JAVA TUTORIAL | #INDEX POSTS | #INTERVIEW QUESTIONS | RESOURCES | HIRE ME | DOWNLOAD ANDROID APP | CONTRIBUTE

Subscribe to Download Java Design Patterns eBook Full name

name@example.com DOWNLOAD NOW

Spring MVC Internationalization (i18n) and Localization (L10n) Example

APRIL 2, 2018 BY PANKAJ — 20 COMMENTS

Welcome to the Spring Internationalization (i18n) tutorial. Any web application with users all around the world, **internationalization** (i18n) or **localization** (L10n) is very important for better user interaction.

Most of the web application frameworks provide easy ways to localize the application based on user locale settings. Spring also follows the pattern and provides extensive support for internationalization (i18n) through the use of Spring interceptors, Locale Resolvers and Resource Bundles for different locales.

Some earlier articles about i18n in java.

- Java Internationalization Example
- Struts2 Internationalization Example

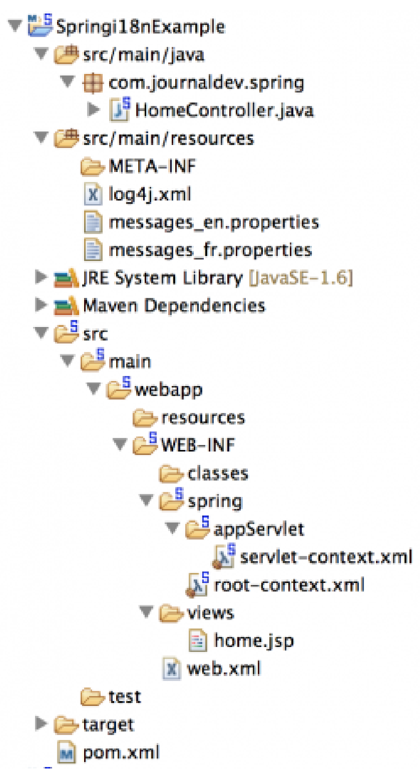
Table of Contents [hide]

Spring Internationalization i18n

Let's create a simple Spring MVC project where we will use request parameter to get the user locale and based on that set the response page label values from locale specific resource bundles.

Create a Spring MVC Project in the Spring Tool Suite to have the base code for our application. If you are not familiar with Spring Tool Suite or Spring MVC Projects, please read Spring MVC Example.

Our final project with localization changes looks like below image. We will look into all the parts of the application one by one.



```
Springi18nExample
  src/main/java
    com.journaldev.spring
      HomeController.java
  src/main/resources
    META-INF
    log4j.xml
    messages_en.properties
    messages_fr.properties
  JRE System Library [JavaSE-1.6]
  Maven Dependencies
  src
    main
      webapp
        resources
        WEB-INF
          classes
          spring
            appServlet
              servlet-context.xml
            root-context.xml
          views
            home.jsp
          web.xml
    test
  target
  pom.xml
```

Spring i18n Maven Configuration

Our Spring MVC pom.xml looks like below.

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/maven-v4_0_0.xsd">
        <modelVersion>4.0.0</modelVersion>
        <groupId>com.journaldev</groupId>
        <artifactId>spring</artifactId>
        <name>Springi18nExample</name>
        <packaging>war</packaging>
        <version>1.0.0-BUILD-SNAPSHOT</version>
        <properties>
                <java-version>1.6</java-version>
                <org.springframework-version>4.0.2.RELEASE</org.springframework-
version>
                <org.aspectj-version>1.7.4</org.aspectj-version>
                <org.slf4j-version>1.7.5</org.slf4j-version>
        </properties>
        <dependencies>
                <!-- Spring -->
                <dependency>
```

Most of the code is auto generated by STS, except that I have updated the Spring version to use the latest one as 4.0.2.RELEASE. We can remove dependencies or update the versions of other dependencies too but I have left them as it is for simplicity.

Spring Resource Bundle

For simplicity, let's assume that our application supports only two locales – **en** and **fr**. If no user locale is specified, we will use english as default locale. Let's create spring resource bundles for both these locales that will be used in the JSP page.

messages_en.properties code:

```
label.title=Login Page
label.firstName=First Name
label.lastName=Last Name
label.submit=Login
```

messages_fr.properties code:

```
label.title=Connectez-vous page
label.firstName=Pr\u00E9nom
label.lastName=Nom
label.submit=Connexion
```

Note that I am using unicode for special characters in the french locale resource bundles, so that it gets interpreted properly in the response HTML sent to client requests.

Another important point to note is that both the resource bundles are in the classpath of the application and their name has pattern as "messages_{locale}.properties". We will see why these are important later on.

Spring i18n Controller Class

Our controller class is very simple, it just logs the user locale and return the home.jsp page as response.

```
package com.journaldev.spring;

import java.util.Locale;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;

/**
 * Handles requests for the application home page.
 */
@Controller
public class HomeController {

        private static final Logger logger =
LoggerFactory.getLogger(HomeController.class);

        /**
         * Simply selects the home view to render by returning its name.
```

Spring i18n JSP Page

Our home.jsp page code looks like below.

```
<%@taglib uri="http://www.springframework.org/tags" prefix="spring"%>
<%@ page session="false"%>
<html>
<head>
<title><spring:message code="label.title" /></title>
</head>
<body>
        <form method="post" action="login">
                <table>
```

```
                        <tr>
                                <td><label> <strong><spring:message

code="label.firstName" /></strong>
                                </label></td>
                                <td><input name="firstName" /></td>
                        </tr>
                        <tr>
                                <td><label> <strong><spring:message
                                                                code="label.lastName"
/></strong>
                                </label></td>
```

The only part worth mentioning is the use of **spring:message** to retrieve the message with the given code. Make sure Spring tag libraries are configured using taglib jsp directive.Spring takes care of loading the appropriate resource bundle messages and make it available for the JSP pages to use.

Spring Internationalization i18n – Bean Configuration File

Spring Bean configuration file is the place where all the magic happens. This is the beauty of Spring framework as it helps us to focus more on business logic rather than coding for trivial tasks. Let's see how our spring bean configuration file looks and we will look at each of the beans one be one.

servlet-context.xml code:

```
<?xml version="1.0" encoding="UTF-8"?>
<beans:beans xmlns="http://www.springframework.org/schema/mvc"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:beans="http://www.springframework.org/schema/beans"
        xmlns:context="http://www.springframework.org/schema/context"
        xsi:schemaLocation="http://www.springframework.org/schema/mvc
http://www.springframework.org/schema/mvc/spring-mvc.xsd
                http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd
                http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context.xsd">

        <!-- DispatcherServlet Context: defines this servlet's request-processing
                infrastructure -->

        <!-- Enables the Spring MVC @Controller programming model -->
        <annotation-driven />

        <!-- Handles HTTP GET requests for /resources/** by efficiently serving
```

```
              up static resources in the ${webappRoot}/resources directory -->
    <resources mapping="/resources/**" location="/resources/" />
```

1. **annotation-driven** tag enables the Controller programming model, without it Spring won't recognize our HomeController as handler for client requests.
2. **context:component-scan** provides the package where Spring will look for the annotated components and register them automatically as Spring bean.
3. **messageSource** bean is configured to enable i18n for our application. **basename** property is used to provide the location of resource bundles. `classpath:messages` means that resource bundles are located in the classpath and follows name pattern as `messages_{locale}.properties`. defaultEncoding property is used to define the encoding used for the messages.
4. **localeResolver** bean of type `org.springframework.web.servlet.i18n.CookieLocaleResolver` is used to set a cookie in the client request so that further requests can easily recognize the user locale. For example, we can ask user to select the locale when he launches the web application for the first time and with the use of cookie, we can identify the user locale and automatically send locale specific response. We can also specify the default locale, cookie name and maximum age of the cookie before it gets expired and deleted by the client browser.
   If your application maintains user sessions, then you can also use `org.springframework.web.servlet.i18n.SessionLocaleResolver` as localeResolver to use a locale attribute in the user's session. The configuration is similar to CookieLocaleResolver.

   ```
   <bean id="localeResolver"
           class="org.springframework.web.servlet.i18n.SessionLocaleResolver">
           <property name="defaultLocale" value="en" />
   </bean>
   ```

   If we don't register any "localeResolver", **AcceptHeaderLocaleResolver** will be used by default, which resolves user locale by checking the `accept-language` header in the client HTTP request.

5. `org.springframework.web.servlet.i18n.LocaleChangeInterceptor` interceptor is configured to intercept the user request and identify the user locale. The parameter name is configurable and we are using request parameter name for locale as "locale". Without this interceptor, we won't be able to change the user locale and send the response based on the new locale settings of the user. It needs to be part of **interceptors** element otherwise Spring won't configure it as an interceptor.

If you are wondering about the configuration that tells Spring framework to load our context configurations, it's present in deployment descriptor of our MVC application.

```
<servlet>
        <servlet-name>appServlet</servlet-name>
        <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-
```

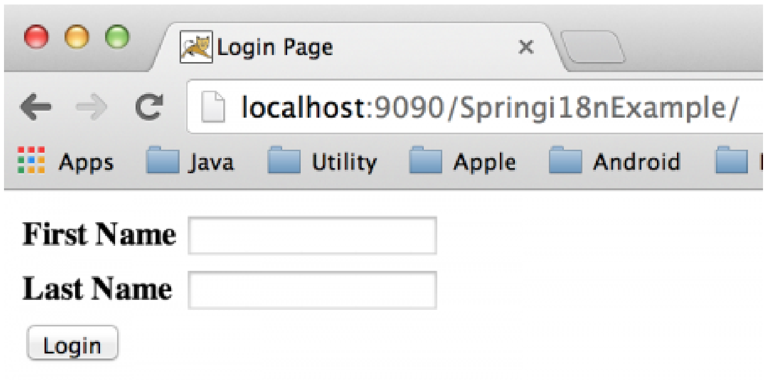
```
class>
        <init-param>
                <param-name>contextConfigLocation</param-name>
                <param-value>/WEB-INF/spring/appServlet/servlet-context.xml</param-
value>
        </init-param>
        <load-on-startup>1</load-on-startup>
</servlet>

<servlet-mapping>
        <servlet-name>appServlet</servlet-name>
        <url-pattern>/</url-pattern>
</servlet-mapping>
```

We can change the location or name of the context file by changing the web.xml configuration.

Our Spring i18n application is ready, just deploy it in any servlet container. Usually I export it as WAR file in a standalone tomcat web server webapps directory.

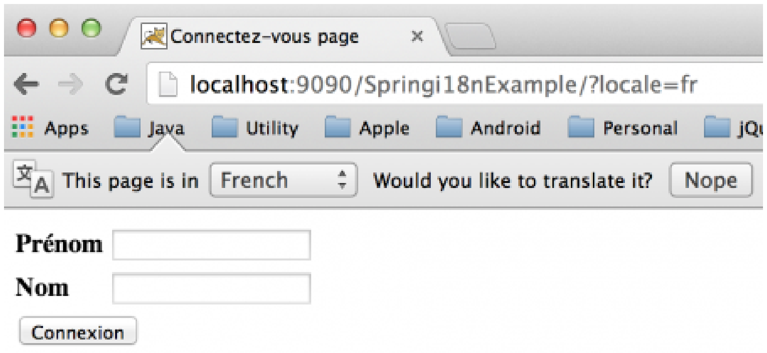
Here are the screenshots of our application home page with different locales.

Default Home Page (en locale):



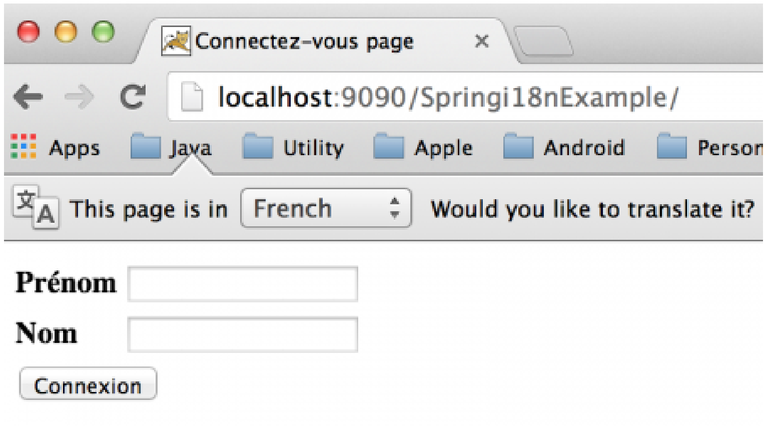
First Name
Last Name
Login

Passing Locale as parameter (fr locale):

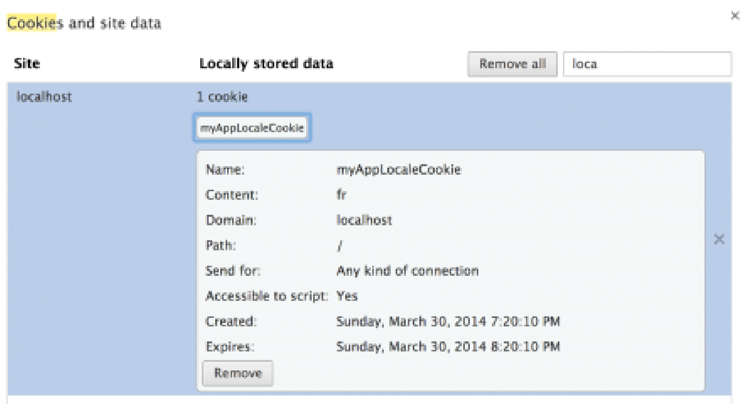

Connectez-vous page
localhost:9090/Springi18nExample/?locale=fr
Apps
Java
Utility
Apple
Android
Personal
This page is in French Would you like to translate it? Nope

Prénom
Nom
Connexion

Further requests without locale:



As you can see in the above image that we are not passing locale information in the client request but still our application identifies the user locale. You must have guessed by now that it's because of the CookieLocaleResolver bean that we configured in our spring bean configuration file. However you can check your browser cookies data to confirm it. I am using chrome and below image shows the cookie data stored by the application.



Name:	myAppLocaleCookie
Content:	fr
Domain:	localhost
Path:	/
Send for:	Any kind of connection
Accessible to script:	Yes
Created:	Sunday, March 30, 2014 7:20:10 PM
Expires:	Sunday, March 30, 2014 8:20:10 PM

Notice that cookie expiry time is one hour i.e 3600 seconds as configured by cookieMaxAge property.

If you will check server logs, you can see that locale is getting logged.

```
INFO : com.journaldev.spring.HomeController - Welcome home! The client locale is en.
INFO : com.journaldev.spring.HomeController - Welcome home! The client locale is fr.
INFO : com.journaldev.spring.HomeController - Welcome home! The client locale is fr.
```

That's all for Spring i18n example application, download the example project from below link and play around with it to learn more.

Download Spring i18n Project

« PREVIOUS
Spring Transaction Management Example JDBC

NEXT »
Spring @Autowired Annotation

About Pankaj
If you have come this far, it means that you liked what you are reading. Why not reach little more and connect with me directly on **Google Plus**, **Facebook** or **Twitter**. I would love to hear your thoughts and opinions on my articles directly.

Recently I started creating video tutorials too, so do check out my videos on **Youtube**.

FILED UNDER: SPRING

Comments

srikanth says
OCTOBER 7, 2016 AT 4:55 AM

suppose if need to support 30 different regional language. Will we creating 30 properites file ? Please advice how will design the application.

Reply

srikanth says

OCTOBER 9, 2016 AT 10:38 PM

Please reply above mentioned query.
suppose if need to support 30 different regional language. Will we creating 30 properites file ? Please advice how will design the application.

Reply

Sean says

DECEMBER 14, 2017 AT 11:24 PM

A viable alternative could be to look into a text translator service such as the Google Cloud Translation API (https://cloud.google.com/translate/docs/), though I believe convention in business is to use native speaker translations rather than relying on a framework in order to account for any linguistic nuances

Reply

SVK PETO says

FEBRUARY 17, 2016 AT 6:58 AM

How to do it, if my site dont allow GET requests, but only POST?

Reply

Vamshi says

JANUARY 6, 2016 AT 4:52 AM

Hi Friends,
I am facing problem in implementing Internationalization in spring, as per my knowledge I have configured all the required resources. Please help me, the following is my project location.
https://github.com/VamshiKrishna2828/SpringI18NDemo
Thanks in advance.

Reply

Jeferson says
OCTOBER 7, 2015 AT 9:01 AM
Hello. How can I retrieve message from properties file programmatically??
Reply

manish gour says
MARCH 4, 2016 AT 11:23 PM
See this https://www.youtube.com/watch?v=cUqquQDZR-c&index=21&list=PLC97BDEFDCDD169D7#t=266.926099
Reply

cindy says
AUGUST 19, 2015 AT 3:25 AM
Hello, how can I change the language of a calendar..datepicker in spring??
Reply

Vishal Wagh says
JULY 2, 2015 AT 12:22 AM
1.7
1.7
…………..
updated the pom.xml with above chages.
But I am getting below error using Jdk 7.
org.apache.maven.lifecycle.LifecycleExecutionException: Failed to execute goal org.apache.maven.plugins:maven-compiler-plugin:2.5.1:compile (default-compile) on project spring: Fatal error compiling
Please help me…
Reply

zoli says
MAY 19, 2015 AT 5:50 AM
I think you don't need the annotation-driven tag in your app config as context:component-scan does the same:
Element : component-scan
Scans the classpath for annotated components that will be auto-registered as Spring beans. By default,

the
Spring-provided @Component, @Repository, @Service, and @Controller stereotypes will be detected.
Note: This
tag implies the effects of the 'annotation-config' tag, activating @Required, @Autowired, @PostConstruct, @PreDestroy, @Resource, @PersistenceContext and @PersistenceUnit annotations in the component classes
Reply

daj says
APRIL 29, 2015 AT 4:18 AM
Thank you very much ! you have just saved my life
Reply

Håvard says
MARCH 2, 2015 AT 1:17 PM
According to Google, it is bad practice to use request parameters and cookies to show translated versions of the page: https://support.google.com/webmasters/answer/182192
Reply

Lluis says
JULY 12, 2015 AT 5:27 AM
So what's your recomendation in order to internationalize your site with spring?
Reply

Vince says
SEPTEMBER 3, 2014 AT 8:05 AM
I got following exceptions when I copied and ran this example using IntelliJ. If I go to my tomcat directory, I can see messages_X.properties are under WEB-INF/classes and no spelling error. Please help. Thanks.
SEVERE: Servlet.service() for servlet [jsp] in context with path [/spring] threw exception [javax.servlet.ServletException: javax.servlet.jsp.JspTagException: No message found under code 'label.title' for locale 'en_US'.] with root cause
javax.servlet.jsp.JspTagException: No message found under code 'label.title' for locale 'en_US'.
at org.springframework.web.servlet.tags.MessageTag.doEndTag(MessageTag.java:202)
at org.apache.jsp.home_jsp._jspx_meth_spring_005fmessage_005f0(home_jsp.java:135)
at org.apache.jsp.home_jsp._jspService(home_jsp.java:73)

```
at org.apache.jasper.runtime.HttpJspBase.service(HttpJspBase.java:70)
at javax.servlet.http.HttpServlet.service(HttpServlet.java:727)
at org.apache.jasper.servlet.JspServletWrapper.service(JspServletWrapper.java:432)
at org.apache.jasper.servlet.JspServlet.serviceJspFile(JspServlet.java:390)
at org.apache.jasper.servlet.JspServlet.service(JspServlet.java:334)
at javax.servlet.http.HttpServlet.service(HttpServlet.java:727)
at org.apache.catalina.core.ApplicationFilterChain.internalDoFilter(ApplicationFilterChain.java:303)
at org.apache.catalina.core.ApplicationFilterChain.doFilter(ApplicationFilterChain.java:208)
at org.apache.tomcat.websocket.server.WsFilter.doFilter(WsFilter.java:52)
at org.apache.catalina.core.ApplicationFilterChain.internalDoFilter(ApplicationFilterChain.java:241)
at org.apache.catalina.core.ApplicationFilterChain.doFilter(ApplicationFilterChain.java:208)
at org.apache.catalina.core.StandardWrapperValve.invoke(StandardWrapperValve.java:220)
at org.apache.catalina.core.StandardContextValve.invoke(StandardContextValve.java:122)
at org.apache.catalina.authenticator.AuthenticatorBase.invoke(AuthenticatorBase.java:501)
at org.apache.catalina.core.StandardHostValve.invoke(StandardHostValve.java:171)
at org.apache.catalina.valves.ErrorReportValve.invoke(ErrorReportValve.java:102)
at org.apache.catalina.valves.AccessLogValve.invoke(AccessLogValve.java:950)
at org.apache.catalina.core.StandardEngineValve.invoke(StandardEngineValve.java:116)
at org.apache.catalina.connector.CoyoteAdapter.service(CoyoteAdapter.java:408)
at org.apache.coyote.http11.AbstractHttp11Processor.process(AbstractHttp11Processor.java:1040)
at org.apache.coyote.AbstractProtocol$AbstractConnectionHandler.process(AbstractProtocol.java:607)
at org.apache.tomcat.util.net.JIoEndpoint$SocketProcessor.run(JIoEndpoint.java:314)
at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1145)
at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:615)
at org.apache.tomcat.util.threads.TaskThread$WrappingRunnable.run(TaskThread.java:61)
at java.lang.Thread.run(Thread.java:744)
```

Reply

Deepak says

AUGUST 18, 2017 AT 5:49 PM

Have you given the id as messageSource for the class-RelodableResourceBundleMessageSource?
Please check if not given, provide the id.

Reply

rajan says

SEPTEMBER 2, 2014 AT 11:16 AM

Hai pankaj,
I compiled this project without any error, what is the url to access the home page?
`http://localhost:8080/spring/` it is not showing any page, How do I access the home

page(controller)
please help
Reply

vivek says
AUGUST 2, 2016 AT 2:57 AM
Change the name of the project.
Reply

Acid Serg says
AUGUST 30, 2014 AT 8:21 AM
Good example! Thanx a lot!
Reply

Anibal Gomez says
JULY 27, 2014 AT 2:22 PM
Thanks for the 100% working example!
Reply

praveen says
APRIL 13, 2014 AT 12:51 PM
it's good , it resolved my problem .. good example . thanks pankaj
Reply

Leave a Reply

Your email address will not be published. Required fields are marked *
Comment

Name *

Email *

Save my name, email, and website in this browser for the next time I comment.

POST COMMENT

Search for tutorials...

DOWNLOAD ANDROID APP

GET IT ON Google Play

SPRING FRAMEWORK

Spring Tutorial

Spring Core

- Spring Framework
- Spring Dependency Injection
- Spring IoC and Bean
- Spring Bean Life Cycle
- Spring REST
- Spring REST XML
- Spring RestTemplate

- Spring AOP
- Spring AOP Method Profiling
- Spring Annotations
- Spring @Autowired
- Spring @RequestMapping

Spring MVC

- Spring MVC Example
- Spring MVC Tutorial
- Spring MVC Exception Handling
- Spring MVC Validator
- Spring MVC Intercepter
- Spring MVC File Upload
- Spring MVC i18n
- Spring MVC Hibernate MqSQL

Spring ORM

- Spring ORM
- Spring ORM JPA
- Spring Data JPA
- Spring Transaction
- Spring JdbcTemplate

Spring Security

- Spring Security Overview
- Spring Security Example Tutorial
- Spring Security UserDetailsService
- Spring MVC Login Logout
- Spring Security Roles

Spring Boot

- Spring Boot Tutorial
- Spring Boot Components
- Spring Boot CLI Hello World
- Spring Boot Initilizr Web
- Spring Boot Initilizr IDE
- Spring Boot Initilizr CLI
- Spring Boot Initilizr Tools
- Spring Boot MongoDB
- Spring Boot Redis Cache
- Spring Boot Interview Questions

Spring Batch

- Spring Batch
- Spring Batch Example

Spring AMQP

- Spring AMQP
- Spring RabbitMQ
- Spring AMQP RabbitMQ
- Apache ActiveMQ
- Spring ActiveMQ Tutorial

- Spring ActiveMQ Example

Spring Integrations

- Spring JDBC
- Spring DataSource JNDI
- Spring Hibernate
- Spring Primefaces JPA
- Spring Primefaces MongoDB
- Spring Primefaces Hibernate
- SpringRoo Primefaces Hibernate
- Spring JSF
- Spring JDF Hibernate

Miscellaneous

- Spring Data MongoDB
- Spring Interview Questions

RECOMMENDED TUTORIALS

Java Tutorials

- Java IO
- Java Regular Expressions
- Multithreading in Java
- Java Logging
- Java Annotations
- Java XML
- Collections in Java
- Java Generics
- Exception Handling in Java
- Java Reflection
- Java Design Patterns
- JDBC Tutorial

Java EE Tutorials

- Servlet JSP Tutorial
- Struts2 Tutorial
- Spring Tutorial
- Hibernate Tutorial
- Primefaces Tutorial
- Apache Axis 2
- JAX-RS
- Memcached Tutorial

Java Programming Courses

Metric Spring Pins

Poland Spring Coupons

Learn Java in 7 Days

Create Your Own Website

Learn Programming Online

Oracle Database Tools

Employer Interview Questions

© 2018 · Privacy Policy · Don't copy, it's Bad Karma · P