wikipedia.org/wiki/Database_system) by a series of [integrity constraints](http://en.wikipedia.org/wiki/Integrity_constraints) or rules. Three types of integrity constraints are an inherent part of the relational data model: entity integrity, referential integrity and domain integrity:

* [*Entity integrity*](http://en.wikipedia.org/wiki/Entity_integrity) concerns the concept of a [primary key](http://en.wikipedia.org/wiki/Primary_key). Entity integrity is an integrity rule which states that every table must have a primary key and that the column or columns chosen to be the primary key should be unique and not null.
* [*Referential integrity*](http://en.wikipedia.org/wiki/Referential_integrity) concerns the concept of a [foreign key](http://en.wikipedia.org/wiki/Foreign_key). The referential integrity rule states that any foreign-key value can only be in one of two states. The usual state of affairs is that the foreign-key value refers to a primary key value of some table in the database. Occasionally, and this will depend on the rules of the data owner, a foreign-key value can be [null](http://en.wikipedia.org/wiki/Null_(SQL)). In this case we are explicitly saying that either there is no relationship between the objects represented in the database or that this relationship is unknown.
* *Domain integrity* specifies that all columns in a relational database must be declared upon a defined domain. The primary unit of data in the relational data model is the data item. Such data items are said to be non-decomposable or atomic. A domain is a set of values of the same type. Domains are therefore pools of values from which actual values appearing in the columns of a table are drawn.
* *User-defined integrity* refers to a set of rules specified by a user, which do not belong to the entity, domain and referential integrity categories.

If a database supports these features it is the responsibility of the database to insure data integrity as well as the [consistency model](http://en.wikipedia.org/wiki/Consistency_model) for the data storage and retrieval. If a database does not support these features it is the responsibility of the applications to ensure data integrity while the database supports the [consistency model](http://en.wikipedia.org/wiki/Consistency_model) for the data storage and retrieval.

Having a single, well-controlled, and well-defined data-integrity system increases

* stability (one centralized system performs all data integrity operations)
* performance (all data integrity operations are performed in the same tier as the [consistency model](http://en.wikipedia.org/wiki/Consistency_model))
* re-usability (all applications benefit from a single centralized data integrity system)
* maintainability (one centralized system for all data integrity administration).

As of 2012, since all modern [databases](http://en.wikipedia.org/wiki/Database) support these features (see [Comparison of relational database management systems](http://en.wikipedia.org/wiki/Comparison_of_relational_database_management_systems)), it has become the de facto responsibility of the database to ensure data integrity. Out-dated and legacy systems that use file systems (text, spreadsheets, ISAM, flat files, etc.) for their[consistency model](http://en.wikipedia.org/wiki/Consistency_model) lack any[[*citation needed*](http://en.wikipedia.org/wiki/Wikipedia:Citation_needed)] kind of data-integrity model. This requires organizations to invest a large amount of time, money and personnel in building data-integrity systems on a per-application basis that needlessly duplicate the existing data integrity systems found in modern databases. Many companies, and indeed many database systems themselves, offer products and services to migrate out-dated and legacy systems to modern databases to provide these data-integrity features. This offers organizations substantial savings in time, money and resources because they do not have to develop per-application data-integrity systems that must be refactored each time the [business requirements](http://en.wikipedia.org/wiki/Business_requirements) change.

**Examples**[[edit](http://en.wikipedia.org/w/index.php?title=Data_integrity&action=edit&section=4)]

An example of a data-integrity mechanism is the parent-and-child relationship of related records. If a parent record owns one or more related child records all of the referential integrity processes are handled by the database itself, which automatically ensures the accuracy and integrity of the data so that no child record can exist without a parent (also called being orphaned) and that no parent loses their child records. It also ensures that no parent record can be deleted while the parent record owns any child records. All of this is handled at the database level and does not require coding integrity checks into each applications.

File systems[[edit](http://en.wikipedia.org/w/index.php?title=Data_integrity&action=edit&section=5)]

Various research results show that neither widespread [filesystems](http://en.wikipedia.org/wiki/Filesystem) (including [UFS](http://en.wikipedia.org/wiki/Unix_File_System), [Ext](http://en.wikipedia.org/wiki/Extended_file_system), [XFS](http://en.wikipedia.org/wiki/XFS), [JFS](http://en.wikipedia.org/wiki/JFS_(file_system)) and [NTFS](http://en.wikipedia.org/wiki/NTFS)) nor [hardware RAID](http://en.wikipedia.org/wiki/RAID#Problems_with_RAID) solutions provide sufficient protection against data integrity problems.[[2]](http://en.wikipedia.org/wiki/Data_integrity#cite_note-2)[[3]](http://en.wikipedia.org/wiki/Data_integrity#cite_note-3)[[4]](http://en.wikipedia.org/wiki/Data_integrity#cite_note-4)[[5]](http://en.wikipedia.org/wiki/Data_integrity#cite_note-5)[[6]](http://en.wikipedia.org/wiki/Data_integrity#cite_note-6)

Some filesystems (including [Btrfs](http://en.wikipedia.org/wiki/Btrfs) and [ZFS](http://en.wikipedia.org/wiki/ZFS)) provide internal data and [metadata](http://en.wikipedia.org/wiki/Metadata) checksumming, what is used for detecting [silent data corruption](http://en.wikipedia.org/wiki/Silent_data_corruption) and improving data integrity. If a corruption is detected that way and internal RAID mechanisms provided by those filesystems are also used, such filesystems can additionally reconstruct corrupted data in a transparent way.[[7]](http://en.wikipedia.org/wiki/Data_integrity#cite_note-7) This approach allows improved data integrity protection covering the entire data paths, which is usually known as[end-to-end data protection](http://en.wikipedia.org/wiki/End-to-end_data_protection).[[8]](http://en.wikipedia.org/wiki/Data_integrity#cite_note-8)

Data storage[[edit](http://en.wikipedia.org/w/index.php?title=Data_integrity&action=edit&section=6)]

*Main article:*[*Data Integrity Field*](http://en.wikipedia.org/wiki/Data_Integrity_Field)

Apart from data in databases, standards exist to address the integrity of data on [storage devices](http://en.wikipedia.org/wiki/Data_storage_device).[[9]](http://en.wikipedia.org/wiki/Data_integrity#cite_note-9)

See also[[edit](http://en.wikipedia.org/w/index.php?title=Data_integrity&action=edit&section=7)]

* [End-to-end data integrity](http://en.wikipedia.org/wiki/End-to-end_data_integrity)
* [National Information Systems Security Glossary](http://en.wikipedia.org/wiki/National_Information_Systems_Security_Glossary)

## The Database Normalization Process

* By [Ronald Plew](http://www.informit.com/authors/bio/52d9db87-6336-4535-b3db-5489cb2f0e01) and [Ryan Stephens](http://www.informit.com/authors/bio/523fbbe1-b936-42a3-8ba6-eadbda1c1bd6)
* Jan 24, 2003

[📄 Contents](http://www.informit.com/articles/article.aspx?p=30646)

[␡](http://www.informit.com/articles/article.aspx?p=30646)

1. Normalizing a Database
2. [Summary](http://www.informit.com/articles/article.aspx?p=30646&seqNum=2)
3. [Q&amp;A](http://www.informit.com/articles/article.aspx?p=30646&seqNum=3)

* [⎙ Print](http://www.informit.com/articles/printerfriendly/30646)
* [+ Share This](http://www.addthis.com/bookmark.php)
* [💬 Discuss](http://www.informit.com/articles/article.aspx?p=30646#articleDiscussion)

**Page 1** of 3 [**Next >**](http://www.informit.com/articles/article.aspx?p=30646&seqNum=2)

[This chapter is from the book ](http://www.informit.com/articles/article.aspx?p=30646)

### This chapter is from the book

[**Sams Teach Yourself SQL in 24 Hours, 3rd Edition**](http://www.informit.com/store/sams-teach-yourself-sql-in-24-hours-9780672324420?w_ptgrevartcl=The+Database+Normalization+Process_30646)   
  
[Learn More](http://www.informit.com/store/sams-teach-yourself-sql-in-24-hours-9780672324420?w_ptgrevartcl=The+Database+Normalization+Process_30646) [Buy](http://www.informit.com/buy.aspx?isbn=9780672324420&w_ptgrevartcl=The+Database+Normalization+Process_30646)

### This chapter is from the book

[Sams Teach Yourself SQL in 24 Hours, 3rd Edition](http://www.informit.com/store/sams-teach-yourself-sql-in-24-hours-9780672324420?w_ptgrevartcl=The+Database+Normalization+Process_30646)   
  
[Learn More](http://www.informit.com/store/sams-teach-yourself-sql-in-24-hours-9780672324420?w_ptgrevartcl=The+Database+Normalization+Process_30646) [Buy](http://www.informit.com/buy.aspx?isbn=9780672324420&w_ptgrevartcl=The+Database+Normalization+Process_30646)

Find out what normalization is and how your database can benefit from it (or suffer from it). Learn the advantages, disadvantages, and some techniques and guidelines to doing it yourself.

In this hour, you learn the process of taking a raw database and breaking it into logical units called tables. This process is referred to as normalization. The normalization process is used by database developers to design databases in which it is easy to organize and manage data while ensuring the accuracy of data throughout the database.

The advantages and disadvantages of both normalization and denormalization of a database are discussed, as well as data integrity versus performance issues that pertain to normalization.

The highlights of this hour include

* What normalization is
* Benefits of normalization
* Advantages of denormalization
* Normalization techniques
* Guidelines of normalization
* The three normal forms
* Database design

## Normalizing a Database

Normalization is a process of reducing redundancies of data in a database. Normalization is a technique that is used when designing and redesigning a database. Normalization is a process or set of guidelines used to optimally design a database to reduce redundant data. The actual guidelines of normalization, called normal forms, will be discussed later in this hour. It was a difficult decision to decide whether to cover normalization in this book because of the complexity involved in understanding the rules of the normal forms this early on in your SQL journey. However, normalization is an important process that, if understood, will increase your understanding of SQL. We have attempted to simplify the process of normalization as much as possible in this hour. At this point, don't be overly concerned with all the specifics of normalization; it is most important to understand the basic concepts.

### The Raw Database

A database that is not normalized may include data that is contained in one or more different tables for no apparent reason. This could be bad for security reasons, disk space usage, speed of queries, efficiency of database updates, and, maybe most importantly, data integrity. A database before normalization is one that has not been broken down logically into smaller, more manageable tables. [Figure 4.1](javascript:popUp('/content/images/chap4_0672324423/elementLinks/04fig01.jpg')) illustrates the database used for this book before it was normalized.

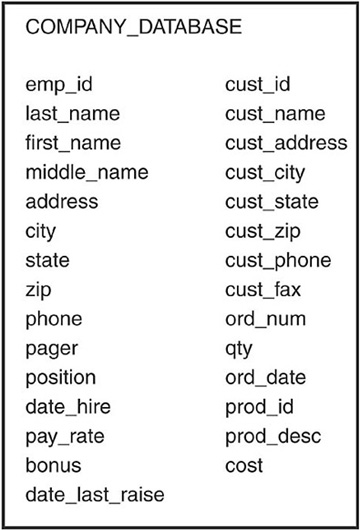


Figure 4.1 **The raw database.**

### Logical Database Design

Any database should be designed with the end user in mind. Logical database design, also referred to as the logical model, is the process of arranging data into logical, organized groups of objects that can easily be maintained. The logical design of a database should reduce data repetition or go so far as to completely eliminate it. After all, why store the same data twice? Naming conventions used in a database should also be standard and logical.

#### What Are the End User's Needs?

The needs of the end user should be one of the top considerations when designing a database. Remember that the end user is the person who ultimately uses the database. There should be ease of use through the user's front-end tool (a client program that allows a user access to a database), but this, along with optimal performance, cannot be achieved if the user's needs are not taken into consideration.

Some user-related design considerations include the following:

* What data should be stored in the database?
* How will the user access the database?
* What privileges does the user require?
* How should the data be grouped in the database?
* What data is the most commonly accessed?
* How is all data related in the database?
* What measures should be taken to ensure accurate data?

#### Data Redundancy

Data should not be redundant, which means that the duplication of data should be kept to a minimum for several reasons. For example, it is unnecessary to store an employee's home address in more than one table. With duplicate data, unnecessary space is used. Confusion is always a threat when, for instance, an address for an employee in one table does not match the address of the same employee in another table. Which table is correct? Do you have documentation to verify the employee's current address? As if data management were not difficult enough, redundancy of data could prove to be a disaster.

### The Normal Forms

The next sections discuss the normal forms, an integral concept involved in the process of database normalization.

Normal form is a way of measuring the levels, or depth, to which a database has been normalized. A database's level of normalization is determined by the normal form.

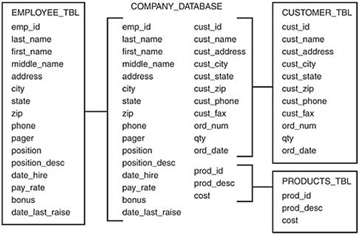
The following are the three most common normal forms in the normalization process:

* The first normal form
* The second normal form
* The third normal form

Of the three normal forms, each subsequent normal form depends on normalization steps taken in the previous normal form. For example, to normalize a database using the second normal form, the database must first be in the first normal form.

#### The First Normal Form

The objective of the first normal form is to divide the base data into logical units called tables. When each table has been designed, a primary key is assigned to most or all tables. Examine [Figure 4.2](javascript:popUp('/content/images/chap4_0672324423/elementLinks/04fig02_alt.jpg')), which illustrates how the raw database shown in the previous figure has been redeveloped using the first normal form.

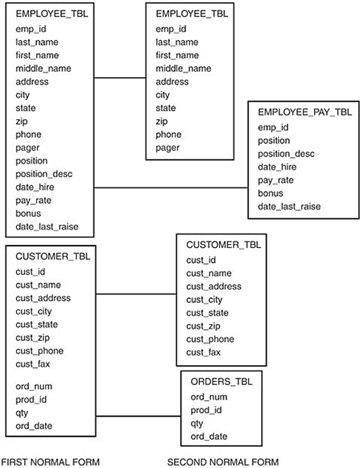
[](javascript:popUp('/content/images/chap4_0672324423/elementLinks/04fig02_alt.jpg'))

[Figure 4.2](javascript:popUp('/content/images/chap4_0672324423/elementLinks/04fig02_alt.jpg')) **The first normal form.**

You can see that to achieve the first normal form, data had to be broken into logical units of related information, each having a primary key and ensuring that there are no repeated groups in any of the tables. Instead of one large table, there are now smaller, more manageable tables: EMPLOYEE\_TBL,CUSTOMER\_TBL, and PRODUCTS\_TBL. The primary keys are normally the first columns listed in a table, in this case: EMP\_ID, CUST\_ID, and PROD\_ID .

#### The Second Normal Form

The objective of the second normal form is to take data that is only partly dependent on the primary key and enter that data into another table. [Figure 4.3](javascript:popUp('/content/images/chap4_0672324423/elementLinks/04fig03_alt.jpg')) illustrates the second normal form.

[](javascript:popUp('/content/images/chap4_0672324423/elementLinks/04fig03_alt.jpg'))

[Figure 4.3](javascript:popUp('/content/images/chap4_0672324423/elementLinks/04fig03_alt.jpg')) **The second normal form.**

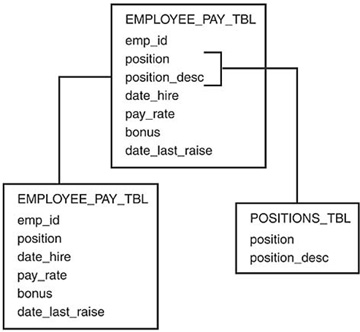
According to the figure, the second normal form is derived from the first normal form by further breaking two tables down into more specific units.

EMPLOYEE\_TBL split into two tables called EMPLOYEE\_TBL and EMPLOYEE\_PAY\_TBL. Personal employee information is dependent on the primary key (EMP\_ID), so that information remained in theEMPLOYEE\_TBL (EMP\_ID, LAST\_NAME, FIRST\_NAME, MIDDLE\_NAME, ADDRESS, CITY, STATE, ZIP,PHONE, and PAGER). On the other hand, the information that is only partly dependent on the EMP\_ID(each individual employee) is used to populate EMPLOYEE\_PAY\_TBL (EMP\_ID, POSITION,POSITION\_DESC, DATE\_HIRE, PAY\_RATE, and DATE\_LAST\_RAISE). Notice that both tables contain the column EMP\_ID. This is the primary key of each table and is used to match corresponding data between the two tables.

CUSTOMER\_TBL split into two tables called CUSTOMER\_TBL and ORDERS\_TBL. What took place is similar to what occurred in the EMPLOYEE\_TBL. Columns that were partly dependent on the primary key were directed to another table. The order information for a customer is dependent on each CUST\_ID, but does not directly depend on the general customer information in the original table.

#### The Third Normal Form

The third normal form's objective is to remove data in a table that is not dependent on the primary key. [Figure 4.4](javascript:popUp('/content/images/chap4_0672324423/elementLinks/04fig04_alt.jpg')) illustrates the third normal form.

[](javascript:popUp('/content/images/chap4_0672324423/elementLinks/04fig04_alt.jpg'))

[Figure 4.4](javascript:popUp('/content/images/chap4_0672324423/elementLinks/04fig04_alt.jpg')) **The third normal form.**

Another table was created to display the use of the third normal form. EMPLOYEE\_PAY\_TBL is split into two tables, one table containing the actual employee pay information and the other containing the position descriptions, which really do not need to reside in EMPLOYEE\_PAY\_TBL. The POSITION\_DESCcolumn is totally independent of the primary key, EMP\_ID .

### Naming Conventions

Naming conventions are one of the foremost considerations when you're normalizing a database. Names are how you will refer to objects in the database. You want to give your tables names that are descriptive of the type of information they contain so that the data you are looking for is easy to find. Descriptive table names are especially important for users querying the database that had no part in the database design. A company-wide naming convention should be set, providing guidance in the naming of not only tables within the database, but users, filenames, and other related objects. Designing and enforcing naming conventions is one of a company's first steps toward a successful database implementation.

### Benefits of Normalization

Normalization provides numerous benefits to a database. Some of the major benefits include the following :

* Greater overall database organization
* Reduction of redundant data
* Data consistency within the database
* A much more flexible database design
* A better handle on database security

Organization is brought about by the normalization process, making everyone's job easier, from the user who accesses tables to the database administrator (DBA) who is responsible for the overall management of every object in the database. Data redundancy is reduced, which simplifies data structures and conserves disk space. Because duplicate data is minimized, the possibility of inconsistent data is greatly reduced. For example, in one table an individual's name could read STEVE SMITH, whereas the name of the same individual reads STEPHEN R. SMITH in another table. Because the database has been normalized and broken into smaller tables, you are provided with more flexibility as far as modifying existing structures. It is much easier to modify a small table with little data than to modify one big table that holds all the vital data in the database. Lastly, security is also provided in the sense that the DBA can grant access to limited tables to certain users. Security is easier to control when normalization has occurred.

Data integrity is the assurance of consistent and accurate data within a database.

#### Referential Integrity

Referential integrity simply means that the values of one column in a table depend on the values of a column in another table. For instance, in order for a customer to have a record in the ORDERS\_TBLtable, there must first be a record for that customer in the CUSTOMER\_TBL table. Integrity constraints can also control values by restricting a range of values for a column. The integrity constraint should be created at the table's creation. Referential integrity is typically controlled through the use of primary and foreign keys.

In a table, a foreign key, normally a single field, directly references a primary key in another table to enforce referential integrity. In the preceding paragraph, the CUST\_ID in ORDERS\_TBL is a foreign key that references CUST\_ID in CUSTOMER\_TBL.

### Drawbacks of Normalization

Although most successful databases are normalized to some degree, there is one substantial drawback of a normalized database: reduced database performance. The acceptance of reduced performance requires the knowledge that when a query or transaction request is sent to the database, there are factors involved, such as CPU usage, memory usage, and input/output (I/O). To make a long story short, a normalized database requires much more CPU, memory, and I/O to process transactions and database queries than does a denormalized database. A normalized database must locate the requested tables and then join the data from the tables to either get the requested information or to process the desired data. A more in-depth discussion concerning database performance occurs in Hour 18, "Managing Database Users."

### Denormalizing a Database

Denormalization is the process of taking a normalized database and modifying table structures to allow controlled redundancy for increased database performance. Attempting to improve performance is the only reason to ever denormalize a database. A denormalized database is not the same as a database that has not been normalized. Denormalizing a database is the process of taking the level of normalization within the database down a notch or two. Remember, normalization can actually slow performance with its frequently occurring table join operations. (Table joins are discussed during Hour 13, "Joining Tables in Queries.") Denormalization may involve recombining separate tables or creating duplicate data within tables to reduce the number of tables that need to be joined to retrieve the requested data, which results in less I/O and CPU time.

There are costs to denormalization, however. Data redundancy is increased in a denormalized database, which can improve performance but requires more extraneous efforts to keep track of related data. Application coding renders more complications, because the data has been spread across various tables and may be more difficult to locate. In addition, referential integrity is more of a chore; related data has been divided among a number of tables. There is a happy medium in both normalization and denormalization, but both require a thorough knowledge of the actual data and the specific business requirements of the pertinent company.

## DB administration console

CREATE SCHEMA `ecare` DEFAULT CHARACTER SET utf8 COLLATE utf8\_general\_ci ;

services.msc

mysql.exe -u root -p

CREATE SCHEMA contour CHARACTER SET utf8;  
CREATE USER 'contour'@'localhost' IDENTIFIED BY 'contour';  
GRANT ALL PRIVILEGES ON contour\_test.\* TO 'contour'@'localhost';

mysql> show grants for ecare@localhost;

+--------------------------------------------------------------------------------------------------------------+

| Grants for ecare@localhost |

+--------------------------------------------------------------------------------------------------------------+

| GRANT USAGE ON \*.\* TO 'ecare'@'localhost' IDENTIFIED BY PASSWORD '\*320F70A89A4A086EFFA612967A8A1233B1F817A5' |

| GRANT ALL PRIVILEGES ON `ecare`.\* TO 'ecare'@'localhost' |

| GRANT ALL PRIVILEGES ON `ecare\_test`.\* TO 'ecare'@'localhost' |

+--------------------------------------------------------------------------------------------------------------+

Update contour.relations\_description

inner join countour.programs progs On relations\_description.objectId = progs.groupId

And relations\_description.space = 'group' and relations\_description.type = "Group"

set relations\_description.objectId = progs.id, relations\_description.type = "Program";

Режим инкогнито в Rest client

Advanced Rest Client мог подхватить мой профиль. Надо зайти из-под режима инкогнито (Ctrl + Shift + N) в Rest Client и запустить все то же самое.

Чтобы убедиться что дело все-таки в правах пользователя "[admin@company.com](mailto:admin@company.com)".

Если какая-то таблица осталась залоченой

show open tables;

UNLOCK TABLES;

# SQL

## My experience

### Query in MySQL:

SELECT projects.id, COUNT(\*), SUM(info.workload)/100.0 FROM contour.projects,

contour.relations

INNER JOIN contour.relations\_description userDes ON (contour.relations.description\_id = userDes.id)

INNER JOIN contour.relations\_description projectDes ON (contour.relations.parentDescription\_id = projectDes.id)

INNER JOIN contour.relations\_optional\_info info ON (contour.relations.id = info.id)

INNER JOIN contour.relations program ON program.description\_id = projectDes.id

INNER JOIN contour.relations\_description programDes ON program.parentDescription\_id = programDes.id AND programDes.type = 'Group'

INNER JOIN contour.groups programGroup ON programGroup.id = programDes.objectId AND programGroup.type = "Program"

WHERE

userDes.deleted <> true

AND userDes.type = 'User'

AND projectDes.type = 'Project'

AND projectDes.objectId = contour.projects.id

GROUP BY projects.id;

### Analog in HQL:

**private** **static** **final** String ***HQL*** = "SELECT "

+ " project.id, "

+ " COUNT(\*), "

+ " SUM(info.workload)/100.0 "

+ "FROM Project project, Relation relation "

+ "INNER JOIN relation.description AS userDes "

+ "INNER JOIN relation.parentDescription AS projectDes "

+ "INNER JOIN relation.optionalInfo AS info "

+ "WHERE "

+ " userDes.deleted <> true "

+ " AND userDes.type = 'User' "

+ " AND projectDes.type = 'Project' "

+ " AND projectDes.objectId = project.id "

+ "GROUP BY project.id";

## Good article in the internet (Queries example)

http://www.ntu.edu.sg/home/ehchua/programming/sql/MySQL\_Beginner.html

# Hibernate

[*A Short Primer On Fetching Strategies*](http://javatalks.ru/out?url=https://community.jboss.org/wiki/AShortPrimerOnFetchingStrategies?_sscc=t)*:*

Criteria respects the laziness settings in your mappings and guarantees that what you want loaded is loaded. This means one Criteria query might result in several SQL immediate SELECT statements to fetch the subgraph with all non-lazy mapped associations and collections. If you want to change the "how" and even the "what", use setFetchMode() to enable or disable outer join fetching for a particular collection or association. Criteria queries also completely respect the fetching strategy (join vs select vs subselect).  
HQL respects the laziness settings in your mappings and guarantees that what you want loaded is loaded. This means one HQL query might result in several SQL immediate SELECT statements to fetch the subgraph with all non-lazy mapped associations and collections. If you want to change the "how" and even the "what", use LEFT JOIN FETCH to enable outer-join fetching for a particular collection or nullable many-to-one or one-to-one association, or JOIN FETCH to enable inner join fetching for a non-nullable many-to-one or one-to-one association. HQL queries do not respect any fetch="join" defined in the mapping document.

## http://www.dil.univ-mrs.fr/~massat/docs/hibernate-2/reference/ru/html\_single/#quickstart-mapping

## Hibernate one-to-one mapping using annotations

[NOVEMBER 15, 2012](http://howtodoinjava.com/2012/11/15/hibernate-one-to-one-mapping-using-annotations/) [LOKESH](http://howtodoinjava.com/author/lokeshgupta1981/)[14 COMMENTS](http://howtodoinjava.com/2012/11/15/hibernate-one-to-one-mapping-using-annotations/#comments)

If you are working on any hibernate project or you are planning to work on any in future, then you can easily understand the one-to-one relationships between several entities in your application. In this post, i will discuss variations of one-to-one mappings supported in hibernate.

[**Download source code**](https://docs.google.com/open?id=0B7yo2HclmjI4VHJnQk4tYjBueDA)

**Sections in this post:**

Various supported techniques

Using foreign key association

Using a common join table

Using shared primary key

For this article, I am extending the example written [**for hello world example**](http://howtodoinjava.com/2012/11/12/hibernate-3-introduction-and-writing-hello-world-application/). We have two entities here: Employeeand Account.

### Various supported techniques

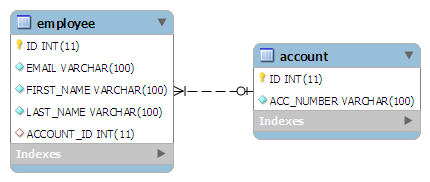
In hibernate there are 3 ways to create one-to-one relationships between two entities. Either way you have to use[**@OneToOne**](http://docs.oracle.com/javaee/5/api/javax/persistence/OneToOne.html) annotation. ***First technique*** is most widely used and uses a foreign key column in one to table. ***Second technique*** uses a rather known solution of having a third table to store mapping between first two tables. ***Third technique*** is something new which uses a common primary key value in both the tables.

Lets see them in action one by one:

### Using foreign key association

**In this association, a foreign key column is created in owner entity.**For example, if we make EmployeeEntity owner, then a extra column “ACCOUNT\_ID” will be created in Employee table. This column will store the foreign key for Account table.

Table structure will be like this:

[](http://howtodoinjava.files.wordpress.com/2012/11/foreign-key-association-one-to-one.png)

To make such association, refer the account entity in EmployeeEntity class as follow:

|  |  |
| --- | --- |
| 1  2  3 | @OneToOne  @JoinColumn(name="ACCOUNT\_ID")  private AccountEntity account; |

The join column is declared with the [**@JoinColumn**](http://docs.oracle.com/javaee/5/api/javax/persistence/JoinColumn.html) annotation which looks like the [**@Column**](http://docs.oracle.com/javaee/5/api/javax/persistence/Column.html) annotation. It has one more parameters named referencedColumnName. This parameter declares the column in the targeted entity that will be used to the join.

If no @JoinColumn is declared on the owner side, the defaults apply. A join column(s) will be created in the owner table and its name will be the concatenation of the name of the relationship in the owner side, \_ (underscore), and the name of the primary key column(s) in the owned side.

In a bidirectional relationship, one of the sides (and only one) has to be the owner: the owner is responsible for the association column(s) update. To declare a side as not responsible for the relationship, the attribute[***mappedBy***](http://docs.oracle.com/javaee/5/api/javax/persistence/OneToOne.html#mappedBy%28%29)is used. mappedBy refers to the property name of the association on the owner side.

|  |  |
| --- | --- |
| 1  2 | @OneToOne(mappedBy="account")  private EmployeeEntity employee; |

**Above “mappedBy” attribute declares that it is dependent on owner entity for mapping.**

Lets test above mappings in running code:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25 | public class TestForeignKeyAssociation {        public static void main(String[] args) {          Session session = HibernateUtil.getSessionFactory().openSession();          session.beginTransaction();            AccountEntity account = new AccountEntity();          account.setAccountNumber("123-345-65454");            // Add new Employee object          EmployeeEntity emp = new EmployeeEntity();          emp.setEmail("demo-user@mail.com");          emp.setFirstName("demo");          emp.setLastName("user");            // Save Account          session.saveOrUpdate(account);          // Save Employee          emp.setAccount(account);          session.saveOrUpdate(emp);            session.getTransaction().commit();          HibernateUtil.shutdown();      }  } |

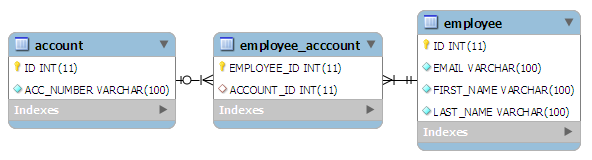
Running above code creates desired schema in database and run these SQL queries.

|  |  |
| --- | --- |
| 1  2 | Hibernate: insert into ACCOUNT (ACC\_NUMBER) values (?)  Hibernate: insert into Employee (ACCOUNT\_ID, EMAIL, FIRST\_NAME, LAST\_NAME) values (?, ?, ?, ?) |

You can verify the data and mappings in both tables when you run above program.. :-)

### ****Using a common join table****

This approach is not new to all of us. Lets start with targeted DB structure in this technique.

[](http://howtodoinjava.files.wordpress.com/2012/11/join-table-one-to-one-mapping.png)

In this technique, main annotation to be used is [**@JoinTable**](http://docs.oracle.com/javaee/5/api/javax/persistence/JoinTable.html). **This annotation is used to define the new table name (mandatory) and foreign keys from both of the tables**. Lets see how it is used:

|  |  |
| --- | --- |
| 1  2  3  4 | @OneToOne(cascade = CascadeType.ALL)  @JoinTable(name="EMPLOYEE\_ACCCOUNT", joinColumns = @JoinColumn(name="EMPLOYEE\_ID"),  inverseJoinColumns = @JoinColumn(name="ACCOUNT\_ID"))  private AccountEntity account; |

@JoinTable annotation is used in EmployeeEntity class. It declares that a new table EMPLOYEE\_ACCOUNT will be created with two columns EMPLOYEE\_ID (primary key of EMPLOYEE table) and ACCOUNT\_ID (primary key of ACCOUNT table).

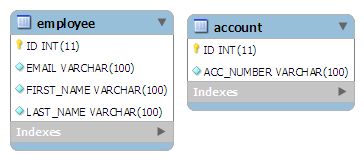
Testing above entities generates following SQL queries in log files:

|  |  |
| --- | --- |
| 1  2  3 | Hibernate: insert into ACCOUNT (ACC\_NUMBER) values (?)  Hibernate: insert into Employee (EMAIL, FIRST\_NAME, LAST\_NAME) values (?, ?, ?)  Hibernate: insert into EMPLOYEE\_ACCCOUNT (ACCOUNT\_ID, EMPLOYEE\_ID) values (?, ?) |

### ****Using shared primary key****

In this technique, hibernate will ensure that **it will use a common primary key value in both the tables**. This way primary key of EmployeeEntity can safely be assumed the primary key of AccountEntity also.

Table structure will be like this:

[](http://howtodoinjava.files.wordpress.com/2012/11/shared-primary-key-one-to-one.png)

In this approach, [**@PrimaryKeyJoinColumn**](http://docs.oracle.com/javaee/5/api/javax/persistence/PrimaryKeyJoinColumn.html) is the main annotation to be used.Let see how to use it.

|  |  |
| --- | --- |
| 1  2  3 | @OneToOne(cascade = CascadeType.ALL)  @PrimaryKeyJoinColumn  private AccountEntity account; |

In AccountEntity side, it will remain dependent on owner entity for the mapping.

|  |  |
| --- | --- |
| 1  2 | @OneToOne(mappedBy="account", cascade=CascadeType.ALL)  private EmployeeEntity employee; |

Testing above entities generates following SQL queries in log files:

|  |  |
| --- | --- |
| 1  2 | Hibernate: insert into ACCOUNT (ACC\_NUMBER) values (?)  Hibernate: insert into Employee (ACCOUNT\_ID, EMAIL, FIRST\_NAME, LAST\_NAME) values (?, ?, ?, ?) |

So, we have seen all 3 types of one to one mappings supported in hibernate. I will suggest you to download the source code and play with it.

Happy Learning !!

### [****Download source code****](https://docs.google.com/open?id=0B7yo2HclmjI4VHJnQk4tYjBueDA)

### Related Posts:

1. [Hibernate one-to-many mapping using annotations](http://howtodoinjava.com/2012/11/17/hibernate-one-to-many-mapping-using-annotations/)
2. [How to Define Association Mappings between Hibernate Entities](http://howtodoinjava.com/2014/09/05/how-to-define-association-mappings-between-hibernate-entities/)
3. [Hibernate many-to-many mapping using annotations](http://howtodoinjava.com/2012/11/17/hibernate-many-to-many-mapping-using-annotations/)
4. [Hibernate @NaturalId example tutorial](http://howtodoinjava.com/2013/08/06/hibernate-naturalid-example-tutorial/)
5. [Hibernate insert query tutorial](http://howtodoinjava.com/2013/06/28/hibernate-insert-query-tutorial/)
6. [Hibernate 4 example to get entity reference for lazy loading](http://howtodoinjava.com/2013/08/06/hibernate-4-example-to-get-entity-reference-for-lazy-loading/)

## Конспект "Hibernate reference manual"

Данная заметка не является мини-учебником или пошаговым руководством по написанию простого приложения с использованием Hibernate, но её можно воспринимать как шпаргалку, позволяющую быстро освежить в памяти знания по данному фреймоврку у тех, кто ими уже обладает, и помочь начинающим программистам, мало работавших с фреймворком, упорядочить свои знания "по полочкам".  
  
Включение поддержки Hibernate в проект  
  
Предположим, что ваше приложение использует систему сборки **Maven**. Тогда для включения поддержки библиотеки Hibernate нужно вставить следующие строки зависимости в ваш**pom.xml**:

[?](http://j4sq.blogspot.ru/2011/09/hibernate-reference-manual.html)

|  |  |
| --- | --- |
| 1  2  3  4 | <dependency>  <groupId>org.hibernate</groupId>  <artifactId>hibernate-core</artifactId>  </dependency> |

Класс для работы с Hibernate  
  
Класс, реализующий сущность базы данных должен удовлетворять **четырём правилам**:

1. Иметь **конструктор**по умолчанию (без аргументов).
2. Иметь **поле-идентификатор**. (не обязательно)
3. Быть **не final-классом** (не обязательно, но существенно для производительности)
4. Иметь**методы доступа** для свойств.

Предположим, что у нас есть следующий класс, который мы хотим сохранить в базе данных:

[?](http://j4sq.blogspot.ru/2011/09/hibernate-reference-manual.html)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23 | public class SimpleClass {      private Long id;      private String text;        public SimpleClass(){}        public Long getId() {          return id;      }        public void setId(Long id) {          this.id = id;      }        public String getText() {          return text;      }        public void setText(String text) {          this.text = text;      }    } |

Свяжем этот класс с таблицой базы данных. Для этого используются **файлы маппинга**. В нашем случае он может иметь вид:

[?](http://j4sq.blogspot.ru/2011/09/hibernate-reference-manual.html)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | <hibernate-mapping package=”com.mycompany.app">      <class name="SimpleClass" table="SIMPLECLASS">          <id name="id" column="SIMPLE\_ID">              <generator class="native"/>          </id>          <property name=”text”/>      </class>  </hibernate-mapping> |

Все тэги этого файла интуитивно очевидны, кроме тэга generator, о котором я расскажу подробнее чуть позже.  
  
В одном файле маппинга можно описывать множество классов. Это применимо для небольших приложений, в которых меньше пяти таблиц, и которые разрабатываются одним разработчиком. В противном случае рекомендуется под каждый класс, который вы хотите хранить в базе, создавать свой маппинг-файл.  
  
Настройка соединения с БД  
  
Существует несколько способов сказать Hibernate с какой БД мы будем работать и как к ней подключиться, из которых самым популярным является **файл конфигурации**, называемый**hibernate.cfg.xml.** Вот как он может выглядеть:

[?](http://j4sq.blogspot.ru/2011/09/hibernate-reference-manual.html)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | <?xml version="1.0" encoding="UTF-8"?>  <!DOCTYPE hibernate-configuration PUBLIC "-//Hibernate/Hibernate Configuration DTD 3.0//EN" "<http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd>">  <hibernate-configuration>    <session-factory>      <property name="hibernate.dialect">org.hibernate.dialect.OracleDialect</property>      <property name="hibernate.connection.driver\_class">oracle.jdbc.OracleDriver</property>      <property name="hibernate.connection.url">jdbc:oracle:thin:@//localhost </property>      <property name="hibernate.connection.username">user</property>      <property name="hibernate.connection.password">pass</property>      <mapping resource="com/mycompany/app/simpleClass.hbm.xml"/>    </session-factory>  </hibernate-configuration> |

Работа с базой данных проходит в транзакциях в пределах сессии. Таким образом перед тем, как начать использовать Hibernate в своем приложении, необходимо создать **фабрику сессий**, обычно оборачиваемую в класс с именем **HibernateUtil**. Вот пример готового класса:

[?](http://j4sq.blogspot.ru/2011/09/hibernate-reference-manual.html)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17 | public class HibernateUtil {      private static final SessionFactory sessionFactory = buildSessionFactory();      private static SessionFactory buildSessionFactory() {          try {              // Create the SessionFactory from hibernate.cfg.xml              return new Configuration().configure().buildSessionFactory();          }          catch (Throwable ex) {              // Make sure you log the exception, as it might be swallowed              System.err.println("Initial SessionFactory creation failed." + ex);              throw new ExceptionInInitializerError(ex);          }      }      public static SessionFactory getSessionFactory() {          return sessionFactory;      }  } |

Базовые операции  
  
Запись  
  
Существует два способа сохранить класс в базе данных: используя методы **persist()** или**save()**.   
Первый не гарантирует создания уникальных идентификаторов на момент вызова. Метод полезен для длительных операций.   
  
Метод **save()**возвращает уникальный идентификатор. Если для получения этого идентификатора должен быть вызван INSERT (т.е. используется генерация «identify», а не «sequence»), то INSERT вызывается сразу, вне зависимости в транзакции мы или нет. Это может создавать проблемы в длительных операциях, но мы можем явно указывать идентификатор перегруженным методом save(Class c, Long id).  
  
Чтобы форсировать запись можно воспользоваться методом**flush()**.   
  
Пример записи:

[?](http://j4sq.blogspot.ru/2011/09/hibernate-reference-manual.html)

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | Session session = HibernateUtil.getSessionFactory().getCurrentSession();  session.beginTransaction();  SimpleClass sc = new SimpleClass();  sc.setText(“some text”);  session.save(sc);  session.getTransaction().commit(); |

Если база данных построена так, что некоторые поля инициализируются триггерами, то правильно сохранять объекты так:

[?](http://j4sq.blogspot.ru/2011/09/hibernate-reference-manual.html)

|  |  |
| --- | --- |
| 1  2  3 | sess.save(sc);  sess.flush(); //форсируем вызов SQL INSERT  sess.refresh(sc); //перечитываем объект из базы, триггеры уже отработали |

По умолчанию метод **commit()**автоматически закрывает сессию, и при следующем обращении к getCurrentSession будет создана новая, так что делать **close()** или **disconnect()** не нужно.  
  
Если неизвестно будет ли создан новый объект или обновлен старый то следует использовать метод **saveOrUpdate()**.Следует помнить, что если вы не изменяете объекты из одной сессии в другой, то методы update(), saveOrUpdate() или merge() использовать не нужно.  
  
Обычно update() или saveOrUpdate() использует в следующем сценарии:

1. Приложение загружает данные в первой сессии
2. Информация показывается пользователю, который её модифицирует
3. Приложение обрабатывает эти изменения и сохраняет модификации вызывая update() в другой сессии.

Метод saveOrUpdate() делает следующее:

* если объект уже существует в этой сессии – не делает ничего;
* если другой объект в этой сессии имеет такой же идентификатор – генерит эксепшн;
* если объект не имеет идентификатора – вызывает save();
* если идентификатор объекта изменился – создает новый объект, т.е. вызывает save();
* если объект обладает версией (свойствами <version>или <timestamp>), то создает новый объект;
* в противном случае вызывается update().

Если объект загружен в одной сессии и сохраняется в другой, и потенциально существует вероятность того, что в базе он уже был изменен, то правильно его сохранять так:

[?](http://j4sq.blogspot.ru/2011/09/hibernate-reference-manual.html)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | // foo is an instance loaded by a previous Session  foo.setProperty("bar");  session = factory.openSession();  Transaction t = session.beginTransaction();  session.saveOrUpdate(foo); // Use merge() if "foo" might have been loaded already  t.commit();  session.close(); |

Hibernate сгенерит эксепшн во время сохранения, если конфликт был обнаружен. Вместо update() можно использовать **lock()** и использовать **LockMode.READ** если вы уверены, что объект не был изменен.  
  
Чтение  
  
Чтение из базы данных можно осуществлятся двумя методами: **get()** и **load()**.  
Пример чтения списка:

[?](http://j4sq.blogspot.ru/2011/09/hibernate-reference-manual.html)

|  |  |
| --- | --- |
| 1  2  3  4 | Session session = HibernateUtil.getSessionFactory().getCurrentSession();  session.beginTransaction();  List result = session.createCriteria(SimpleClass.class).list();  session.getTransaction().commit(); |

Пример чтения записи по ключу:

[?](http://j4sq.blogspot.ru/2011/09/hibernate-reference-manual.html)

|  |  |
| --- | --- |
| 1  2  3  4 | Session session = HibernateUtil.getSessionFactory().getCurrentSession();  session.beginTransaction();  SimpleClass sc = (SimpleClass) session.load(SimpleClass.class, Id);  session.getTransaction().commit(); |

Если объекта с данным идентификатором в базе не окажется, Hibernate сгенерирует эксепшн. Для решения этой ситуации можно воспользоваться методом **get(Class c, Long id)**, к примеру:

[?](http://j4sq.blogspot.ru/2011/09/hibernate-reference-manual.html)

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | SimpleClass sc = (SimpleClass) sess.get(SimpleClass.class, id);  if (sc ==null) {      sc = new SimpleClass ();      sess.save(sc, id);  }  return sc; |

Для массива вместо метода list(), который возвращает сразу все данные, можно использовать**итератор**. Его использование аткуально для повышения производительности, если мы предпологаем, что объект, с которым мы собираемся работать, находится в кеше. Иначе такой запрос будет медленнее, чем list(). Пример использования из hibernate reference:

[?](http://j4sq.blogspot.ru/2011/09/hibernate-reference-manual.html)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | // fetch ids  Iterator iter = sess.createQuery("from eg.Qux q order by q.likeliness").iterate();  while ( iter.hasNext() ) {      Qux qux = (Qux) iter.next(); // fetch the object      // something we couldnt express in the query      if ( qux.calculateComplicatedAlgorithm() ) {          // delete the current instance          iter.remove();          // dont need to process the rest          break;      }  } |

Удаление  
  
Чтобы удалить объект из базы данных используется метод **delete()**. Нужно быть внимательным, потому что объекты удаляются без учёта ключей, т.е. связи нужно удалять вручную. Помните: сначла удаляем «детей», а уже потом их «родителей». :)  
  
Критерий выборки  
  
Существует два способа задать критерий выборки: **HQL**и **критерион**. Первый похож на native-SQL и вероятно больше понравится программистам, которые не хотят расставаться с написанием старых SQL-запросов вручную, второй более объектно-ориентирован.  
  
Ниже приведу несколько примеров критерионов из hibernate reference:  
Пример 1:

[?](http://j4sq.blogspot.ru/2011/09/hibernate-reference-manual.html)

|  |  |
| --- | --- |
| 1  2  3  4 | Criteria crit = session.createCriteria(Cat.class);  crit.add( Restrictions.eq( "color", eg.Color.BLACK ) );  crit.setMaxResults(10);  List cats = crit.list(); |

Пример 2:

[?](http://j4sq.blogspot.ru/2011/09/hibernate-reference-manual.html)

|  |  |
| --- | --- |
| 1  2  3  4  5 | List cats = sess.createCriteria(Cat.class)      .add( Restrictions.like("name", "F%") )      .createCriteria("kittens")      .add( Restrictions.like("name", "F%") )      .list(); |

Пример 3:

[?](http://j4sq.blogspot.ru/2011/09/hibernate-reference-manual.html)

|  |  |
| --- | --- |
| 1  2  3  4  5 | List cats = sess.createCriteria(Cat.class)      .createAlias("kittens", "kt")      .createAlias("mate", "mt")      .add( Restrictions.eqProperty("kt.name", "mt.name") )      .list(); |

Семантику выборки можно установить метдом**setFetchMode()**, например этот код создаст запрос с **outer join**:

[?](http://j4sq.blogspot.ru/2011/09/hibernate-reference-manual.html)

|  |  |
| --- | --- |
| 1  2  3  4  5 | List cats = sess.createCriteria(Cat.class)      .add( Restrictions.like("name", "Fritz%") )      .setFetchMode("mate", FetchMode.EAGER)      .setFetchMode("kittens", FetchMode.EAGER)      .list(); |

О параметре **FetchMode**мы поговорим чуть ниже.  
  
Запросы «по примеру»  
  
Можно создать класс, заполнить в нем некоторые поля и затем искать в базе те сущности, у которых эти заполненные поля исходного класса будут равны. Например:

[?](http://j4sq.blogspot.ru/2011/09/hibernate-reference-manual.html)

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | Cat cat = new Cat();  cat.setSex('F');  cat.setColor(Color.BLACK);  List results = session.createCriteria(Cat.class)      .add( Example.create(cat) )      .list(); |

Идентификаторы и версии будут игнорированы. По умолчанию null valued свойства так же игнорируются. Можно задавать какие свойства игнорировать, например:

[?](http://j4sq.blogspot.ru/2011/09/hibernate-reference-manual.html)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | Example example = Example.create(cat)      .excludeZeroes() //exclude zero valued properties      .excludeProperty("color") //exclude the property named "color"      .ignoreCase() //perform case insensitive string comparisons      .enableLike(); //use like for string comparisons  List results = session.createCriteria(Cat.class)      .add(example)      .list(); |

Примеры можно использовать и в ассоциированных объектах:

[?](http://j4sq.blogspot.ru/2011/09/hibernate-reference-manual.html)

|  |  |
| --- | --- |
| 1  2  3  4  5 | List results = session.createCriteria(Cat.class)      .add( Example.create(cat) )      .createCriteria("mate")      .add( Example.create( cat.getMate() ) )      .list(); |

Проекции, аггрегация и группировка  
  
Тут всё более, чем очевидно, только взгляните на этот пример:

[?](http://j4sq.blogspot.ru/2011/09/hibernate-reference-manual.html)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | List results = session.createCriteria(Cat.class)      .setProjection( Projections.projectionList()          .add( Projections.rowCount() )          .add( Projections.avg("weight") )          .add( Projections.max("weight") )          .add( Projections.groupProperty("color") )      )      .list(); |

Или с группировкой:

[?](http://j4sq.blogspot.ru/2011/09/hibernate-reference-manual.html)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | List results = session.createCriteria(Cat.class)      .setProjection( Projections.projectionList()          .add( Projections.rowCount(), "catCountByColor" )          .add( Projections.avg("weight"), "avgWeight" )          .add( Projections.max("weight"), "maxWeight" )          .add( Projections.groupProperty("color"), "color" )      )      .addOrder( Order.desc("catCountByColor") )      .addOrder( Order.desc("avgWeight") )      .list(); |

Общие сведения  
  
Сессии  
  
Существует**три типа сессий** (устанавливается свойством hibernate.current\_session\_context\_class в hibernate.cfg.xml):

* **jta**- org.hibernate.context.JTASessionContext: сессии отслеживаются и ограничены JTA транзакциями. Работа с ними происходит так же, как при использовании старого JTA-подхода;
* **thread**- org.hibernate.context.ThreadLocalSessionContext:current сесии обрабатываются потоком;
* **managed**-  org.hibernate.context.ManagedSessionContext: сесии обрабатываются потоком. В отличии от thread программист сам отвечает за связывание сесии. Для сесии не делается open, flush и close.

Первые два типа реализуют модель "*one session - one database transaction*", так же известную как *session-per-request*.  
  
Если hibernate.current\_session\_context\_class установлен в **jta**, то пользоваться сессиями так:

[?](http://j4sq.blogspot.ru/2011/09/hibernate-reference-manual.html)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | // Non-managed environment idiom  Session sess = factory.openSession();  Transaction tx = null;  try {      tx = sess.beginTransaction();      // do some work      ...      tx.commit();  }  catch (RuntimeException e) {      if (tx != null) tx.rollback();      throw e; // or display error message  }  finally {      sess.close();  } |

Если установлен в **thread**, то так:

[?](http://j4sq.blogspot.ru/2011/09/hibernate-reference-manual.html)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | // Non-managed environment idiom with getCurrentSession()  try {      factory.getCurrentSession().beginTransaction();      // do some work      ...      factory.getCurrentSession().getTransaction().commit();  }  catch (RuntimeException e) {      factory.getCurrentSession().getTransaction().rollback();      throw e; // or display error message  } |

 Запросы можно формировать не в сессии, например:

[?](http://j4sq.blogspot.ru/2011/09/hibernate-reference-manual.html)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | DetachedCriteria query = DetachedCriteria.forClass(Cat.class)      .add( Property.forName("sex").eq('F') );  Session session = ....;  Transaction txn = session.beginTransaction();  List results = query.getExecutableCriteria(session).setMaxResults(100).list();  txn.commit();  session.close(); |

Постраничная выборка  
  
Постраничная выборка реализуется методами **setFirstResult()** и **setMaxResult()**, например так:

[?](http://j4sq.blogspot.ru/2011/09/hibernate-reference-manual.html)

|  |  |
| --- | --- |
| 1  2  3  4 | Query q = sess. createCriteria(SimpleClass.class);  q.setFirstResult(20);  q.setMaxResults(10);  List scList = q.list(); |

Методы equals() и hashCode()  
  
Чтобы класс мог полноценно поддерживаться в **many-to-one** и **many-to-many** и использовать**композитные ключи** (в этом случае класс еще и должен реализовывать интерфейс**Serializable**) нужно реализовать в нем методы **equals()** и **hashCode()**. Самым простым вариантом реализации сравнения двух сущностей является проверка их **идентификационных**номеров, однако следует помнить, что при использовании автогенерации айдишников они будут недоступны при работе, если объект не был сохранен в базе. В документации по гибернейт рекомендуют сравнивать объекты основываясь на семантике бизнесс-логики. К примеру:

[?](http://j4sq.blogspot.ru/2011/09/hibernate-reference-manual.html)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20 | public class Cat {      ...      public boolean equals(Object other) {          if (this == other) return true;          if ( !(other instanceof Cat) ) return false;            final Cat cat = (Cat) other;            if ( !cat.getLitterId().equals( getLitterId() ) ) return false;          if ( !cat.getMother().equals( getMother() ) ) return false;          return true;      }        public int hashCode() {          int result;          result = getMother().hashCode();          result = 29 \* result + getLitterId();          return result;      }  } |

Локировки  
  
Сущестуют разные виды локировки:

* LockMode.**WRITE**устанавливается автоматически, когда Hibernate обновляет или вставляет строку.
* LockMode.**UPGRADE**устанавливается, когда программист хочет использовать конструкцию SELECT ... FOR UPDATE на базах данных, поддерживающих этот синтаксис
* LockMode.**UPGRADE\_NOWAIT** устанавливается, когда программист хочет использовать конструкцию SELECT ... FOR UPDATE NOWAIT используя Oracle.
* LockMode.**READ**устанавливается автоматически, когда Hibernate читает данные из-под Repeatable Read или Serializable уровня изоляции.
* LockMode.**NONE**означает отсутствие локировки.  Все объекты переключаются на этот вид локировки в конце транзакции. Объекты, ассоциированные с сессией вызовом update()или saveOrUpdate()так же стартуют с этим видом.

Если выбранный тип локировки не поддерживается базой данных – используется ближайший «по смыслу» к нему и эксепшн не генерируется. Это позволяет сохранять приложение портируемым.  
  
Генератор идентификаторов  
  
Чтобы избежать проблемы с одинаковыми идентификаторами при работе приложения в кластере нужно в hibernate-mapping конфиге изменить **генератор**(тэг generator). Могут быть следующие значения:

* **increment**- генерирует идентификатор типа long, short или int, которые будет уникальным только в том случае, если другой процесс не добавляет запись в эту же таблицу в это же время.
* **identity**- генерирует идентификатор типа long, short или int. Поддерживается в DB2, MySQL, MS SQL Server, Sybase и HypersonicSQL.
* **sequence**- использует последовательности в DB2, PostgreSQL, Oracle, SAP DB, McKoi или генератор Interbase. Возвращает идентификатор типа long, short или int.
* **hilo**- использует алгоритм hi/lo для генерации идентификаторов типа long, short или int. Алгоритм гарантирует генерацию идентификаторов, которые уникальны только в данной базе данных.
* **seqhilo**- использует алгоритм hi/lo для генерации идентификаторов типа long, short или int учитывая последовательность базы данных.
* **uuid**- использует для генерации идентификатора алгоритм 128-bit UUID. Идентификатор будет уникальным в пределах сети. UUID представляется строкой из 32 чисел.
* **guid**- использует сгенерированую БД строку GUID в MS SQL Server и MySQL.
* **native**- использует identity, sequence или hilo в завимисимости от типа БД, с которой работает приложение
* **assigned**- позволяет приложению устанавливать идентификатор объекту, до вызова метода save(). Используется по умолчанию, если тег <generator> не указан.
* **select**- получает первичный ключ, присвоенный триггером БД
* **foreign**- использует идентификатор другого, связанного с данным объекта. Используется в <one-to-one> ассоциации первичных ключей.
* **sequence-identity** - специализированный генератор идентификатора. Используется только с драйевром Oracle 10g дл JDK 1.4.

Hibernate exceptions  
  
Если гибернейт бросает эксепшн, то нет никаких шансов восстановить работу сессии, потому что соответствие объектов в памяти и в базе более не гарантируется. Таким образом при любых эксепшенах вы должны закрыть сессию вызвав **Session.close()**.  
  
Версии  
  
Для предотвращения конфликтов используется механизм версий. Например:

[?](http://j4sq.blogspot.ru/2011/09/hibernate-reference-manual.html)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | // foo is an instance loaded by a previous Session  session = factory.openSession();  Transaction t = session.beginTransaction();  int oldVersion = foo.getVersion();  session.load( foo, foo.getKey() ); // load the current state  if ( oldVersion != foo.getVersion() ) throw new StaleObjectStateException();  foo.setProperty("bar");  t.commit();  session.close(); |

Свойство **version**замапленно используя **<version>** и гибернейт будет его автоматически инкрементировать согласно выбранной политике (даты, таймштампы, итераторы) **во время сохранения**. Если механизм версий не используется, то в базе будет та информация, которую сохранили последней. Так делать не рекомендуется, потому что пользователь потенциально будет терять свою информацию без всяких предупреждений и сообщений об ошибках. Ручное сравнение версий так же возможно, однако нерационально для большинства приложений.  
  
Производительность  
  
Политика получения данных  
  
При использовании связывания **many-to-one** по умолчанию используется выборка данных через **select**, а не через **join**. Переключается это свойством **fetch**в тэге many-to-one файла настройки связывания класса (\*.hbm.xml).  
Использование стратегии доступа по умолчанию к связанным сущностям через **select**потенциально добавляет проблем с производительностью. Их можно решив прописав что-то вроде:

[?](http://j4sq.blogspot.ru/2011/09/hibernate-reference-manual.html)

|  |  |
| --- | --- |
| 1  2  3  4 | <set name="permissions" fetch="join">      <key column="userId"/>      <one-to-many class="Permission"/>  </set> |

или

[?](http://j4sq.blogspot.ru/2011/09/hibernate-reference-manual.html)

|  |  |
| --- | --- |
| 1 | <many-to-one name="mother" class="Cat" fetch="join"/> |

Существуют следующие **типа fetch'a**:

* **Join fetching**: hibernate получает ассоциированные объекты и коллекции одним SELECT используя OUTER JOIN
* **Select fetching**: использует уточняющий SELECT чтобы получить ассоциированные объекты и коллекции. Если вы не установите lazy fetching определив lazy="false", уточняющий SELECT будет выполнен только когда вы запрашиваете доступ к ассоциированным объектам
* **Subselect fetching**: поведение такое же, как у предыдущего типа, за тем исключением, что будут загружены ассоциации для все других коллекций, «родительским» для которых является сущность, которую вы загрузили первым SELECT’ом.
* **Batch fetching**: оптимизированная стратегия вида select fetching. Получает группу сущностей или коллекций в одном SELECT’е.

Fetch определяет **как**будут получены объекты, а вот некоторые из стратегий, отвечающих за то **когда**они будут получены:

* **Immediate fetching**: ассоцииации, коллекции или аттрибуты загружаются вместе с загрузкой родительской записи.
* **Lazy collection fetching**:  коллекции загружаются тогда, когда над ними производятся операции. Этот тип используется по умолчанию для коллекций.
* **"Extra-lazy" collection fetching**: отдельные элементы коллекций загружаются из БД. Hibernate пытается не загружать всю коллекцию целиком. Применяется для больших коллекций.

Стратегиям выборки посвящена почти вся **19.1** глава документа **Hibernate Reference**.  
  
Кеш  
  
Каждый объект, который был получен в сессии кешируется. Если сессия очень длинная или мы получаем слишком много объектов, то чтобы не получить **OutOfMemoryException**нужно периодически вызывать **clear()** и **evict()**.  
  
Таймаут для сессии  
  
Существует механизм, позволяющий выставить таймаут для сессии (стоит учесть, что он не может быть использован в CMT бинах). Работает так:

[?](http://j4sq.blogspot.ru/2011/09/hibernate-reference-manual.html)

|  |  |
| --- | --- |
| 1  2  3 | //set transaction timeout to 3 seconds  sess.getTransaction().setTimeout(3);  sess.getTransaction().begin(); |

SQL “FOR UPDATE”  
  
Можно использовать правило SQL “FOR UPDATE” для получения объектов, например:

[?](http://j4sq.blogspot.ru/2011/09/hibernate-reference-manual.html)

|  |  |
| --- | --- |
| 1 | Cat cat = (Cat) sess.get(Cat.class, id, LockMode.UPGRADE); |

Если сервер базы данных не поддерживает операцию FOR UPDATE, то второй параметр метода get() будет проигнорирован. Это сделано для обеспечения переносимости кода между разными серверами баз данных.  
  
HQL  
Альтернативой использованию критерионов является язык HQL.  
Примеры запросов на языке HQL из Hibernate Reference:

[?](http://j4sq.blogspot.ru/2011/09/hibernate-reference-manual.html)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19 | List cats = session.createQuery(      "from Cat as cat where cat.birthdate < ?")      .setDate(0, date)      .list();  List mothers = session.createQuery(      "select mother from Cat as cat join cat.mother as mother where cat.name = ?")      .setString(0, name)      .list();  List kittens = session.createQuery(      "from Cat as cat where cat.mother = ?")      .setEntity(0, pk)      .list();  Cat mother = (Cat) session.createQuery(      "select cat.mother from Cat as cat where cat = ?")      .setEntity(0, izi)      .uniqueResult();]]  Query mothersWithKittens = (Cat) session.createQuery(      "select mother from Cat as mother left join fetch mother.kittens");      Set uniqueMothers = new HashSet(mothersWithKittens.list()); |

Параметры для запросов можно подставлять в **createQuery**через символ «**?**» или именовать. Второе лучше, потому что:

1. параметры можно перечислять в произвольном порядке;
2. параметры могут использоваться несколько раз в запросе;
3. параметры становятся самодокументируемыми.

Пример использования:

[?](http://j4sq.blogspot.ru/2011/09/hibernate-reference-manual.html)

|  |  |
| --- | --- |
| 1  2  3  4 | //named parameter (preferred)  Query q = sess.createQuery("from DomesticCat cat where cat.name = :name");  q.setString("name", "Fritz");  Iterator cats = q.iterate(); |

Или даже массивом:

[?](http://j4sq.blogspot.ru/2011/09/hibernate-reference-manual.html)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | //named parameter list  List names = new ArrayList();  names.add("Izi");  names.add("Fritz");  Query q = sess.createQuery("from DomesticCat cat where cat.name in (:namesList)");  q.setParameterList("namesList", names);  List cats = q.list(); |

Если запрос возврщает кортедж, то правильно его обрабатывать так:

[?](http://j4sq.blogspot.ru/2011/09/hibernate-reference-manual.html)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | Iterator kittensAndMothers = sess.createQuery(      "select kitten, mother from Cat kitten join kitten.mother mother")      .list()      .iterator();  while ( kittensAndMothers.hasNext() ) {      Object[] tuple = (Object[]) kittensAndMothers.next();      Cat kitten = (Cat) tuple[0];      Cat mother = (Cat) tuple[1];      ....  } |

## MyExperience in one-to-one relations in Hibernate

### 5.1.11. one-to-one

Ассоциация "один к одному" с другим персистентным классом можно объявить, используя элемент one-to-one.

<one-to-one

name="propertyName" (1)

class="ClassName" (2)

cascade="all|none|save-update|delete" (3)

constrained="true|false" (4)

outer-join="true|false|auto" (5)

property-ref="propertyNameFromAssociatedClass" (6)

access="field|property|ClassName" (7)

/>

|  |  |
| --- | --- |
| (1) | name: Имя свойства. |
| (2) | class (необязательно - по умолчанию определяется рефлексией исходя из типа поля): Имя ассоциированного класса. |
| (3) | cascade (необязательно) определяет какая операция будет выполняться каскадом от родительского объекта к ассоциированному. |
| (4) | constrained (необязательно) определяет то, что внешний ключ, ссылающийся на таблицу ассоциированного класса, ограничен первичным ключом этой таблицы. Эта опция влияет на порядок, в котором выполняются каскадные операции save() и delete() (а так же используется утилитой экспортирующей схему - schema export tool). |
| (5) | outer-join (необязательно - по умолчанию auto): Задействует извлечение ассоциированных объектов, используя объединения outer-join если опция hibernate.use\_outer\_join конфигурационного файла включена. |
| (6) | property-ref: (необязательно) Имя свойства ассоциированного класса, которое входит в первичный ключ данного класса. Если не указано, то используется первичный ключ ассоциированного класса. |
| (7) | access (необязательно, по умолчанию property): Стратегия, которую должен использовать Hibernate для доступа к данному полю. |

Существует два вида ассоциаций "один к одному":

* связь по первичному ключу
* связь по уникальному внешнему ключу

Для организации ассоциации по первичному ключу нет надобности в дополнительных столбцах; если две записи связаны такой ассоциацией, то это значит, что две записи в двух таблицах имеют одно и то же значение первичного ключа. Следовательно, если вы хотите ассоциировать два объекта, чтобы они были связаны по первичному ключу, то вы должны убедиться в том, что их идентификаторам присвоены одинаковые значения!

Для ассоциации по первичному ключу добавьте следующее отображение для классов Employee и Person соответственно.

<one-to-one name="person" class="Person"/>

<one-to-one name="employee" class="Employee" constrained="true"/>

Здесь Employee – родительская таблица

Теперь мы должны убедиться в том, что первичные ключи связанных записей в таблицах идентичны. Мы используем специальный генератор Hibernate foreign:

<class name="person" table="PERSON">

<id name="id" column="PERSON\_ID">

<generator class="foreign">

<param name="property">employee</param>

</generator>

</id>

...

<one-to-one name="employee"

class="Employee"

constrained="true"/>

</class>

Сохраняемому экземпляру класса Person присваивается тоже значение первичного ключа, которое присвоено экземпляру класса Employee на который ссылается свойство employee класса Person.

Какова же аналогия <one-to-one name="employee" class="Employee" constrained="true"/> ?

Какие аннотации нужно указать над свойством employee дочернего класса Person?

Ответ:

Hi, constrained="true" is use in case of shared primary key. (in case of parent child relationship, when child's primary key works as foreign key pointing to primary key of parent)..  
  
Equivalent annotation is  
  
@PrimaryKeyJoinColumn  
  
(for more information look at chapter 7 of "Java Persistence with Hibernate").

I know this is a bit late, but we found that setting optional=false on the @OneToOne annotation did what you're looking for.  
  
It's the opposite of constrained="true" from HBM, in that this association is required, such that null can never be returned for the association and a proxy is used for lazy loading.

\_\_\_\_\_\_

Файлы связи Relation.java – родительский класс

RelationOptionalInfo.java – дочерний класс

**package** com.t\_systems.contour.entity;

**import** javax.persistence.Column;

**import** javax.persistence.Entity;

**import** javax.persistence.FetchType;

**import** javax.persistence.GeneratedValue;

**import** javax.persistence.Id;

**import** javax.persistence.JoinColumn;

**import** javax.persistence.ManyToOne;

**import** javax.persistence.OneToOne;

**import** javax.persistence.Table;

@Entity

@Table(name = "relations")

**public** **class** Relation {

@Id

@GeneratedValue

@Column(name = "id", unique = **true**, nullable = **false**)

**private** Long id;

@ManyToOne

@JoinColumn(name = "description\_id")

**private** RelationDescription description;

@ManyToOne

@JoinColumn(name = "parentDescription\_id")

**private** RelationDescription parentDescription;

@OneToOne(mappedBy = "relation", fetch = FetchType.***LAZY***)

**private** RelationOptionalInfo optionalInfo;

@Column(name = "level")

**private** Integer level;

**public** Long getId() {

**return** id;

}

**public** Relation setId(Long id) {

**this**.id = id;

**return** **this**;

}

**public** RelationDescription getDescription() {

**return** description;

}

**public** Relation setDescription(RelationDescription description) {

**this**.description = description;

**return** **this**;

}

**public** RelationDescription getParentDescription() {

**return** parentDescription;

}

**public** Relation setParentDescription(RelationDescription parentDescription) {

**this**.parentDescription = parentDescription;

**return** **this**;

}

**public** Integer getLevel() {

**return** level;

}

**public** Relation setLevel(Integer level) {

**this**.level = level;

**return** **this**;

}

**public** RelationOptionalInfo getOptionalInfo() {

**return** optionalInfo;

}

**public** **void** setOptionalInfo(RelationOptionalInfo optionalInfo) {

**this**.optionalInfo = optionalInfo;

}

}

**package** com.t\_systems.contour.entity;

**import** javax.persistence.Column;

**import** javax.persistence.Entity;

**import** javax.persistence.FetchType;

**import** javax.persistence.GeneratedValue;

**import** javax.persistence.Id;

**import** javax.persistence.OneToOne;

**import** javax.persistence.PrimaryKeyJoinColumn;

**import** javax.persistence.Table;

**import** org.hibernate.annotations.GenericGenerator;

**import** org.hibernate.annotations.Parameter;

**import** com.fasterxml.jackson.annotation.JsonAutoDetect;

**import** com.fasterxml.jackson.annotation.JsonIgnore;

**import** com.fasterxml.jackson.annotation.JsonIgnoreProperties;

**import** com.fasterxml.jackson.annotation.JsonProperty;

@Entity

@Table(name = "relations\_optional\_info")

@JsonAutoDetect

@JsonIgnoreProperties(ignoreUnknown = **true**)

**public** **class** RelationOptionalInfo {

@GenericGenerator(name = "generator", strategy = "foreign", parameters = @Parameter(name = "property", value = "relation"))

@Id

@GeneratedValue(generator = "generator")

@Column(name = "id")

@JsonProperty(value = "id")

**private** Long id;

@Column(name = "workload")

@JsonProperty(value = "workload")

**private** Byte workload;

// Adding Relation to RelationOptionalInfo with @OneToOne &

// @PrimaryKeyJoinColumn

// annotations tells Hibernate that the RelationOptionalInfo has a

// one-to-one mapping with

// Relation by using the Primary Key as the join column.

@PrimaryKeyJoinColumn

@OneToOne(optional = **false**, fetch = FetchType.***LAZY***)

@JsonIgnore

**private** Relation relation;

// /

// / Getters and setters

// /

**public** Byte getWorkload() {

**return** workload;

}

**public** **void** setWorkload(Byte workload) {

**this**.workload = workload;

}

**public** Relation getRelation() {

**return** relation;

}

**public** **void** setRelation(Relation relation) {

**this**.relation = relation;

}

**public** Long getId() {

**return** id;

}

**public** **void** setId(Long relationId) {

**this**.id = relationId;

}

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Как альтернативный вариант описания связи "один к одному" от Employee к Person, через уникальный внешний ключ можно использовать следующую запись:

<many-to-one name="person" class="Person" column="PERSON\_ID" unique="true"/>

Эта ассоциация может быть двунаправленной после добавления следующего выражения к маппингу класса Person:

<one-to-one name"employee" class="Employee" property-ref="person"/>

## HIBERNATE - Relational Persistence for Idiomatic Java

http://www.dil.univ-mrs.fr/~massat/docs/hibernate-2/reference/ru/html\_single/#quickstart-mapping

# Debugging

## Running Tomcat in Debug mode

To run tomcat in debug mode Script is located in:

c:\Users\btn\Documents\projects\

1. Firstly you should start server in debug mode. Run it in special mode

Sasha’s file:

We should start Tomcat with the following settings:

Create run-tomcat.bat and use it to run Tomcat

SET CATALINA\_HOME=C:\workspace\Contour\_workspace\apache-tomcat-8.0.9  
SET JAVA\_HOME=C:\Program Files\Java\jdk1.8.0\_11  
SET JRE\_HOME=C:\Program Files\Java\jdk1.8.0\_11\jre  
  
SET CATALINA\_OPTS=-Xdebug -Xrunjdwp:transport=dt\_socket,address=9995,server=y,suspend=n  
  
C:\workspace\Contour\_workspace\apache-tomcat-8.0.9\bin\catalina.bat run > C:\workspace\Contour\_workspace\tomcat.log

My file located in C:\Users\btn\Documents\projects\run-tomcat.bat

SET CATALINA\_HOME=C:\Program Files\Tomcat\apache-tomcat-8.0.14-windows-x64\apache-tomcat-8.0.14

SET JAVA\_HOME=C:\Program Files\Java\jdk1.8.0\_25

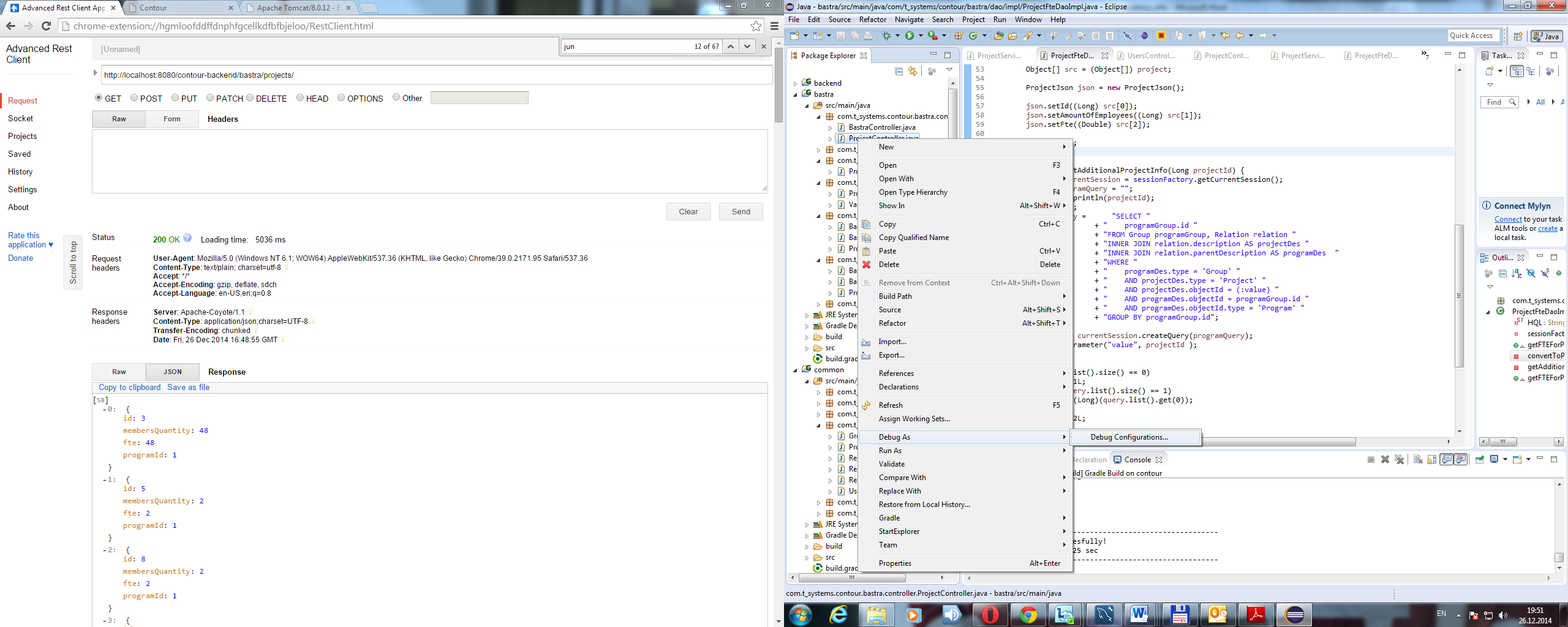
SET JRE\_HOME=C:\Program Files\Java\jre1.8.0\_25

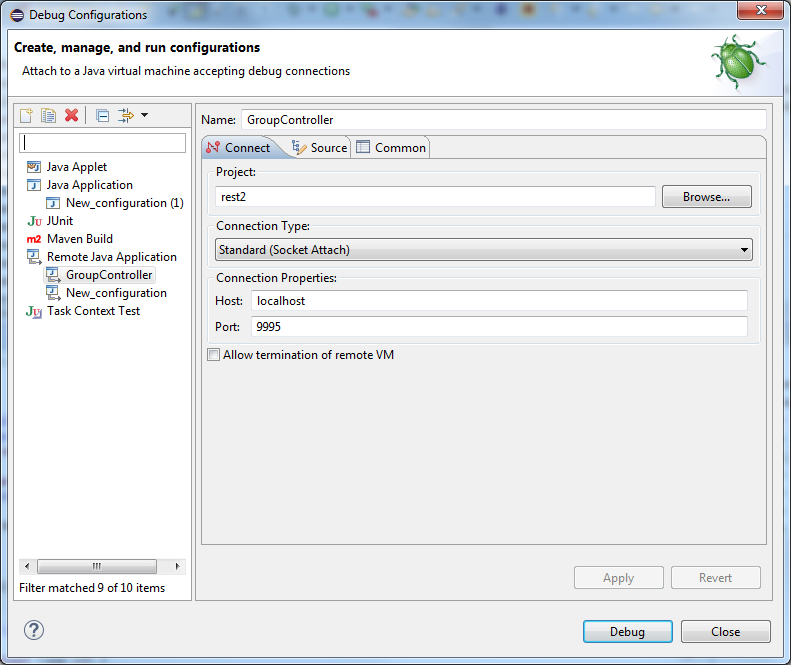
SET CATALINA\_OPTS=-Xdebug -Xrunjdwp:transport=dt\_socket,address=9995,server=y,suspend=n

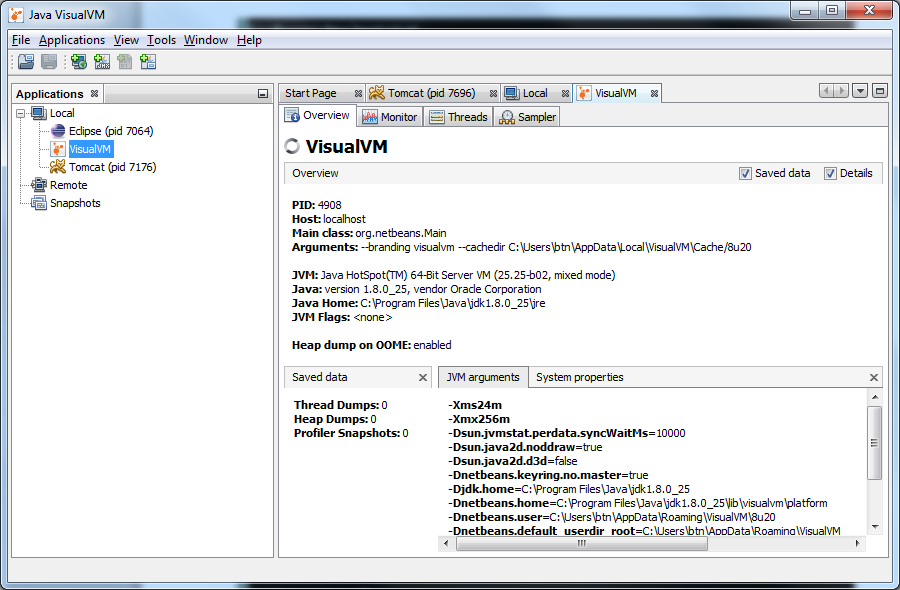
cd C:\Program Files\Tomcat\apache-tomcat-8.0.14-windows-x64\apache-tomcat-8.0.14\bin\

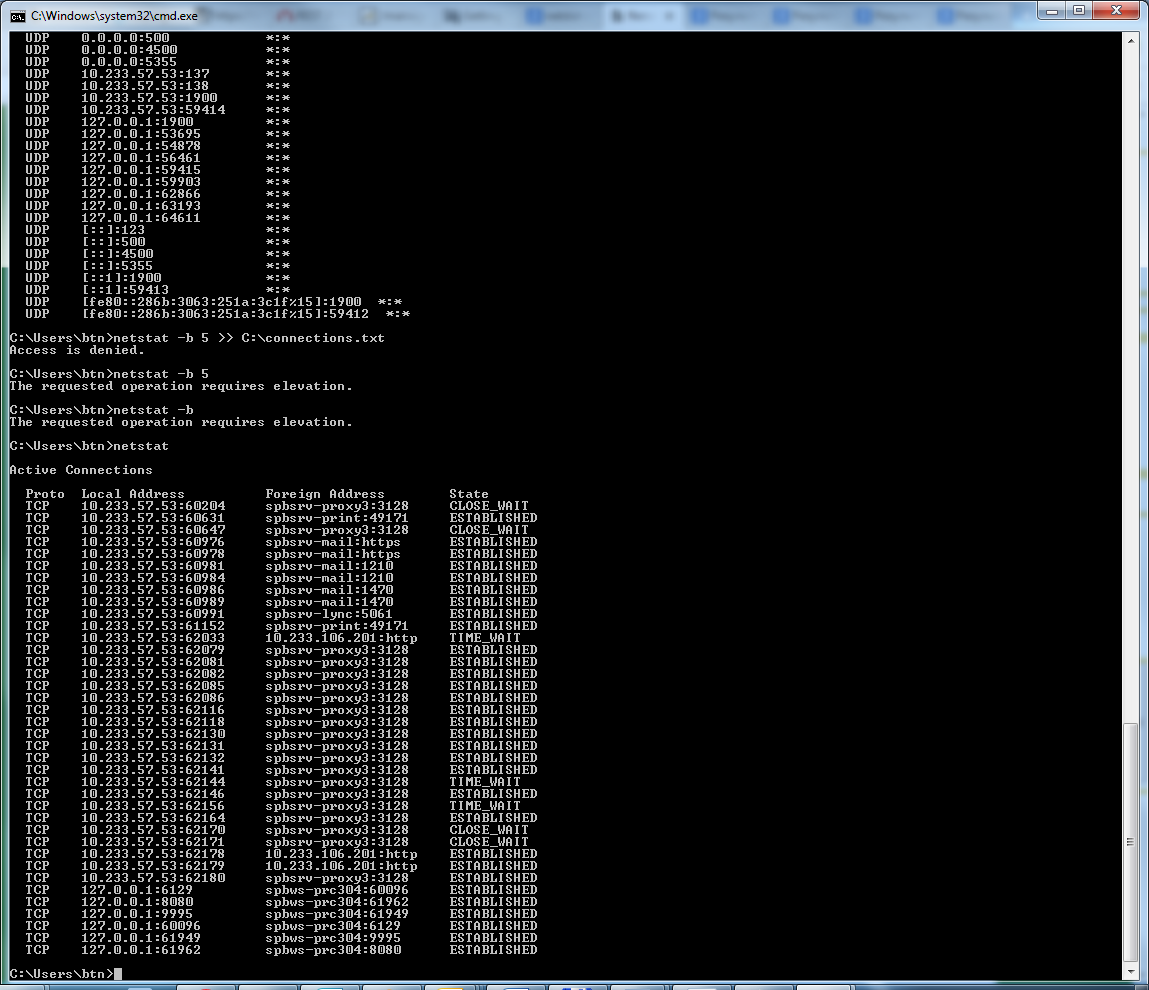
catalina.bat run > C:\Users\btn\Documents\projects\tomcat.log

1. We need to set port and add all projects that we needed.





1. Any problems? Go to c:\Program Files\Java\jdk1.8.0\_25\bin\ and run jvisualvm.exe
2. 



*netstat -ntpl | grep 9995*

# Как разрешить удаленный дебаг на CentOS

Да, можно задизейблить службу фаервола.

Для сервера в подсети разработки она необязательна.

Если бы этот сервер находился в подесети ДМЗ, т.е. порты были выставлены бы в открытую сеть интернет, то тогда имеет смысл включать фаервол.

**From:** Poskochinova, Tatiana   
**Sent:** Wednesday, October 07, 2015 5:25 PM  
**To:** Ivanov, Oleg  
**Subject:** RE: уточнение к заявке № IM-CL-40334 НАШЛА ПРИЧИНУ!!!

Получается что я точечно разрешила, а можно просто убрать все эти правила.

**From:** Ivanov, Oleg   
**Sent:** Wednesday, October 07, 2015 5:24 PM  
**To:** Poskochinova, Tatiana  
**Subject:** RE: уточнение к заявке № IM-CL-40334 НАШЛА ПРИЧИНУ!!!

Молодец! 

В моем предыдущем письме рекомендации!

**From:** Poskochinova, Tatiana   
**Sent:** Wednesday, October 07, 2015 4:20 PM  
**To:** Poskochinova, Tatiana  
**Cc:** Ivanov, Oleg  
**Subject:** RE: уточнение к заявке № IM-CL-40334 НАШЛА ПРИЧИНУ!!!

Олег!!!

Я нашла в чем причина!

Это SE linux. У CentOS своя security дополнительная.

Я сделала

iptables -I INPUT -p tcp --dport 9997 -j ACCEPT и удаленный дебаг заработал

а если сделать iptables -I INPUT -p tcp --dport 9997 -j REJECT

то опять работать не будет

только вот странно, что service iptables stop у меня не работает.

[root@fmip-vm1 home]# service iptables stop

Redirecting to /bin/systemctl stop  iptables.service

Failed to issue method call: Unit iptables.service not loaded.

**From:** Poskochinova, Tatiana   
**Sent:** Wednesday, October 07, 2015 3:09 PM  
**To:** Ivanov, Oleg  
**Subject:** RE: уточнение к заявке № IM-CL-40334

[root@fmip-vm1 home]# iptables -L|more

Chain INPUT (policy ACCEPT)

target     prot opt source               destination

ACCEPT     all  --  anywhere             anywhere             ctstate RELATED,ES

TABLISHED

ACCEPT     all  --  anywhere             anywhere

INPUT\_direct  all  --  anywhere             anywhere

INPUT\_ZONES\_SOURCE  all  --  anywhere             anywhere

INPUT\_ZONES  all  --  anywhere             anywhere

ACCEPT     icmp --  anywhere             anywhere

REJECT     all  --  anywhere             anywhere             reject-with icmp-h

ost-prohibited

Chain FORWARD (policy ACCEPT)

target     prot opt source               destination

ACCEPT     all  --  anywhere             anywhere             ctstate RELATED,ES

TABLISHED

ACCEPT     all  --  anywhere             anywhere

FORWARD\_direct  all  --  anywhere             anywhere

FORWARD\_IN\_ZONES\_SOURCE  all  --  anywhere             anywhere

FORWARD\_IN\_ZONES  all  --  anywhere             anywhere

FORWARD\_OUT\_ZONES\_SOURCE  all  --  anywhere             anywhere

FORWARD\_OUT\_ZONES  all  --  anywhere             anywhere

ACCEPT     icmp --  anywhere             anywhere

REJECT     all  --  anywhere             anywhere             reject-with icmp-host-prohibited

Chain OUTPUT (policy ACCEPT)

target     prot opt source               destination

OUTPUT\_direct  all  --  anywhere             anywhere

Chain FORWARD\_IN\_ZONES (1 references)

target     prot opt source               destination

FWDI\_public  all  --  anywhere             anywhere            [goto]

FWDI\_public  all  --  anywhere             anywhere            [goto]

Chain FORWARD\_IN\_ZONES\_SOURCE (1 references)

target     prot opt source               destination

Chain FORWARD\_OUT\_ZONES (1 references)

target     prot opt source               destination

FWDO\_public  all  --  anywhere             anywhere            [goto]

FWDO\_public  all  --  anywhere             anywhere            [goto]

Chain FORWARD\_OUT\_ZONES\_SOURCE (1 references)

target     prot opt source               destination

Chain FORWARD\_direct (1 references)

target     prot opt source               destination

Chain FWDI\_public (2 references)

target     prot opt source               destination

FWDI\_public\_log  all  --  anywhere             anywhere

FWDI\_public\_deny  all  --  anywhere             anywhere

FWDI\_public\_allow  all  --  anywhere             anywhere

Chain FWDI\_public\_allow (1 references)

target     prot opt source               destination

Chain FWDI\_public\_deny (1 references)

target     prot opt source               destination

Chain FWDI\_public\_log (1 references)

target     prot opt source               destination

Chain FWDO\_public (2 references)

target     prot opt source               destination

FWDO\_public\_log  all  --  anywhere             anywhere

FWDO\_public\_deny  all  --  anywhere             anywhere

FWDO\_public\_allow  all  --  anywhere             anywhere

Chain FWDO\_public\_allow (1 references)

target     prot opt source               destination

Chain FWDO\_public\_deny (1 references)

target     prot opt source               destination

Chain FWDO\_public\_log (1 references)

target     prot opt source               destination

Chain INPUT\_ZONES (1 references)

target     prot opt source               destination

IN\_public  all  --  anywhere             anywhere            [goto]

IN\_public  all  --  anywhere             anywhere            [goto]

Chain INPUT\_ZONES\_SOURCE (1 references)

target     prot opt source               destination

Chain INPUT\_direct (1 references)

target     prot opt source               destination

Chain IN\_public (2 references)

target     prot opt source               destination

IN\_public\_log  all  --  anywhere             anywhere

IN\_public\_deny  all  --  anywhere             anywhere

IN\_public\_allow  all  --  anywhere             anywhere

Chain IN\_public\_allow (1 references)

target     prot opt source               destination

ACCEPT     tcp  --  anywhere             anywhere             tcp dpt:ssh ctstate NEW

Chain IN\_public\_deny (1 references)

target     prot opt source               destination

Chain IN\_public\_log (1 references)

target     prot opt source               destination

Chain OUTPUT\_direct (1 references)

target     prot opt source               destination

**From:** Ivanov, Oleg   
**Sent:** Wednesday, October 07, 2015 2:41 PM  
**To:** Poskochinova, Tatiana  
**Subject:** RE: уточнение к заявке № IM-CL-40334

Вижу что есть успешные подключения по 9997 и 17871

Т.о. с моей стороны проблем нет.

На локальной машине проверь iptables –L

Если там пусто, то никаких правил не применено.

Если что то есть,

Пришли!

**From:** Poskochinova, Tatiana   
**Sent:** Wednesday, October 07, 2015 2:24 PM  
**To:** Ivanov, Oleg  
**Subject:** RE: уточнение к заявке № IM-CL-40334

Привет.

Повторила, то что делала вчера и сегодня где-то в 14.08

Я запустила два раза с разными портами.

Первый раз с портом 9997 и второй с существующим.

У меня такой вопрос, может ты знаешь, откуда я должна брать порт для дебага из головы?

Допустим я хочу получать и отправлять дебажную информацию по порту 9997.

Тогда на виртуальной машине я запускаю свою программу с соответствующими опциями:

[root@fmip-vm1 home]# java -agentlib:jdwp=transport=dt\_socket,server=y,suspend=y,address=9997 -jar messageArchiver.jar

Я потом с Windows удаленной машины  пытаюсь подсоединиться дебаггером.

Но если попросить вывести порт на виртуалке, то он не отображается. Его не существует.

[root@fmip-vm1 home]# netstat -al | grep 9997

А если попробовать с существующим, то все равно дебаггер не подключается.

[root@fmip-vm1 home]# netstat -al | grep 17871

unix  3      [ ]         STREAM     CONNECTED     17871

Что значит «Проверь локальный фаервол на виртуалке».

Мне туда вроде CentOs поставили и джава и больше ничего не ставили. Как его проверить? Куда зайти? Что посмотреть?

**From:** Ivanov, Oleg   
**Sent:** Wednesday, October 07, 2015 12:54 PM  
**To:** Poskochinova, Tatiana  
**Subject:** RE: уточнение к заявке № IM-CL-40334

Привет.

Вчера было несколько удачных подключений по SSH

Сегодня мониторинг был выключен.

Включил снова,

Попробуй, пожалуйста, что ты там делала утром

Но сразу скажу что ограничений на ферволе нет.

Если SSH прошел, то и другие пройдут.

Проверь локальный фаервол на виртуалке.

**From:** Poskochinova, Tatiana   
**Sent:** Wednesday, October 07, 2015 9:53 AM  
**To:** Ivanov, Oleg  
**Subject:** RE: уточнение к заявке № IM-CL-40334

Привет.

Где-то минуту назад попробовала достучаться до виртуалки.

Но опять получила Failed to connect to remote VM. Connection timed out.

org.eclipse.jdi.TimeoutException

**From:** Ivanov, Oleg   
**Sent:** Tuesday, October 06, 2015 4:15 PM  
**To:** Poskochinova, Tatiana  
**Subject:** RE: уточнение к заявке № IM-CL-40334

Татьяна привет,

Если я правильно тебя понял твой eclipse находится на одном из хостов в офисных подсетях и

Ты подключаешься к ВМ в подсети разработки 10.233.53.0.

С моей стороны такой доступ не ограничен.

Включил журнализирование на сетевом оборудовании,

Попробуй подключиться, я посмотрю что видно из обращений к 10.233.53.80 например

С Уважением

PQIT

Олег Иванов

Инфраструктура, данные и безопасность

Старший специалист

тел. 2007

**From:** Poskochinova, Tatiana   
**Sent:** Tuesday, October 06, 2015 2:55 PM  
**To:** Ivanov, Oleg  
**Subject:** уточнение к заявке № IM-CL-40334

У меян такой вопрос действительно не коннектится мой eclipse к порту виртуальной машины из-за того, что не соединены физические порты и виртуальные.

Если это действительно надо делать, то мне для проверки функциональности ещё потребуются порты

164, 162, 515 мжно ещё 516, 163 и 514

Ну и для дебага какой-нибудь 9997 или что-то в этом духе

И на второй виртуалке тоже надо это сделать.

# Testing

## Эти ссылки могут быть полезными:

<http://www.petrikainulainen.net/programming/spring-framework/unit-testing-of-spring-mvc-controllers-normal-controllers#writing-unit-tests>

<http://nat.truemesh.com/archives/000714.html>

# JSON

## JSON validator:

http://jsonformatter.curiousconcept.com/

http://wiki.fasterxml.com/JacksonAnnotations

## Every day Jackson usage, part 3: Filtering properties

(part of continuing series on covering common usage patterns with [Jackson JSON processor](http://wiki.fasterxml.com/JacksonHome); previous entry covered [handling open (extensible) content](http://www.cowtowncoder.com/blog/archives/2010/09/entry_414.html))

One of first things users eventually want to configure when using Jackson is (re)defining which Java object properties to serialize (written out) and which not.

**1. What gets serialized by default?**

Properties of an object are initially determined by process called auto-detection: all member methods and fields are checked to find:

1. **"Getter" methods: all no-argument public member methods which return a value, and conform to naming convention of "getXxx" (or "isXxx", iff return type is boolean; called "is-getter") are considered to infer existence of property with name "xxx" (where property name is inferred using bean convention, i.e. the leading capitali letter(s) is changed to lower case**
2. **field properties: all public member fields are considered to represent properties, using field name as is.**

In case both a getter and a field is found for same logical property, getter method has precedence and is used (field is ignored).

Set of properties introspected using this process are considered to be the base set of properties. But the auto-detection process itself can be configured differently, and there are multiple annotations and configuration settings that can further change actual effective set of properties to serialize.

**2. Changing auto-detection defaults: @JsonAutoDetect**

If the default auto-detection visibility limits (fields and member methods needing to be public) are not to your liking, it is easy to change them by using one of following method:

* **@JsonAutoDetect annotation can be defined for classes; properties "fieldVisibility", "getterVisibility" and "isGetterVisibility" define minimum visibility needed to include property (for fields, getters and is-getters, respectively). It is possible to, for example, include all field properties regardess of visibility (@JsonAutoDetect(fieldVisibility=Visibility.ANY)); or to disable getter-method auto-detection (@JsonAutoDetect(getterVisibility=Visibility.NONE)), and combinations there-of. Note that this annotation (just like any other Jackson annotation) can be applied as a**[**mix-in annotation**](http://wiki.fasterxml.com/JacksonMixInAnnotations)**, without having to modify type directly; and can be added to a base type to apply to all subtypes.**
* **ObjectMapper.setVisibilityChecker() can be used to define customized minimum visibility detection**

Changing minimum auto-detection visibility limits is an easy to way to increase number of properties discovered (for example by exposing all member fields; similar to how libraries like XStream and Gson work by default), or to prevent any and all auto-detection (i.e. to force explicit annotations using @JsonProperty or @JsonGetter annotation).

As an example, to serialize all fields (and use no getter methods), you could do:

@JsonAutoDetect(fieldVisibility=Visibility.ANY,

getterVisibility=Visibility.NONE, isGetterVisibility=Visibility.NONE)

public class FieldsOnlyBean {

private String name; // will now be used instead of getName()

public String getName() { throw new Error(); } // never used!

}

**3. Explicitly ignoring properties: @JsonIgnore, @JsonIgnoreProperties**

Given set of auto-detect potential properties is the starting point; and it can further be modified by per-property annotations:

* **@JsonProperty (and @JsonGetter, @JsonAnyGetter) can be used to indicate that a field or method is to be consider a property field or getter method, even if it isn't auto-detected.**
* **@JsonIgnore can be used to forcibly prevent inclusion, regardless of auto-detection (or other annotations)**

In addition, there is per-class annotation **@JsonIgnoreProperties** that can be used alternatively to list names of logical properties NOT to include for serialization; it may be easier to use via mix-in annotations than per-property annotations (although both can be used via mix-in annotations).

So you could do:

@JsonIgnoreProperties({ "internal" })

public class Bean {

public Settings getInternal() { ... } // ignored

@JsonIgnore public Settinger getBogus(); // likewise ignored

public String getName(); // but this would be serialized

}

**4. Defining profiles for dynamic ignoral: JSON Views (@JsonView)**

So far configuration methods have been applied statically; meaning that a property will either always be included (except for special case of possibly suppressing null-values), or never included.

JSON views are a way to define more dynamic inclusion/exclusion strategy. The idea is to define inclusion rules for properties, by associating logical views (classes used as identifiers; this allows use of hierarchic views) with properties, using **@JsonView** annotations; and then specifying which view is to be used for serialization. This is often used to define smaller "public" set of properties, and larger "private" or "confidential" set of properties. See[@JsonView wiki page](http://wiki.fasterxml.com/JacksonJsonViews) for usage example.

**5. Ignoring all properties with specified type: @JsonIgnoreType**

In addition to defining rules on per-property basis, there are times when it makes sense to just prevent serialization of any auto-detected properties for given type(s). For example, many frameworks add specific accessors for types they generate, which return objects that should not be serialized.

For example, let's say that an Object-Relational Mapper always adds "public Schema getSchema()" accessors for all value classes. And if this is metadata that is not part of serializable state, we can prevent its inclusion in serialization by adding **@JsonIgnoreType**annotation on Schema type (or its supertype). This is often easiest done using mix-in annotations.

**6. Fully dynamic filtering: @JsonFilter**

Although JSON views allow somewhat dynamic filtering, definitions of filters are still static. This means that it is only possible to dynamically choose from a static set of views.

JSON Filters are a way to implement fully dynamic filtering. The way this is done is by defining logical filter a property uses, with @JsonFilter("id") annotation, but specifying actual filter (and its configuration) using ObjectWriter. Filters themselves are obtained via FilterProviders which can be fully custom, or based on simple implementations. Check out[JSON Filter](http://wiki.fasterxml.com/JacksonFeatureJsonFilter) wiki page for details.

**7 . Most extreme way to filter out properties: BeanSerializerModifier**

And if ability to define custom filters is not enough, the ultimate in configurability is ability to modify configuration and configuration of **BeanSerializer** instances. This makes it possible to do all kinds of modifications (changing order in which properties are serialized; adding, removing or renaming properties; replacing serializer altogether with a custom instance and so on): you can completely re-wire or replace serialization of regular POJO ("bean") types.

This is achieved by adding a BeanSerializerModifier: the simplest way to do this is by using[Module interface](http://wiki.fasterxml.com/JacksonFeatureModules). Details of using BeanSerializerModifier are more advanced topic; I hope to cover it separately in future. The basic idea is that BeanSerializerModifier instance defines callbacks that Jackson BeanSerializerFactory calls during construction of a serializer.

## Jackson Annotations

This page lists all general-purpose Jackson 2.0 annotations, grouped by functionality.

For older (Jackson 1.x) annotations, refer to [FasterXML Wiki](http://wiki.fasterxml.com/JacksonAnnotations).

All annotations include a brief explanation, and (in near future!) a link to full explanation with basic usage examples.

**NOTE**: Contributions welcome!!!

### Property Naming

* @JsonProperty (also indicates that property is to be included) is used to indicate external property name, name used in data format (JSON or one of other supported data formats)

### Property Inclusion

* @JsonAutoDetect: class annotation used for overriding property introspection definitions
* @JsonIgnore: simple annotation to use for ignoring specified properties
* @JsonIgnoreProperties: per-class annotation to list properties to ignore, or to indicate that any unknown properties are to be ignored.
  + On serialization, @JsonIgnoreProperties({"prop1", "prop2"}) ignores listed properties
  + On deserialization, @JsonIgnoreProperties(ignoreUnknown=true) ignores properties that don't have getter/setters
* @JsonIgnoreType: per-class annotation to indicate that all properties of annotated type are to be ignored.
* @JsonInclude: annotation used to define if certain "non-values" (nulls or empty values) should not be included when serializing; can be used on per-property basis as well as default for a class (to be used for all properties of a class)

### Deserialization and Serialization details

* @JsonFormat: general annotation that has per-type behavior; can be used for example to specify format to use when serializing Date/Time values.
* @JsonUnwrapped: property annotation used to define that value should be "unwrapped" when serialized (and wrapped again when deserializing), resulting in flattening of data structure, compared to POJO structure.
* @JsonView: property annotation used for defining View(s) in which property will be included for serialization, deserialization.

### Deserialization details

* @JacksonInject: annotation to indicate that property should get its value via "injection", and not from data (JSON).
* @JsonAnySetter: annotation used for defining a two-argument method as "any setter", used for deserializing values of otherwise unmapped JSON properties
* @JsonCreator: annotation used for indicating that a constructor or static factory method should be used for creating value instances during deserialization.
* @JsonSetter: alternative to @JsonProperty, for marking that specified method is a "setter-method"

### Serialization details

* @JsonAnyGetter: annotation used to define a getter as "any getter", which returns ajava.util.Map, contents of which will be serialized as additional properties for JSON Object, along with regular properties that the Object may have.
* @JsonGetter: alternative to @JsonProperty, for marking that specified method is a "getter-method"
* @JsonPropertyOrder: annotation for specifying order in which properties are serialized
* @JsonRawValue: per-property marker that can be used to specify that the value of property is to be included in serialization ''exactly'' as is, with no escaping or decoration -- useful for embedding pre-serialized JSON (or whatever data format is being used) in output
* @JsonValue: per-property marker to indicate that the POJO should serialization is to be done using value of the property, often a java.lang.String (like annotationtoString() method)
* @JsonRootName: class annotation used to indicate name of "wrapper" entry used for root value, if root-wrapping is enabled

### Type handling

* @JsonSubTypes: class annotation used to indicate sub-types of annotated type; necessary when deserializing polymorphic types using logical type names (and not class names)
* @JsonTypeId: property annotation used to indicate that the property value should be used as the Type Id for object, instead of using class name or external type name.
* @JsonTypeInfo: class/property annotation used to indicate details of what type information is included in serialization, as well as how.
* @JsonTypeName: class annotation used to define logical type name to use for annotated class; type name can be used as Type Id (depending on settings of@JsonTypeInfo)

### Object references, identity

* @JsonManagedReference, @JsonBackReference: pair of annotations used to indicate and handle parent/child relationships expressed with pair of matching properties
* @JsonIdentityInfo: class/property annotation used to indicate that Object Identity is to be used when serializing/deserializing values, such that multiple references to a single Java Object can be properly deserialized. This can used to properly deal with cyclic object graphs and directed-acyclic graphs.

### Meta-annotations

This group includes annotations used on other annotations.

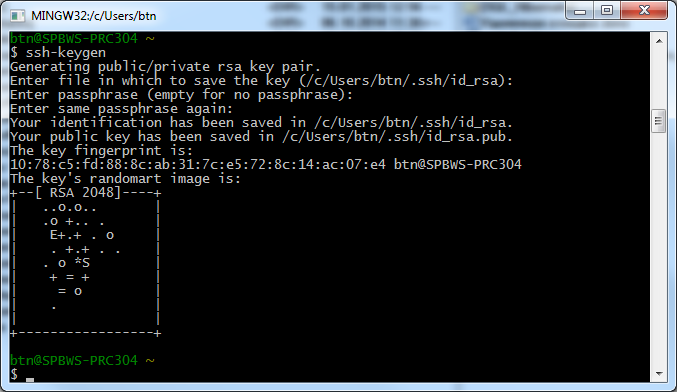
* @JacksonAnnotation: marker annotation added to all Jackson-defined annotations (which includes all other annotations contained in this package)
* @JacksonAnnotationsInside: marked annotation used to indicate that a custom annotation contains Jackson annotations; used to allow "annotation bundles", custom annotations that are annotated with Jackson annotations (why? to allow adding just a single annotation to represent set of multiple Jackson annotations)

### Related

It is also possible to use [JAXB annotations](https://github.com/FasterXML/jackson-module-jaxb-annotations) in addition to or instead of these core annotations.

Ссылка на мануал по jaksony:http://fasterxml.github.io/jackson-annotations/javadoc/2.2.0/com/fasterxml/jackson/annotation/JsonTypeInfo.html

# Public keys

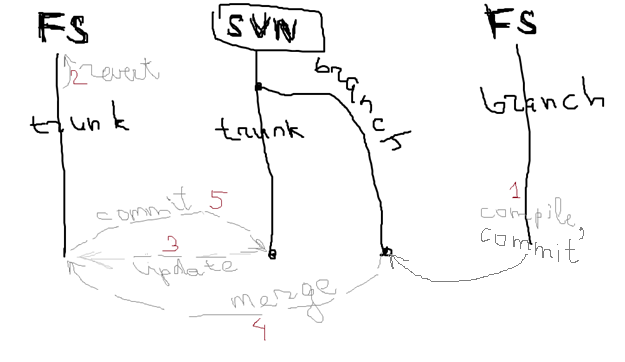


# SVN commit

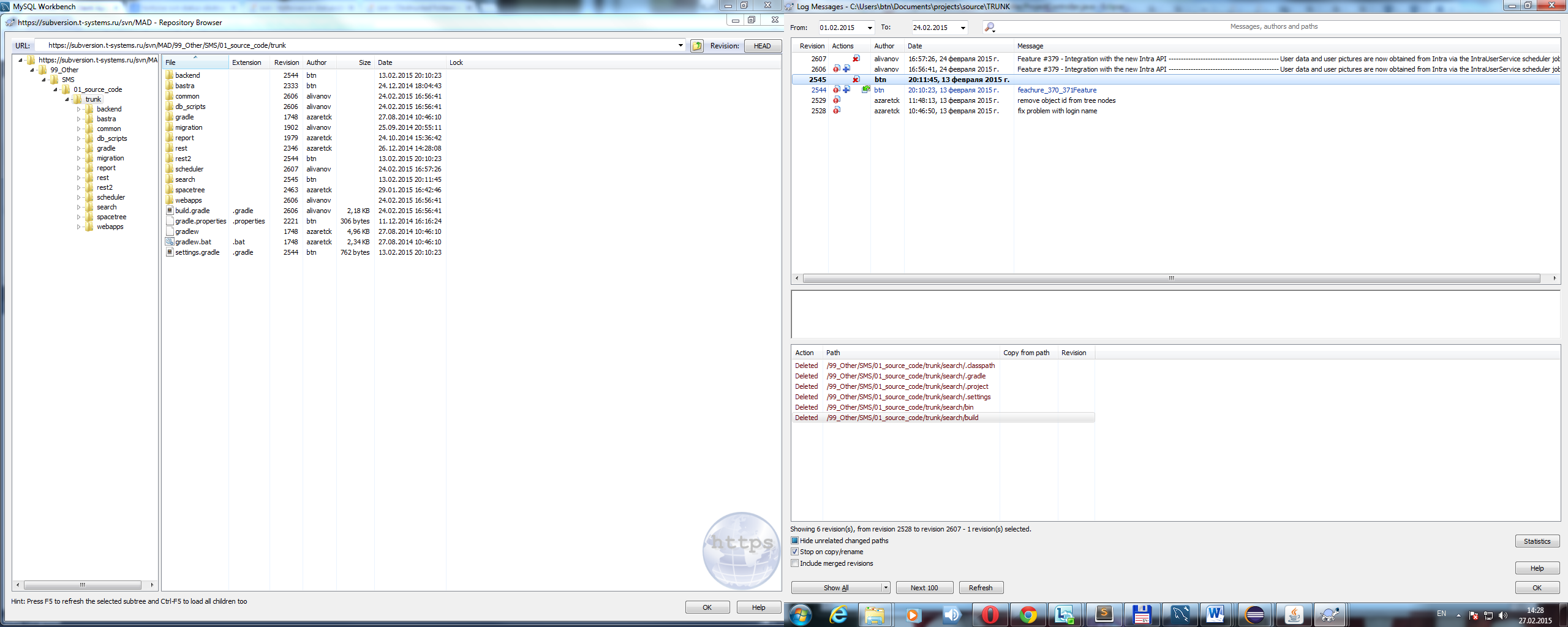
## Ссылка на ресурс из сети:

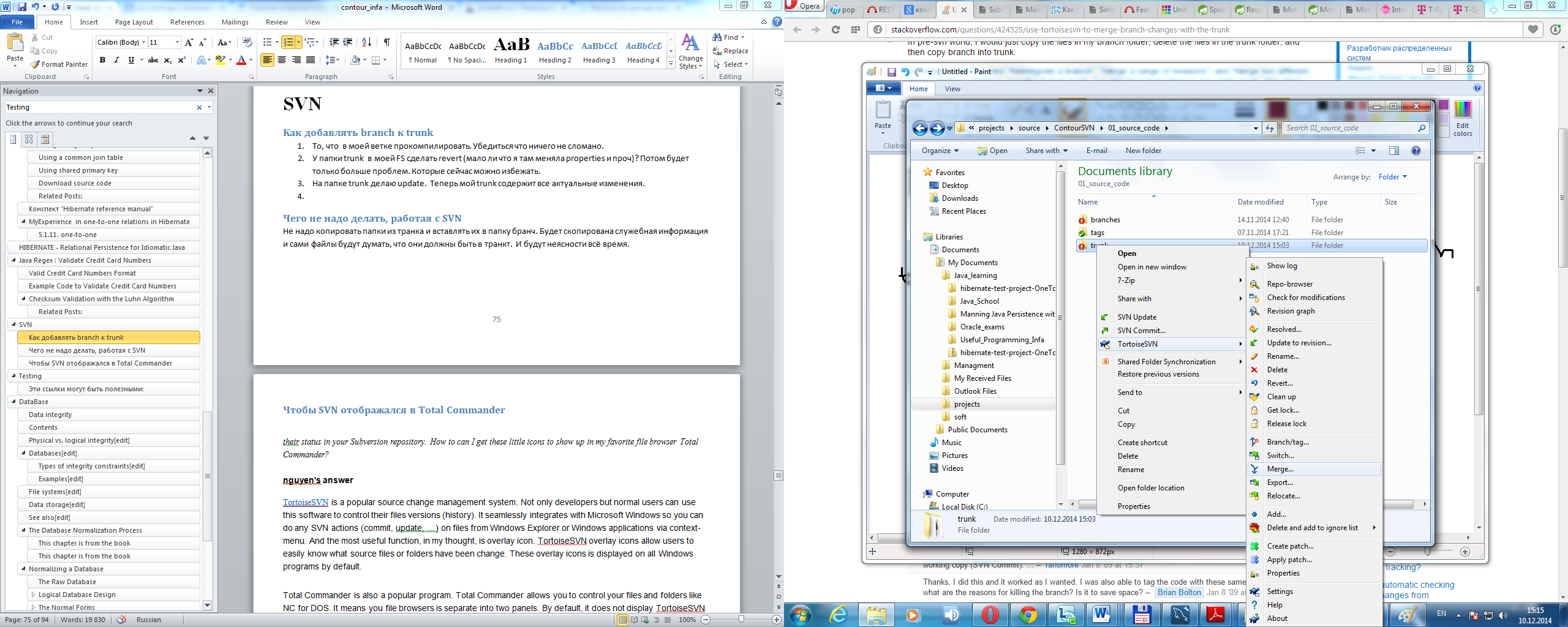
http://tortoisesvn.net/docs/nightly/TortoiseSVN\_ru/tsvn-dug-branchtag.html

## Как добавлять branch к trunk и создавать новый бранч



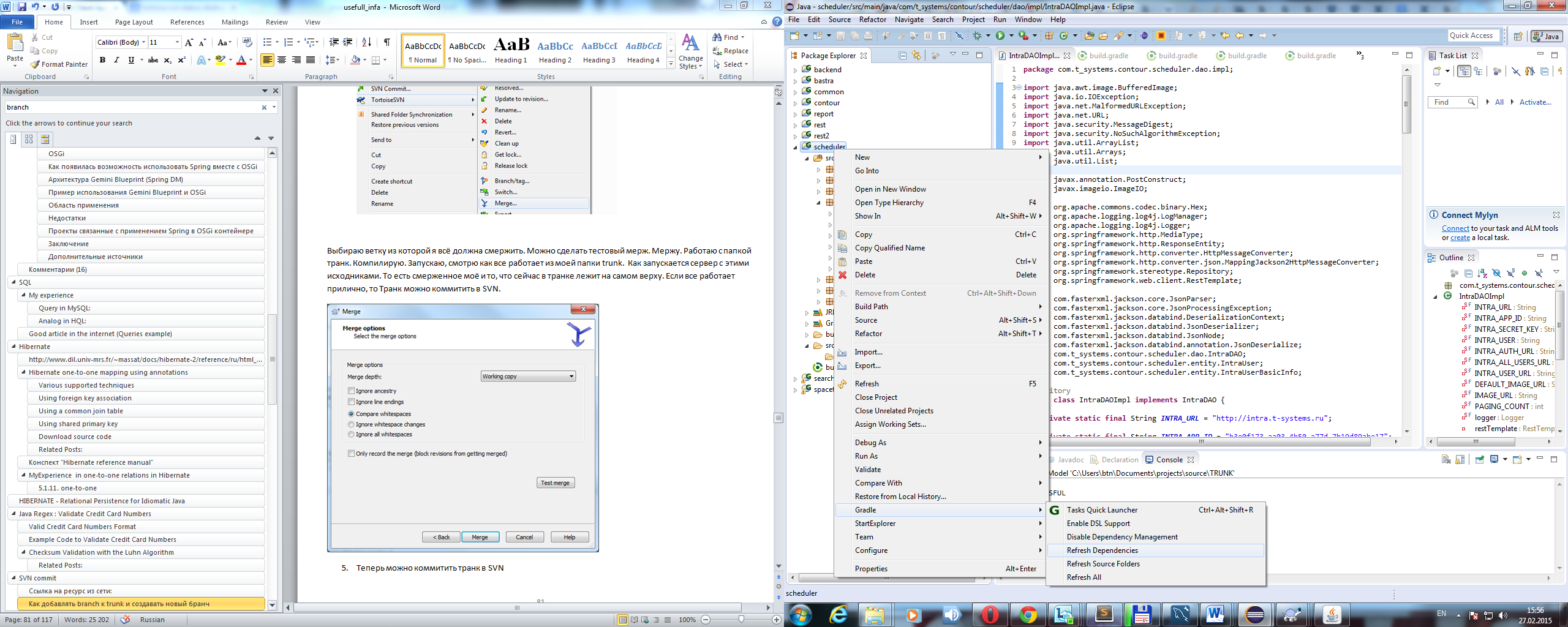
1. То, что в моей ветке прокомпилировать. Убедиться, что ничего не сломано. И закомиттить в бранч.  
   не все папки и файлы нужно комиттить. Нижеприведенные папки и файлы комиттить не надо.



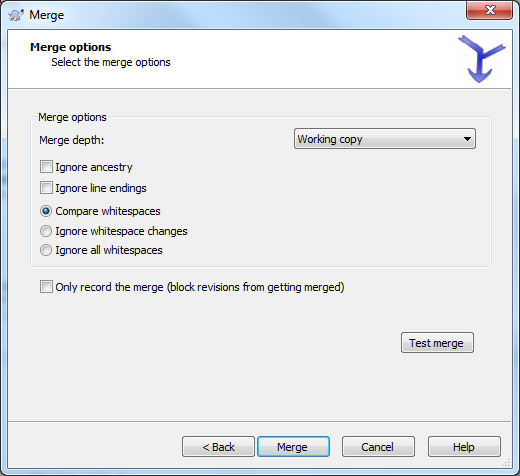
1. У папки trunk в моей FS сделать revert (мало ли что я там меняла properties и проч)? Потом будет только больше проблем. Которые сейчас можно избежать.
2. На папке trunk делаю update. Теперь мой trunk содержит все актуальные изменения.
3. На транке делаю merge  
     
   

Выбираю ветку из которой я всё должна смержить. Можно сделать тестовый мерж. Мержу. Работаю с папкой транк. Компилирую.

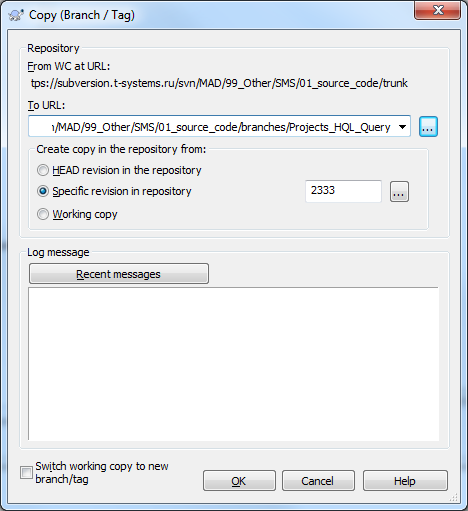
Если красное на каких-то проектах всё равно горит, то надо индивидуально уговорить gradle:



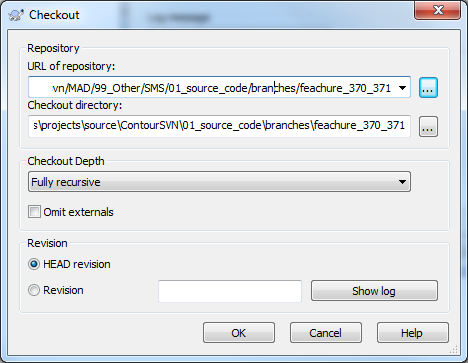
Запускаю, смотрю как все работает из моей папки trunk. Как запускается сервер с этими исходниками. То есть смерженное моё и то, что сейчас в транке лежит на самом верху. Если все работает прилично, то Транк можно коммитить в SVN.



1. Теперь можно коммитить транк в SVN
2. Посмотреть, добавились ли мои записи в транк. Можно опять извлечь транк. И собрать транк, как это будут делать другие программисты.
3. Теперь можно создать новую ветку
   1. Создать новую папку в FS под новую ветку
   2. На папке транк (она ведь проапдейчена? – надо проапдейтить!) правой кнопкой мыши –> tortoise -> Branch/Tag
   3. В поле To URL модифицирую URL . Новый URL отображает путь к новой ветке в хранилище. Чтобы не было ошибок путь <https://subversion.t-systems.ru/svn/MAD/99_Other/SMS/01_source_code/branches/> должен реально существовать на диске, иначе метку Projects\_HQL\_Query создать не получится. На данном этапе рабочей копией является транк. Если поставить галочку переключить рабочую копию на новое ответвление, то папка транк будет указывать на новую созданную ветку. Поэтому мы не ставим галочку «Switch working copy to the new branch». Отметить копию какой ревизии вы хотите сделать. HEAD или какую конкретную.



* 1. Сделать checkout из только что созданной ветки в папку в FS.



1. Старые ветки желательно убивать.
2. Создать папку-воркспейс для эклипса
3. В eclipse сменить воркспейс, импортировать проект, убедиться, что он компилируется.

## Чего не надо делать, работая с SVN

Не надо копировать папки из транка и вставлять их в папку бранч. Будет скопирована служебная информация и сами файлы будут думать, что они должны быть в транк. И будут неясности всё время.

## Чтобы SVN отображался в Total Commander

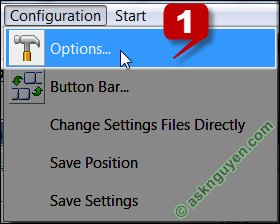
*their status in your Subversion repository. How to can I get these little icons to show up in my favorite file browser Total Commander?*

**nguyen's answer**

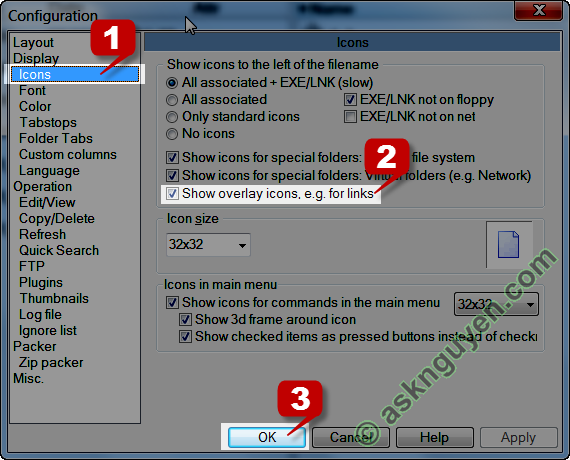
[TortoiseSVN](http://tortoisesvn.tigris.org/) is a popular source change management system. Not only developers but normal users can use this software to control their files versions (history). It seamlessly integrates with Microsoft Windows so you can do any SVN actions (commit, update, …) on files from Windows Explorer or Windows applications via context-menu. And the most useful function, in my thought, is overlay icon. TortoiseSVN overlay icons allow users to easily know what source files or folders have been change. These overlay icons is displayed on all Windows programs by default.

Total Commander is also a popular program. Total Commander allows you to control your files and folders like NC for DOS. It means you file browsers is separate into two panels. By default, it does not display TortoiseSVN overlay icons for files and folders. But you can easily turn this option on by following below steps. This tutorial is done with Total Commander version 7.57.

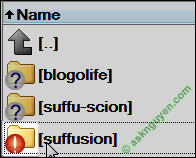
**Step 1.** In your Total Commander windows, click on **Configuration**on menu bar and then choose **Options…**



**Step 2.** In the **Configuration**screen, click on **Icons** then click on checkbox near **Show overlay icons, e.g. for links** to check it.



**Step 3.** Click **OK** to apply it. Then you will see the overlay icons that you want to show.



**P/S:**Follow **rop**‘s comment**,**On Windows 64-bit, Total Commander will enable “Show overlay icons” by default after installation. But the Tortoise icons were not showing. To make it working, you should **disable it & clicking OK, then re-enabling it**. That’s all.

Read more at <http://www.asknguyen.com/display-tortoisesvn-overlay-icon-total-commander.html#WFqBaFu8MwiQEp16.99>

# Git hub

http://tutorials.assembla.com/git-guide-for-windows-users/tour.ru.html

<http://git-scm.com/book/ru/v1>

## Нечто важное для любого пользователя:

## индексируем файлы

1. git add "filename" либо git add   
   перед коммитом надо обязательно обновить репозиторий
2. git pull  
   киммитим файлы git commit -m "commit message"
3. отправляем на сервер (потом пушим в репозиторий и все изменения будут видны на bastra.t-systems.ru:8000)  
   git push origin master

# My github

https://github.com/sunny-prog/docs.git

[https://github.com/sunny-prog/ecare-git.git](http://vk.com/away.php?to=https%3A%2F%2Fgithub.com%2Fsunny-prog%2Fecare-git.git)

git config --global http.proxy http://proxyuser:proxypwd@proxy.server.com:8080

(set proxy.t-systems.ru:3128, user login and password from T-Systems domain )

Command to use :

git config --global http.proxy http://proxyuser:proxypwd@proxy.server.com:8080

git config --global https.proxy https://proxyuser:proxypwd@proxy.server.com:8080

* change proxyuser to your proxy user
* change proxypwd to your proxy password
* change proxy.server.com to the URL of your proxy server
* change 8080 to the proxy port configured on your proxy server

If you decide at any time to reset this proxy and work without (no proxy):

Commands to use:

git config --global --unset http.proxy

git config --global --unset https.proxy

Practise

1

Создаем на сайте <https://github.com/sunny-prog/> новый репозиторий

Пусть new\_repo

2

Заходим в Far Managere в ту папку, в которой надо создать папку с содержимым нового репозитория

И выполняем в этой папке git clone <https://github.com/sunny-prog/new_repo> github\_repo

Последнее – это имя папки, содержащей репозиторий.

3

Добавляем туда файлы, модифицирем их

Git add .

Git status

Git commit

4

Отправляем на удаленное хранилище. В far заходим в папку, где создалась папка юпше

Индексируем файлы

git add "filename" либо git add .

коммитим файлы git commit -m "commit message"

отправляем на сервер. В far заходим в папку корень хранилища – github\_repo и там:

git push origin master

вводим логин и пароль

# Linux

## Useful commands

*netstat -ntpl | grep 9995*

## Скачать и закачать файл

<http://hutpu4.net/linux-open-source/kopirovanie-fajlov-po-ssh.html>

Скачать файл себе из Linux:

pscp.exe -r -pw bastra-dev-01 root@bastra.t-systems.ru:/opt/bastra\_dev C:\Bastra

## Как отключить secure boot на разных компах:

<http://remontcompa.ru/392-kak-otklyuchit-secure-boot.html>

## Команды Linux: расширенный справочник команд Unix / Linux / PuTTY SSH

Большинство UNIX-like систем обладают встроенной справкой, которая подробно описывает все доступные команды. Однако чтобы воспользоваться этой справкой, вы должны знать, по крайней мере, название команды, о которой вы хотите получить информацию. Поскольку большинство пользователей только в общих чертах понимают, что они хотят сделать, то, как правило, встроенная справка мало полезна новичкам.

Этот справочник поможет пользователям, знающим, что они хотят сделать, найти соответствующую команду Linux по краткому описанию.

* [Системная информация](http://putty.org.ru/articles/unix-linux-ref.html#system-info)
* [Остановка системы](http://putty.org.ru/articles/unix-linux-ref.html#shutdown-restart-logout)
* [Файлы и директории](http://putty.org.ru/articles/unix-linux-ref.html#files-directory)
* [Поиск файлов](http://putty.org.ru/articles/unix-linux-ref.html#file-search)
* [Монтирование файловых систем](http://putty.org.ru/articles/unix-linux-ref.html#mounting-filesystem)
* [Дисковое пространство](http://putty.org.ru/articles/unix-linux-ref.html#disk-space)
* [Пользователи и группы](http://putty.org.ru/articles/unix-linux-ref.html#users-groups)
* [Выставление/изменение полномочий на файлы](http://putty.org.ru/articles/unix-linux-ref.html#permits-files)
* [Специальные атрибуты файлов](http://putty.org.ru/articles/unix-linux-ref.html#special-attributes-files)
* [Архивирование и сжатие файлов](http://putty.org.ru/articles/unix-linux-ref.html#archives-compressed-files)
* [RPM пакеты (Fedora, Red Hat и тому подобное)](http://putty.org.ru/articles/unix-linux-ref.html#rpm-packages)
* [YUM — средство обновления пакетов(Fedora, RedHat и тому подобное)](http://putty.org.ru/articles/unix-linux-ref.html#yum-packages)
* [DEB пакеты (Debian, Ubuntu и тому подобное)](http://putty.org.ru/articles/unix-linux-ref.html#deb-packages)
* [APT — средство управление пакетами (Debian, Ubuntu и тому подобное)](http://putty.org.ru/articles/unix-linux-ref.html#apt-packages)
* [Pacman — средство управление пакетами (Arch, Frugalware и alike)](http://putty.org.ru/articles/unix-linux-ref.html#pacman-packages)
* [Просмотр содержимого файлов](http://putty.org.ru/articles/unix-linux-ref.html#view-file-content)
* [Манипуляции с текстом](http://putty.org.ru/articles/unix-linux-ref.html#text-manipulation)
* [Преобразование наборов символов и файловых форматов](http://putty.org.ru/articles/unix-linux-ref.html#charset-format-conversion)
* [Анализ файловых систем](http://putty.org.ru/articles/unix-linux-ref.html#filesystem-analysis)
* [Форматирование файловых систем](http://putty.org.ru/articles/unix-linux-ref.html#format-filesystem)
* [swap-пространство](http://putty.org.ru/articles/unix-linux-ref.html#filesystem-swap)
* [Создание резервных копий (backup)](http://putty.org.ru/articles/unix-linux-ref.html#backup)
* [CDROM](http://putty.org.ru/articles/unix-linux-ref.html#cdrom)
* [Сеть (LAN и WiFi)](http://putty.org.ru/articles/unix-linux-ref.html#networking-lan-wifi)
* [Microsoft Windows networks(SAMBA)](http://putty.org.ru/articles/unix-linux-ref.html#windows-samba)
* [IPTABLES (firewall)](http://putty.org.ru/articles/unix-linux-ref.html#iptables-firewall)
* [Мониторинг и отладка](http://putty.org.ru/articles/unix-linux-ref.html#monitoring-debugging)
* [Другие полезные команды](http://putty.org.ru/articles/unix-linux-ref.html#other)

### Системная информация:

**arch** или **uname -m** — отобразить архитектуру компьютера

**uname -r** — отобразить используемую версию ядра

**dmidecode -q** — показать аппаратные системные компоненты — (SMBIOS / DMI)

**hdparm -i /dev/hda** — вывести характеристики жёсткого диска

**hdparm -tT /dev/sda** — протестировать производительность чтения данных с жёсткого диска

**cat /proc/cpuinfo** — отобразить информацию о процессоре

**cat /proc/interrupts** — показать прерывания

**cat /proc/meminfo** — проверить использование памяти

**cat /proc/swaps** — показать файл(ы) подкачки

**cat /proc/version** — вывести версию ядра

**cat /proc/net/dev** — показать сетевые интерфейсы и статистику по ним

**cat /proc/mounts** — отобразить смонтированные файловые системы

**lspci -tv** — показать в виде дерева PCI устройства

**lsusb -tv** — показать в виде дерева USB устройства

**date** — вывести системную дату

**cal 2007** — вывести таблицу-календарь 2007-го года

**date 041217002007.00\*** — установить системные дату и время ММДДЧЧммГГГГ.СС (МесяцДеньЧасМинутыГод.Секунды)

**clock -w** — сохранить системное время в BIOS

### Остановка системы:

**shutdown -h now** или **init 0** или **telinit 0** — остановить систему

**shutdown -h hours:minutes &** — запланировать остановку системы на указанное время

**shutdown -c** — отменить запланированную по расписанию остановку системы

**shutdown -r now** или **reboot** — перегрузить систему

**logout** — выйти из системы

### Файлы и директории:

**cd /home** — перейти в директорию '/home'

**cd ..** — перейти в директорию уровнем выше

**cd ../..** — перейти в директорию двумя уровнями выше

**cd** — перейти в домашнюю директорию

**cd ~user** — перейти в домашнюю директорию пользователя user

**cd -** — перейти в директорию, в которой находились до перехода в текущую директорию

**pwd** — показать текущую директорию

**ls** — отобразить содержимое текущей директории

**ls -F** — отобразить содержимое текущей директории с добавлением к именам символов, характеризующих тип

**ls -l** — показать детализированное представление файлов и директорий в текущей директории

**ls -a** — показать скрытые файлы и директории в текущей директории

**ls \*[0-9]\*** — показать файлы и директории содержащие в имени цифры

**tree** или **lstree** — показать дерево файлов и директорий, начиная от корня (/)

**mkdir dir1** — создать директорию с именем 'dir1'

**mkdir dir1 dir2** — создать две директории одновременно

**mkdir -p /tmp/dir1/dir2** — создать дерево директорий

**rm -f file1** — удалить файл с именем 'file1'

**rmdir dir1**— удалить директорию с именем 'dir1'

**rm -rf dir1** — удалить директорию с именем 'dir1' и рекурсивно всё её содержимое

**rm -rf dir1 dir2** — удалить две директории и рекурсивно их содержимое

**mv dir1 new\_dir** — переименовать или переместить файл или директорию

**cp file1 file2** — скопировать файл file1 в файл file2

**cp dir/\* .** — копировать все файлы директории dir в текущую директорию

**cp -a /tmp/dir1 .** — копировать директорию dir1 со всем содержимым в текущую директорию

**cp -a dir1 dir2** — копировать директорию dir1 в директорию dir2

**ln -s file1 lnk1\*** — создать символическую ссылку на файл или директорию

**ln file1 lnk1** — создать «жёсткую» (физическую) ссылку на файл или директорию

**touch -t 0712250000 fileditest** — модифицировать дату и время создания файла, при его отсутствии, создать файл с указанными датой и временем (YYMMDDhhmm)

### Поиск файлов:

**find / -name file1** — найти файлы и директории с именем file1. Поиск начать с корня (/)

**find / -user user1** — найти файл и директорию принадлежащие пользователю user1. Поиск начать с корня (/)

**find /home/user1 -name "\*.bin"** — найти все файлы и директории, имена которых оканчиваются на '. bin'. Поиск начать с '/ home/user1'\*

**find /usr/bin -type f -atime +100** — найти все файлы в '/usr/bin', время последнего обращения к которым более 100 дней

**find /usr/bin -type f -mtime -10** — найти все файлы в '/usr/bin', созданные или изменённые в течении последних 10 дней

**find / -name \*.rpm -exec chmod 755 '{}' \;** — найти все фалы и директории, имена которых оканчиваются на '.rpm', и изменить права доступа к ним

**find / -xdev -name "\*.rpm"** — найти все фалы и директории, имена которых оканчиваются на '.rpm', игнорируя съёмные носители, такие как cdrom, floppy и т.п.

**locate "\*.ps"** — найти все файлы, содержащие в имени '.ps'. Предварительно рекомендуется выполнить команду 'updatedb'

**whereis halt** — показывает размещение бинарных файлов, исходных кодов и руководств, относящихся к файлу 'halt'

**which halt** — отображает полный путь к файлу 'halt'

### Монтирование файловых систем:

**mount /dev/hda2 /mnt/hda2** — монтирует раздел 'hda2' в точку монтирования '/mnt/hda2'. Убедитесь в наличии директории-точки монтирования '/mnt/hda2'

**umount /dev/hda2** — размонтирует раздел 'hda2'. Перед выполнением, покиньте '/mnt/hda2'

**fuser -km /mnt/hda2** — принудительное размонтирование раздела. Применяется в случае, когда раздел занят каким-либо пользователем

**umount -n /mnt/hda2** — выполнить размонтирование без занесения информации в /etc/mtab. Полезно когда файл имеет атрибуты «только чтение» или недостаточно места на диске

**mount /dev/fd0 /mnt/floppy** — монтировать флоппи-диск

**mount /dev/cdrom /mnt/cdrom** — монтировать CD или DVD

**mount /dev/hdc /mnt/cdrecorder** — монтировать CD-R/CD-RW или DVD-R/DVD-RW(+-)

**mount -o loop file.iso /mnt/cdrom** — смонтировать ISO-образ

**mount -t vfat /dev/hda5 /mnt/hda5** — монтировать файловую систему Windows FAT32

**mount -t smbfs -o username=user,password=pass //winclient/share /mnt/share** — монтировать сетевую файловую систему Windows (SMB/CIFS)

**mount -o bind /home/user/prg /var/ftp/user** — «монтирует» директорию в директорию (binding). Доступна с версии ядра 2.4.0. Полезна, например, для предоставления содержимого пользовательской директории через ftp при работе ftp-сервера в «песочнице» (chroot), когда симлинки сделать невозможно. Выполнение данной команды сделает копию содержимого /home/user/prg в /var/ftp/user

### Дисковое пространство:

**df -h** — отображает информацию о смонтированных разделах с отображением общего, доступного и используемого пространства (Прим. переводчика. ключ -h работает не во всех \*nix системах)

**ls -lSr |more** — выдаёт список файлов и директорий рекурсивно с сортировкой по возрастанию размера и позволяет осуществлять постраничный просмотр

**du -sh dir1** — подсчитывает и выводит размер, занимаемый директорией 'dir1' (Прим. переводчика. ключ -h работает не во всех \*nix системах)

**du -sk \* | sort -rn** — отображает размер и имена файлов и директорий, с соритровкой по размеру

**rpm -q -a --qf '%10{SIZE}t%{NAME}n' | sort -k1,1n** — показывает размер используемого дискового пространства, занимаемое файлами rpm-пакета, с сортировкой по размеру (fedora, redhat и т.п.)

**dpkg-query -W -f='${Installed-Size;10}t${Package}n' | sort -k1,1n** — показывает размер используемого дискового пространства, занимаемое файлами deb-пакета, с сортировкой по размеру (ubuntu, debian т.п.)

### Пользователи и группы:

**groupadd group\_name** — создать новую группу с именем group\_name

**groupdel group\_name** — удалить группу group\_name

**groupmod -n new\_group\_name old\_group\_name** — переименовать группу old\_group\_name в new\_group\_name

**useradd -c "Nome Cognome" -g admin -d /home/user1 -s /bin/bash user1** — создать пользователя user1, назначить ему в качестве домашнего каталога /home/user1, в качестве shell'а /bin/bash, включить его в группу admin и добавить комментарий Nome Cognome

**useradd user1** — создать пользователя user1

**userdel -r user1** — удалить пользователя user1 и его домашний каталог

**usermod -c "User FTP" -g system -d /ftp/user1 -s /bin/nologin user1** — изменить атрибуты пользователя

**passwd** — сменить пароль

**passwd user1** — сменить пароль пользователя user1 (только root)

**chage -E 2005-12-31 user1** — установить дату окончания действия учётной записи пользователя user1

**pwck** — проверить корректность системных файлов учётных записей. Проверяются файлы /etc/passwd и /etc/shadow

**grpck** — проверяет корректность системных файлов учётных записей. Проверяется файл/etc/group

**newgrp [-] group\_name** — изменяет первичную группу текущего пользователя. Если указать «-», ситуация будет идентичной той, в которой пользователь вышил из системы и снова вошёл. Если не указывать группу, первичная группа будет назначена из /etc/passwd

### Выставление/изменение полномочий на файлы:

**ls -lh** — просмотр полномочий на файлы и директории в текущей директории

**ls /tmp | pr -T5 -W$COLUMNS** — вывести содержимое директории /tmp и разделить вывод на пять колонок

**chmod ugo+rwx directory1** — добавить полномочия на директорию directory1 ugo(User Group Other)+rwx(Read Write eXecute) — всем полные права. Аналогичное можно сделать таким образом chmod 777 directory1

**chmod go-rwx directory1** — отобрать у группы и всех остальных все полномочия на директорию directory1.

**chown user1 file1** — назначить владельцем файла file1 пользователя user1

**chown -R user1 directory1** — назначить рекурсивно владельцем директории directory1 пользователя user1

**chgrp group1 file1** — сменить группу-владельца файла file1 на group1

**chown user1:group1 file1** — сменить владельца и группу владельца файла file1

**find / -perm -u+s** — найти, начиная от корня, все файлы с выставленным SUID

**chmod u+s /bin/binary\_file** — назначить SUID-бит файлу /bin/binary\_file. Это даёт возможность любому пользователю запускать на выполнение файл с полномочиями владельца файла.

**chmod u-s /bin/binary\_file** — снять SUID-бит с файла /bin/binary\_file.

**chmod g+s /home/public** — назначить SGID-бит директории /home/public.

**chmod g-s /home/public** — снять SGID-бит с директории /home/public.

**chmod o+t /home/public** — назначить STIKY-бит директории /home/public. Позволяет удалять файлы только владельцам

**chmod o-t /home/public** — снять STIKY-бит с директории /home/public

### Специальные атрибуты файлов:

**chattr +a file1** — позволить открывать файл на запись только в режиме добавления

**chattr +c file1** — позволяет ядру автоматически сжимать/разжимать содержимое файла.

**chattr +d file1**— указывает утилите dump игнорировать данный файл во время выполнения backup'а

**chattr +i file1** — делает файл недоступным для любых изменений: редактирование, удаление, перемещение, создание линков на него.

**chattr +s file1** — позволяет сделать удаление файла безопасным, т.е. выставленный атрибут s говорит о том, что при удалении файла, место, занимаемое файлом на диске заполняется нулями, что предотвращает возможность восстановления данных.

**chattr +S file1** — указывает, что, при сохранении изменений, будет произведена синхронизация, как при выполнении команды sync

**chattr +u file1** — данный атрибут указывает, что при удалении файла содержимое его будет сохранено и при необходимости пользователь сможет его восстановить

**lsattr** — показать атрибуты файлов

### Архивирование и сжатие файлов:

**bunzip2 file1.bz2** — разжимает файл 'file1.gz'

**gunzip file1.gz** —

**gzip file1** или **bzip2 file1** — сжимает файл 'file1'

**gzip -9 file1** — сжать файл file1 с максимальным сжатием

**rar a file1.rar test\_file** — создать rar-архив 'file1.rar' и включить в него файл test\_file

**rar a file1.rar file1 file2 dir1** — создать rar-архив 'file1.rar' и включить в него file1, file2 и dir1

**rar x file1.rar** — распаковать rar-архив

**unrar x file1.rar** —

**tar -cvf archive.tar file1** — создать tar-архив archive.tar, содержащий файл file1

**tar -cvf archive.tar file1 file2 dir1** — создать tar-архив archive.tar, содержащий файл file1, file2 и dir1

**tar -tf archive.tar** — показать содержимое архива

**tar -xvf archive.tar** — распаковать архив

**tar -xvf archive.tar -C /tmp** — распаковать архив в /tmp

**tar -cvfj archive.tar.bz2 dir1** — создать архив и сжать его с помощью bzip2(Прим. переводчика. ключ -j работает не во всех \*nix системах)

**tar -xvfj archive.tar.bz2** — разжать архив и распаковать его(Прим. переводчика. ключ -j работает не во всех \*nix системах)

**tar -cvfz archive.tar.gz dir1** — создать архив и сжать его с помощью gzip

**tar -xvfz archive.tar.gz** — разжать архив и распаковать его

**zip file1.zip file1** — создать сжатый zip-архив

**zip -r file1.zip file1 file2 dir1** — создать сжатый zip-архив и со включением в него нескольких файлов и/или директорий

**unzip file1.zip** — разжать и распаковать zip-архив

### RPM пакеты (Fedora, Red Hat и тому подобное):

**rpm -ivh package.rpm** — установить пакет с выводом сообщений и прогресс-бара

**rpm -ivh --nodeps package.rpm** — установить пакет с выводом сообщений и прогресс-бара без контроля зависимостей

**rpm -U package.rpm** — обновить пакет без изменений конфигурационных файлов, в случае отсутствия пакета, он будет установлен

**rpm -F package.rpm** — обновить пакет только если он установлен

**rpm -e package\_name.rpm** — удалить пакет

**rpm -qa** — отобразить список всех пакетов, установленных в системе

**rpm -qa | grep httpd** — среди всех пакетов, установленных в системе, найти пакет содержащий в своём имени «httpd»

**rpm -qi package\_name** — вывести информацию о конкретном пакете

**rpm -qg "System Environment/Daemons"** — отобразить пакеты входящие в группу пакетов

**rpm -ql package\_name** — вывести список файлов, входящих в пакет

**rpm -qc package\_name** — вывести список конфигурационных файлов, входящих в пакет

**rpm -q package\_name --whatrequires** — вывести список пакетов, необходимых для установки конкретного пакета по зависимостям

**rpm -q package\_name --whatprovides** — show capability provided by a rpm package

**rpm -q package\_name --scripts** — отобразит скрипты, запускаемые при установке/удалении пакета

**rpm -q package\_name --changelog** — вывести историю ревизий пакета

**rpm -qf /etc/httpd/conf/httpd.conf** — проверить какому пакету принадлежит указанный файл. Указывать следует полный путь и имя файла.

**rpm -qp package.rpm -l** — отображает список файлов, входящих в пакет, но ещё не установленных в систему

**rpm --import /media/cdrom/RPM-GPG-KEY** — импортировать публичный ключ цифровой подписи

**rpm --checksig package.rpm** — проверит подпись пакета

**rpm -qa gpg-pubkey** — проверить целостность установленного содержимого пакета

**rpm -V package\_name** — проверить размер, полномочия, тип, владельца, группу, MD5-сумму и дату последнего изменеия пакета

**rpm -Va** — проверить содержимое всех пакеты установленные в систему. Выполняйте с осторожностью!

**rpm -Vp package.rpm** — проверить пакет, который ещё не установлен в систему

**rpm2cpio package.rpm | cpio --extract --make-directories \*bin\*** — извлечь из пакета файлы содержащие в своём имени bin

**rpm -ivh /usr/src/redhat/RPMS/`arch`/package.rpm** — установить пакет, собранный из исходных кодов

**rpmbuild --rebuild package\_name.src.rpm** — собрать пакет из исходных кодов

### YUM — средство обновления пакетов(Fedora, RedHat и тому подобное):

**yum install package\_name** — закачать и установить пакет

**yum update** — обновить все пакеты, установленные в систему

**yum update package\_name** — обновить пакет

**yum remove package\_name** — удалить пакет

**yum list** — вывести список всех пакетов, установленных в систему

**yum search package\_name** — найти пакет в репозиториях

**yum clean packages** — очисть rpm-кэш, удалив закачанные пакеты

**yum clean headers** — удалить все заголовки файлов, которые система использует для разрешения зависимостей

**yum clean all** — очисть rpm-кэш, удалив закачанные пакеты и заголовки

### DEB пакеты (Debian, Ubuntu и тому подобное):

**dpkg -i package.deb** — установить / обновить пакет

**dpkg -r package\_name** — удалить пакет из системы

**dpkg -l** — показать все пакеты, установленные в систему

**dpkg -l | grep httpd** — среди всех пакетов, установленных в системе, найти пакет содержащий в своём имени «httpd»

**dpkg -s package\_name** — отобразить информацию о конкретном пакете

**dpkg -L package\_name** — вывести список файлов, входящих в пакет, установленный в систему

**dpkg --contents package.deb** — отобразить список файлов, входящих в пакет, который ещё не установлен в систему

**dpkg -S /bin/ping** — найти пакет, в который входит указанный файл.

### APT — средство управление пакетами (Debian, Ubuntu и тому подобное):

**apt-get update** — получить обновлённые списки пакетов

**apt-get upgrade** — обновить пакеты, установленные в систему

**apt-get install package\_name** — установить / обновить пакет

**apt-cdrom install package\_name** — установить / обновить пакет с cdrom'а

**apt-get remove package\_name** — удалить пакет, установленный в систему с сохранением файлов конфигурации

**apt-get purge package\_name** — удалить пакет, установленный в систему с удалением файлов конфигурации

**apt-get check** — проверить целостность зависимостей

**apt-get clean** — удалить загруженные архивные файлы пакетов

**apt-get autoclean** — удалить старые загруженные архивные файлы пакетов

### Pacman — средство управление пакетами (Arch, Frugalware и alike)

**pacman -S name** — install package «name» with dependencies

**pacman -R name** — delete package «name» and all files of it

### Просмотр содержимого файлов:

**cat file1** — вывести содержимое файла file1 на стандартное устройство вывода

**tac file1** — вывести содержимое файла file1 на стандартное устройство вывода в обратном порядке (последняя строка становиться первой и т.д.)

**more file1** — постраничный вывод содержимого файла file1 на стандартное устройство вывода

**less file1** — постраничный вывод содержимого файла file1 на стандартное устройство вывода, но с возможностью пролистывания в обе стороны (вверх-вниз), поиска по содержимому и т.п.

**head -2 file1** — вывести первые две строки файла file1 на стандартное устройство вывода. По умолчанию выводится десять строк

**tail -2 file1** — вывести последние две строки файла file1 на стандартное устройство вывода. По умолчанию выводится десять строк

**tail -f /var/log/messages** — выводить содержимое файла /var/log/messages на стандартное устройство вывода по мере появления в нём текста.

### Манипуляции с текстом:

**grep -HR OLDTEXT ./ | awk '{print $1}' | sed 's/:.\*$//' | grep -v '~' | sort | uniq | xargs perl -i -pe "s/OLD\_TEXT/NEW\_TEXT/g;"** — Поиск и замена текста OLDTEXT на NEW\_TEXT во многих файлах одновременно с рекурсивным обходом директорий.

**cat file\_originale | [operation: sed, grep, awk, grep и т.п.] > result.txt** — общий синтаксис выполнения действий по обработке содержимого файла и вывода результата в новый

**cat file\_originale | [operazione: sed, grep, awk, grepи т.п.] >> result.txt** — общий синтаксис выполнения действий по обработке содержимого файла и вывода результата в существующий файл. Если файл не существует, он будет создан

**grep Aug /var/log/messages** — из файла '/var/log/messages' отобрать и вывести на стандартное устройство вывода строки, содержащие «Aug»

**grep ^Aug /var/log/messages** — из файла '/var/log/messages' отобрать и вывести на стандартное устройство вывода строки, начинающиеся на «Aug»

**grep [0-9] /var/log/messages** — из файла '/var/log/messages' отобрать и вывести на стандартное устройство вывода строки, содержащие цифры

**grep Aug -R /var/log/\*** — отобрать и вывести на стандартное устройство вывода строки, содержащие «Augr», во всех файлах, находящихся в директории /var/log и ниже

**sed 's/stringa1/stringa2/g' example.txt** — в файле example.txt заменить «string1» на «string2», результат вывести на стандартное устройство вывода.

**sed '/^$/d' example.txt** — удалить пустые строки из файла example.txt

**sed '/ \*#/d; /^$/d' example.txt** — удалить пустые строки и комментарии из файла example.txt

**echo 'esempio' | tr '[:lower:]' '[:upper:]'** — преобразовать символы из нижнего регистра в верхний

**sed -e '1d' result.txt** — удалить первую строку из файла example.txt

**sed -n '/string1/p'** — отобразить только строки, содержащие «string1»

**sed -e 's/ \*$//' example.txt** — удалить пустые символы в конце каждой строки

**sed -e 's/string1//g' example.txt** — удалить строку «string1» из текста не изменяя всего остального

**sed -n '1,8p;5q' example.txt** — взять из файла с первой по восьмую строки и из них вывести первые пять

**sed -n '5p;5q' example.txt** — вывести пятую строку

**sed -e 's/0\*/0/g' example.txt** — заменить последовательность из любого количества нулей одним нулём

**cat -n file1** — пронумеровать строки при выводе содержимого файла

**cat example.txt | awk 'NR%2==1'** — при выводе содержимого файла, не выводить чётные строки файла

**echo a b c | awk '{print $1}'** — вывести первую колонку. Разделение, по умолчанию, по пробелу/пробелам или символу/символам табуляции

**echo a b c | awk '{print $1,$3}'** — вывести первую и третью колонки. Разделение, по умолчанию, по проблелу/пробелам или символу/символам табуляции

**paste file1 file2** — объединить содержимое file1 и file2 в виде таблицы: строка 1 из file1 = строка 1 колонка 1-n, строка 1 из file2 = строка 1 колонка n+1-m

**paste -d '+' file1 file2** — объединить содержимое file1 и file2 в виде таблицы с разделителем «+»

**sort file1 file2** — отсортировать содержимое двух файлов

**sort file1 file2 | uniq** — отсортировать содержимое двух файлов, не отображая повторов

**sort file1 file2 | uniq -u** — отсортировать содержимое двух файлов, отображая только уникальные строки (строки, встречающиеся в обоих файлах, не выводятся на стандартное устройство вывода)

**sort file1 file2 | uniq -d** — отсортировать содержимое двух файлов, отображая только повторяющиеся строки

**comm -1 file1 file2** — сравнить содержимое двух файлов, не отображая строки принадлежащие файлу 'file1'

**comm -2 file1 file2** — сравнить содержимое двух файлов, не отображая строки принадлежащие файлу 'file2'

**comm -3 file1 file2** — сравнить содержимое двух файлов, удаляя строки встречающиеся в обоих файлах

### Преобразование наборов символов и файловых форматов:

**dos2unix filedos.txt fileunix.txt** — конвертировать файл текстового формата из MSDOS в UNIX (разница в символах возврата каретки)

**unix2dos fileunix.txt filedos.txt** — конвертировать файл текстового формата из UNIX в MSDOS (разница в символах возврата каретки)

**recode ..HTML < page.txt > page.html** — конвертировать содержимое тестового файла page.txt в html-файл page.html

**recode -l | more** — вывести список доступных форматов

### Анализ файловых систем:

**badblocks -v /dev/hda1** — проверить раздел hda1 на наличие bad-блоков

**fsck /dev/hda1** — проверить/восстановить целостность linux-файловой системы раздела hda1

**fsck.ext2 /dev/hda1** или **e2fsck /dev/hda1** — проверить/восстановить целостность файловой системы ext2 раздела hda1

**e2fsck -j /dev/hda1** — проверить/восстановить целостность файловой системы ext3 раздела hda1 с указанием, что журнал расположен там же

**fsck.ext3 /dev/hda1** — проверить/восстановить целостность файловой системы ext3 раздела hda1

**fsck.vfat /dev/hda1** или **fsck.msdos /dev/hda1** или **dosfsck /dev/hda1** — проверить/восстановить целостность файловой системы fat раздела hda11

### Форматирование файловых систем:

**mkfs /dev/hda1** — создать linux-файловую систему на разделе hda1

**mke2fs /dev/hda1** — создать файловую систему ext2 на разделе hda1

**mke2fs -j /dev/hda1** — создать журналирующую файловую систему ext3 на разделе hda1

**mkfs -t vfat 32 -F /dev/hda1** — создать файловую систему FAT32 на разделе hda1

**fdformat -n /dev/fd0** — форматирование флоппи-диска без проверки

**mkswap /dev/hda3** — создание swap-пространства на разделе hda3

### swap-пространство:

**mkswap /dev/hda3** — создание swap-пространства на разделе hda3

**swapon /dev/hda3** — активировать swap-пространство, расположенное на разделе hda3

**swapon /dev/hda2 /dev/hdb3** — активировать swap-пространства, расположенные на разделах hda2 и hdb3

### Создание резервных копий (backup):

**dump -0aj -f /tmp/home0.bak /home** — создать полную резервную копию директории /home в файл /tmp/home0.bak

**dump -1aj -f /tmp/home0.bak /home**— создать инкрементальную резервную копию директории /home в файл /tmp/home0.bak

**restore -if /tmp/home0.bak** — восстановить из резервной копии /tmp/home0.bak

**rsync -rogpav --delete /home /tmp** — синхронизировать /tmp с /home

**rsync -rogpav -e ssh --delete /home ip\_address:/tmp**— синхронизировать через SSH-туннель

**rsync -az -e ssh --delete ip\_addr:/home/public /home/local** — синхронизировать локальную директорию с удалённой директорией через ssh-туннель со сжатием

**rsync -az -e ssh --delete /home/local ip\_addr:/home/public** — синхронизировать удалённую директорию с локальной директорией через ssh-туннель со сжатием

**dd bs=1M if=/dev/hda | gzip | ssh user@ip\_addr 'dd of=hda.gz'** — сделать «слепок» локального диска в файл на удалённом компьютере через ssh-туннель

**tar -Puf backup.tar /home/user** — создать инкрементальную резервную копию директории '/home/user' в файл backup.tar с сохранением полномочий

**( cd /tmp/local/ && tar c . ) | ssh -C user@ip\_addr 'cd /home/share/ && tar x -p'** — копирование содержимого /tmp/local на удалённый компьютер через ssh-туннель в /home/share/

**( tar c /home ) | ssh -C user@ip\_addr 'cd /home/backup-home && tar x -p'**— копирование содержимого /home на удалённый компьютер через ssh-туннель в /home/backup-home

**tar cf - . | (cd /tmp/backup ; tar xf - )** — копирование одной директории в другую с сохранением полномочий и линков

**find /home/user1 -name '\*.txt' | xargs cp -av --target-directory=/home/backup/ --parents** — поиск в /home/user1 всех файлов, имена которых оканчиваются на '.txt', и копирование их в другую директорию

**find /var/log -name '\*.log' | tar cv --files-from=- | bzip2 > log.tar.bz2** — поиск в /var/log всех файлов, имена которых оканчиваются на '.log', и создание bzip-архива из них

**dd if=/dev/hda of=/dev/fd0 bs=512 count=1** — создать копию MBR (Master Boot Record) с /dev/hda на флоппи-диск

**dd if=/dev/fd0 of=/dev/hda bs=512 count=1** — восстановить MBR с флоппи-диска на /dev/hda

### CDROM:

**cdrecord -v gracetime=2 dev=/dev/cdrom -eject blank=fast -force** — clean a rewritable cdrom

**mkisofs /dev/cdrom > cd.iso** — create an iso image of cdrom on disk

**mkisofs /dev/cdrom | gzip > cd\_iso.gz** — create a compressed iso image of cdrom on disk

**mkisofs -J -allow-leading-dots -R -V "Label CD" -iso-level 4 -o ./cd.iso data\_cd** — create an iso image of a directory

**cdrecord -v dev=/dev/cdrom cd.iso** — burn an ISO image

**gzip -dc cd\_iso.gz | cdrecord dev=/dev/cdrom** — burn a compressed ISO image

**mount -o loop cd.iso /mnt/iso** — mount an ISO image

**cd-paranoia -B** — rip audio tracks from a CD to wav files

**cd-paranoia -- "-3"** — rip first three audio tracks from a CD to wav files

**cdrecord --scanbus** — scan bus to identify the channel scsi

### Сеть (LAN и WiFi):

**ifconfig eth0** — показать конфигурацию сетевого интерфейса eth0

**ifup eth0** — активировать (поднять) интерфейс eth0

**ifdown eth0** — деактивировать (опустить) интерфейс eth0

**ifconfig eth0 192.168.1.1 netmask 255.255.255.0** — выставить интерфейсу eth0 IP-адрес и маску подсети

**ifconfig eth0 promisc** — перевести интерфейс eth0 в promiscuous-режим для «отлова» пакетов (sniffing)

**ifconfig eth0 -promisc** — отключить promiscuous-режим на интерфейсе eth0

**dhclient eth0** — активировать интерфейс eth0 в dhcp-режиме.

**route -n**

**netstat -rn** — вывести локальную таблицу маршрутизации

**route add -net 0/0 gw IP\_Gateway** — задать IP-адрес шлюза по умолчанию (default gateway)

**route add -net 192.168.0.0 netmask 255.255.0.0 gw 192.168.1.1** — добавить статический маршрут в сеть 192.168.0.0/16 через шлюз с IP-адресом 192.168.1.1

**route del 0/0 gw IP\_gateway** — удалить IP-адрес шлюза по умолчанию (default gateway)

**echo "1" > /proc/sys/net/ipv4/ip\_forward** — разрешить пересылку пакетов (forwarding)

**hostname** — отобразить имя компьютера

**host www.example.com** или **host 192.0.43.10** — разрешить имя www.example.com хоста в IP-адрес и наоборот

**ip link show** — отобразить состояние всех интерфейсов

**mii-tool eth0** — отобразить статус и тип соединения для интерфейса eth0

**ethtool eth0** — отображает статистику интерфейса eth0 с выводом такой информации, как поддерживаемые и текущие режимы соединения

**netstat -tupn** — отображает все установленные сетевые соединения по протоколам TCP и UDP без разрешения имён в IP-адреса и PID'ы и имена процессов, обеспечивающих эти соединения

**netstat -tupln** — отображает все сетевые соединения по протоколам TCP и UDP без разрешения имён в IP-адреса и PID'ы и имена процессов, слушающих порты

**tcpdump tcp port 80** — отобразить весь трафик на TCP-порт 80 (обычно — HTTP)

**iwlist scan** — просканировать эфир на предмет, доступности беспроводных точек доступа

**iwconfig eth1** — показать конфигурацию беспроводного сетевого интерфейса eth1

### Microsoft Windows networks(SAMBA):

**nbtscan ip\_addr**

**nmblookup -A ip\_addr** — разрешить netbios-имя nbtscan не во всех системах ставится по умолчанию, возможно, придётся доустанавливать вручную. nmblookup включён в пакет samba.

**smbclient -L ip\_addr/hostname** — отобразить ресурсы, предоставленные в общий доступ на windows-машине

**smbget -Rr smb://ip\_addr/share** — подобно wget может получить файлы с windows-машин через smb-протокол

**mount -t smbfs -o username=user,password=pass //winclient/share /mnt/share** — смонтировать smb-ресурс, предоставленный на windows-машине, в локальную файловую систему

### IPTABLES (firewall):

**iptables -t filter -nL**

**iptables -nL** — отобразить все цепочки правил

**iptables -t nat -L** — отобразить все цепочки правил в NAT-таблице

**iptables -t filter -F** или **iptables -F** — очистить все цепочки правил в filter-таблице

**iptables -t nat -F** — очистить все цепочки правил в NAT-таблице

**iptables -t filter -X** — удалить все пользовательские цепочки правил в filter-таблице

**iptables -t filter -A INPUT -p tcp --dport telnet -j ACCEPT** — позволить входящее подключение telnet'ом

**iptables -t filter -A OUTPUT -p tcp --dport http -j DROP** — блокировать исходящие HTTP-соединения

**iptables -t filter -A FORWARD -p tcp --dport pop3 -j ACCEPT** — позволить «прокидывать» (forward) POP3-соединения

**iptables -t filter -A INPUT -j LOG --log-prefix "DROP INPUT"** — включить журналирование ядром пакетов, проходящих через цепочку INPUT, и добавлением к сообщению префикса «DROP INPUT»

**iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE** — включить NAT (Network Address Translate) исходящих пакетов на интерфейс eth0. Допустимо при использовании с динамически выделяемыми IP-адресами.

**iptables -t nat -A PREROUTING -d 192.168.0.1 -p tcp -m tcp --dport 22 -j DNAT --to-destination 10.0.0.2:22**— перенаправление пакетов, адресованных одному хосту, на другой хост

### Мониторинг и отладка:

**top** — отобразить запущенные процессы, используемые ими ресурсы и другую полезную информацию (с автоматическим обновлением данных)

**ps -eafw** — отобразить запущенные процессы, используемые ими ресурсы и другую полезную информацию (единожды)

**ps -e -o pid,args --forest** — вывести PID'ы и процессы в виде дерева

**pstree** — отобразить дерево процессов

**kill -9 98989** или **kill -KILL 98989** — «убить» процесс с PID 98989 «на смерть» (без соблюдения целостности данных)

**kill -TERM 98989** — корректно завершить процесс с PID 98989

**kill -1 98989** или **kill -HUP 98989** — заставить процесс с PID 98989 перепрочитать файл конфигурации

**lsof -p 98989** — отобразить список файлов, открытых процессом с PID 98989

**lsof /home/user1** — отобразить список открытых файлов из директории /home/user1

**strace -c ls >/dev/null** — вывести список системных вызовов, созданных и полученных процессом ls

**strace -f -e open ls >/dev/null** — вывести вызовы библиотек

**watch -n1 'cat /proc/interrupts'** — отображать прерывания в режиме реального времени

**last reboot** — отобразить историю перезагрузок системы

**last user1** — отобразить историю регистрации пользователя user1 в системе и время его нахождения в ней

**lsmod** — вывести загруженные модули ядра

**free -m** — показать состояние оперативной памяти в мегабайтах

**smartctl -A /dev/hda** — контроль состояния жёсткого диска /dev/hda через SMART

**smartctl -i /dev/hda** — проверить доступность SMART на жёстком диске /dev/hda

**tail /var/log/dmesg** — вывести десять последних записей из журнала загрузки ядра

**tail /var/log/messages** — вывести десять последних записей из системного журнала

### Другие полезные команды:

**apropos …keyword** — выводит список команд, которые так или иначе относятся к ключевым словам. Полезно, когда вы знаете что делает программа, но не помните команду

**man ping**— вызов руководства по работе с программой, в данном случае, — ping

**whatis …keyword**— отображает описание действий указанной программы

**mkbootdisk --device /dev/fd0 `uname -r`** — создаёт загрузочный флоппи-диск

**gpg -c file1** — шифрует файл file1 с помощью GNU Privacy Guard

**gpg file1.gpg** — дешифрует файл file1 с помощью GNU Privacy Guard

**wget -r http://www.example.com** — загружает рекурсивно содержимое сайта http://www.example.com

**wget -c http://www.example.com/file.iso** — загрузить файл http://www.example.com/file.iso с возможностью останова и продолжения в последствии

**echo 'wget -c http://www.example.com/files.iso' | at 09:00** — начать закачку в указанное время

**ldd /usr/bin/ssh**— вывести список библиотек, необходимых для работы ssh

**alias hh='history'** — назначить алиас hh команде history

# General

## naming conventions

https://developers.google.com/java-dev-tools/codepro/doc/features/audit/audit\_rules\_com.instantiations.assist.eclipse.auditGroup.namingConventions?hl=en#com.instantiations.assist.eclipse.audit.booleanMethodNamingConvention