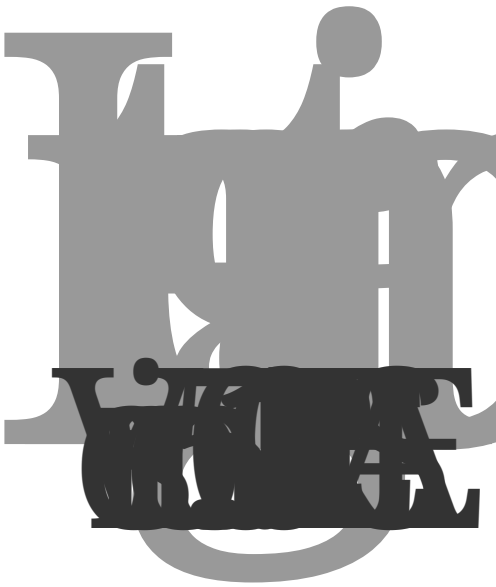
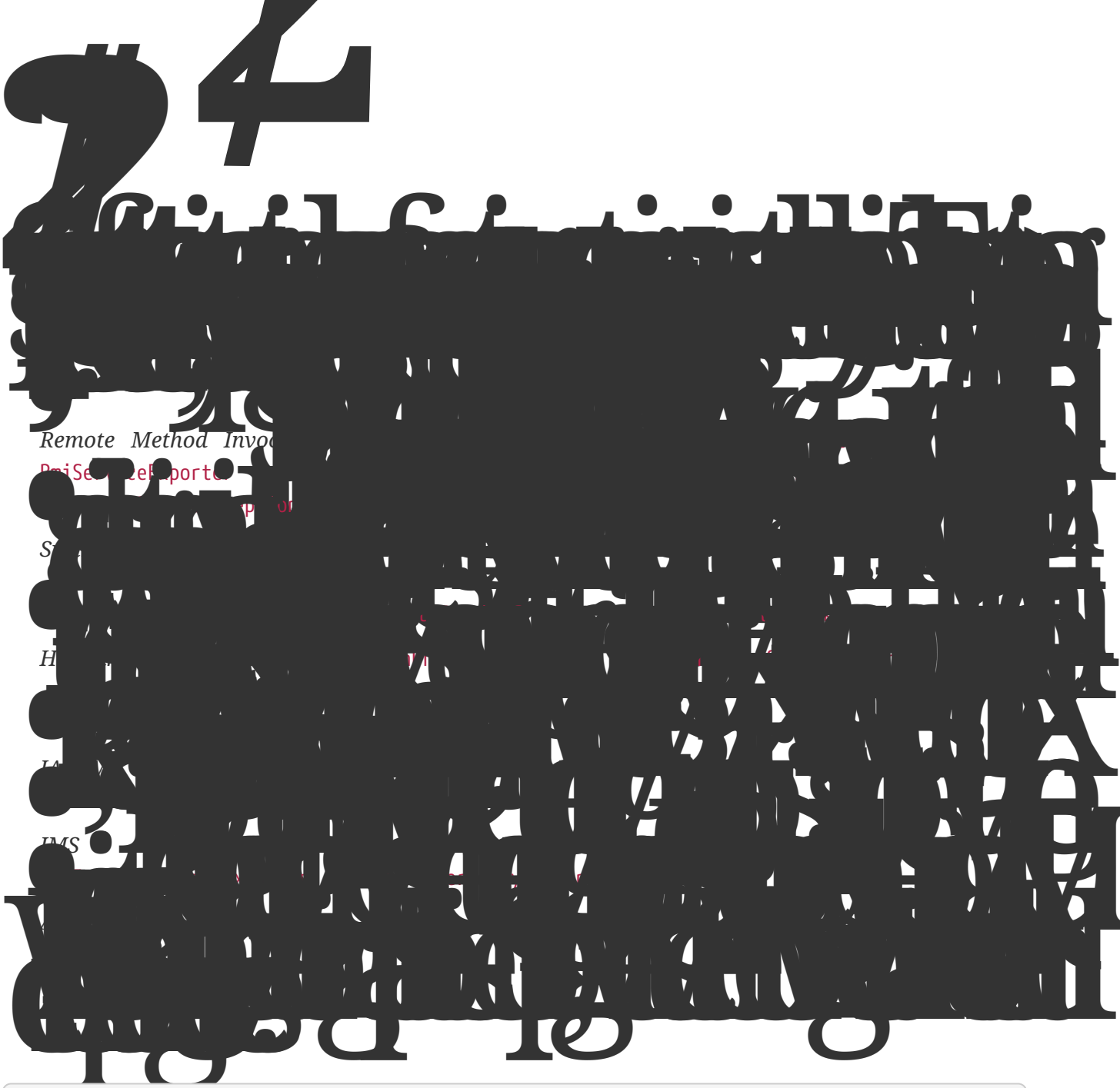


[redacted]

[redacted]

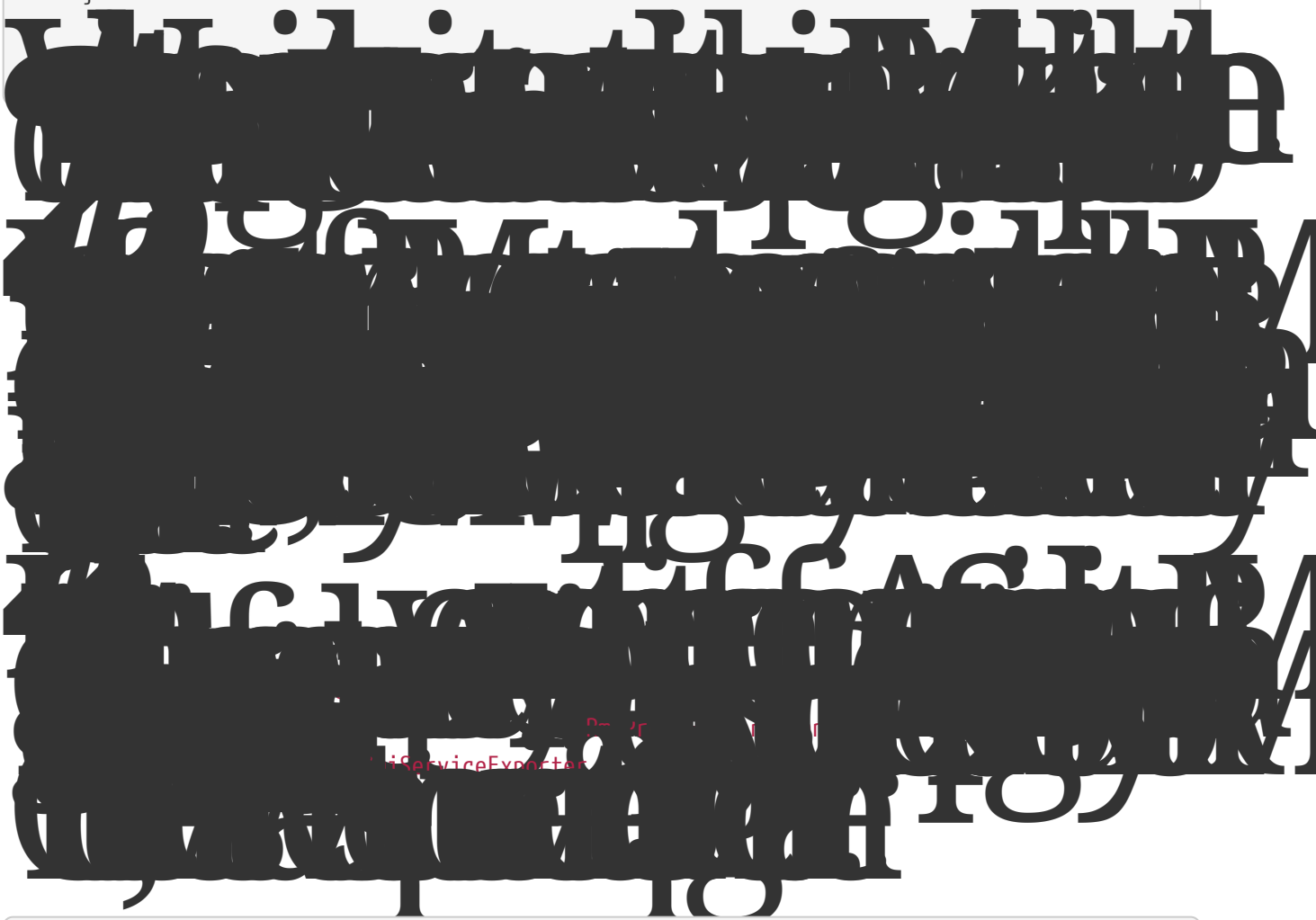
[redacted]



```
public class Account implements Serializable{

    private String name;

    public String getName(){
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

}
```

```java
public interface AccountService {

    public void insertAccount(Account account);

    public List<Account> getAccounts(String name);

}
```

```java
// the implementation doing nothing at the moment
public class AccountServiceImpl implements AccountService {

    public void insertAccount(Account acc) {
        // do something...
    }

    public List<Account> getAccounts(String name) {
        // do something...
    }
```



```xml
<bean id="accountService" class="example.AccountServiceImpl">
    <!-- any additional properties, maybe a DAO? -->
</bean>
```

RmiServiceExporter :

```
<bean class="org.springframework.remoting.rmi.RmiServiceExporter">
    <!-- does not necessarily have to be the same name as the bean to be exported -->
    <property name="serviceName" value="AccountService"/>
    <property name="service" ref="accountService"/>
    <property name="serviceInterface" value="example.AccountService"/>
    <!-- defaults to 1099 -->
    <property name="registryPort" value="1199"/>
</bean>
```

'rmi://HOST:1199/AccountService

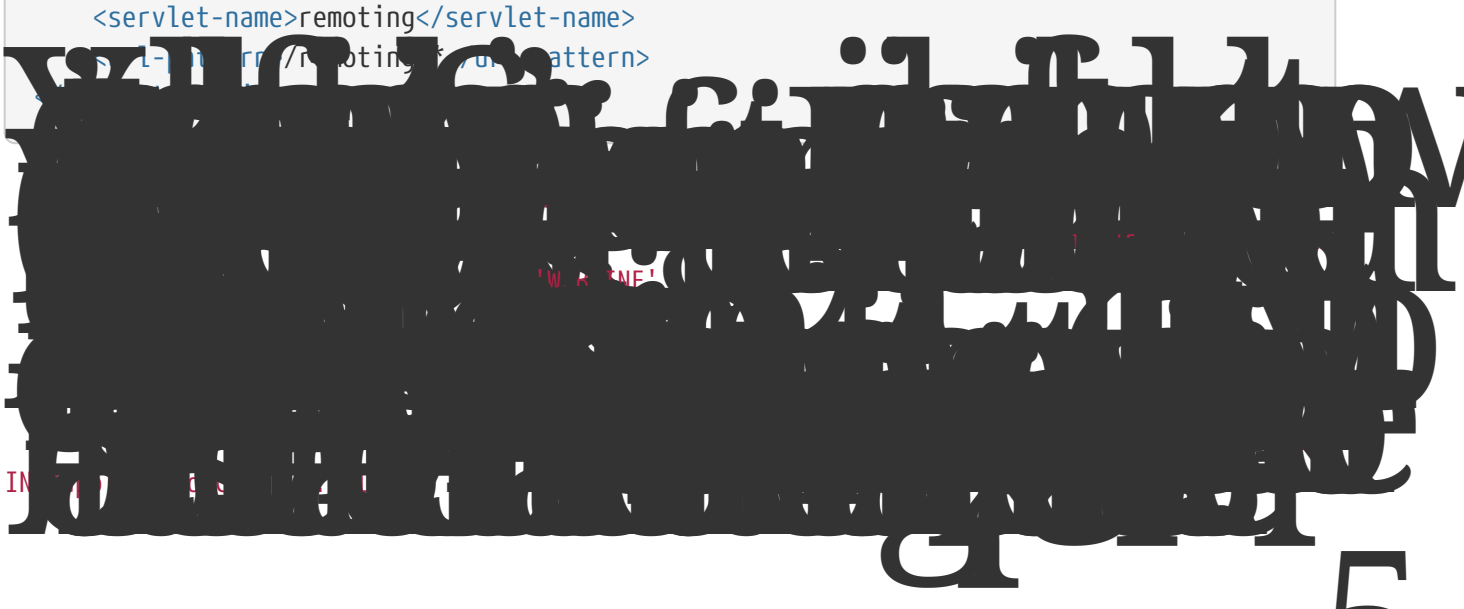
AccountService

```
public class SimpleObject {

    private AccountService accountService;

    public void setAccountService(AccountService accountService) {
        this.accountService = accountService;
    }

    // additional methods using the accountService
}
```

4

```
<bean class="example.SimpleObject">
    <property name="accountService" ref="accountService"/>
</bean>

<bean id="accountService" class="org.springframework.remoting.rmi.RmiProxyFactoryBean
">
    <property name="serviceUrl" value="rmi://HOST:1199/AccountService"/>
    <property name="serviceInterface" value="example.AccountService"/>
</bean>
```

```
<servlet>
    <servlet-name>remoting</servlet-name>
    <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
    <load-on-startup>1</load-on-startup>
</servlet>

<servlet-mapping>
    <servlet-name>remoting</servlet-name>
```



remoting-servlet.xml

HessianServiceExporter

```
<bean id="accountService" class="example.AccountServiceImpl">
    <!-- any additional properties, maybe a DAO? -->
</bean>

<bean name="/AccountService" class=
"org.springframework.remoting.caucho.HessianServiceExporter">
    <property name="service" ref="accountService"/>
    <property name="serviceInterface" value="example.AccountService" />
</bean>
```

HessianServiceExporter 'WEB-INF/applicationContext.xml'

```
<bean name="accountExporter" class=
"org.springframework.remoting.caucho.HessianServiceExporter">
    <property name="service" ref="accountService"/>
    <property name="serviceInterface" value="example.AccountService"/>
</bean>
```

web.xml'

AccountService

```
<servlet>
    <servlet-name>accountExporter</servlet-name>
    <servlet-class>
org.springframework.web.context.support.HttpRequestHandlerServlet</servlet-class>
</servlet>

<servlet-mapping>
    <servlet-name>accountExporter</servlet-name>
    <url-pattern>/remoting/AccountService</url-pattern>
</servlet-mapping>
```

HessianProxyFactoryBean

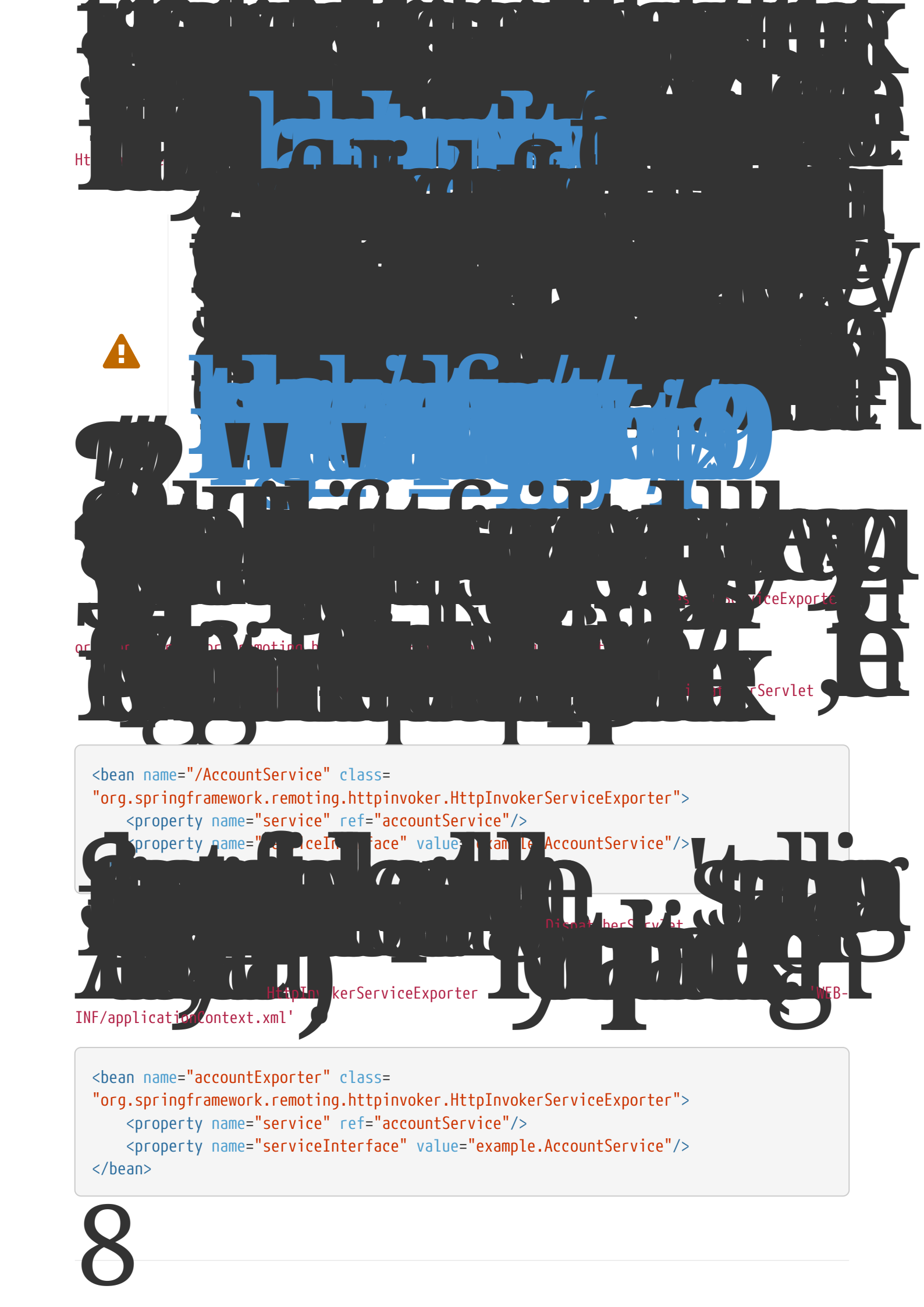
SimpleObject AccountService

```
<bean class="example.SimpleObject">
    <property name="accountService" ref="accountService"/>
</bean>

<bean id="accountService" class=
"org.springframework.remoting.caucho.HessianProxyFactoryBean">
    <property name="serviceUrl" value="http://remotehost:8080/remoting/AccountService
"/>
    <property name="serviceInterface" value="example.AccountService"/>
</bean>
```

HessianProxyFactoryBean

DataSource

```
<bean class="org.springframework.web.servlet.handler.BeanNameUrlHandlerMapping">
    <property name="interceptors" ref="authorizationInterceptor"/>
</bean>

<bean id="authorizationInterceptor"
        class=
"org.springframework.web.servlet.handler.UserRoleAuthorizationInterceptor">
    <property name="authorizedRoles" value="administrator,operator"/>
</bean>
```



```
<bean name="/AccountService" class=
"org.springframework.remoting.httpinvoker.HttpInvokerServiceExporter">
    <property name="service" ref="accountService"/>
    <property name="serviceInterface" value="example.AccountService"/>
</bean>
```

```
<bean name="accountExporter" class=
"org.springframework.remoting.httpinvoker.HttpInvokerServiceExporter">
    <property name="service" ref="accountService"/>
    <property name="serviceInterface" value="example.AccountService"/>
</bean>
```

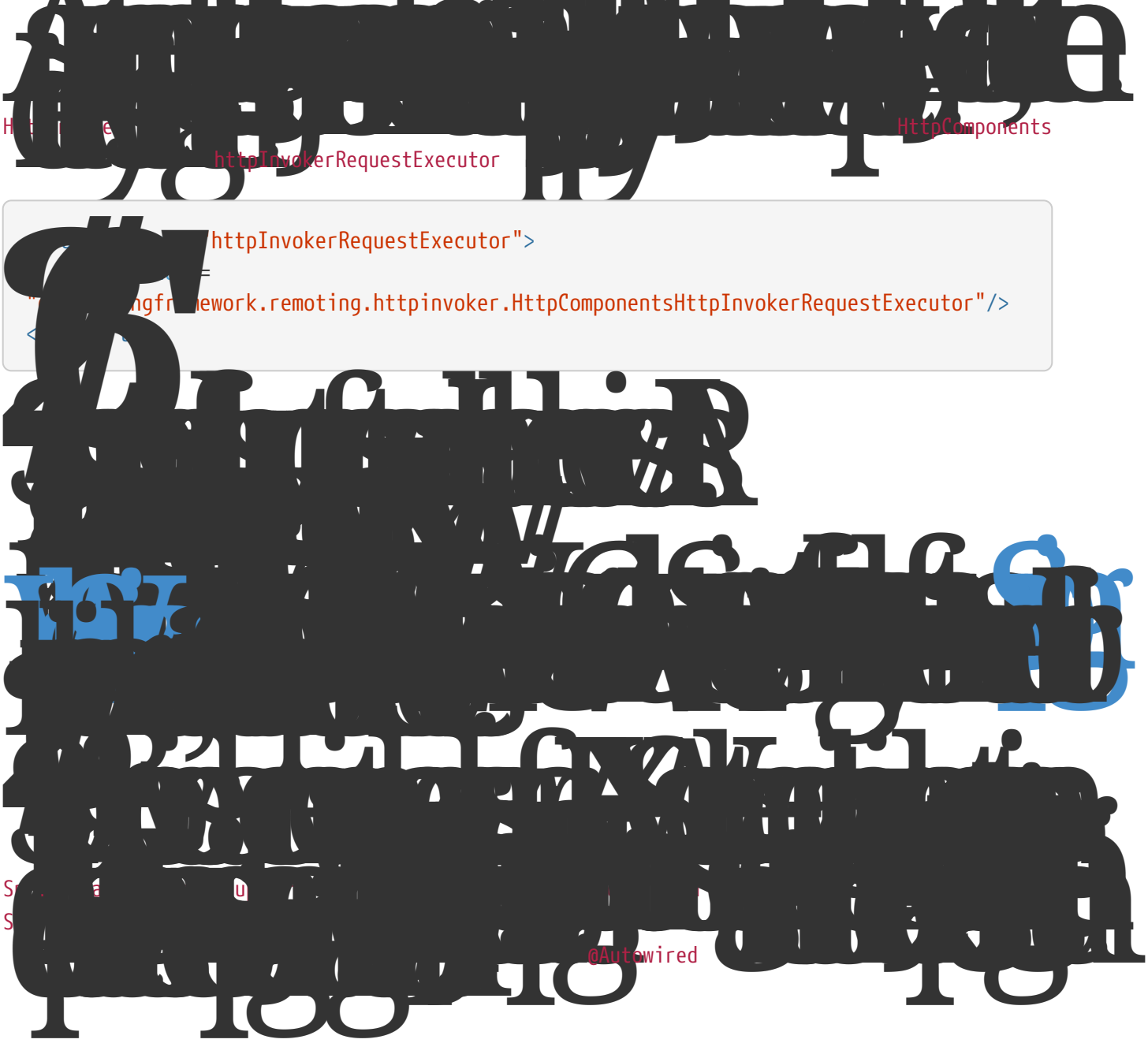
'web.xml'

```
<servlet>
    <servlet-name>accountExporter</servlet-name>
    <servlet-class>
org.springframework.web.context.support.HttpRequestHandlerServlet</servlet-class>
</servlet>

<servlet-mapping>
    <servlet-name>accountExporter</servlet-name>
    <url-pattern>/remoting/AccountService</url-pattern>
</servlet-mapping>
```

SimpleHttpInvokerServiceExporter

```
<bean name="accountExporter"
        class=
"org.springframework.remoting.httpinvoker.SimpleHttpInvokerServiceExporter">
    <property name="service" ref="accountService"/>
    <property name="serviceInterface" value="example.AccountService"/>
</bean>

<bean id="httpServer"
        class="org.springframework.remoting.support.SimpleHttpServerFactoryBean">
    <property name="contexts">
        <util:map>
            <entry key="/remoting/AccountService" value-ref="accountExporter"/>
        </util:map>
    </property>
    <property name="port" value="8080"/>
</bean>
```

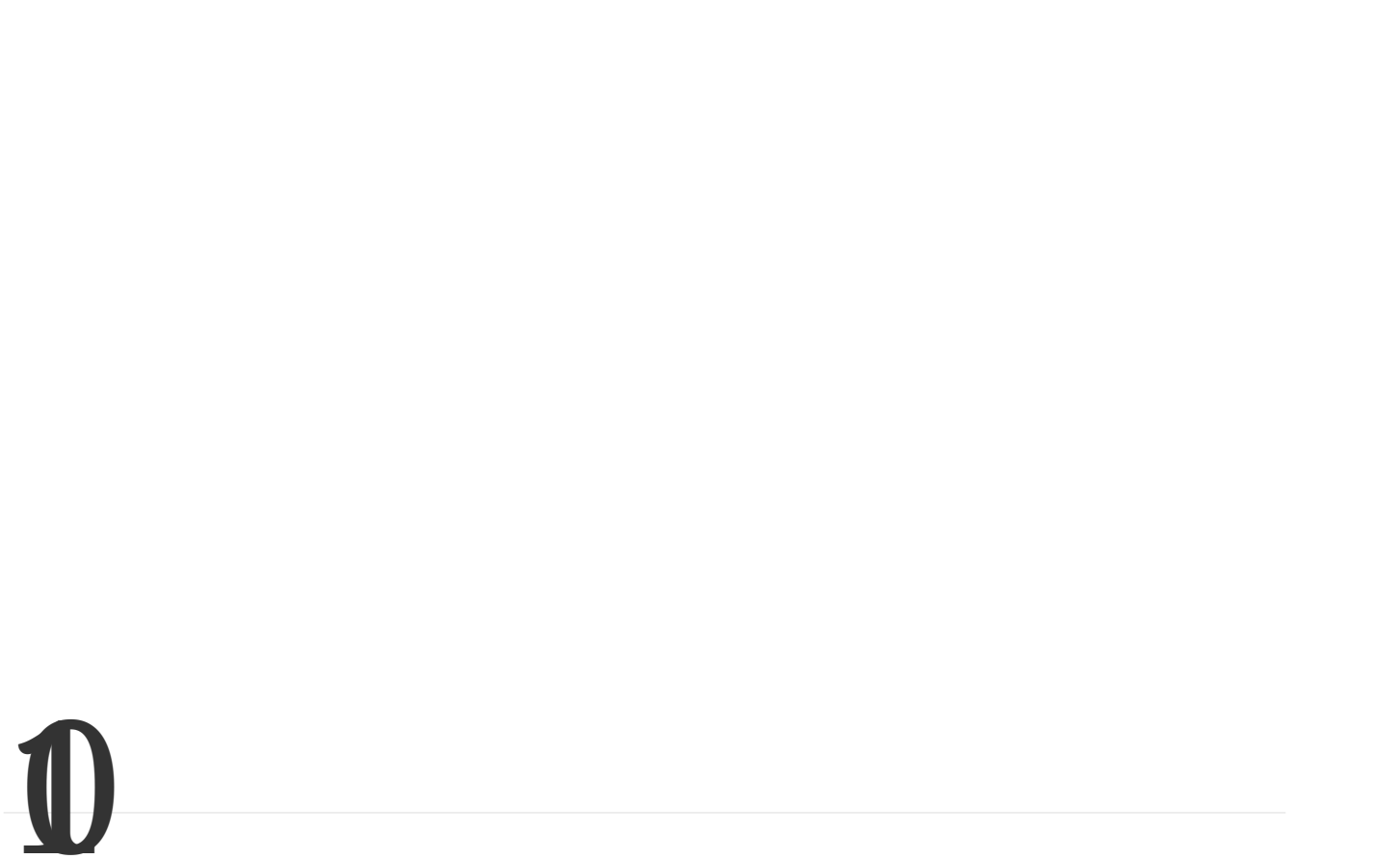
```
<bean id="httpInvokerProxy" class=
"org.springframework.remoting.httpinvoker.HttpInvokerProxyFactoryBean">
    <property name="serviceUrl" value="http://remotehost:8080/remoting/AccountService
"/>
    <property name="serviceInterface" value="example.AccountService"/>
</bean>
```



HttpComponents

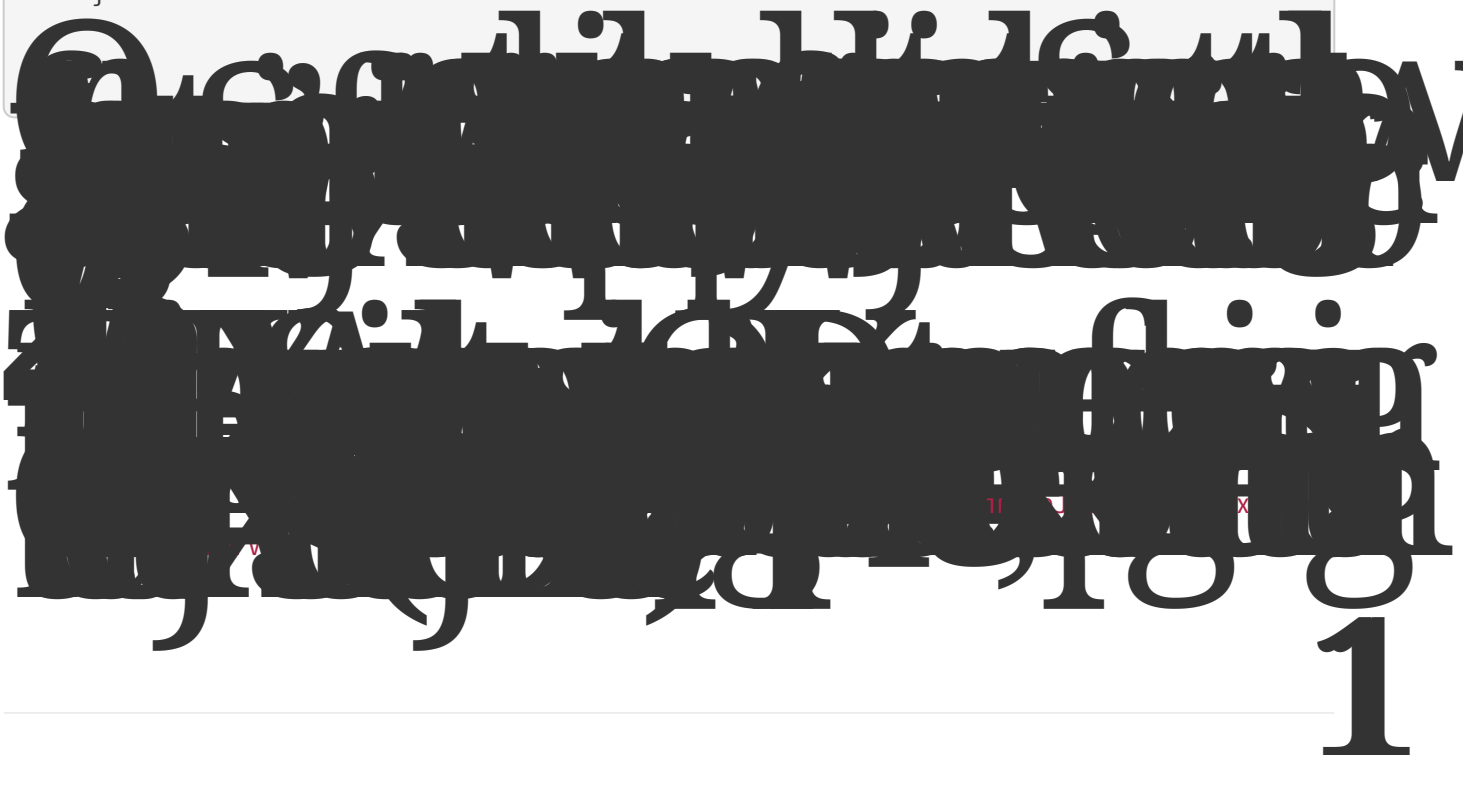
httpInvokerRequestExecutor

```
<bean id="httpInvokerRequestExecutor"
class="org.springframework.remoting.httpinvoker.HttpComponentsHttpInvokerRequestExecutor"/>
```



@Autowired

```java
/**
 * JAX-WS compliant AccountService implementation that simply delegates
 * to the AccountService implementation in the root web application context.
 *
 * This wrapper class is necessary because JAX-WS requires working with dedicated
 * endpoint classes. If an existing service needs to be exported, a wrapper that
 * extends SpringBeanAutowiringSupport for simple Spring bean autowiring (through
 * the @Autowired annotation) is the simplest JAX-WS compliant way.
 *
 * This is the class registered with the server-side JAX-WS implementation.
 * In the case of a Java EE 5 server, this would simply be defined as a servlet
 * in web.xml, with the server detecting that this is a JAX-WS endpoint and reacting
 * accordingly. The servlet name usually needs to match the specified WS service name.
 *
 * The web service engine manages the lifecycle of instances of this class.
 * Spring bean references will just be wired in here.
 */
import org.springframework.web.context.support.SpringBeanAutowiringSupport;

@WebService(serviceName="AccountService")
public class AccountServiceEndpoint extends SpringBeanAutowiringSupport {

    @Autowired
    private AccountService biz;

    @WebMethod
    public void insertAccount(Account acc) {
        biz.insertAccount(acc);
    }

    @WebMethod
    public Account[] getAccounts(String name) {
        return biz.getAccounts(name);
    }
```



@Autowired

```
<bean class="org.springframework.remoting.jaxws.SimpleJaxWsServiceExporter">
    <property name="baseAddress" value="http://localhost:8080/"/>
</bean>

<bean id="accountServiceEndpoint" class="example.AccountServiceEndpoint">
    ...
</bean>
```

@Autowired

```
@WebService(serviceName="AccountService")
public class AccountServiceEndpoint {

    @Autowired
    private AccountService biz;

    @WebMethod
    public void insertAccount(Account acc) {
        biz.insertAccount(acc);
    }

    @WebMethod
    public List<Account> getAccounts(String name) {
        return biz.getAccounts(name);
    }
```

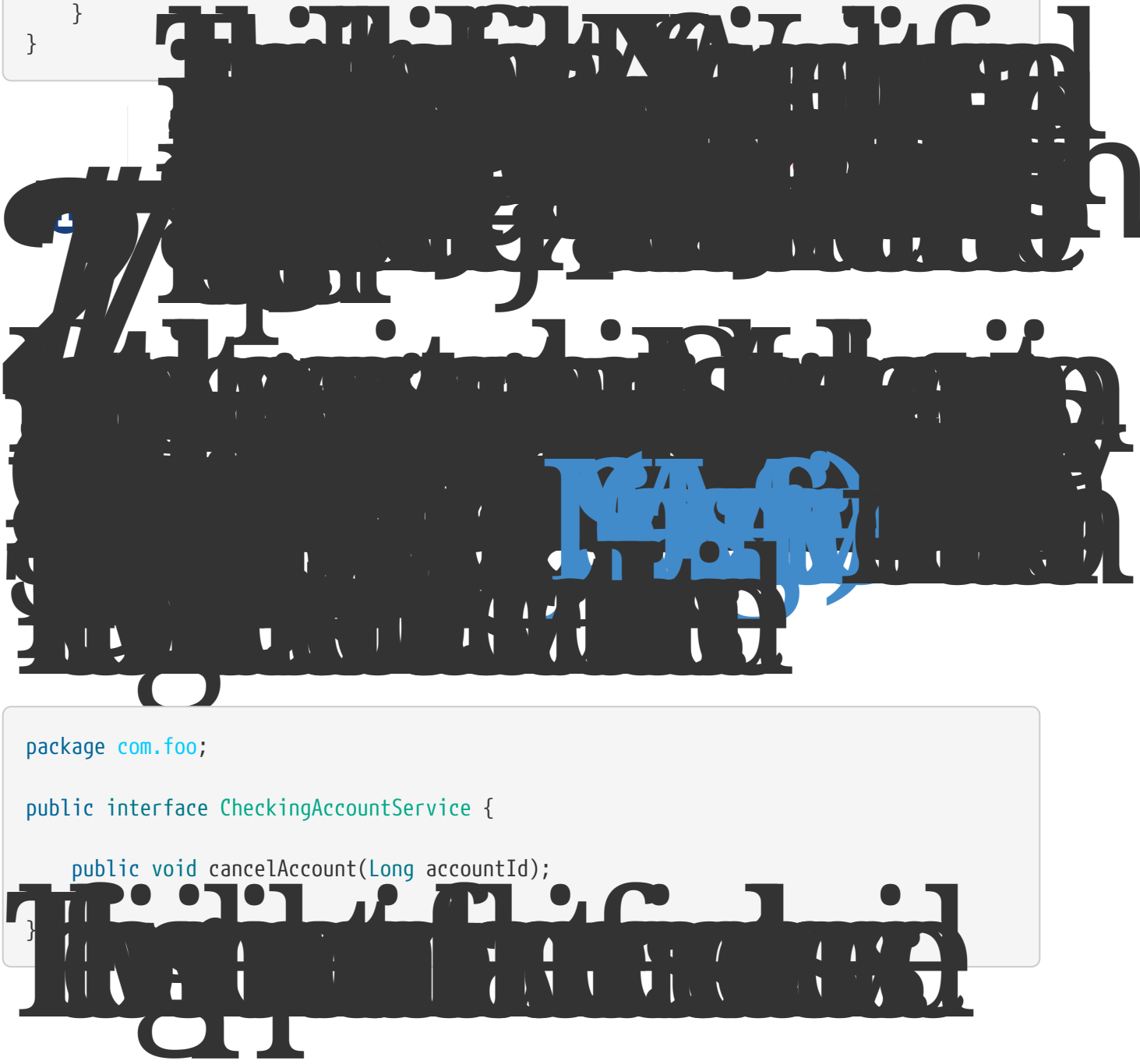
EE

```
<bean id="accountWebService" class=
"org.springframework.remoting.jaxws.JaxWsPortProxyFactoryBean">
    <property name="serviceInterface" value="example.AccountService"/>
    <property name="wsdlDocumentUrl" value=
"http://localhost:8888/AccountServiceEndpoint?WSDL"/>
    <property name="namespaceUri" value="http://example/"/>
    <property name="serviceName" value="AccountService"/>
    <property name="portName" value="AccountServiceEndpointPort"/>
```



```
<bean id="client" class="example.AccountClientImpl">
    ...
    <property name="service" ref="accountWebService"/>
```

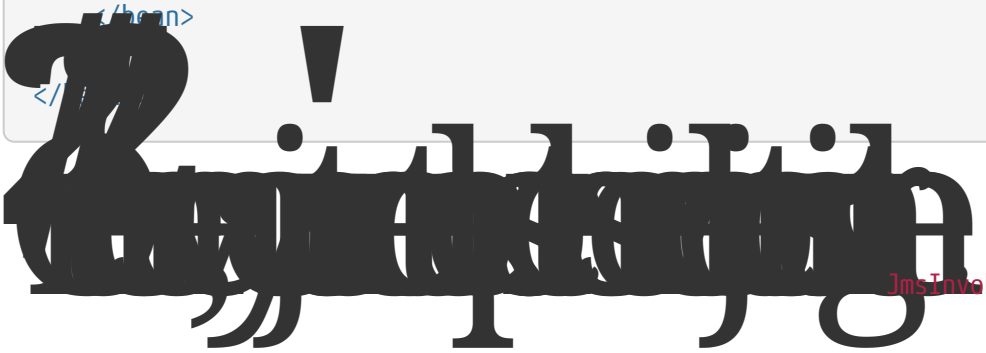
```
public class AccountClientImpl {

    private AccountService service;

    public void setService(AccountService service) {
        this.service = service;
    }

    public void foo() {
        service.insertAccount(...);
    }
}
```



```
package com.foo;

public interface CheckingAccountService {

    public void cancelAccount(Long accountId);
}
```

```
package com.foo;

public class SimpleCheckingAccountService implements CheckingAccountService {

    public void cancelAccount(Long accountId) {
        System.out.println("Cancelling account [" + accountId + "]");
    }
}
```

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

    <bean id="connectionFactory" class="org.apache.activemq.ActiveMQConnectionFactory
">
        <property name="brokerURL" value="tcp://ep-t43:61616"/>
    </bean>

    <bean id="queue" class="org.apache.activemq.command.ActiveMQQueue">
        <constructor-arg value="mmm"/>
    </bean>
</
```



JmsInvokerServiceExporter ●

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

    <bean id="checkingAccountService"
            class="org.springframework.jms.remoting.JmsInvokerServiceExporter">
        <property name="serviceInterface" value="com.foo.CheckingAccountService"/>
        <property name="service">
            <bean class="com.foo.SimpleCheckingAccountService"/>
        </property>
    </bean>

    <bean class="org.springframework.jms.listener.SimpleMessageListenerContainer">
        <property name="connectionFactory" ref="connectionFactory"/>
        <property name="destination" ref="queue"/>
        <property name="concurrentConsumers" value="3"/>
        <property name="messageListener" ref="checkingAccountService"/>
    </bean>

</beans>
```

```
package com.foo;

import org.springframework.context.support.ClassPathXmlApplicationContext;

public class Server {

    public static void main(String[] args) throws Exception {
        new ClassPathXmlApplicationContext(new String[]{"com/foo/server.xml",
"com/foo/jms.xml"});
    }
}
```

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

    <bean id="checkingAccountService"
            class="org.springframework.jms.remoting.JmsInvokerProxyFactoryBean">
        <property name="serviceInterface" value="com.foo.CheckingAccountService"/>
        <property name="connectionFactory" ref="connectionFactory"/>
        <property name="queue" ref="queue"/>
    </bean>

</beans>
```

```
package com.foo;

import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;

public class Client {

    public static void main(String[] args) throws Exception {
        ApplicationContext ctx = new ClassPathXmlApplicationContext(
                new String[] {"com/foo/client.xml", "com/foo/jms.xml"});
        CheckingAccountService service = (CheckingAccountService) ctx.getBean(
"checkingAccountService");
        service.cancelAccount(new Long(10));


}
```

[redacted]

Table 1. RestTemplate method

getForObject	
getForEntity	
headForHeaders	
postForLocation	
postForObject	
postForEntity	
put	
patchForObject	
delete	
optionsForAllow	
exchange	
execute	

```
RestTemplate template = new RestTemplate(new HttpComponentsClientHttpRequestFactory());
```

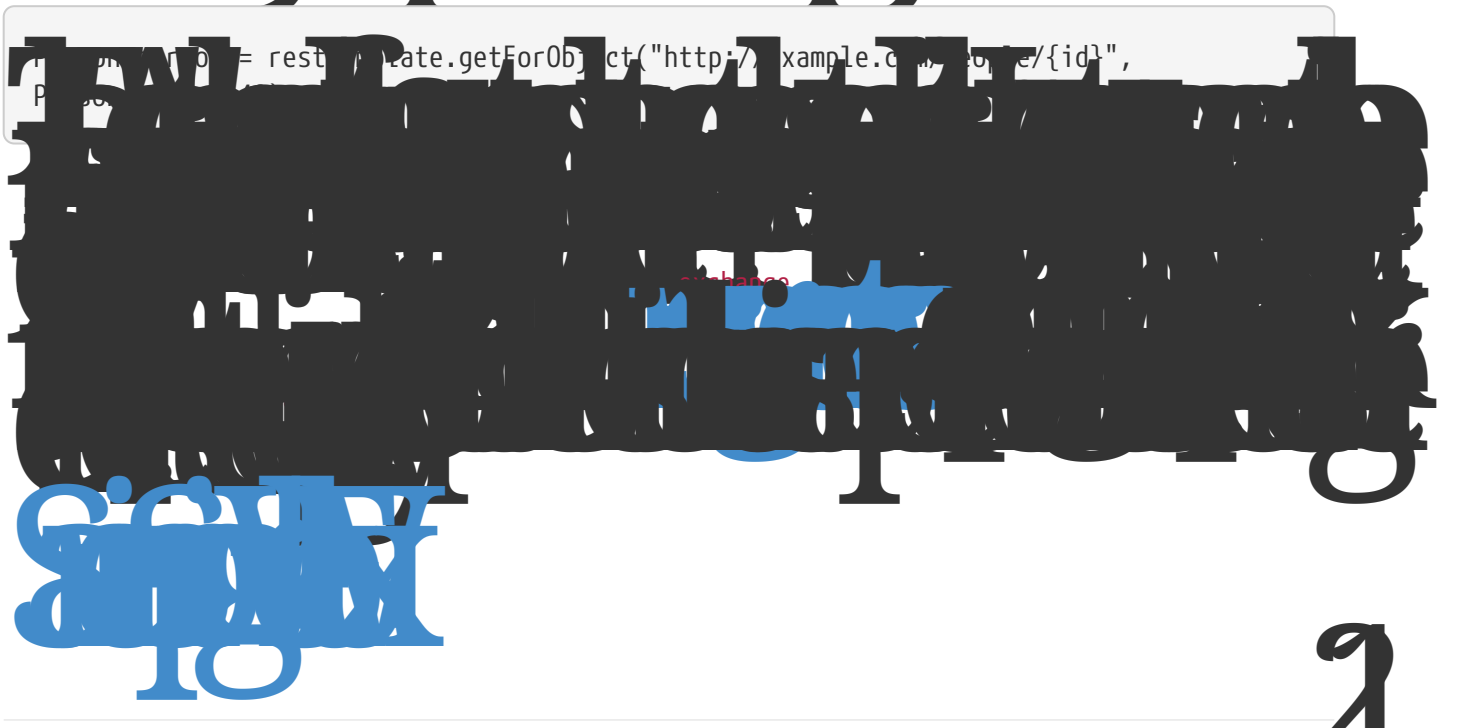
```
String result = restTemplate.getForObject(
        "http://example.com/hotels/{hotel}/bookings/{booking}", String.class, "42",

Map<String, String>
```

```
Map<String, String> vars = Collections.singletonMap("hotel", "42");

String result = restTemplate.getForObject(
        "http://example.com/hotels/{hotel}/bookings/{booking}", String.class, vars);
```

```
restTemplate.getForObject("http://example.com/hotel list", String.class);
```

```
exchange()
```

```
String uriTemplate = "http://example.com/hotels/{hotel}";
URI uri = UriComponentsBuilder.fromUriString(uriTemplate).build(42);

RequestEntity<Void> requestEntity = RequestEntity.get(uri)
        .header(("MyRequestHeader", "MyValue")
        .build();

ResponseEntity<String> response = template.exchange(requestEntity, String.class);

String responseHeader = response.getHeaders().getFirst("MyResponseHeader");
```



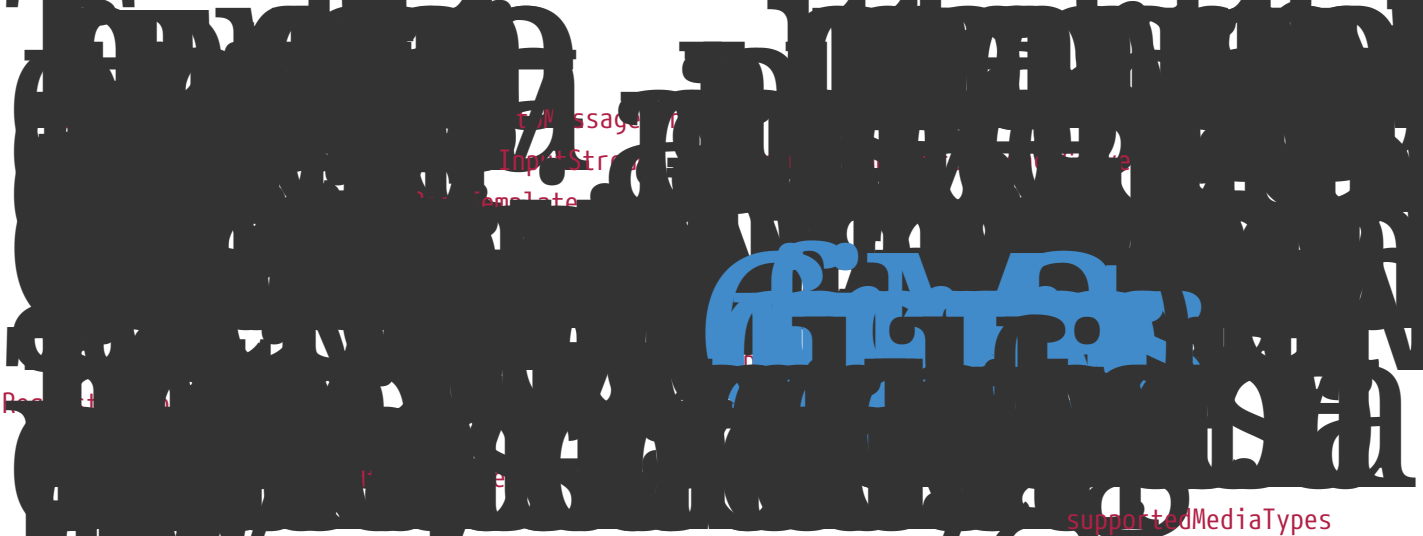
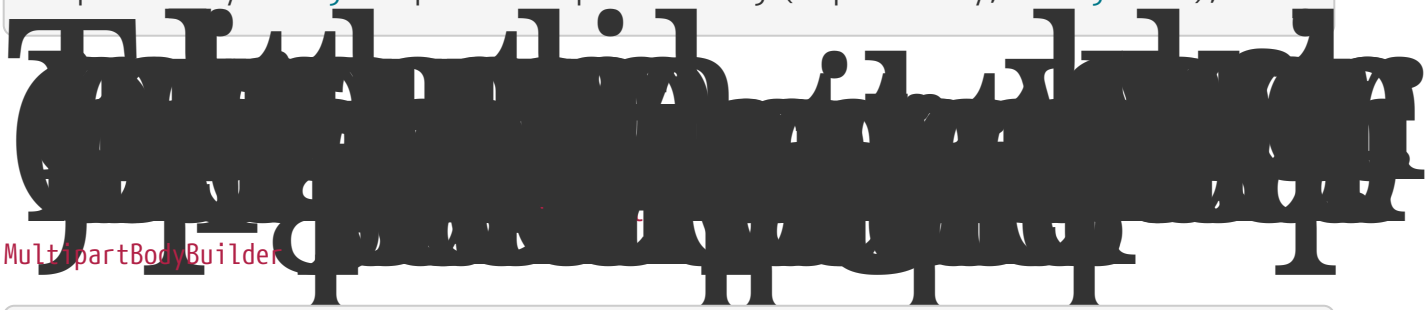


Table 2. HttpMessageConverter implementations

StringHttpMessageConverter	
FormHttpMessageConverter	
ByteArrayHttpMessageConverter	
MarshallingHttpMessageConverter	
MappingJackson2HttpMessageConverter	
MappingJackson2XmlHttpMessageConverter	application/xml

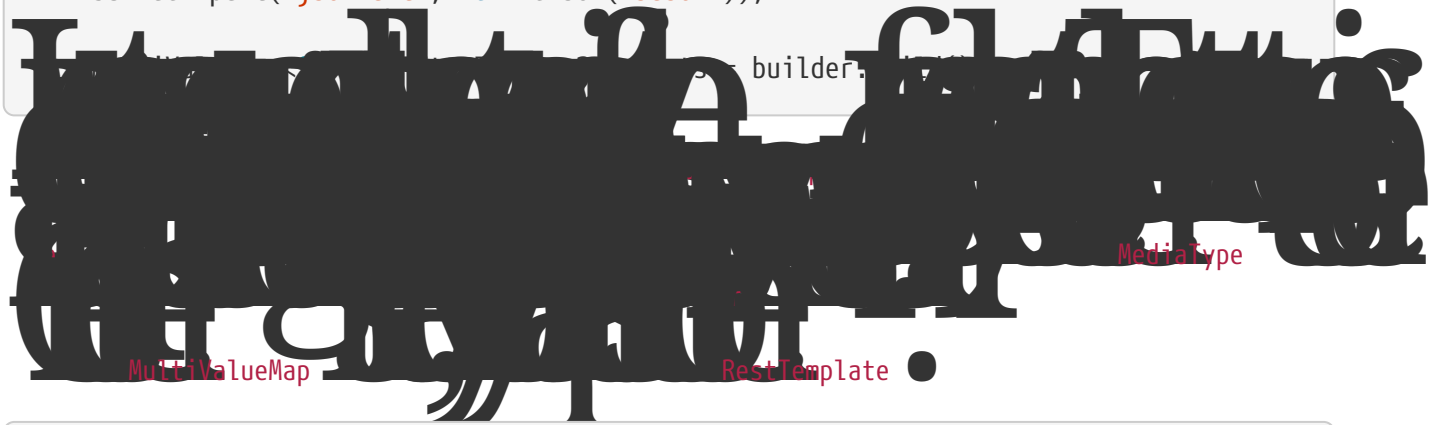
SourceHttpMessageConverter	
BufferedImageHttpMessageConverter	

```
MappingJacksonValue value = new MappingJacksonValue(new User("eric", "7!jd#h23"));
value.setSerializationView(User.WithoutPasswordView.class);

RequestEntity<MappingJacksonValue> requestEntity =
    RequestEntity.post(new URI("http://example.com/user")).body(value);

ResponseEntity<String> response = template.exchange(requestEntity, String.class);
```



MultipartBodyBuilder

```
    MultipartBodyBuilder builder = new MultipartBodyBuilder();
    builder.part("fieldPart", "fieldValue");
    builder.part("filePart", new FileSystemResource("...logo.png"));
    builder.part("jsonPart", new Person("Jason"));
```

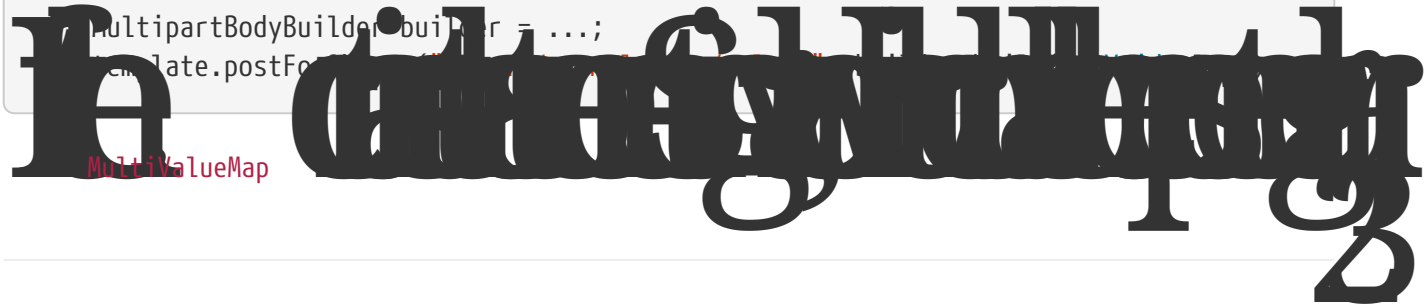


MediaType

MultiValueMap

RestTemplate

```
MultipartBodyBuilder builder = ...;
template.postFo
```



MultiValueMap


MultipartBodyBuilder
AsyncRestTemplate

[redacted]

MyComponent

```
public interface MyComponent {
```

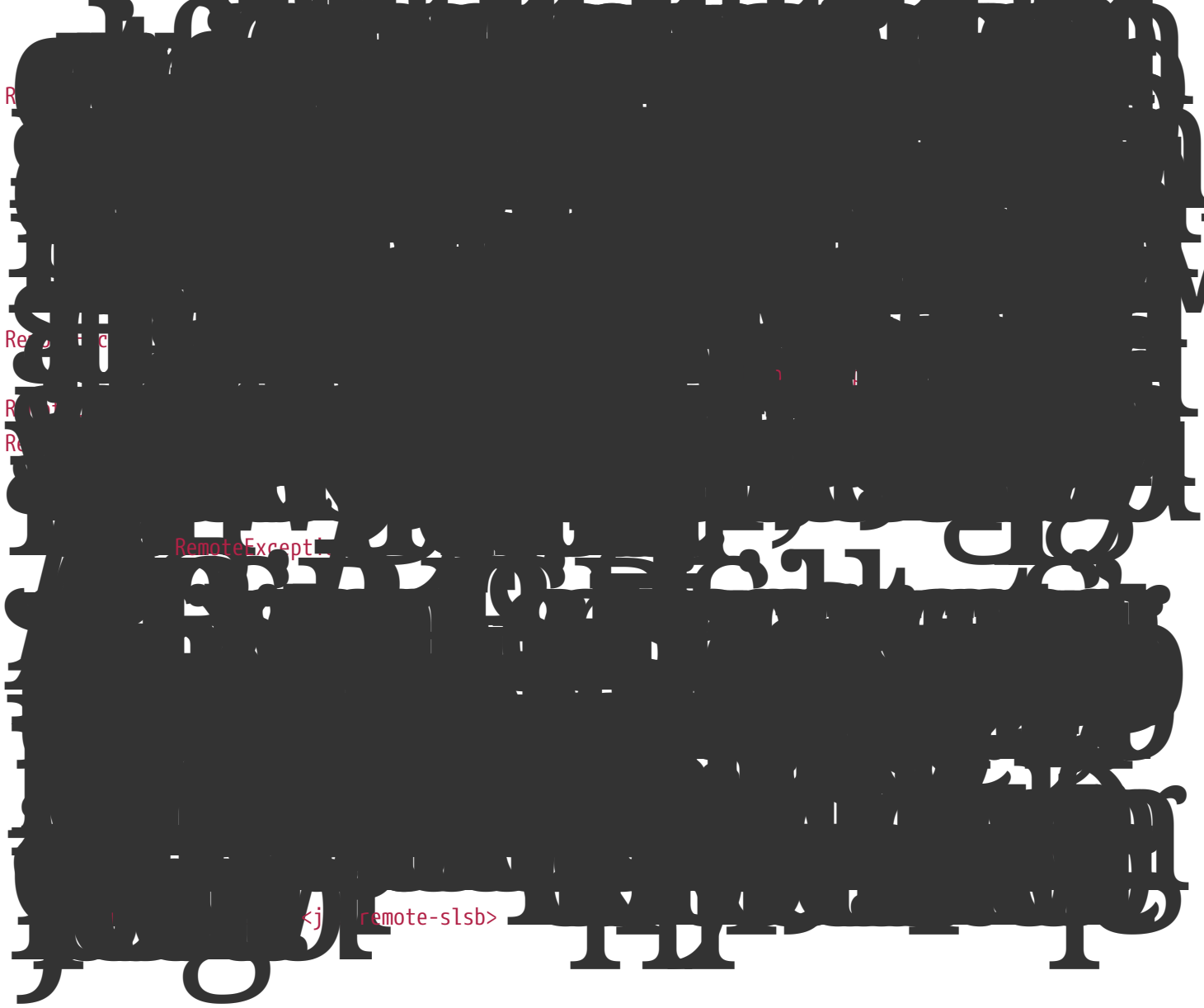
MyComponent

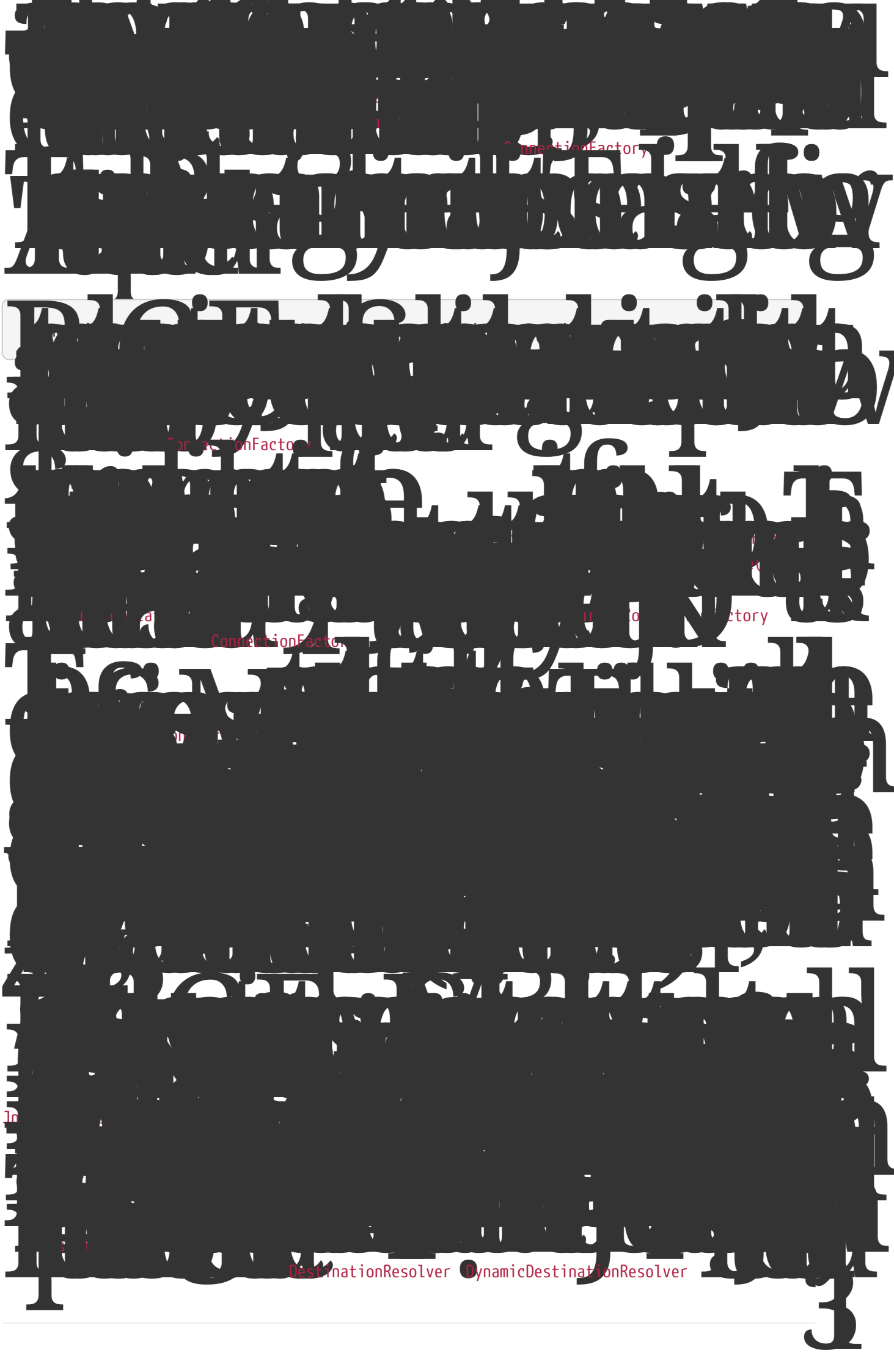
```
private MyComponent myComponent;

public void setMyComponent(MyComponent myComponent) {
    this.myComponent = myComponent;
```

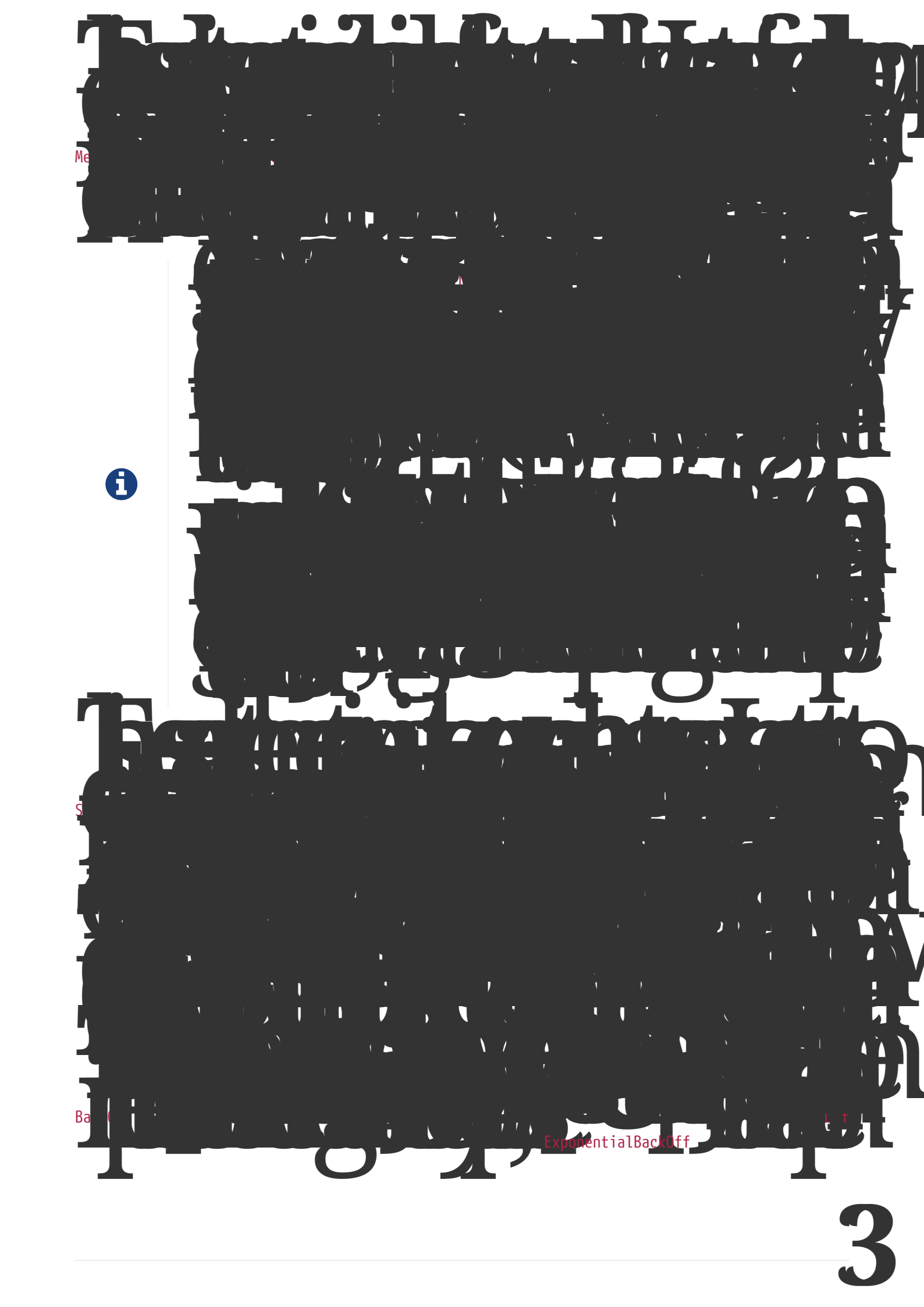
myComponent

```
<bean id="myComponent"
        class="org.springframework.ejb.access.LocalStatelessSessionProxyFactoryBean">
    <property name="jndiName" value="ejb/myBean"/>
    <property name="businessInterface" value="com.mycom.MyComponent"/>
</bean>

<bean id="myController" class="com.mycom.myController">
```

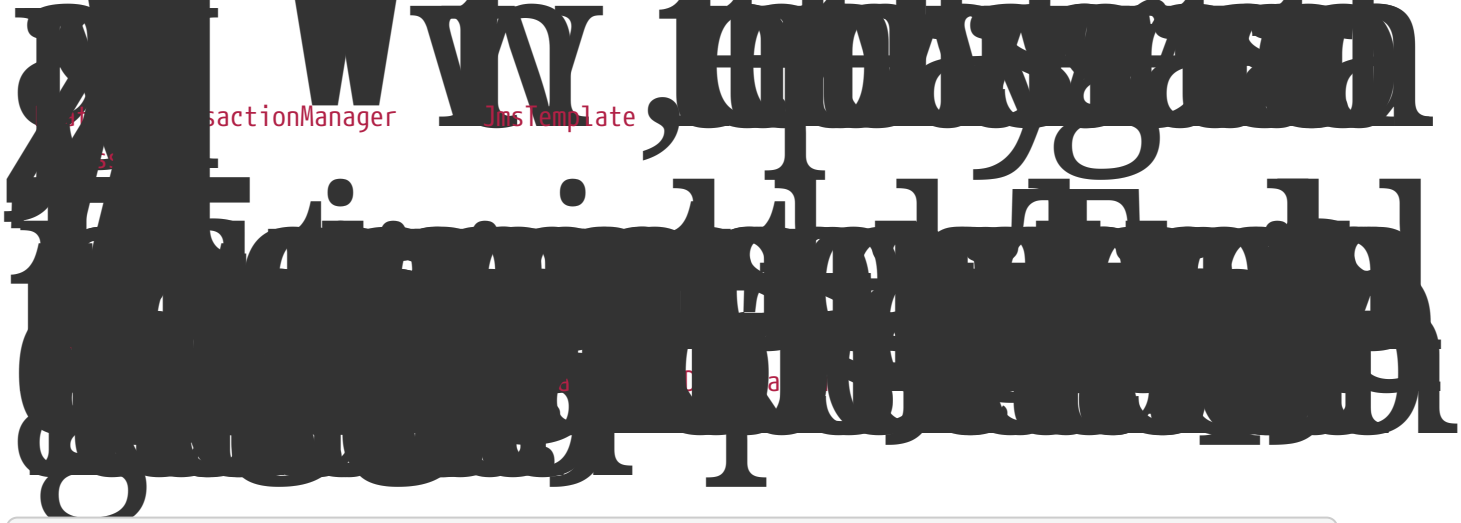
classname


RemoteExcept
<j remote-slsb>




ExponentialBackOff

sessionTransacted sessionAcknowledgeMode

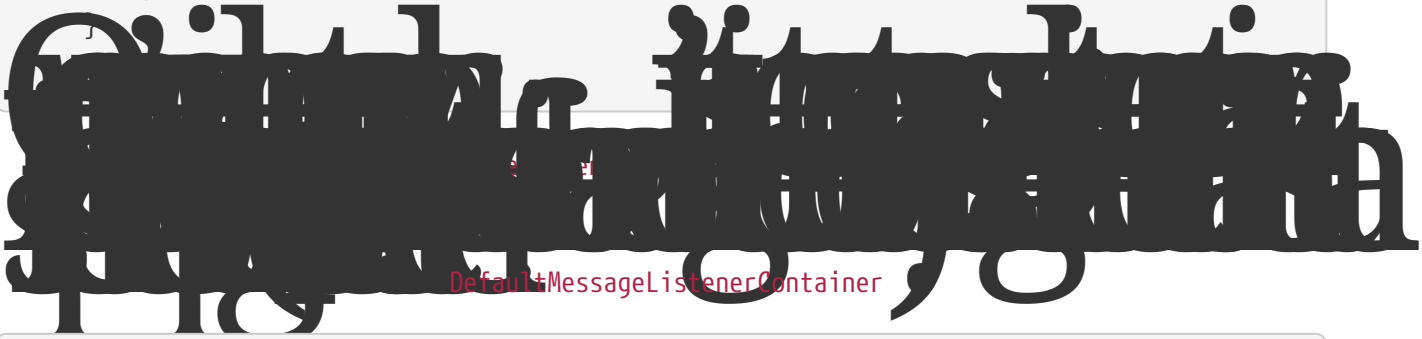


```
import javax.jms.ConnectionFactory;
import javax.jms.JMSException;
import javax.jms.Message;
import javax.jms.Queue;
import javax.jms.Session;

import org.springframework.jms.core.MessageCreator;
import org.springframework.jms.core.JmsTemplate;

public class JmsQueueSender {

    private JmsTemplate jmsTemplate;
    private Queue queue;

    public void setConnectionFactory(ConnectionFactory cf) {
        this.jmsTemplate = new JmsTemplate(cf);
    }

    public void setQueue(Queue queue) {
        this.queue = queue;
    }

    public void simpleSend() {
        this.jmsTemplate.send(this.queue, new MessageCreator() {
            public Message createMessage(Session session) throws JMSException {
                return session.createTextMessage("hello queue world");
            }
        });
    }
}
```


```
public void sendWithConversion() {
    Map map = new HashMap();
    map.put("Name", "Mark");
    map.put("Age", new Integer(47));
    jmsTemplate.convertAndSend("testQueue", map, new MessagePostProcessor() {
        public Message postProcessMessage(Message message) throws JMSException {
            message.setIntProperty("AccountID", 1234);
            message.setJMSCorrelationID("123-00001");
            return message;
        }
    });
}
```

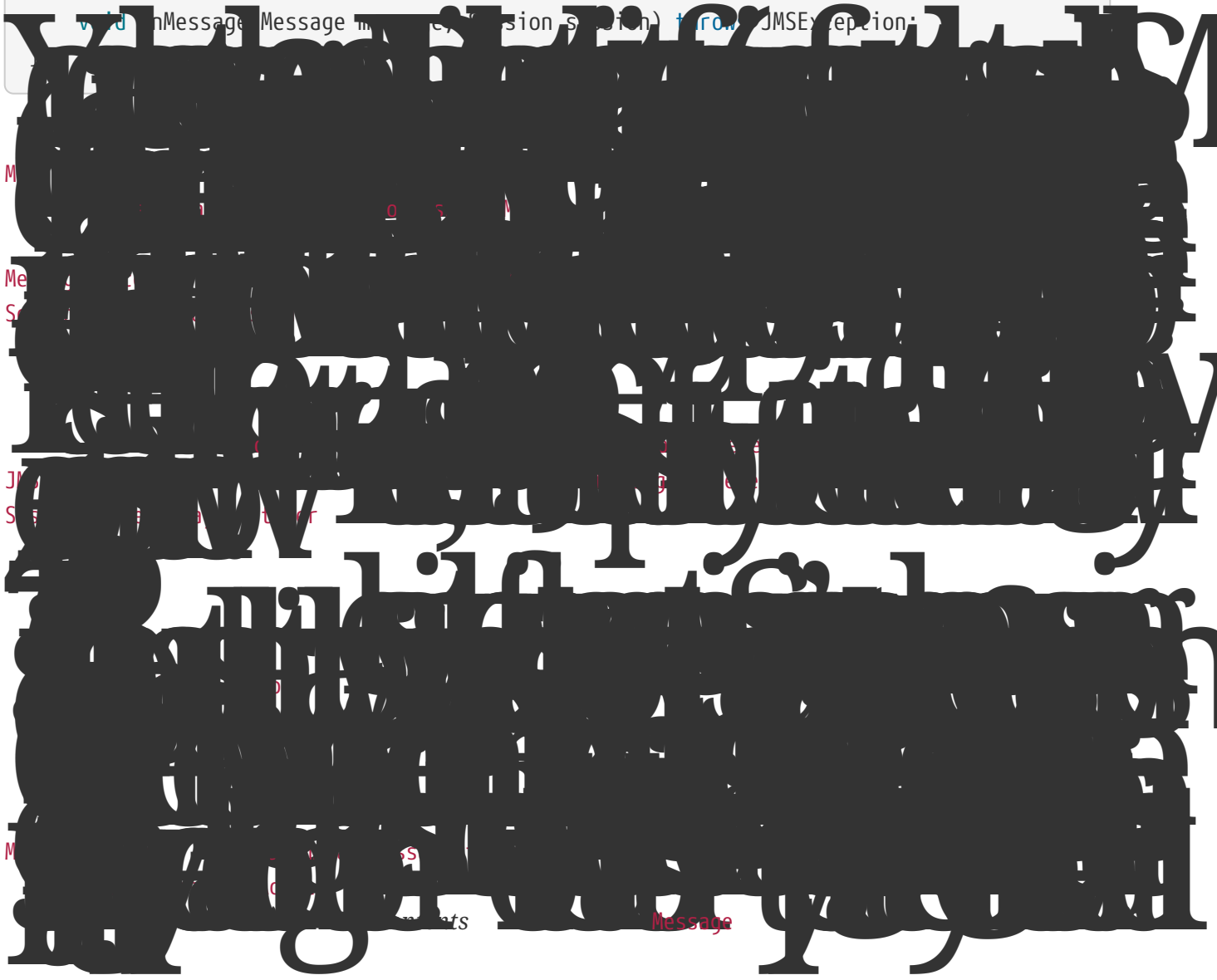
```
MapMessage={
    Header={
        ... standard headers ...
        CorrelationID={123-00001}
    }
    Properties={
        AccountID={Integer:1234}
    }
    Fields={
        Name={String:Mark}
        Age={Integer:47}
}
```

```
import javax.jms.JMSException;
import javax.jms.Message;
import javax.jms.MessageListener;
import javax.jms.TextMessage;

public class ExampleListener implements MessageListener {

    public void onMessage(Message message) {
        if (message instanceof TextMessage) {
            try {
                System.out.println(((TextMessage) message).getText());
            }
            catch (JMSException ex) {
                throw new RuntimeException(ex);
            }
        }
        else {
            throw new IllegalArgumentException("Message must be of type TextMessage");
        }
```



```
<!-- this is the Message Driven POJO (MDP) -->
<bean id="messageListener" class="jmsexample.ExampleListener"/>

<!-- and this is the message listener container -->
<bean id="jmsContainer" class=
"org.springframework.jms.listener.DefaultMessageListenerContainer">
    <property name="connectionFactory" ref="connectionFactory"/>
    <property name="destination" ref="destination"/>
```

```
package org.springframework.jms.listener;

public interface SessionAwareMessageListener {
```



```
public interface MessageDelegate {

    void handleMessage(String message);

    void handleMessage(Map message);

    void handleMessage(byte[] message);

    void handleMessage(Serializable message);
}
```

```
public class DefaultMessageDelegate implements MessageDelegate {
    // implementation elided for clarity...
}
```

```
<!-- this is the Message Driven POJO (MDP) -->
<strong><bean id="messageListener" class=
"org.springframework.jms.listener.adapter.MessageListenerAdapter">
    <constructor-arg>
        <bean class="jmsexample.DefaultMessageDelegate"/>
    </constructor-arg>
</bean></strong>

<!-- and this is the message listener container... -->
<bean id="jmsContainer" class=
"org.springframework.jms.listener.DefaultMessageListenerContainer">
    <property name="connectionFactory" ref="connectionFactory"/>
    <property name="destination" ref="destination"/>
```

```
public interface TextMessageDelegate {

    void receive(TextMessage message);
}
```

```
public class DefaultTextMessageDelegate implements TextMessageDelegate {
    // implementation elided for clarity...
}
```

```
<bean id="messageListener" class=
"org.springframework.jms.listener.adapter.MessageListenerAdapter">
    <constructor-arg>
        <bean class="jmsexample.DefaultTextMessageDelegate"/>
    </constructor-arg>
    <property name="defaultListenerMethod" value="receive"/>
    <!-- we don't want automatic message context extraction -->
    <property name="messageConverter">
        <null/>
    </proper
```

[illegible]

```
public interface ResponsiveTextMessageDelegate {

    // notice the return type...
    String receive(TextMessage message);
}
```

```
public class DefaultResponsiveTextMessageDelegate implements
ResponsiveTextMessageDelegate {
    // implementation elided for clarity...
```

[illegible]

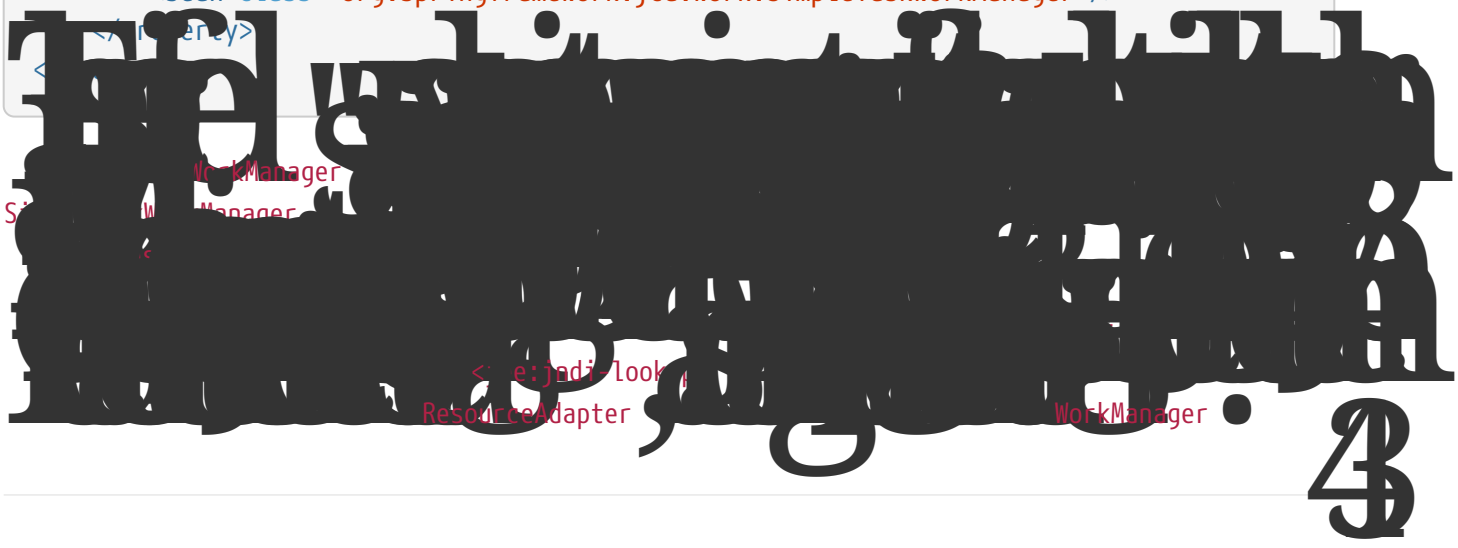
```
<bean id="jmsContainer" class=
"org.springframework.jms.listener.DefaultMessageListenerContainer">
    <property name="connectionFactory" ref="connectionFactory"/>
    <property name="destination" ref="destination"/>
    <property name="messageListener" ref="messageListener"/>
    <strong><property name="sessionTransacted" value="true"/></strong>
</bean>
```

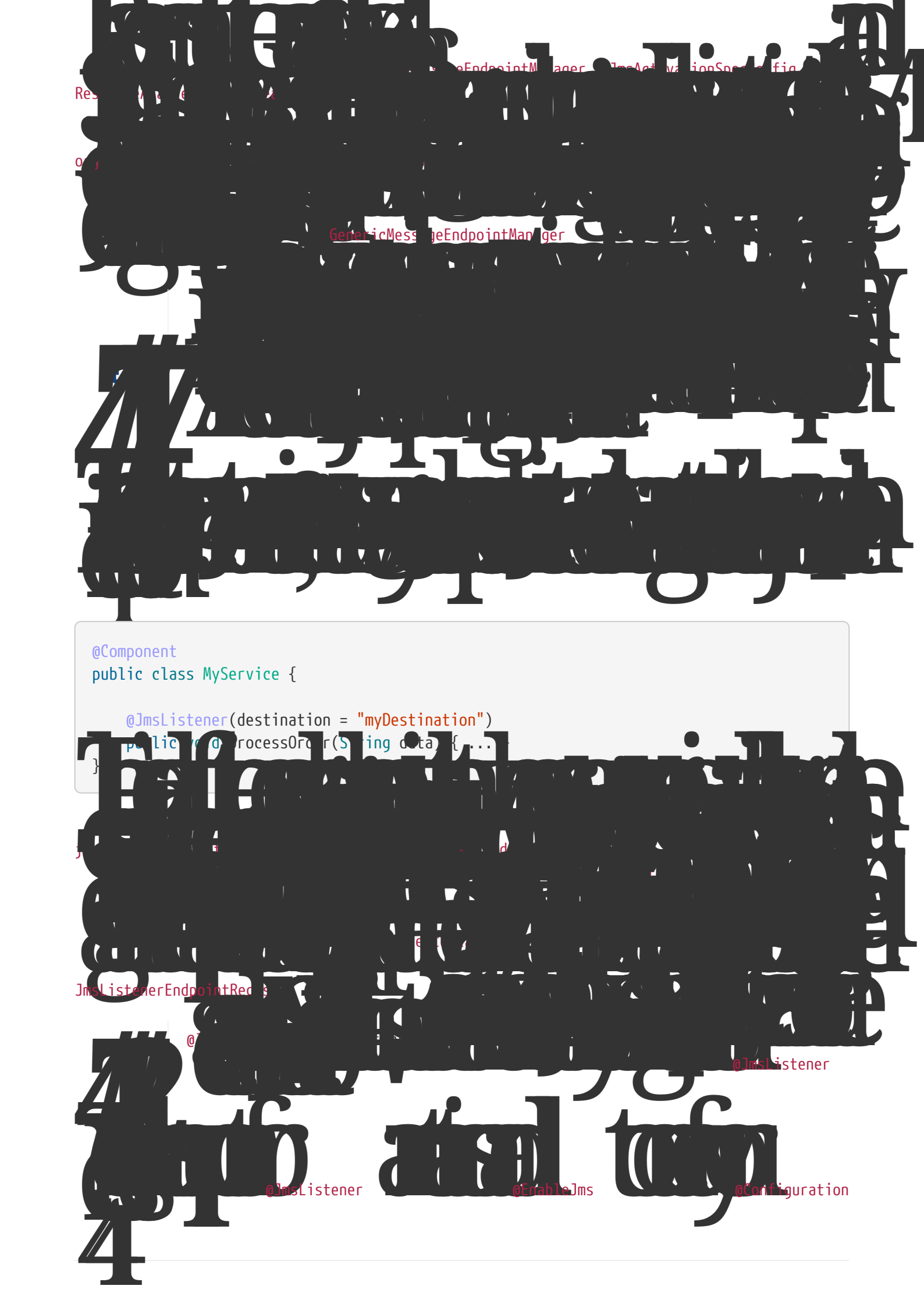
```
<bean id="jmsContainer" class=
"org.springframework.jms.listener.DefaultMessageListenerContainer">
    <property name="connectionFactory" ref="connectionFactory"/>
    <property name="destination" ref="destination"/>
    <property name="messageListener" ref="messageListener"/>
    <strong><property name="transactionManager" ref="transactionManager"/></strong>
</bean>
```

```
<bean class="org.springframework.jms.listener.endpoint.JmsMessageEndpointManager">
    <property name="resourceAdapter" ref="resourceAdapter"/>
    <property name="activationSpecConfig">
        <bean class="
org.springframework.jms.listener.endpoint.JmsActivationSpecConfig">
            <property name="destinationName" value="myQueue"/>
        </bean>
    </property>
    <property name="messageListener" ref="myMessageListener"/>
</bean>
```

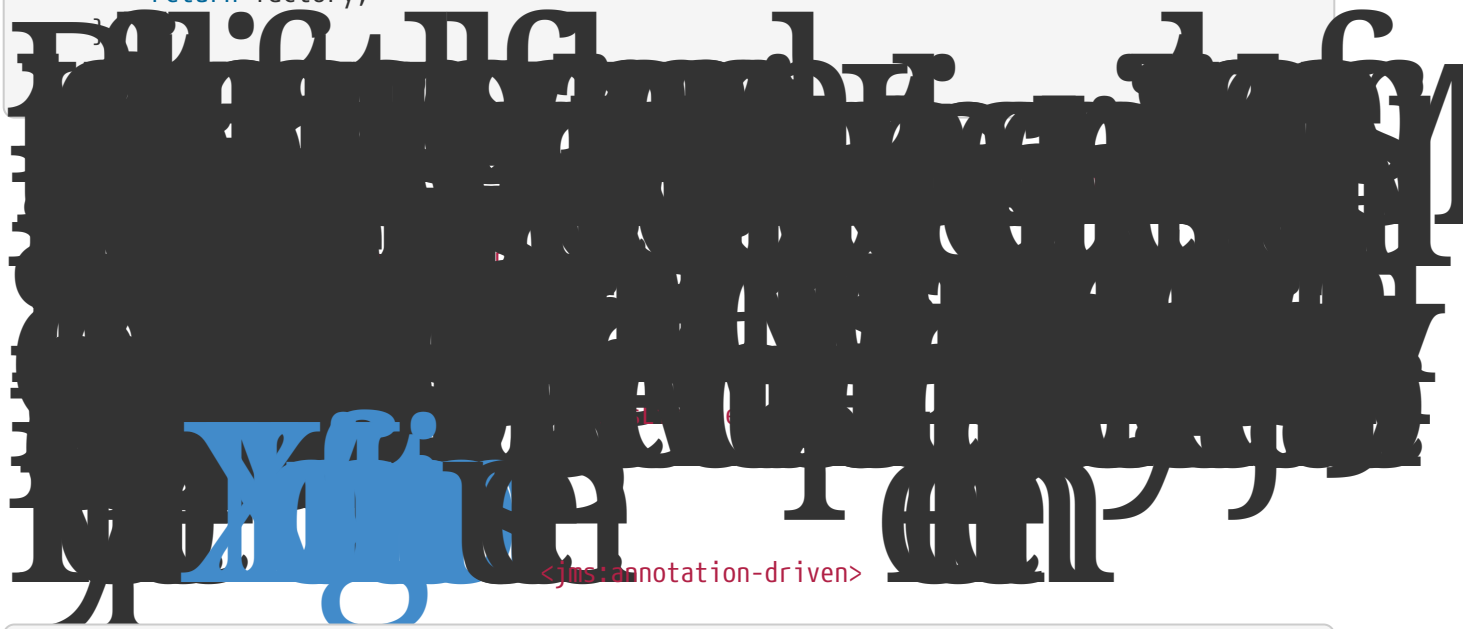
```
<bean class="org.springframework.jms.listener.endpoint.JmsMessageEndpointManager">
    <property name="resourceAdapter" ref="resourceAdapter"/>
    <property name="activationSpec">
        <bean class="org.apache.activemq.ra.ActiveMQActivationSpec">
            <property name="destination" value="myQueue"/>
            <property name="destinationType" value="javax.jms.Queue"/>
        </bean>
    </property>
    <property name="messageListener" ref="myMessageListener"/>
</bean>
```

```
<bean id="resourceAdapter" class=
"org.springframework.jca.support.ResourceAdapterFactoryBean">
    <property name="resourceAdapter">
        <bean class="org.apache.activemq.ra.ActiveMQResourceAdapter">
            <property name="serverUrl" value="tcp://localhost:61616"/>
        </bean>
    </property>
    <property name="workManager">
        <bean class="org.springframework.jca.work.SimpleTaskWorkManager"/>
    </property>
</bean>
```

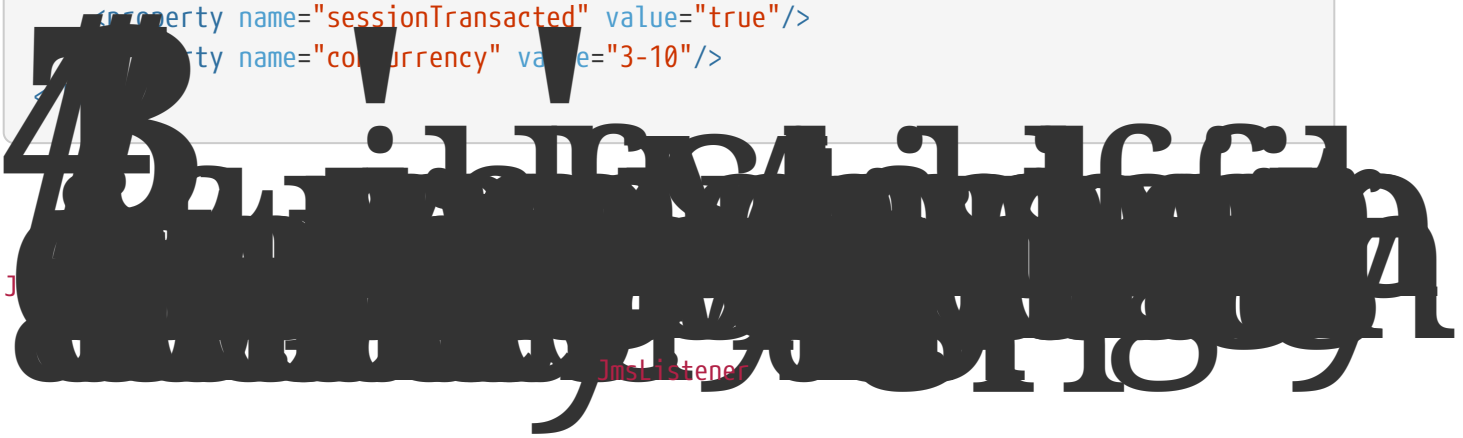




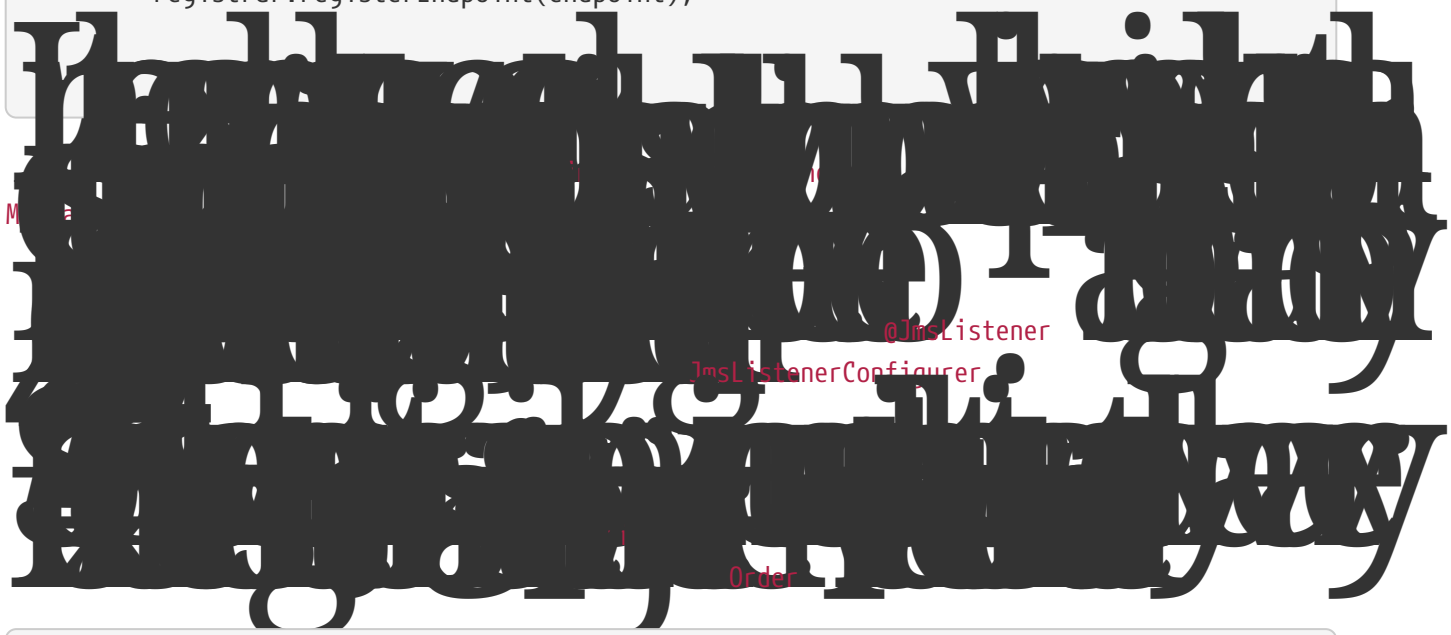
```
@Configuration
@EnableJms
public class AppConfig {

    @Bean
    public DefaultJmsListenerContainerFactory jmsListenerContainerFactory() {
        DefaultJmsListenerContainerFactory factory = new
DefaultJmsListenerContainerFactory();
        factory.setConnectionFactory(connectionFactory());
        factory.setDestinationResolver(destinationResolver());
        factory.setSessionTransacted(true);
        factory.setConcurrency("3-10");
        return factory;
```



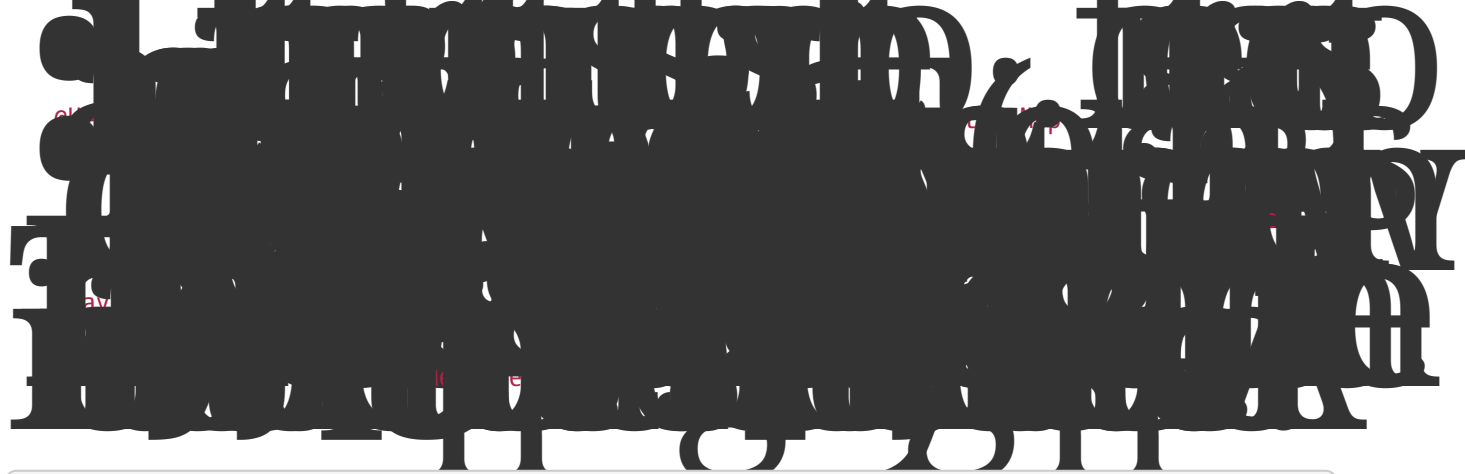
```
<jms:annotation-driven/>

<bean id="jmsListenerContainerFactory"
        class="org.springframework.jms.config.DefaultJmsListenerContainerFactory">
    <property name="connectionFactory" ref="connectionFactory"/>
    <property name="destinationResolver" ref="destinationResolver"/>
    <property name="sessionTransacted" value="true"/>
    <property name="concurrency" value="3-10"/>
</bean>
```

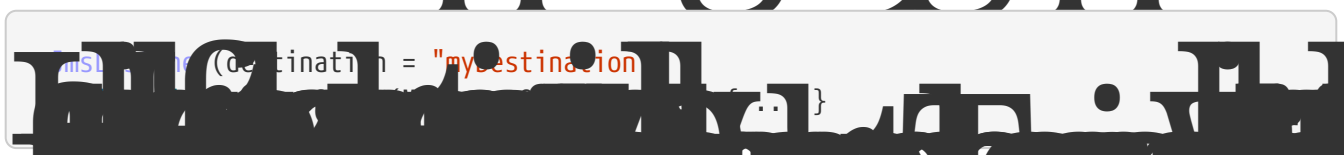


```java
@Configuration
@EnableJms
public class AppConfig implements JmsListenerConfigurer {

    @Override
    public void configureJmsListeners(JmsListenerEndpointRegistrar registrar) {
        SimpleJmsListenerEndpoint endpoint = new SimpleJmsListenerEndpoint();
        endpoint.setId("myJmsEndpoint");
        endpoint.setDestination("anotherQueue");
        endpoint.setMessageListener(message -> {
            // processing
        });
        registrar.registerEndpoint(endpoint);
```

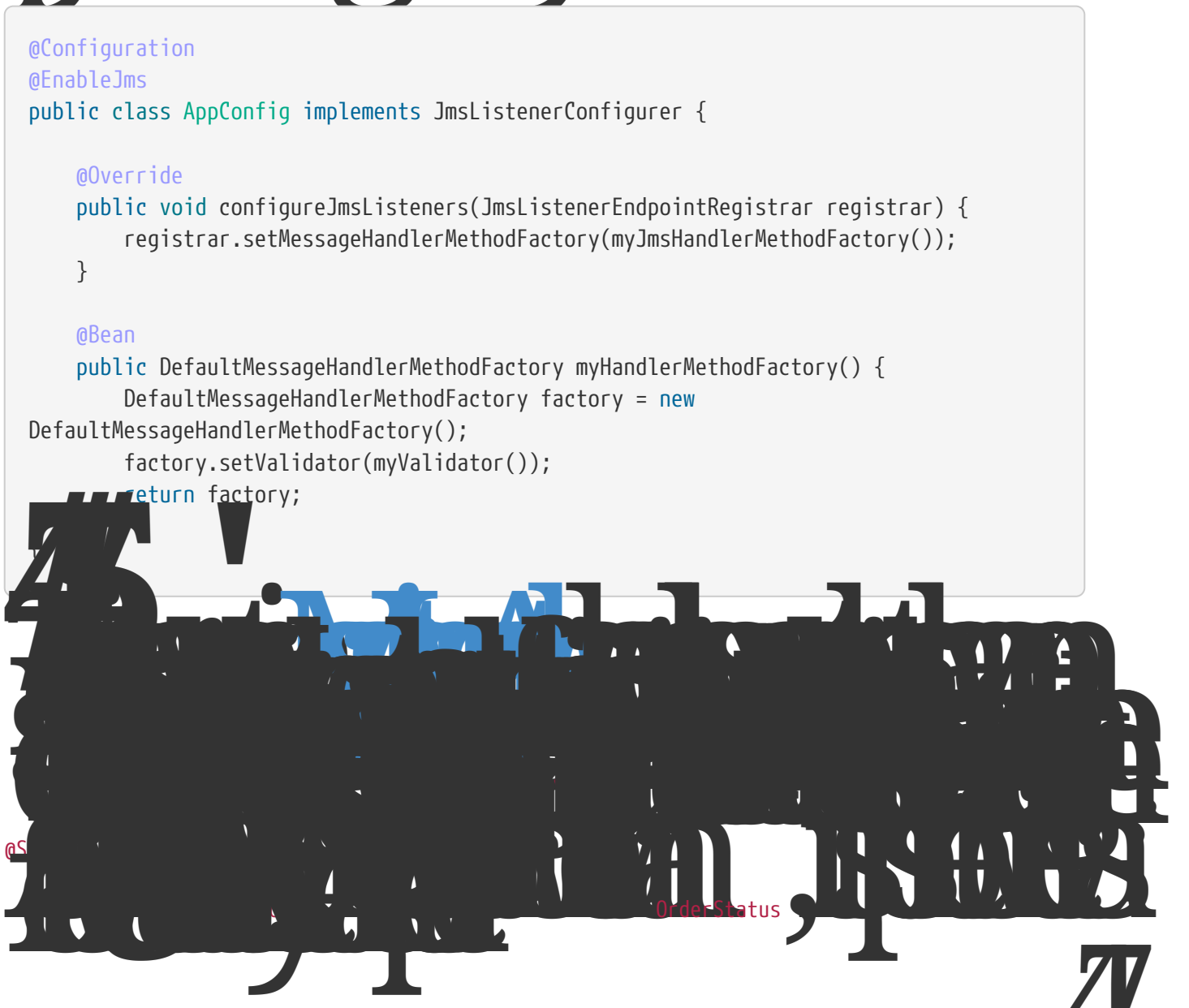


```java
@Component
public class MyService {

    @JmsListener(destination = "myDestination")
    public void processOrder(Order order, @Header("order_type") String orderType) {
        ...
}
```

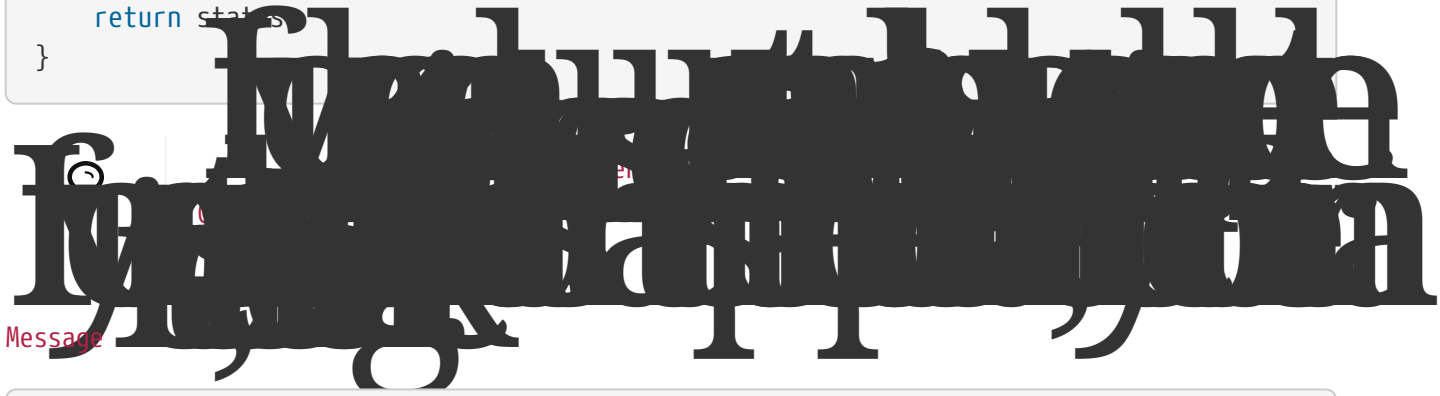




```
@Configuration
@EnableJms
public class AppConfig implements JmsListenerConfigurer {

    @Override
    public void configureJmsListeners(JmsListenerEndpointRegistrar registrar) {
        registrar.setMessageHandlerMethodFactory(myJmsHandlerMethodFactory());
    }

    @Bean
    public DefaultMessageHandlerMethodFactory myHandlerMethodFactory() {
        DefaultMessageHandlerMethodFactory factory = new
DefaultMessageHandlerMethodFactory();
        factory.setValidator(myValidator());
        return factory;
```



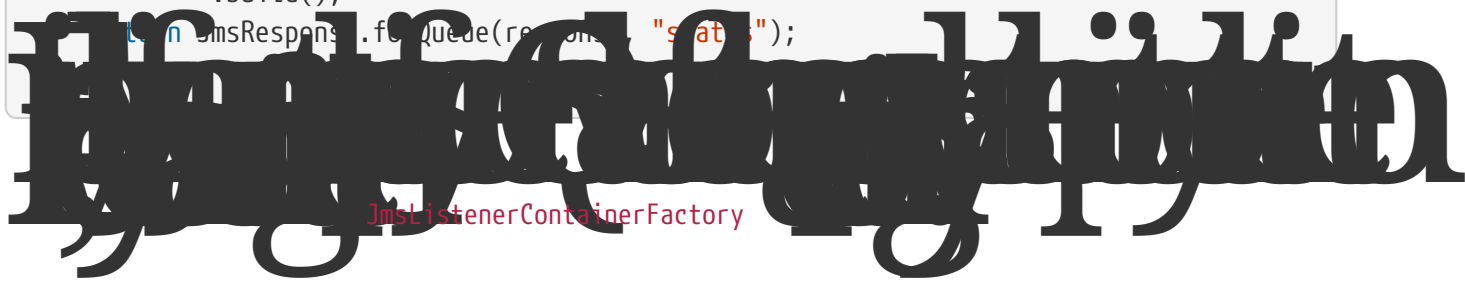
```
@JmsListener(destination = "myDestination")
@SendTo("status")
public OrderStatus processOrder(Order order) {
    // order processing
    return status
}
```



Message

```
@JmsListener(destination = "myDestination")
@SendTo("status")
public Message<OrderStatus> processOrder(Order order) {
    // order processing
    return MessageBuilder
            .withPayload(status)
            .setHeader("code", 1234)
            .build();
```

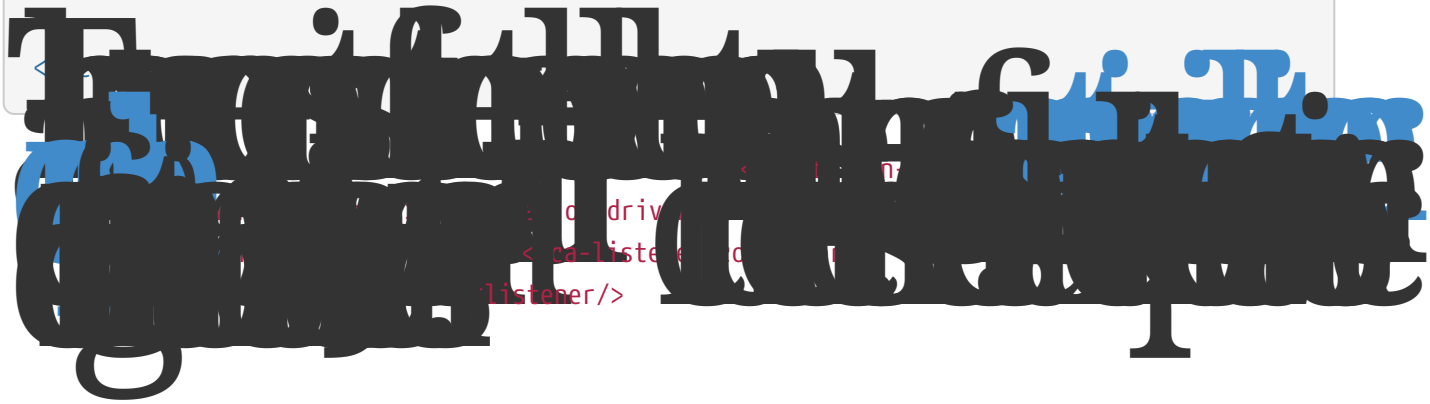
```
@JmsListener(destination = "myDestination")
public JmsResponse<Message<OrderStatus>> processOrder(Order order) {
    // order processing
    Message<OrderStatus> response = MessageBuilder
            .withPayload(status)
            .setHeader("code", 1234)
            .build();
    return JmsResponse.forQueue(response, "status");
```



JmsListenerContainerFactory

```
@Configuration
@EnableJms
public class AppConfig {

    @Bean
    public DefaultJmsListenerContainerFactory jmsListenerContainerFactory() {
        DefaultJmsListenerContainerFactory factory = new
DefaultJmsListenerContainerFactory();
        factory.setConnectionFactory(connectionFactory());
        QosSettings replyQosSettings = new QosSettings();
        replyQosSettings.setPriority(2);
        replyQosSettings.setTimeToLive(10000);
        [illegible].setReplyQosSet[illegible]s(replyQosSettings);
        [illegible] factory;
```

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

    <!-- bean definitions here -->
```



```
<jms:listener-container>

    <jms:listener destination="queue.orders" ref="orderService" method="placeOrder"/>

    <jms:listener destination="queue.confirmations" ref="confirmationLogger" method=
"log"/>

</jms:listener-container>
```

[illegible]

Table 2. Attributes of the JMS [illegible] container

[illegible]

`DestinationResolver`

[illegible]

`...nerContainerFactory`

`factory-id`

[illegible]

```
<jms:listener-container connection-factory="myConnectionFactory"
        task-executor="myTaskExecutor"
        destination-resolver="myDestinationResolver"
        transaction-manager="myTransactionManager"
        concurrency="10">

    <jms:listener destination="queue.orders" ref="orderService" method="placeOrder"/>

    <jms:listener destination="queue.confirmations" ref="confirmationLogger" method=
"log"/>
```

```
<jms:jca-listener-container resource-adapter="myResourceAdapter"
        destination-resolver="myDestinationResolver"
        transaction-manager="myTransactionManager"
        concurrency="10">

    <jms:listener destination="queue.orders" ref="myMessageListener"/>
</jms:jca-listener-container>
```

Publishing


MBeanServer

```
package org.springframework.jmx;

public class JmxTestBean implements IJmxTestBean {

    private String name;
    private int age;
    private boolean isSuperman;

    public int getAge() {
        return age;
    }

    public void setAge(int age) {
        this.age = age;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getName() {
        return name;
    }

    public int add(int x, int y) {
        return x + y;
    }

    public void dontExposeMe() {
        throw new RuntimeException();
    }
}
```

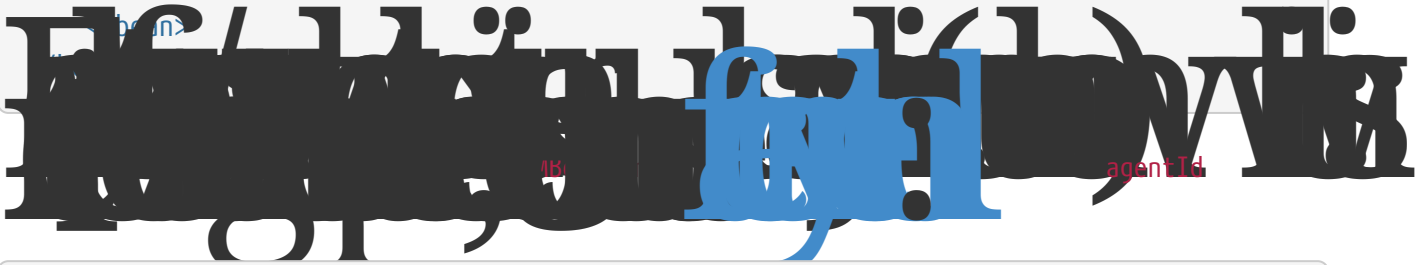
MBeanExporter

```
<beans>
    <!-- this bean must not be lazily initialized if the exporting is to happen -->
    <bean id="exporter" class="org.springframework.jmx.export.MBeanExporter" lazy-
init="false">
        <property name="beans">
            <map>
                <entry key="bean:name=testBean1" value-ref="testBean"/>
            </map>
        </property>
    </bean>
    <bean id="testBean" class="org.springframework.jmx.JmxTestBean">
        <property name="name" value="TEST"/>
        <property name="age" value="100"/>
    </bean>
```

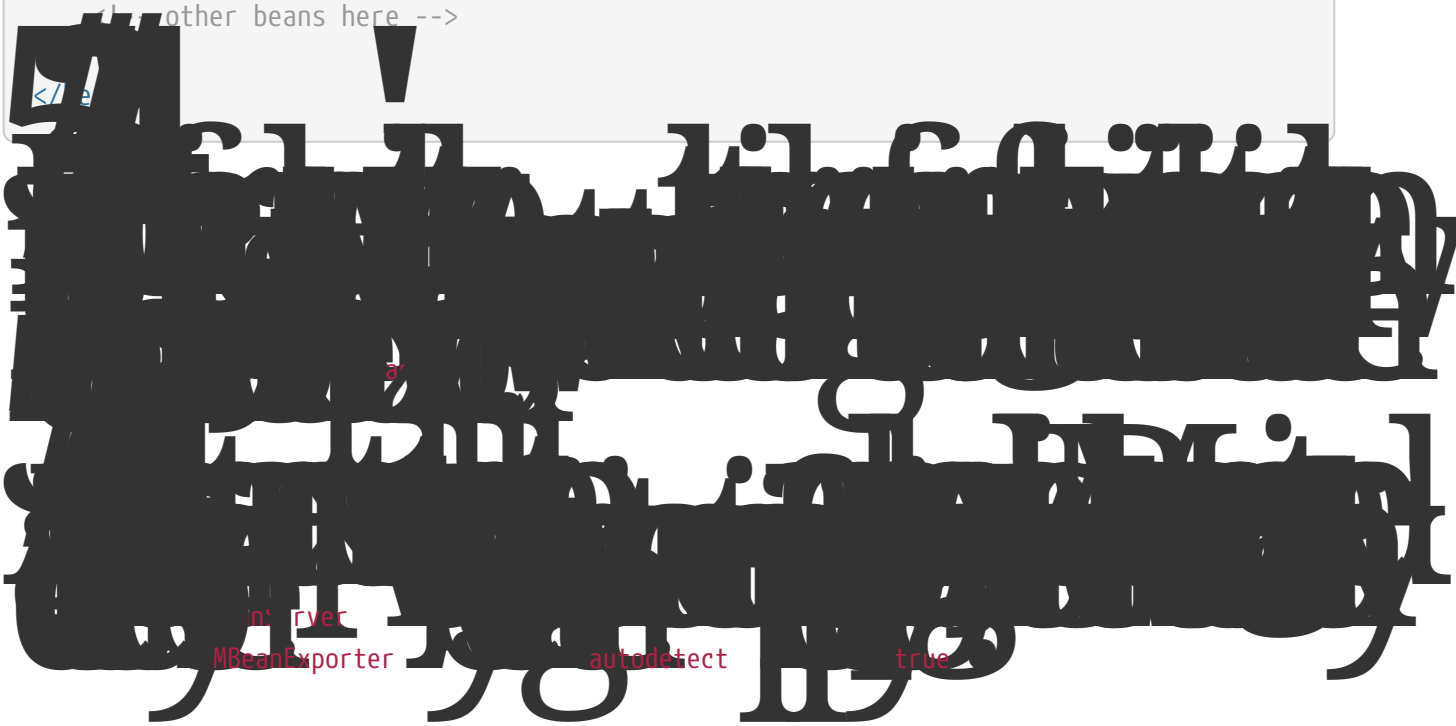
```
<beans>

    <bean id="mbeanServer" class=
"org.springframework.jmx.support.MBeanServerFactoryBean"/>

    <!--
    this bean needs to be eagerly pre-instantiated in order for the exporting to
occur;
    this means that it must not be marked as lazily initialized
    -->
    <bean id="exporter" class="org.springframework.jmx.export.MBeanExporter">
        <property name="beans">
            <map>
                <entry key="bean:name=testBean1" value-ref="testBean"/>
            </map>
        </property>
        <property name="server" ref="mbeanServer"/>
    </bean>

    <bean id="testBean" class="org.springframework.jmx.JmxTestBean">
        <property name="name" value="TEST"/>
        <property name="age" value="100"/>
    </bean>
```

MBeanServer agentId

```
<beans>
    <bean id="mbeanServer" class=
"org.springframework.jmx.support.MBeanServerFactoryBean">
        <!-- indicate to first look for a server -->
        <property name="locateExistingServerIfPossible" value="true"/>
        <!-- search for the MBeanServer instance with the given agentId -->
        <property name="agentId" value="MBeanServer_instance_agentId>"/>
    </bean>
    <bean id="exporter" class="org.springframework.jmx.export.MBeanExporter">
        <property name="server" ref="mbeanServer"/>
        ...
    </bean>
</beans>
```



```
<beans>
    <bean id="exporter" class="org.springframework.jmx.export.MBeanExporter">
        <property name="server">
            <!-- Custom MBeanServerLocator -->
            <bean class="platform.package.MBeanServerLocator" factory-method=
"locateMBeanServer"/>
        </property>
    </bean>

    <!-- other beans here -->

</beans>
```

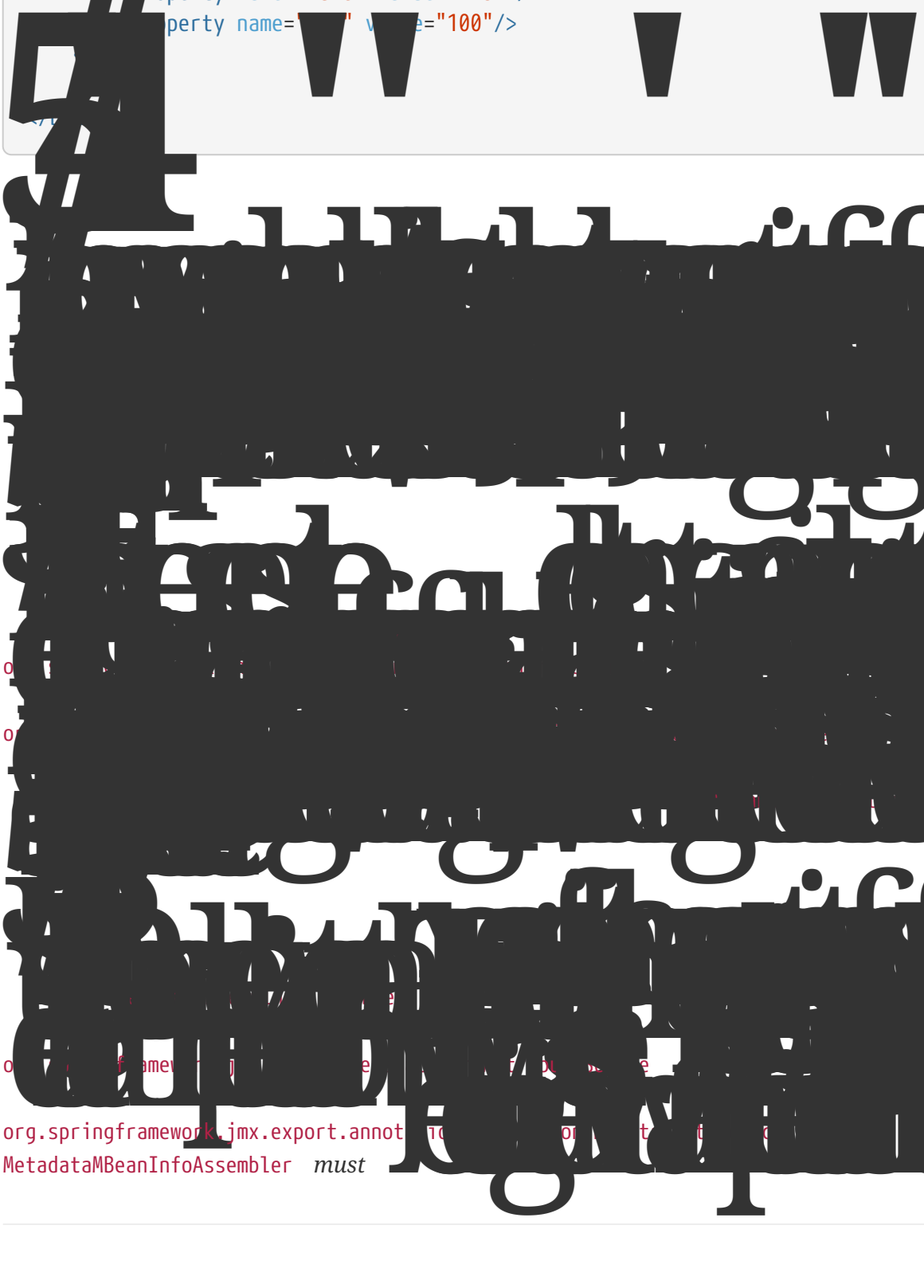


```
<bean id="exporter" class="org.springframework.jmx.export.MBeanExporter">
    <property name="autodetect" value="true"/>
</bean>

<bean id="spring:mbean=true" class="org.springframework.jmx.export.TestDynamicMBean"/>
```

Table 6. Registration Be

FAIL_ON_EXISTING
IGNORE_EXISTING
REPLACE_EXISTING

```
<beans>

    <bean id="exporter" class="org.springframework.jmx.export.MBeanExporter">
        <property name="beans">
            <map>
                <entry key="bean:name=testBean1" value-ref="testBean"/>
            </map>
        </property>
        <property name="registrationPolicy" value="REPLACE_EXISTING"/>
    </bean>

    <bean id="testBean" class="org.springframework.jmx.JmxTestBean">
        <property name="name" value="TEST"/>
        <property name="age" value="100"/>
```



org.springframework.jmx.export.annot
MetadataMBeanInfoAssembler *must*

JmxTestBean

```
package org.springframework.jmx;

import org.springframework.jmx.export.annotation.ManagedResource;
import org.springframework.jmx.export.annotation.ManagedOperation;
import org.springframework.jmx.export.annotation.ManagedAttribute;

@ManagedResource(
        objectName="bean:name=testBean4",
        description="My Managed Bean",
        log=true,
        logFile="jmx.log",
        currencyTimeLimit=15,
        persistPolicy="OnUpdate",
        persistPeriod=200,
        persistLocation="foo",
        persistName="bar")
public class AnnotationTestBean implements IJmxTestBean {

    private String name;
    private int age;

    @ManagedAttribute(description="The Age Attribute", currencyTimeLimit=15)
    public int getAge() {
        return age;
    }

    public void setAge(int age) {
        this.age = age;
    }

    @ManagedAttribute(description="The Name Attribute",
            currencyTimeLimit=20,
            defaultValue="bar",
            persistPolicy="OnUpdate")
    public void setName(String name) {
        this.name = name;
```

```
    @ManagedAttribute(defaultValue="foo", persistPeriod=300)
    public String getName() {
        return name;
    }

    @ManagedOperation(description="Add two numbers")
    @ManagedOperationParameters({
        @ManagedOperationParameter(name = "x", description = "The first number"),
        @ManagedOperationParameter(name = "y", description = "The second number")})
    public int add(int x, int y) {
        return x + y;
    }

    public void dontExposeMe() {
        throw new RuntimeException();
    }
```

```
<beans>
    <bean id="exporter" class="org.springframework.jmx.export.MBeanExporter">
        <property name="assembler" ref="assembler"/>
        <property name="namingStrategy" ref="namingStrategy"/>
        <property name="autodetect" value="true"/>
    </bean>

    <bean id="jmxAttributeSource"
            class=
"org.springframework.jmx.export.annotation.AnnotationJmxAttributeSource"/>

    <!-- will create management interface using annotation metadata -->
    <bean id="assembler"
            class="
org.springframework.jmx.export.assembler.MetadataMBeanInfoAssembler">
        <property name="attributeSource" ref="jmxAttributeSource"/>
    </bean>

    <!-- will pick up the ObjectName from the annotation -->
    <bean id="namingStrategy"
            class="org.springframework.jmx.export.naming.MetadataNamingStrategy">
        <property name="attributeSource" ref="jmxAttributeSource"/>
    </bean>

    <bean id="testBean" class="org.springframework.jmx.AnnotationTestBean">
        <property name="name" value="TEST"/>
        <property name="age" value="100"/>
    </bean>
```

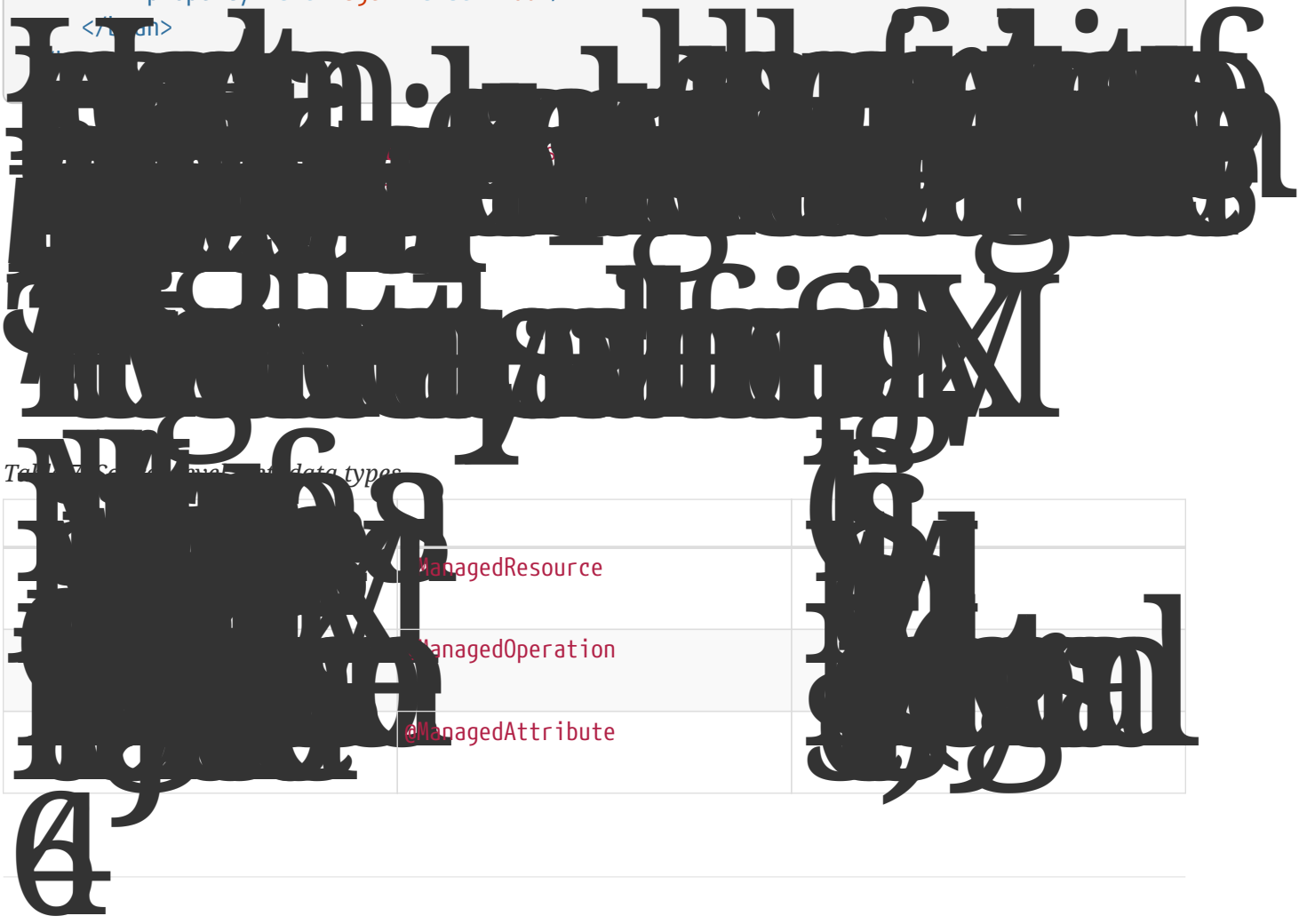


Table of annotation types

	@ManagedResource	
	@ManagedOperation	
	@ManagedAttribute	

	@ManagedOperationParameter

Table 8. Source-level metadata annotations

ObjectName		ManagedResource
description		ManagedResource ManagedAttribute ManagedOperation ManagedOperationParameter
currencyTimeLimit		ManagedResource ManagedAttribute
defaultValue		ManagedAttribute
log		ManagedResource
logFile		ManagedResource
persistPolicy		ManagedResource
persistPeriod		ManagedResource
persistLocation		ManagedResource
persistName		ManagedResource
name		ManagedOperationParameter
index		ManagedOperationParameter

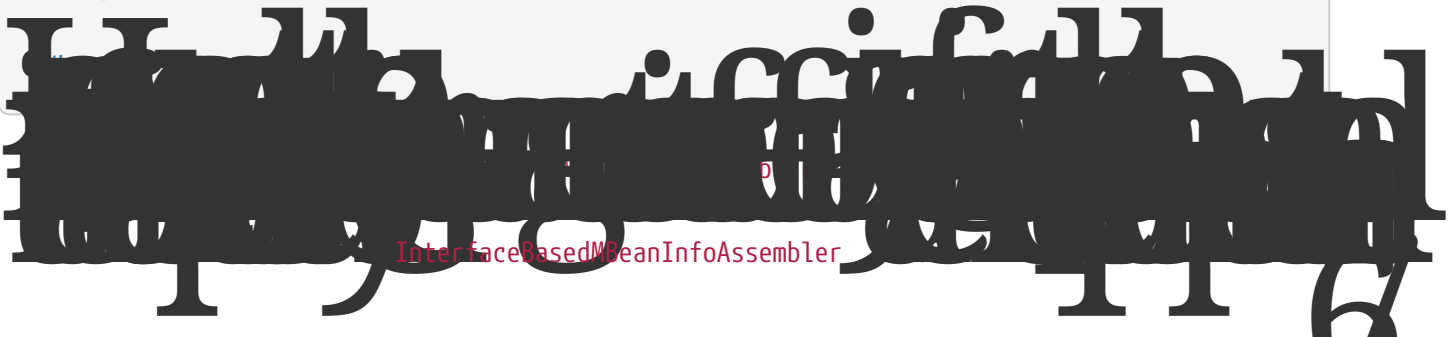
MetadataMBean
ManagedResource
ObjectName

```
<beans>

    <bean id="exporter" class="org.springframework.jmx.export.MBeanExporter">
        <!-- notice how no 'beans' are explicitly configured here -->
        <property name="autodetect" value="true"/>
        <property name="assembler" ref="assembler"/>
    </bean>

    <bean id="testBean" class="org.springframework.jmx.JmxTestBean">
        <property name="name" value="TEST"/>
        <property name="age" value="100"/>
    </bean>

    <bean id="assembler" class=
"org.springframework.jmx.export.assembler.MetadataMBeanInfoAssembler">
        <property name="attributeSource">
            <bean class=
"org.springframework.jmx.export.annotation.AnnotationJmxAttributeSource"/>
        </property>
    </bean>
```

6

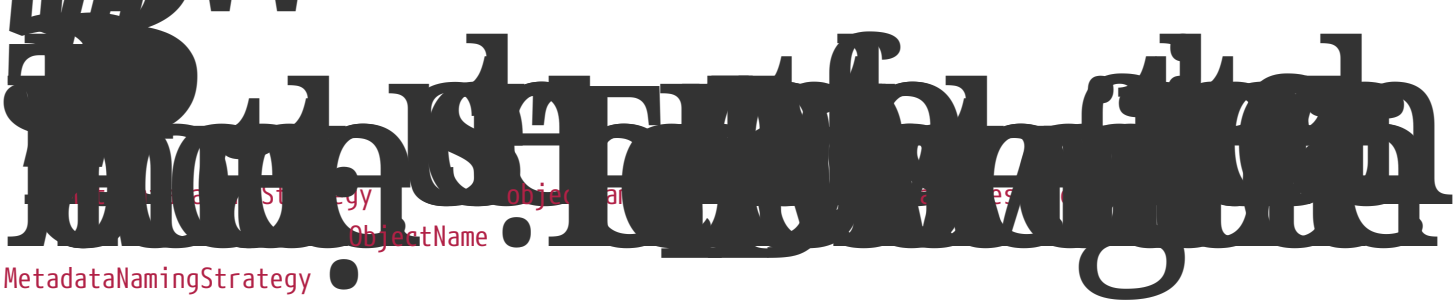
```java
public interface IJmxTestBean {

    public int add(int x, int y);

    public long myOperation();

    public int getAge();

    public void setAge(int age);

    public void setName(String name);

    public String getName();
}
```

```xml
<beans>

    <bean id="exporter" class="org.springframework.jmx.export.MBeanExporter">
        <property name="beans">
            <map>
                <entry key="bean:name=testBean5" value-ref="testBean"/>
            </map>
        </property>
        <property name="assembler">
            <bean class=
"org.springframework.jmx.export.assembler.InterfaceBasedMBeanInfoAssembler">
                <property name="managedInterfaces">
                    <value>org.springframework.jmx.IJmxTestBean</value>
                </property>
            </bean>
        </property>
    </bean>

    <bean id="testBean" class="org.springframework.jmx.JmxTestBean">
        <property name="name" value="TEST"/>
        <property name="age" value="100"/>
    </bean>
```



```
<bean id="exporter" class="org.springframework.jmx.export.MBeanExporter">
    <property name="beans">
        <map>
            <entry key="bean:name=testBean5" value-ref="testBean"/>
        </map>
    </property>
    <property name="assembler">
        <bean class=
"org.springframework.jmx.export.assembler.MethodNameBasedMBeanInfoAssembler">
            <property name="managedMethods">
                <value>add,myOperation,getName,setName,getAge</value>
            </property>
        </bean>
    </property>
```

```
<beans>

    <bean id="exporter" class="org.springframework.jmx.export.MBeanExporter">
        <property name="beans">
            <map>
                <entry key="testBean" value-ref="testBean"/>
            </map>
        </property>
        <property name="namingStrategy" ref="namingStrategy"/>
    </bean>

    <bean id="testBean" class="org.springframework.jmx.JmxTestBean">
        <property name="name" value="TEST"/>
        <property name="age" value="100"/>
    </bean>

    <bean id="namingStrategy" class=
"org.springframework.jmx.export.naming.KeyNamingStrategy">
        <property name="mappings">
            <props>
                <prop key="testBean">bean:name=testBean1</prop>
            </props>
        </property>
        <property name="mappingLocations">
            <value>names1.properties,names2.properties</value>
        </property>
    </bean>
```



ObjectName

MetadataNamingStrategy

```
<beans>

    <bean id="exporter" class="org.springframework.jmx.export.MBeanExporter">
        <property name="beans">
            <map>
                <entry key="testBean" value-ref="testBean"/>
            </map>
        </property>
        <property name="namingStrategy" ref="namingStrategy"/>
    </bean>

    <bean id="testBean" class="org.springframework.jmx.JmxTestBean">
        <property name="name" value="TEST"/>
        <property name="age" value="100"/>
    </bean>

    <bean id="namingStrategy" class=
"org.springframework.jmx.export.naming.MetadataNamingStrategy">
        <property name="attributeSource" ref="attributeSource"/>
    </bean>

    <bean id="attributeSource"
            class=
"org.springframework.jmx.export.annotation.AnnotationJmxAttributeSource"/>
```

ObjectName

cl snam e],name=[bean-name]

foo:type=MyClass,name=myBean

```
<bean id="myBean" class="com.foo.MyClass"/>
```

@EnableMBeanExport @Configuration

```
@Configuration
@EnableMBeanExport
public class AppConfig {
```

```
'context:mbean-export'
```

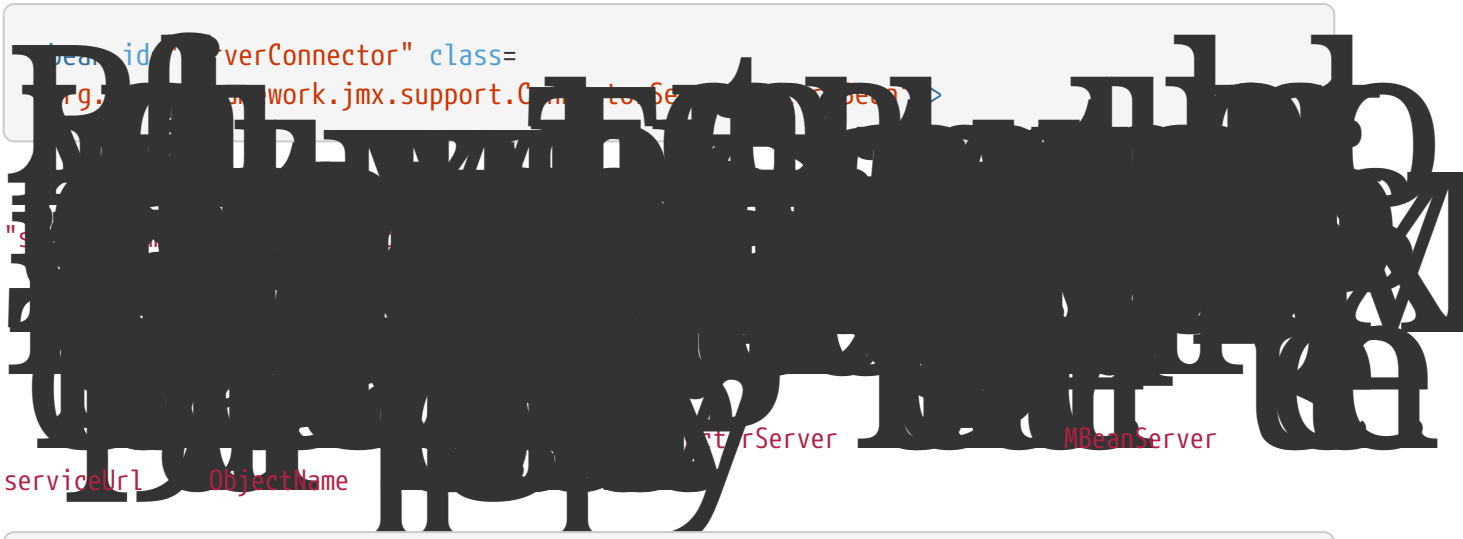
```
@EnableMBeanExport(server="myMBeanServer", defaultDomain="myDomain")
@Configuration
ContextConfiguration {

}
```

```
<context:mbea
```

```
iven/>
```

```
amework.jmx.support
```

```
JMXConnectorServer
```



```
<bean id="registry" class="org.springframework.remoting.rmi.RmiRegistryFactoryBean">
    <property name="port" value="1099"/>
</bean>
```

MBeanServerConnectionFactoryBean

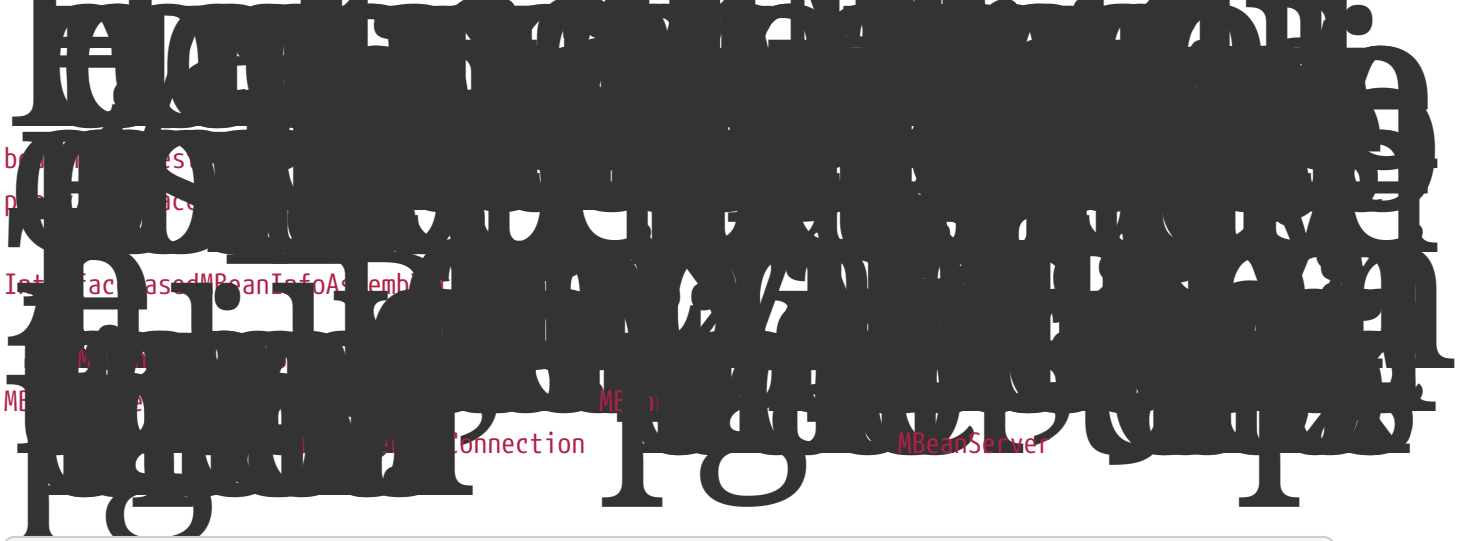
MBeanServer

```
<bean id="clientConnector" class=
"org.springframework.jmx.support.MBeanServerConnectionFactoryBean">
    <property name="serviceUrl" value=
    "service:jmx:rmi://localhost/jndi/rmi://localhost:1099/jmxrmi"/>
</bean>
```

```
<bean id="serverConnector" class=
"org.springframework.jmx.support.ConnectorServerFactoryBean">
    <property name="objectName" value="connector:name=burlap"/>
    <property name="serviceUrl" value="service:jmx:burlap://localhost:9874"/>
</bean>
```

MBeanServer

```
<bean id="proxy" class="org.springframework.jmx.access.MBeanProxyFactoryBean">
    <property name="objectName" value="bean:name=testBean"/>
    <property name="proxyInterface" value="org.springframework.jmx.IJmxTestBean"/>
</bean>
```



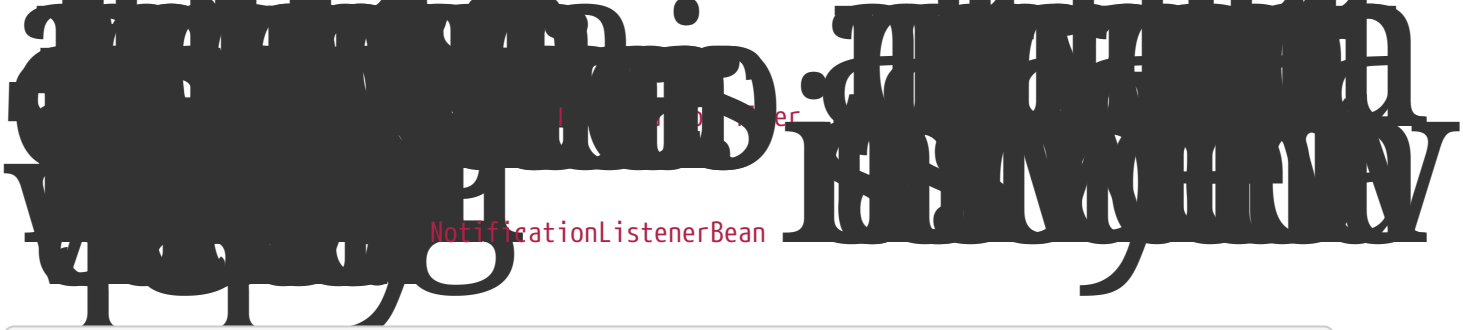
```
<bean id="clientConnector"
        class="org.springframework.jmx.support.MBeanServerConnectionFactoryBean">
    <property name="serviceUrl" value="service:jmx:rmi://remotehost:9875"/>
</bean>

<bean id="proxy" class="org.springframework.jmx.access.MBeanProxyFactoryBean">
    <property name="objectName" value="bean:name=testBean"/>
    <property name="proxyInterface" value="org.springframework.jmx.IJmxTestBean"/>
    <property name="server" ref="clientConnector"/>
```

```
package com.example;

import javax.management.AttributeChangeNotification;
import javax.management.Notification;
import javax.management.NotificationFilter;
import javax.management.NotificationListener;

public class ConsoleLoggingNotificationListener
        implements NotificationListener, NotificationFilter {

    public void handleNotification(Notification notification, Object handback) {
        System.out.println(notification);
        System.out.println(handback);
    }

    public boolean isNotificationEnabled(Notification notification) {
        return AttributeChangeNotification.class.isAssignableFrom(notification
.getClass());
    }

}
```

```
<beans>

    <bean id="exporter" class="org.springframework.jmx.export.MBeanExporter">
        <property name="beans">
            <map>
                <entry key="bean:name=testBean1" value-ref="testBean"/>
            </map>
        </property>
        <property name="notificationListenerMappings">
            <map>
                <entry key="bean:name=testBean1">
                    <bean class="com.example.ConsoleLoggingNotificationListener"/>
                </entry>
            </map>
        </property>
    </bean>

    <bean id="testBean" class="org.springframework.jmx.JmxTestBean">
        <property name="name" value="TEST"/>
        <property name="age" value="100"/>
    </bean>
```

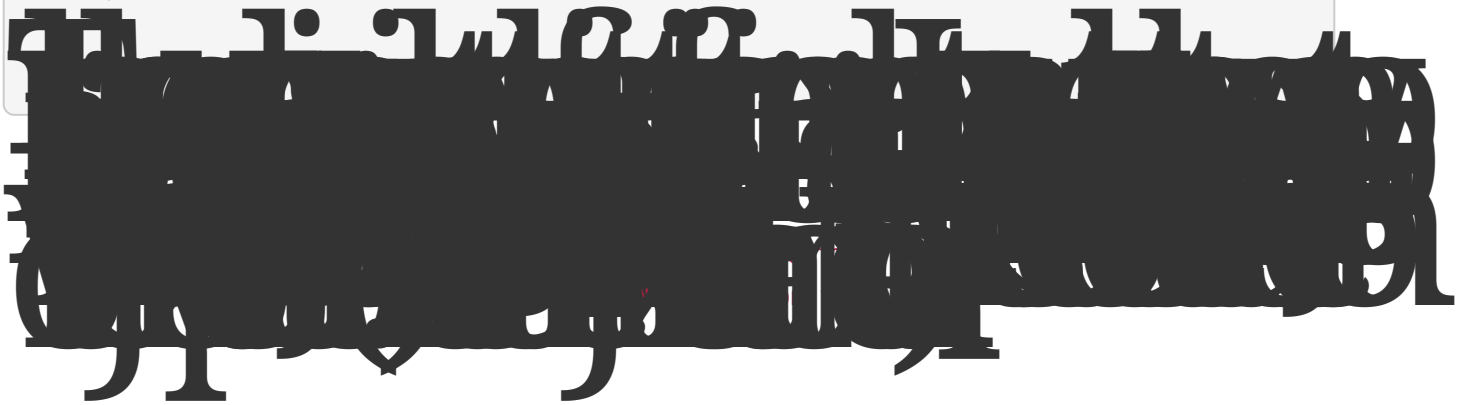
```
<beans>

    <bean id="exporter" class="org.springframework.jmx.export.MBeanExporter">
        <property name="beans">
            <map>
                <entry key="bean:name=testBean1" value-ref="testBean"/>
            </map>
        </property>
        <property name="notificationListenerMappings">
            <map>
                <entry key="<em>testBean</em>">
                    <bean class="com.example.ConsoleLoggingNotificationListener"/>
                </entry>
            </map>
        </property>
    </bean>

    <bean id="<em>testBean</em>" class="org.springframework.jmx.JmxTestBean">
        <property name="name" value="TEST"/>
        <property name="age" value="100"/>
    </bean>
```

```
<property name="notificationListenerMappings">
    <map>
        <entry key="*">
            <bean class="com.example.ConsoleLoggingNotificationListener"/>
        </entry>
    </map>
```



NotificationListenerBean

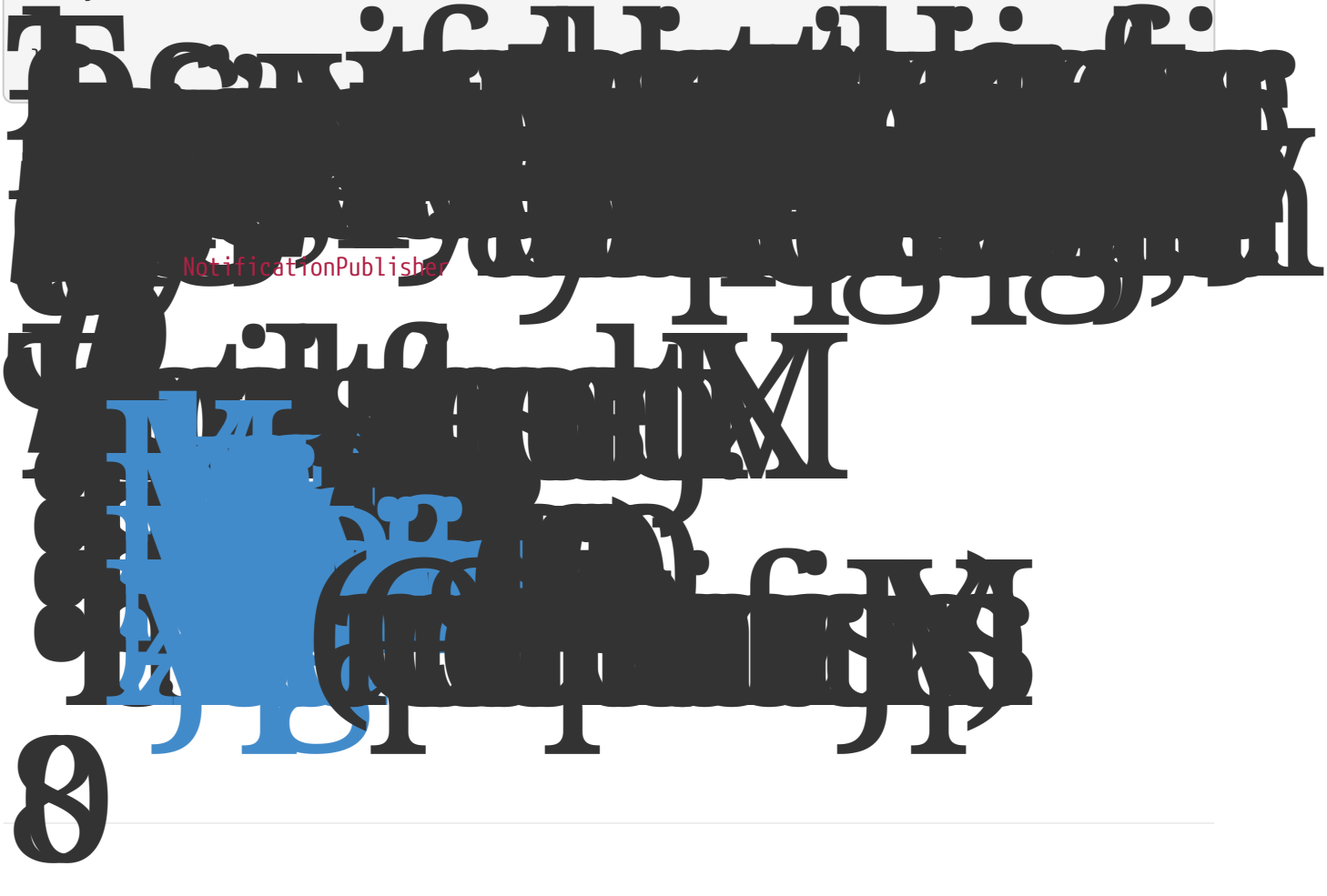
```
<beans>

    <bean id="exporter" class="org.springframework.jmx.export.MBeanExporter">
        <property name="beans">
            <map>
                <entry key="bean:name=testBean1" value-ref="testBean"/>
            </map>
        </property>
        <property name="notificationListeners">
            <list>
                <bean class="org.springframework.jmx.export.NotificationListenerBean">
                    <constructor-arg>
                        <bean class="com.example.ConsoleLoggingNotificationListener"/>
                    </constructor-arg>
                    <property name="mappedObjectNames">
                        <list>
                            <value>bean:name=testBean1</value>
                        </list>
                    </property>
                </bean>
            </list>
        </property>
    </bean>

    <bean id="testBean" class="org.springframework.jmx.JmxTestBean">
        <property name="name" value="TEST"/>
        <property name="age" value="100"/>
    </bean>
```



```
<beans>

    <bean id="exporter" class="org.springframework.jmx.export.MBeanExporter">
        <property name="beans">
            <map>
                <entry key="bean:name=testBean1" value-ref="testBean1"/>
                <entry key="bean:name=testBean2" value-ref="testBean2"/>
            </map>
        </property>
        <property name="notificationListeners">
            <list>
                <bean class="org.springframework.jmx.export.NotificationListenerBean">
                    <constructor-arg ref="customerNotificationListener"/>
                    <property name="mappedObjectNames">
                        <list>
                            <!-- handles notifications from two distinct MBeans -->
                            <value>bean:name=testBean1</value>
                            <value>bean:name=testBean2</value>
                        </list>
                    </property>
                    <property name="handback">
                        <bean class="java.lang.String">
                            <constructor-arg value="This could be anything..."/>
                        </bean>
                    </property>
                    <property name="notificationFilter" ref=
"customerNotificationListener"/>
                </bean>
            </list>
        </property>
    </bean>

    <!-- implements both the NotificationListener and NotificationFilter interfaces
-->
    <bean id="customerNotificationListener" class=
"com.example.ConsoleLoggingNotificationListener"/>

    <bean id="testBean1" class="org.springframework.jmx.JmxTestBean">
        <property name="name" value="TEST"/>
        <property name="age" value="100"/>
    </bean>

    <bean id="testBean2" class="org.springframework.jmx.JmxTestBean">
        <property name="name" value="ANOTHER TEST"/>
        <property name="age" value="200"/>
    </bean>

</beans>
```


Notifications
Notifications
NotificationEvent
add(int, int)

```
package org.springframework.jmx;

import org.springframework.jmx.export.notification.NotificationPublisherAware;
import org.springframework.jmx.export.notification.NotificationPublisher;
import javax.management.Notification;

public class JmxTestBean implements IJmxTestBean, NotificationPublisherAware {

    private String name;
    private int age;
    private boolean isSuperman;
    private NotificationPublisher publisher;

    // other getters and setters omitted for clarity

    public int add(int x, int y) {
        int answer = x + y;
        this.publisher.sendNotification(new Notification("add", this, 0));
        return answer;
    }

    public void dontExposeMe() {
        throw new RuntimeException();
    }

    public void setNotificationPublisher(NotificationPublisher notificationPublisher)
{
        this.publisher = notificationPublisher;
    }
```



```
<bean id="eciManagedConnectionFactory" class=
"com.ibm.connector2.cics.ECIManagedConnectionFactory">
    <property name="serverName" value="TXSERIES"/>
    <property name="connectionURL" value="tcp://localhost/"/>
    <property name="portNumber" value="2006"/>
</bean>

<bean id="eciConnectionFactory" class=
"org.springframework.jca.support.LocalConnectionFactoryBean">
    <property name="managedConnectionFactory" ref="eciManagedConnectionFactory"/>
</bean>
```

ManagedConnectionFactory

```
public interface ConnectionFactory implements Serializable, Referenceable {
    ...
    Connection getConnection() throws ResourceException;
    Connection getConnection(ConnectionSpec connectionSpec) throws ResourceException;
    ...
```

```
<bean id="managedConnectionFactory"
        class="com.sun.connector.cciblackbox.CciLocalTxManagedConnectionFactory">
    <property name="connectionURL" value="jdbc:hsqldb:hsql://localhost:9001"/>
    <property name="driverName" value="org.hsqldb.jdbcDriver"/>
</bean>

<bean id="targetConnectionFactory"
        class="org.springframework.jca.support.LocalConnectionFactoryBean">
    <property name="managedConnectionFactory" ref="managedConnectionFactory"/>
</bean>

<bean id="connectionFactory"
        class=
"org.springframework.jca.cci.connection.ConnectionSpecConnectionFactoryAdapter">
    <property name="targetConnectionFactory" ref="targetConnectionFactory"/>
    <property name="connectionSpec">
        <bean class="com.sun.connector.cciblackbox.CciConnectionSpec">
            <property name="user" value="sa"/>
            <property name="password" value=""/>
        </bean>
    </property>
</bean>
```

Connection

```
<bean id="eciManagedConnectionFactory"
        class="com.ibm.connector2.cics.ECIManagedConnectionFactory">
    <property name="serverName" value="TEST"/>
    <property name="connectionURL" value="tcp://localhost/"/>
    <property name="portNumber" value="2006"/>
</bean>

<bean id="targetEciConnectionFactory"
        class="org.springframework.jca.support.LocalConnectionFactoryBean">
    <property name="managedConnectionFactory" ref="eciManagedConnectionFactory"/>
</bean>

<bean id="eciConnectionFactory"
        class="org.springframework.jca.cci.connection.SingleConnectionFactory">
    <property name="targetConnectionFactory" ref="targetEciConnectionFactory"/>
</bean>
```

[illegible]

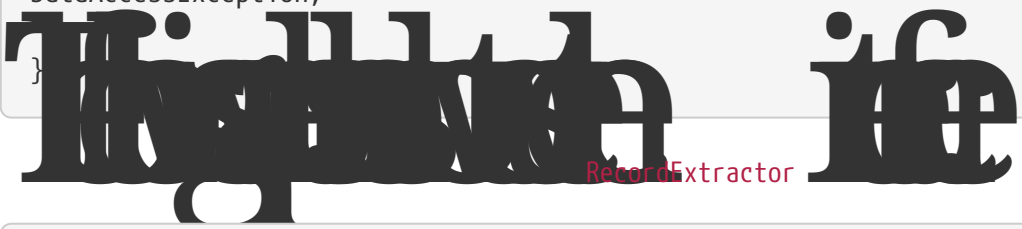
RecordCreator

```
public interface RecordCreator {

    Record createRecord(RecordFactory recordFactory) throws ResourceException,
DataAccessException;

}
```

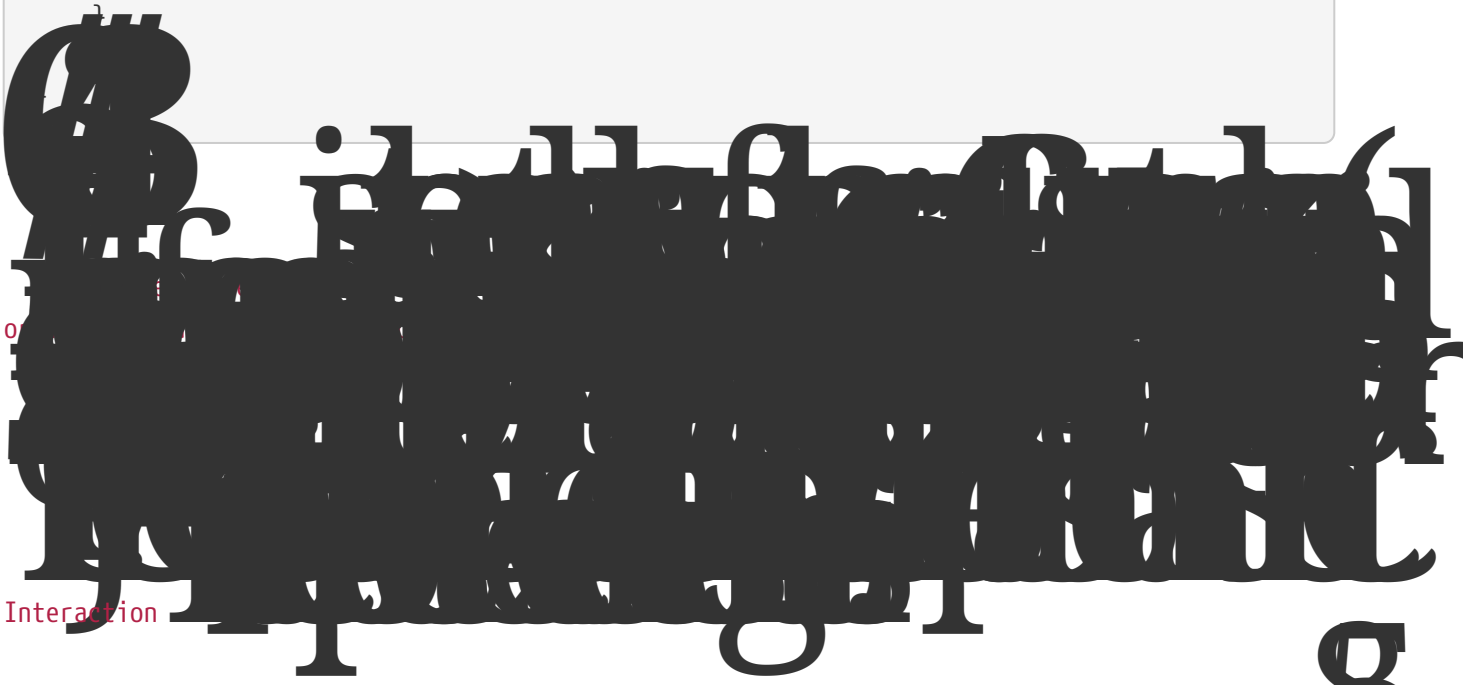
[illegible]

```
public class MyRecordCreator implements RecordCreator {

    public Record createRecord(RecordFactory recordFactory) throws ResourceException {
        IndexedRecord input = recordFactory.createIndexedRecord("input");
        input.add(new Integer(id));
        return input;
    }
```

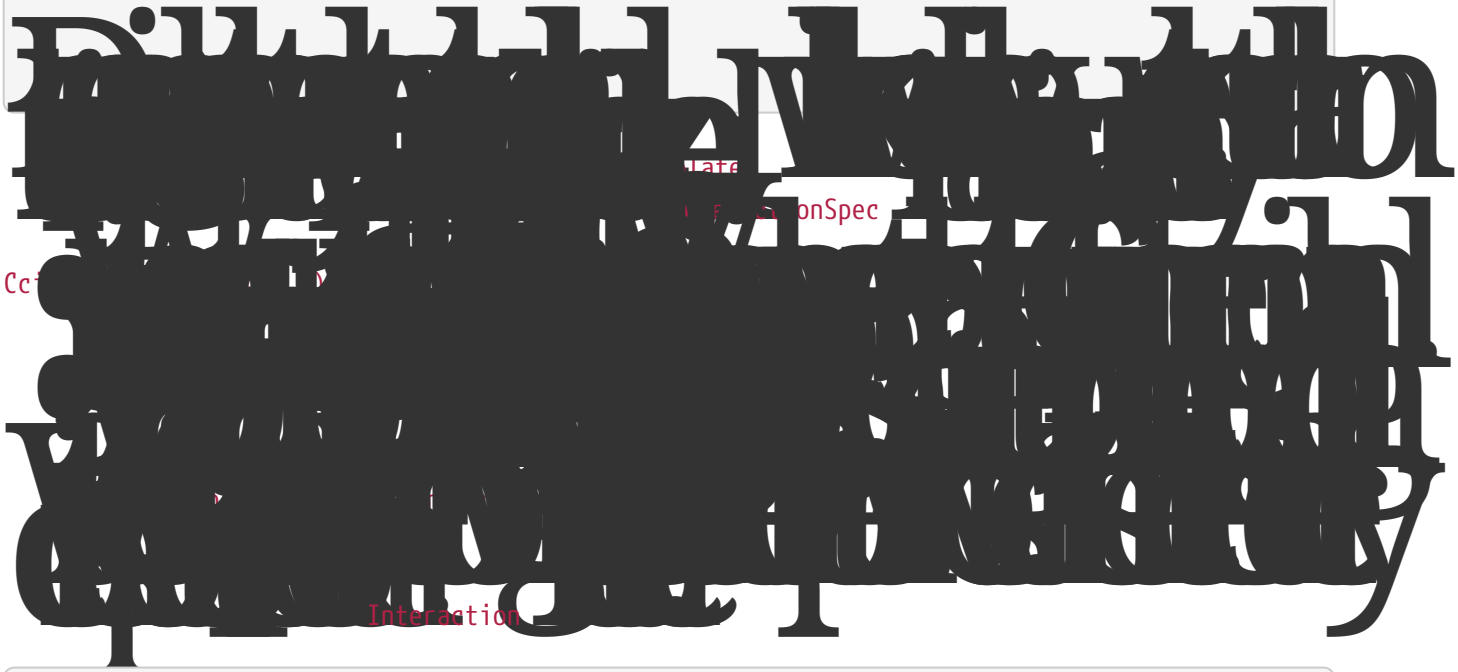
```
public interface RecordExtractor {

    Object extractData(Record record) throws ResourceException, SQLException,
DataAccessException;
}
```



```
public class MyRecordExtractor implements RecordExtractor {

    public Object extractData(Record record) throws ResourceException {
        CommAreaRecord commAreaRecord = (CommAreaRecord) record;
        String str = new String(commAreaRecord.toByteArray());
        String field1 = string.substring(0,6);
        String field2 = string.substring(6,1);
        return new OutputObject(Long.parseLong(field1), field2);
    }
```



```
public interface javax.resource.cci.Interaction {

    ...

    boolean execute(InteractionSpec spec, Record input, Record output) throws
ResourceException;

    Record execute(InteractionSpec spec, Record input) throws ResourceException;

    ...
```



```
public class CciTemplate implements CciOperations {

    public Record execute(InteractionSpec spec, Record inputRecord)
            throws DataAccessException { ... }

    public void execute(InteractionSpec spec, Record inputRecord, Record outputRecord)
            throws DataAccessException { ... }
```

```
public class CciTemplate implements CciOperations {

    public Record execute(InteractionSpec spec,
            RecordCreator inputCreator) throws DataAccessException {
        // ...
    }

    public Object execute(InteractionSpec spec, Record inputRecord,
            RecordExtractor outputExtractor) throws DataAccessException {
        // ...
    }

    public Object execute(InteractionSpec spec, RecordCreator creator,
            RecordExtractor extractor) throws DataAccessException {
        // ...
    }
```

CciTemplate.execute(..)

```
public class CciTemplate implements CciOperations {

    public IndexedRecord createIndexedRecord(String name) throws DataAccessException {
... }

    public MappedRecord createMappedRecord(String name) throws DataAccessException {
...
```

```
public abstract class CciDaoSupport {

    public void setConnectionFactory(ConnectionFactory connectionFactory) {
        // ...
    }

    public ConnectionFactory getConnectionFactory() {
        // ...
    }

    public void setCciTemplate(CciTemplate cciTemplate) {
        // ...
    }

    public CciTemplate getCciTemplate() {
        // ...
    }
```

```
<bean id="eciOutputRecordCreator" class="eci.EciOutputRecordCreator"/>

<bean id="cciTemplate" class="org.springframework.jca.cci.core.CciTemplate">
    <property name="connectionFactory" ref="eciConnectionFactory"/>
    <property name="outputRecordCreator" ref="eciOutputRecordCreator"/>
</bean>
```

Table 9. Usage of Interaction execute methods

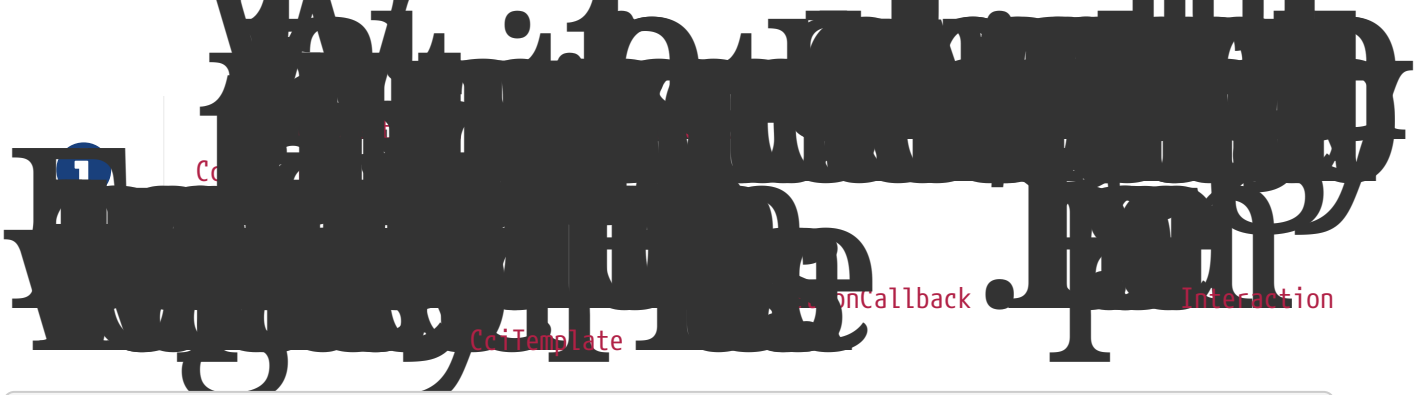
```
public interface ConnectionCallback {

    Object doInConnection(Connection connection, ConnectionFactory connectionFactory)
            throws ResourceException, SQLException, DataAccessException;

}
```

```
public interface InteractionCallback {

    Object doInInteraction(Interaction interaction, ConnectionFactory
connectionFactory)
        throws ResourceException, SQLException, DataAccessException;

}
```

```
ECIInteractionSpec interactionSpec = new ECIInteractionSpec();
```

```
public class MyDaoImpl extends CciDaoSupport implements MyDao {

    public OutputObject getData(InputObject input) {
        ECIInteractionSpec interactionSpec = ...;

    OutputObject output = (ObjectOutput) getCciTemplate().execute(interactionSpec,
        new RecordCreator() {
            public Record createRecord(RecordFactory recordFactory) throws
ResourceException {
                return new CommAreaRecord(input.toString().getBytes());
            }
        },
        new RecordExtractor() {
            public Object extractData(Record record) throws ResourceException {
                CommAreaRecord commAreaRecord = (CommAreaRecord)record;
                String str = new String(commAreaRecord.toByteArray());
                String field1 = string.substring(0,6);
                String field2 = string.substring(6,1);
                return new OutputObject(Long.parseLong(field1), field2);
            }
        });

        return output;
}
```

```
public class MyDaoImpl extends CciDaoSupport implements MyDao {

    public OutputObject getData(InputObject input) {
        ObjectOutput output = (ObjectOutput) getCciTemplate().execute(
            new ConnectionCallback() {
                public Object doInConnection(Connection connection,
                        ConnectionFactory factory) throws ResourceException {

                    // do something...

                }
            });
        }
        return output;
    }

}
```



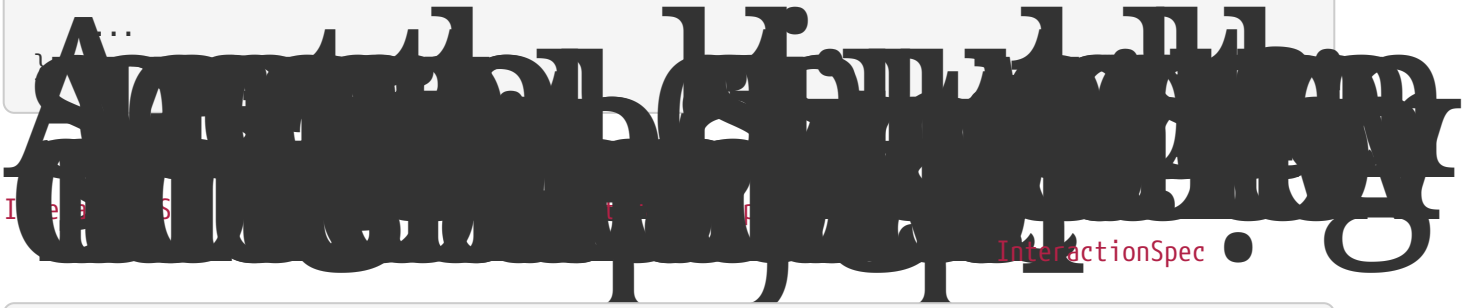
```
public class MyDaoImpl extends CciDaoSupport implements MyDao {

    public String getData(String input) {
        ECIInteractionSpec interactionSpec = ...;
        String output = (String) getCciTemplate().execute(interactionSpec,
            new InteractionCallback() {
                public Object doInInteraction(Interaction interaction,
                        ConnectionFactory factory) throws ResourceException {
                    Record input = new CommAreaRecord(inputString.getBytes());
                    Record output = new CommAreaRecord();
                    interaction.execute(holder.getInteractionSpec(), input, output);
                    return new String(output.toByteArray());
                }
            });
        return output;
    }
```

```
<bean id="managedConnectionFactory" class=
"com.ibm.connector2.cics.ECIManagedConnectionFactory">
    <property name="serverName" value="TXSERIES"/>
    <property name="connectionURL" value="local:"/>
    <property name="userName" value="CICSUSER"/>
    <property name="password" value="CICS"/>
</bean>

<bean id="connectionFactory" class=
"org.springframework.jca.support.LocalConnectionFactoryBean">
    <property name="managedConnectionFactory" ref="managedConnectionFactory"/>
</bean>

<bean id="component" class="mypackage.MyDaoImpl">
    <property
```

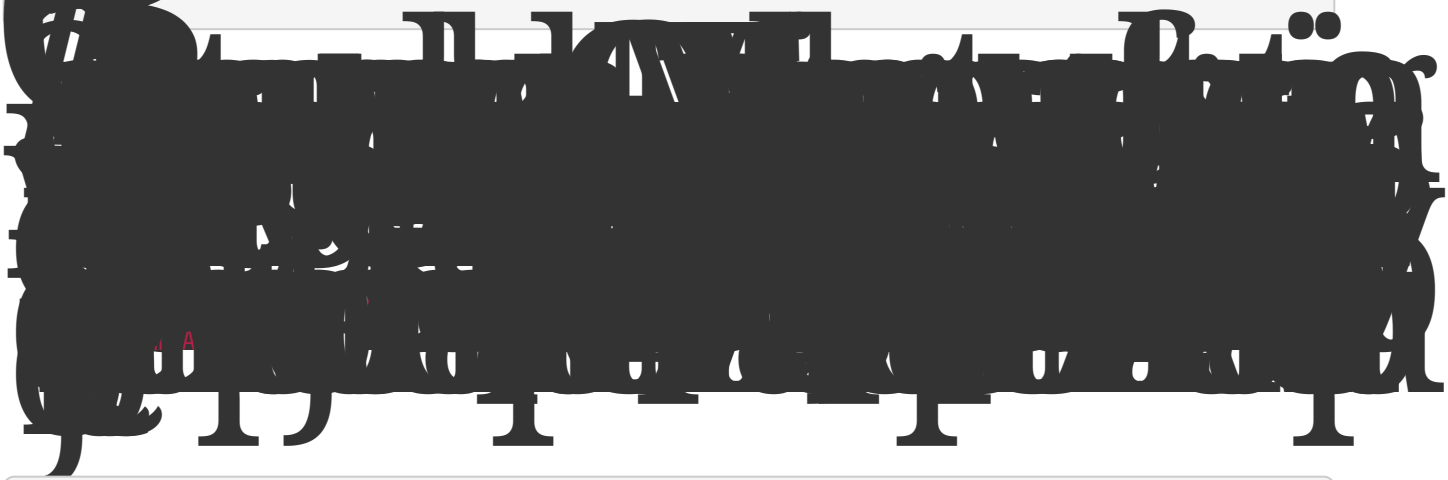
```
<jee:jndi-lookup id="connectionFactory" jndi-name="eis/cicseci"/>

<bean id="customerDao" class="...CustomerDaoImpl">
    <property name="connectionFactory" ref="connectionFactory"/>
</bean>
```


```
public abstract class MappingRecordOperation extends EisOperation {

    ...

    protected abstract Record createInputRecord(RecordFactory recordFactory,
            Object inputObject) throws ResourceException, DataAccessException {
        // ...
    }

    protected abstract Object extractOutputData(Record outputRecord)
            throws ResourceException, SQLException, DataAccessException {
        // ...
    }

    ...

}
```

```
public abstract class MappingRecordOperation extends EisOperation {

    ...

    public Object execute(Object inputObject) throws DataAccessException {
    }

    ...
}
```



```
InteractionSpec spec = ...;
MyMappingRecordOperation eisOperation = new MyMappingRecordOperation
    ionFactory(), spec);
```



```
public abstract class MappingCommAreaOperation extends MappingRecordOperation {

    ...

    protected abstract byte[] objectToBytes(Object inObject)
            throws IOException, DataAccessException;

    protected abstract Object bytesToObject(byte[] bytes)
        throws IOException, DataAccessException;

    ...

}
```

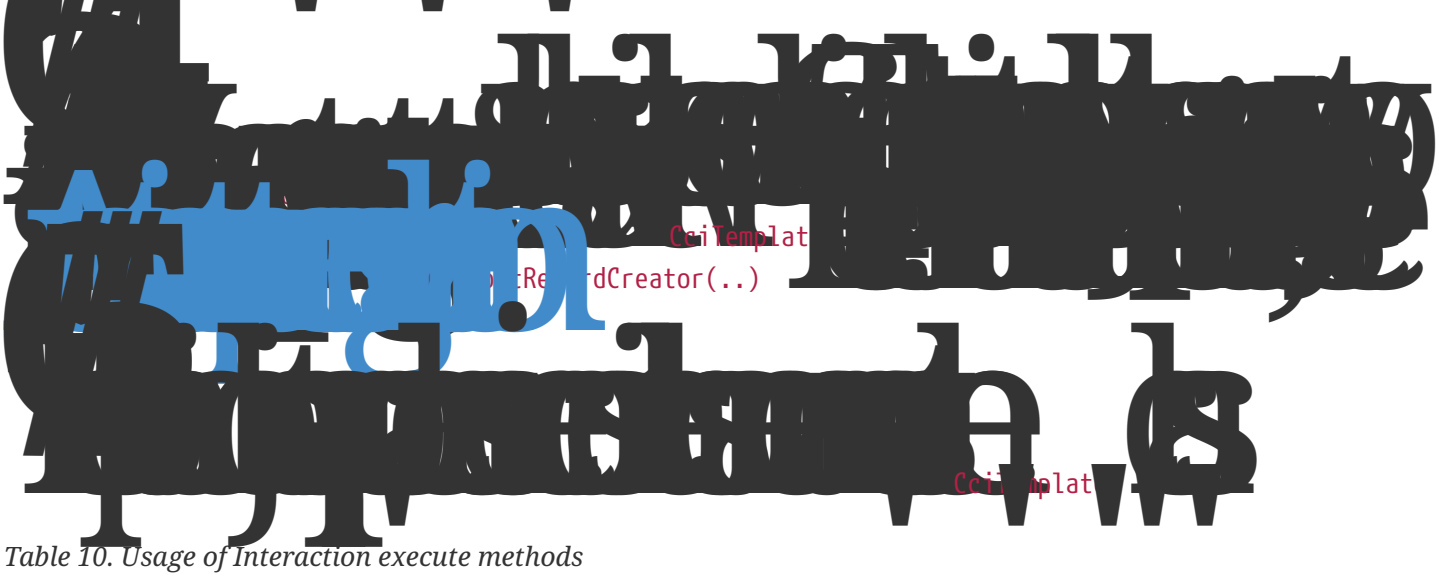
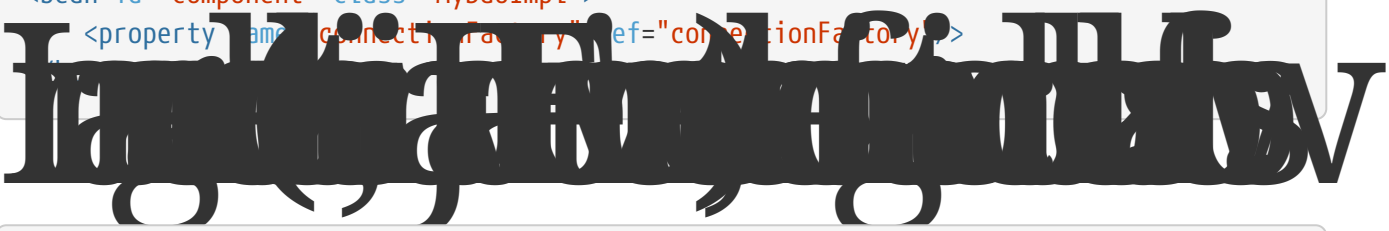


Table 10. Usage of Interaction execute methods

```
public class PersonMappingOperation extends MappingRecordOperation {

    public PersonMappingOperation(ConnectionFactory connectionFactory) {
        setConnectionFactory(connectionFactory);
        CciInteractionSpec interactionSpec = new CciConnectionSpec();
        interactionSpec.setSql("select * from person where person_id=?");
        setInteractionSpec(interactionSpec);
    }

    protected Record createInputRecord(RecordFactory recordFactory,
            Object inputObject) throws ResourceException {
        Integer id = (Integer) inputObject;
        IndexedRecord input = recordFactory.createIndexedRecord("input");
        input.add(new Integer(id));
        return input;
    }

    protected Object extractOutputData(Record outputRecord)
            throws ResourceException, SQLException {
        ResultSet rs = (ResultSet) outputRecord;
        Person person = null;
        if (rs.next()) {
            Person person = new Person();
            person.setId(rs.getInt("person_id"));
            person.setLastName(rs.getString("person_last_name"));
            person.setFirstName(rs.getString("person_first_name"));
        }
        return person;
```

```
public class MyDaoImpl extends CciDaoSupport implements MyDao {

    public Person getPerson(int id) {
        PersonMappingOperation query = new PersonMappingOperation(
getConnectionFactory());
        Person person = (Person) query.execute(new Integer(id));
        return person;
    }
}
```

```
<bean id="managedConnectionFactory"
        class="com.sun.connector.cciblackbox.CciLocalTxManagedConnectionFactory">
    <property name="connectionURL" value="jdbc:hsqldb:hsql://localhost:9001"/>
    <property name="driverName" value="org.hsqldb.jdbcDriver"/>
</bean>

<bean id="targetConnectionFactory"
        class="org.springframework.jca.support.LocalConnectionFactoryBean">
    <property name="managedConnectionFactory" ref="managedConnectionFactory"/>
</bean>

<bean id="connectionFactory"
        class=
"org.springframework.jca.cci.connection.ConnectionSpecConnectionFactoryAdapter">
    <property name="targetConnectionFactory" ref="targetConnectionFactory"/>
    <property name="connectionSpec">
        <bean class="com.sun.connector.cciblackbox.CciConnectionSpec">
            <property name="user" value="sa"/>
            <property name="password" value=""/>
        </bean>
    </property>
</bean>

<bean id="component" class="MyDaoImpl">
    <property name="connectionFactory" ref="connectionFactory"/>
</bean>
```

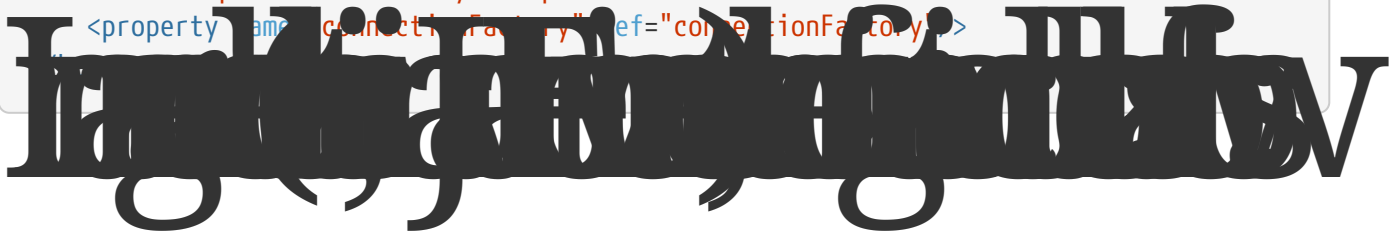


```
<jee:jndi-lookup id="targetConnectionFactory" jndi-name="eis/blackbox"/>

<bean id="connectionFactory"
        class=
"org.springframework.jca.cci.connection.ConnectionSpecConnectionFactoryAdapter">
    <property name="targetConnectionFactory" ref="targetConnectionFactory"/>
    <property name="connectionSpec">
        <bean class="com.sun.connector.cciblackbox.CciConnectionSpec">
            <property name="user" value="sa"/>
            <property name="password" value=""/>
        </bean>
    </property>
</bean>

<bean id="component" class="MyDaoImpl">
    <property name="connectionFactory" ref="connectionFactory"/>
</bean>
```

```
public abstract class EciMappingOperation extends MappingCommAreaOperation {

    public EciMappingOperation(ConnectionFactory connectionFactory, String
programName) {
        setConnectionFactory(connectionFactory);
        ECIInteractionSpec interactionSpec = new ECIInteractionSpec(),
        interactionSpec.setFunctionName(programName);
        interactionSpec.setInteractionVerb(ECIInteractionSpec.SYNC_SEND_RECEIVE);
        interactionSpec.setCommareaLength(30);
        setInteractionSpec(interactionSpec);
        setOutputRecordCreator(new EciOutputRecordCreator());
    }

    private static class EciOutputRecordCreator implements RecordCreator {
        public Record createRecord(RecordFactory recordFactory) throws
ResourceException {
            return new CommAreaRecord();
        }
    }
}
```

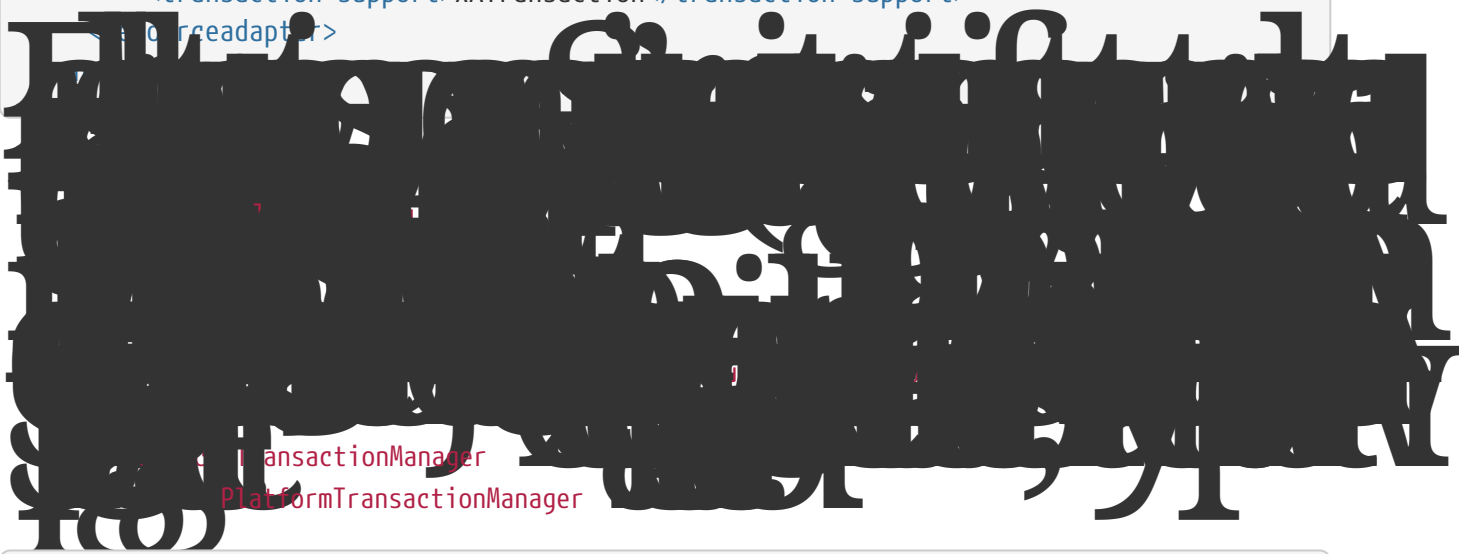
```
public class MyDaoImpl extends CciDaoSupport implements MyDao {

    public OutputObject getData(Integer id) {
        EciMappingOperation query = new EciMappingOperation(getConnectionFactory(),
"MYPROG") {

            protected abstract byte[] objectToBytes(Object inObject) throws
IOException {
                Integer id = (Integer) inObject;
                return String.valueOf(id);
            }

            protected abstract Object bytesToObject(byte[] bytes) throws IOException;
                String str = new String(bytes);
                String field1 = str.substring(0,6);
                String field2 = str.substring(6,1);
                String field3 = str.substring(7,1);
                return new OutputObject(field1, field2, field3);
            }
        });

        return (OutputObject) query.execute(new Integer(id));
    }
}
```

```
<bean id="managedConnectionFactory" class=
"com.ibm.connector2.cics.ECIManagedConnectionFactory">
    <property name="serverName" value="TXSERIES"/>
    <property name="connectionURL" value="local:"/>
    <property name="userName" value="CICSUSER"/>
    <property name="password" value="CICS"/>
</bean>

<bean id="connectionFactory" class=
"org.springframework.jca.support.LocalConnectionFactoryBean">
    <property name="managedConnectionFactory" ref="managedConnectionFactory"/>
</bean>

<bean id="component" class="MyDaoImpl">
    <property name="connectionFactory" ref="connectionFactory"/>
```



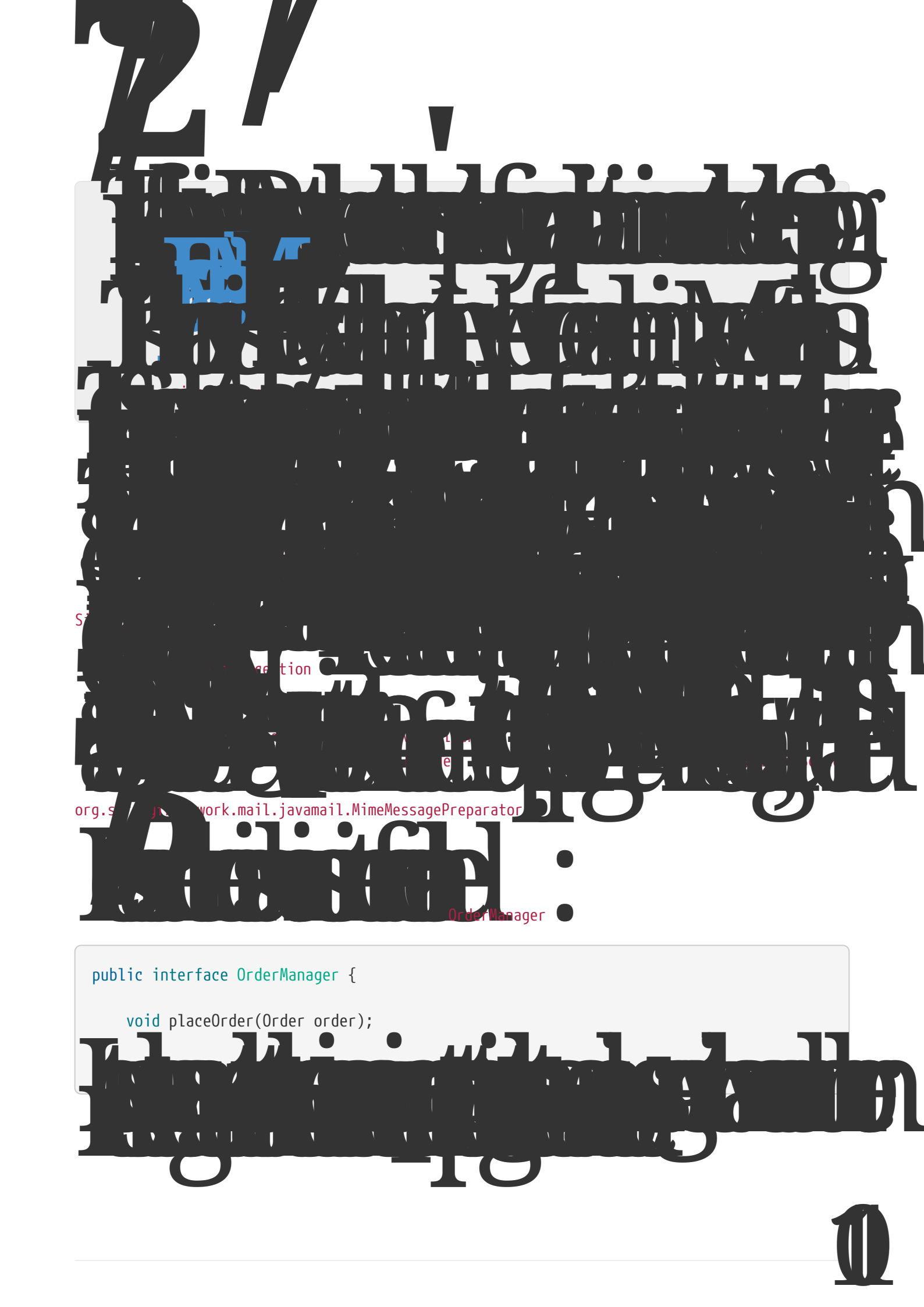
```
<jee:jndi-lookup id="connectionFactory" jndi-name="eis/cicseci"/>

    mponent" class="MyDaoImpl">
    perty name="connectionFactory" ref="connectionFactory"/>
```

```
<connector>
    <resourceadapter>
        <!-- <transaction-support>NoTransaction</transaction-support> -->
        <!-- <transaction-support>LocalTransaction</transaction-support> -->
        <transaction-support>XATransaction</transaction-support>
    rceadapter>
```

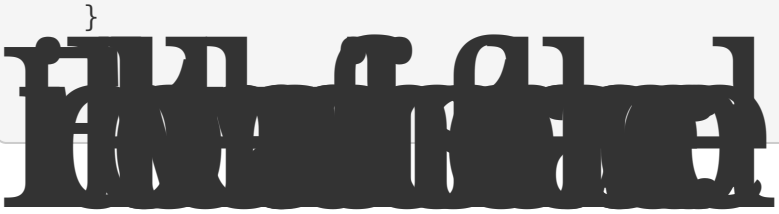


```
    ansactionManager
    PlatformTransactionManager
```

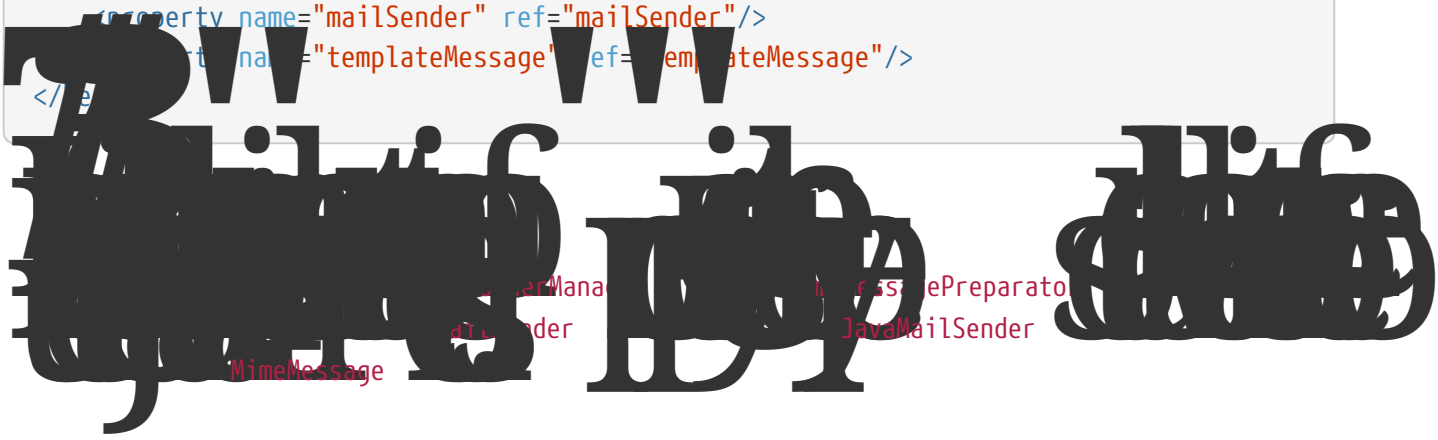
```
<jee:jndi-lookup id="eciConnectionFactory" jndi-name="eis/cicseci"/>

<bean id="eciTransactionManager"
        class="org.springframework.jca.cci.connection.CciLocalTransactionManager">
    roperty name="co    Factory"  ef="ec    ionFacto y"/>
```



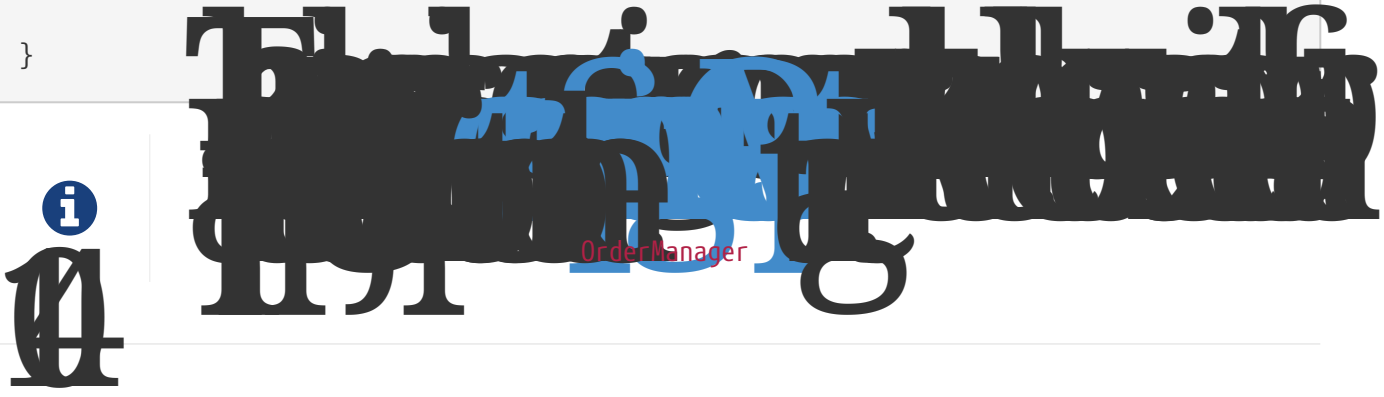
```
import org.springframework.mail.MailException;
import org.springframework.mail.MailSender;
import org.springframework.mail.SimpleMailMessage;

public class SimpleOrderManager implements OrderManager {

    private MailSender mailSender;
    private SimpleMailMessage templateMessage;

    public void setMailSender(MailSender mailSender) {
        this.mailSender = mailSender;
    }

    public void setTemplateMessage(SimpleMailMessage templateMessage) {
        this.templateMessage = templateMessage;
    }

    public void placeOrder(Order order) {

        // Do the business calculations...

        // Call the collaborators to persist the order...

        // Create a thread safe "copy" of the template message and customize it
        SimpleMailMessage msg = new SimpleMailMessage(this.templateMessage);
        msg.setTo(order.getCustomer().getEmailAddress());
        msg.setText(
            "Dear " + order.getCustomer().getFirstName()
                + order.getCustomer().getLastName()
                + ", thank you for placing order. Your order number is "
                + order.getOrderNumber());
        try{
            this.mailSender.send(msg);
        }
        catch (MailException ex) {
            // simply log it and go on...
            System.err.println(ex.getMessage());
        }
    }
```



```
<bean id="mailSender" class="org.springframework.mail.javamail.JavaMailSenderImpl">
    <property name="host" value="mail.mycompany.com"/>
</bean>

<!-- this is a template message that we can pre-load with default state -->
<bean id="templateMessage" class="org.springframework.mail.SimpleMailMessage">
    <property name="from" value="customerservice@mycompany.com"/>
    <property name="subject" value="Your order"/>
</bean>

<bean id="orderManager" class="com.mycompany.businessapp.support.SimpleOrderManager">
    <property name="mailSender" ref="mailSender"/>
    <property name="templateMessage" ref="templateMessage"/>
</bean>
```



```
import javax.mail.Message;
import javax.mail.MessagingException;
import javax.mail.internet.InternetAddress;
import javax.mail.internet.MimeMessage;

import javax.mail.internet.MimeMessage;
import org.springframework.mail.MailException;
import org.springframework.mail.javamail.JavaMailSender;
import org.springframework.mail.javamail.MimeMessagePreparator;

public class SimpleOrderManager implements OrderManager {

    private JavaMailSender mailSender;

    public void setMailSender(JavaMailSender mailSender) {
        this.mailSender = mailSender;
    }

    public void placeOrder(final Order order) {
        // Do the business calculations...
        // Call the collaborators to persist the order...

        MimeMessagePreparator preparator = new MimeMessagePreparator() {
            public void prepare(MimeMessage mimeMessage) throws Exception {
                mimeMessage.setRecipient(Message.RecipientType.TO,
                        new InternetAddress(order.getCustomer().getEmailAddress()));
                mimeMessage.setFrom(new InternetAddress("mail@mycompany.com"));
                mimeMessage.setText("Dear " + order.getCustomer().getFirstName() + " "
+
                        order.getCustomer().getLastName() + ", thanks for your order.
" +
                        "Your order number is " + order.getOrderNumber() + ".");
            }
        };

        try {
            this.mailSender.send(preparator);
        }
        catch (MailException ex) {
            // simply log it and go on...
            System.err.println(ex.getMessage());
        }
    }

}
```



MimeMessageHelper

MimeMessageHelper

MimeMessage

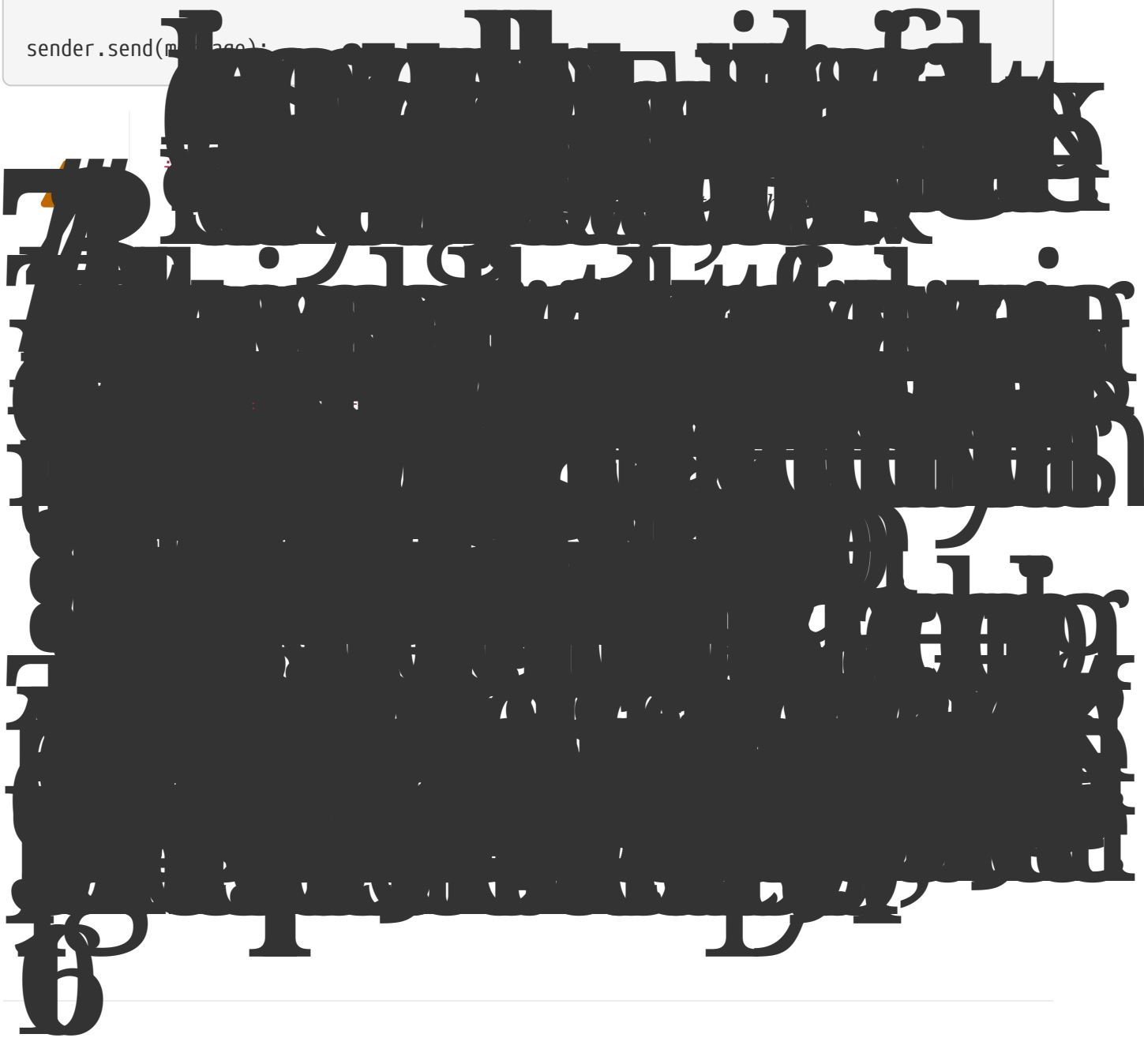
```
// of course you would use DI in any real-world cases
JavaMailSenderImpl sender = new JavaMailSenderImpl();
sender.setHost("mail.host.com");

MimeMessage message = sender.createMimeMessage();
MimeMessageHelper helper = new MimeMessageHelper(message);
helper.setTo("test@host.com");
helper.setText("Thank you for ordering!");

sender.send(message);
```

MimeMessageHelper

```
JavaMailSenderImpl sender = new JavaMailSenderImpl();
sender.setHost("mail.host.com");

MimeMessage message = sender.createMimeMessage();

// use the true flag to indicate you need a multipart message
MimeMessageHelper helper = new MimeMessageHelper(message, true);
helper.setTo("test@host.com");

helper.setText("Check out this image!");

// let's attach the infamous windows Sample file (this time copied to c:/)
FileSystemResource file = new FileSystemResource(new File("c:/Sample.jpg"));
helper.addAttachment("CoolImage.jpg", file);

sender.send(message);
```

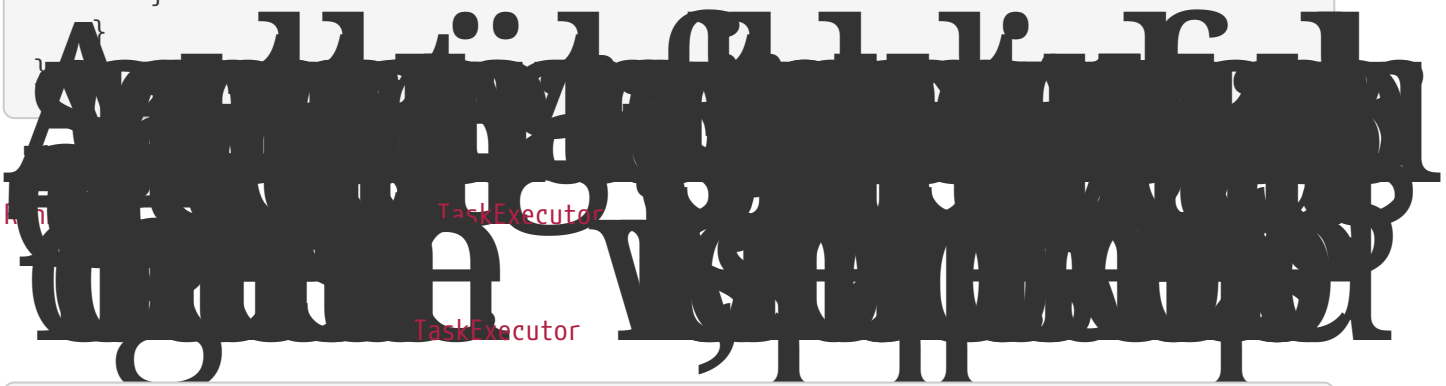
MimeMessageHelper

```
JavaMailSenderImpl sender = new JavaMailSenderImpl();
sender.setHost("mail.host.com");

MimeMessage message = sender.createMimeMessage();

// use the true flag to indicate you need a multipart message
MimeMessageHelper helper = new MimeMessageHelper(message, true);
helper.setTo("test@host.com");

// use the true flag to indicate the text included is HTML
helper.setText("<html><body><img src='cid:identifier1234'></body></html>", true);

// let's include the infamous windows Sample file (this time copied to c:/)
FileSystemResource res = new FileSystemResource(new File("c:/Sample.jpg"));
helper.addInline("identifier1234", res);

sender.send(m
```




ThreadPoolTaskExecutor

```
import org.springframework.core.task.TaskExecutor;

public class TaskExecutorExample {

    private class MessagePrinterTask implements Runnable {

        private String message;

        public MessagePrinterTask(String message) {
            this.message = message;
        }

        public void run() {
            System.out.println(message);
        }
    }

    private TaskExecutor taskExecutor;

    public TaskExecutorExample(TaskExecutor taskExecutor) {
        this.taskExecutor = taskExecutor;
    }

    public void printMessages() {
        for(int i = 0; i < 25; i++) {
            taskExecutor.execute(new MessagePrinterTask("Message" + i));
        }
    }
}
```



```
<bean id="taskExecutor" class=
"org.springframework.scheduling.concurrent.ThreadPoolTaskExecutor">
    <property name="corePoolSize" value="5"/>
    <property name="maxPoolSize" value="10"/>
    <property name="queueCapacity" value="25"/>
</bean>

<bean id="taskExecutorExample" class="TaskExecutorExample">
    <constructor-arg ref="taskExecutor"/>
</bean>
```

TaskScheduler

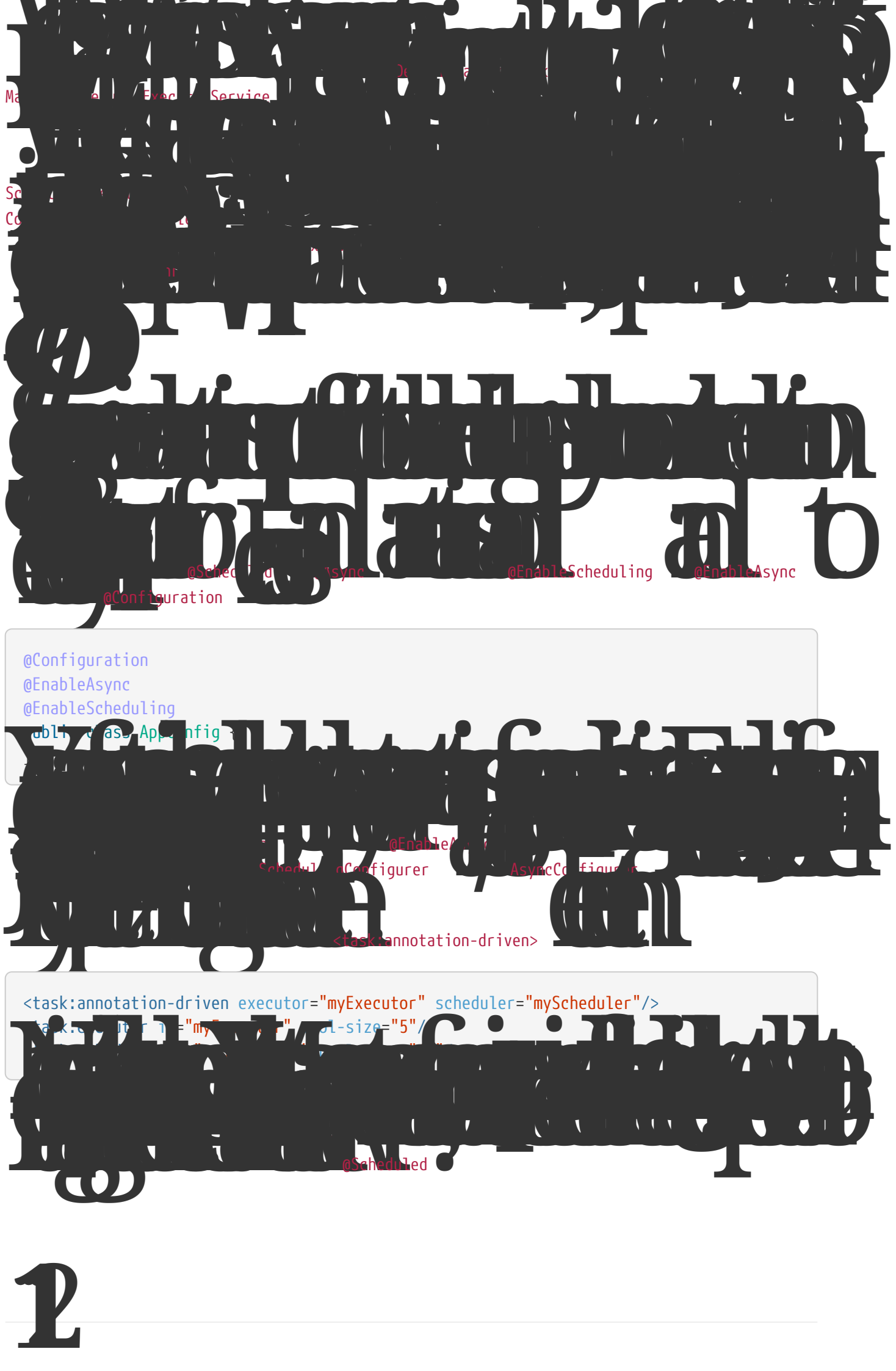
```
public interface TaskScheduler {

    ScheduledFuture schedule(Runnable task, Trigger trigger);

    ScheduledFuture schedule(Runnable task, Instant startTime);

    ScheduledFuture schedule(Runnable task, Date startTime);

    ScheduledFuture scheduleAtFixedRate(Runnable task, Instant startTime, Duration
period);

    ScheduledFuture scheduleAtFixedRate(Runnable task, Date startTime, long period);

    ScheduledFuture scheduleAtFixedRate(Runnable task, Duration period);

    ScheduledFuture scheduleAtFixedRate(Runnable task, long period);

    ScheduledFuture scheduleWithFixedDelay(Runnable task, Instant startTime, Duration
delay);

    ScheduledFuture scheduleWithFixedDelay(Runnable task, Date startTime, long delay);

    ScheduledFuture scheduleWithFixedDelay(Runnable task, Duration delay);

    ScheduledFuture scheduleWithFixedDelay(Runnable task, long delay);
```

TriggerContext

Trigger

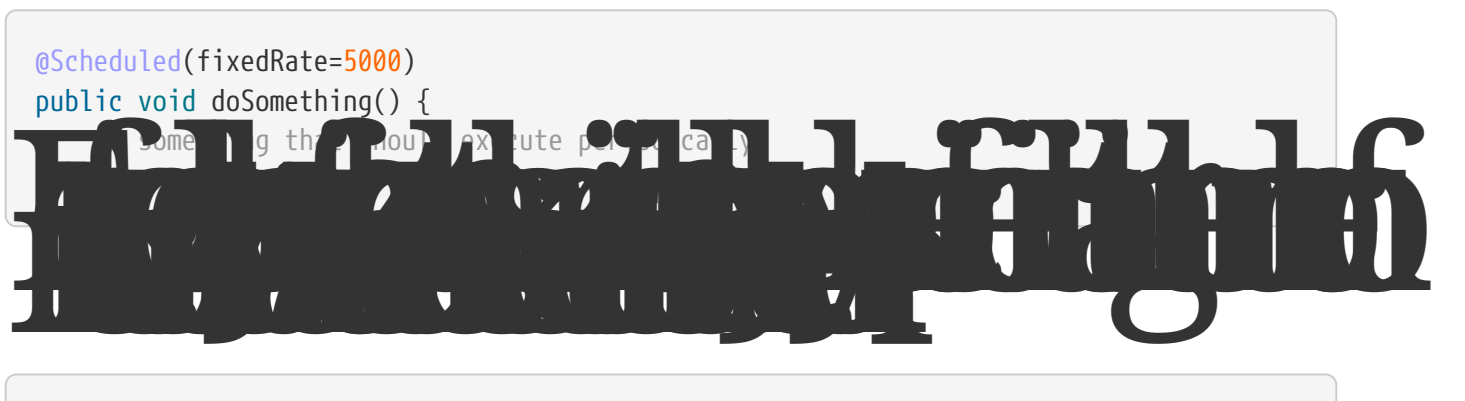
```
public interface Trigger {

    Date nextExecutionTime(TriggerContext triggerContext);
}
```

```
public interface TriggerContext {

    Date lastScheduledExecutionTime();

    Date lastActualExecutionTime();

    Date lastCompletionTime();
}
```



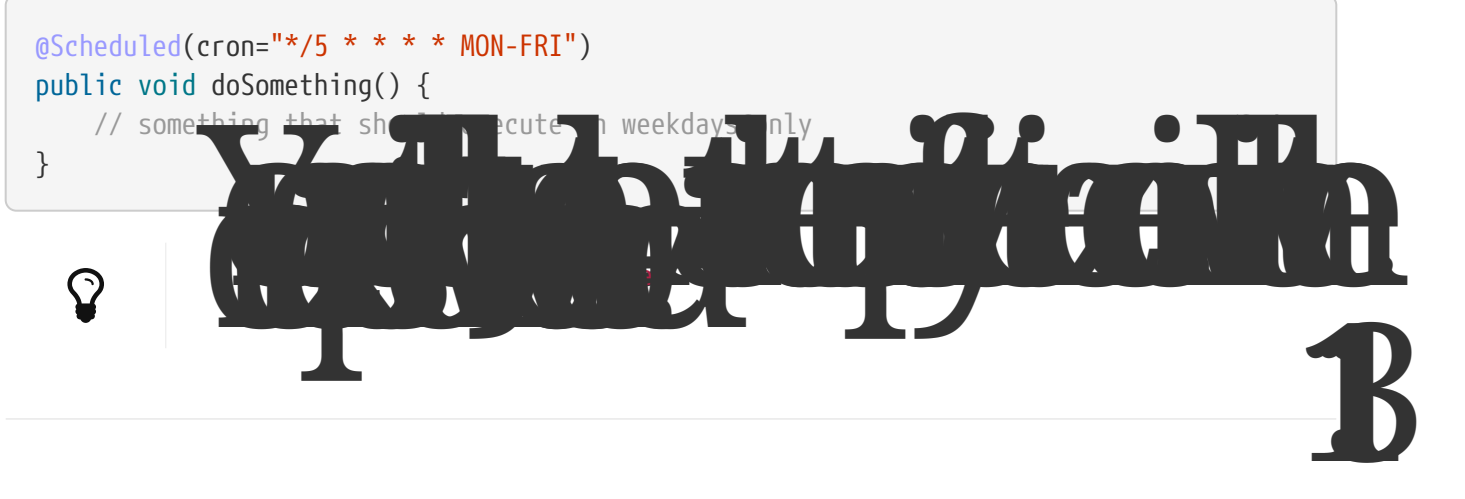
```
@Scheduled(fixedDelay=5000)
public void doSomething() {
    // something that should execute periodically
}
```

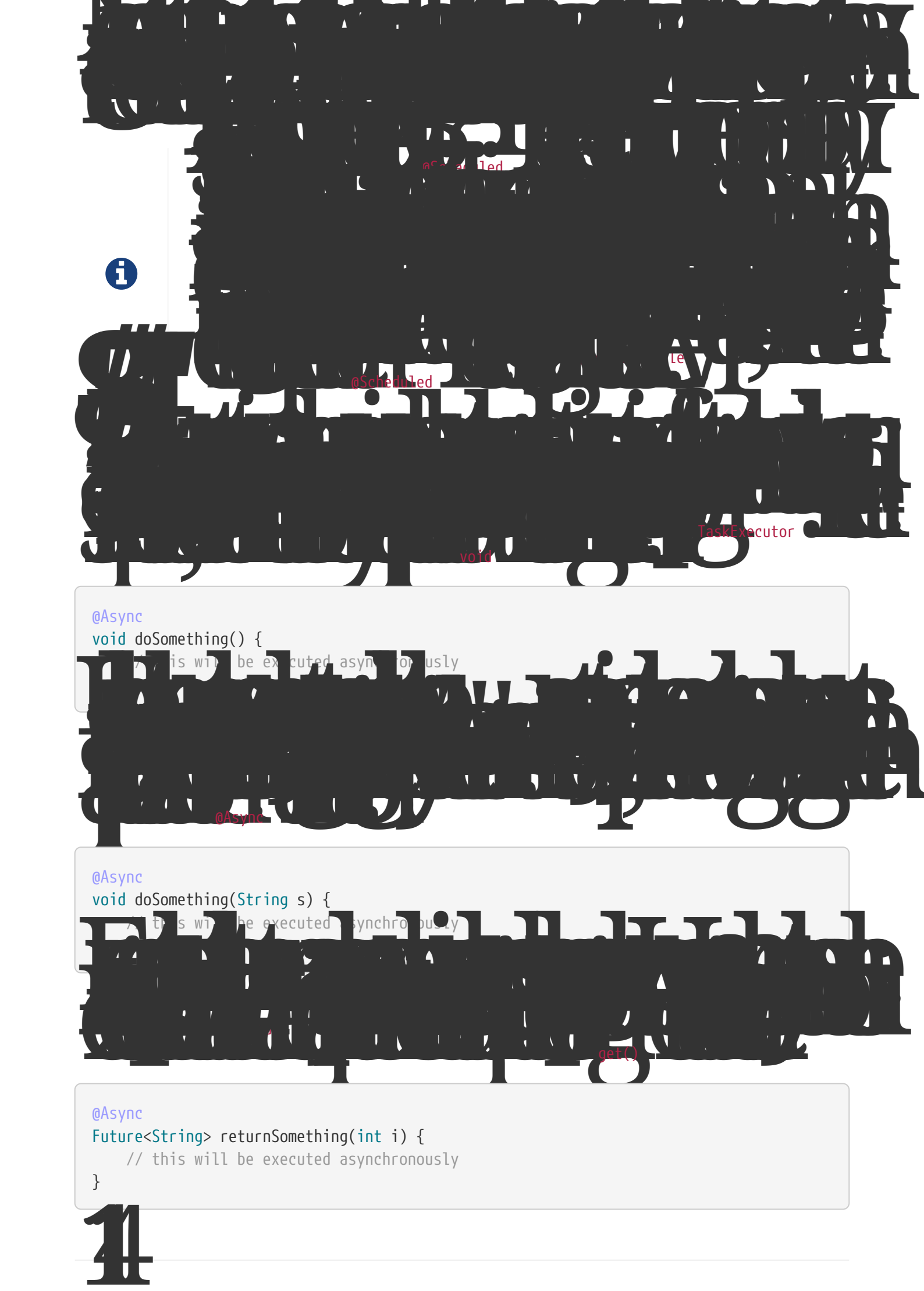
```
@Scheduled(fixedRate=5000)
public void doSomething() {
    // something that should execute periodically
}
```



```
@Scheduled(initialDelay=1000, fixedRate=5000)
public void doSomething() {
    // something that should execute periodically
}
```

```
@Scheduled(cron="*/5 * * * * MON-FRI")
public void doSomething() {
    // something that should execute on weekdays only
}
```





```
@Async
void doSomething() {
    // this will be executed asynchronously
}
```

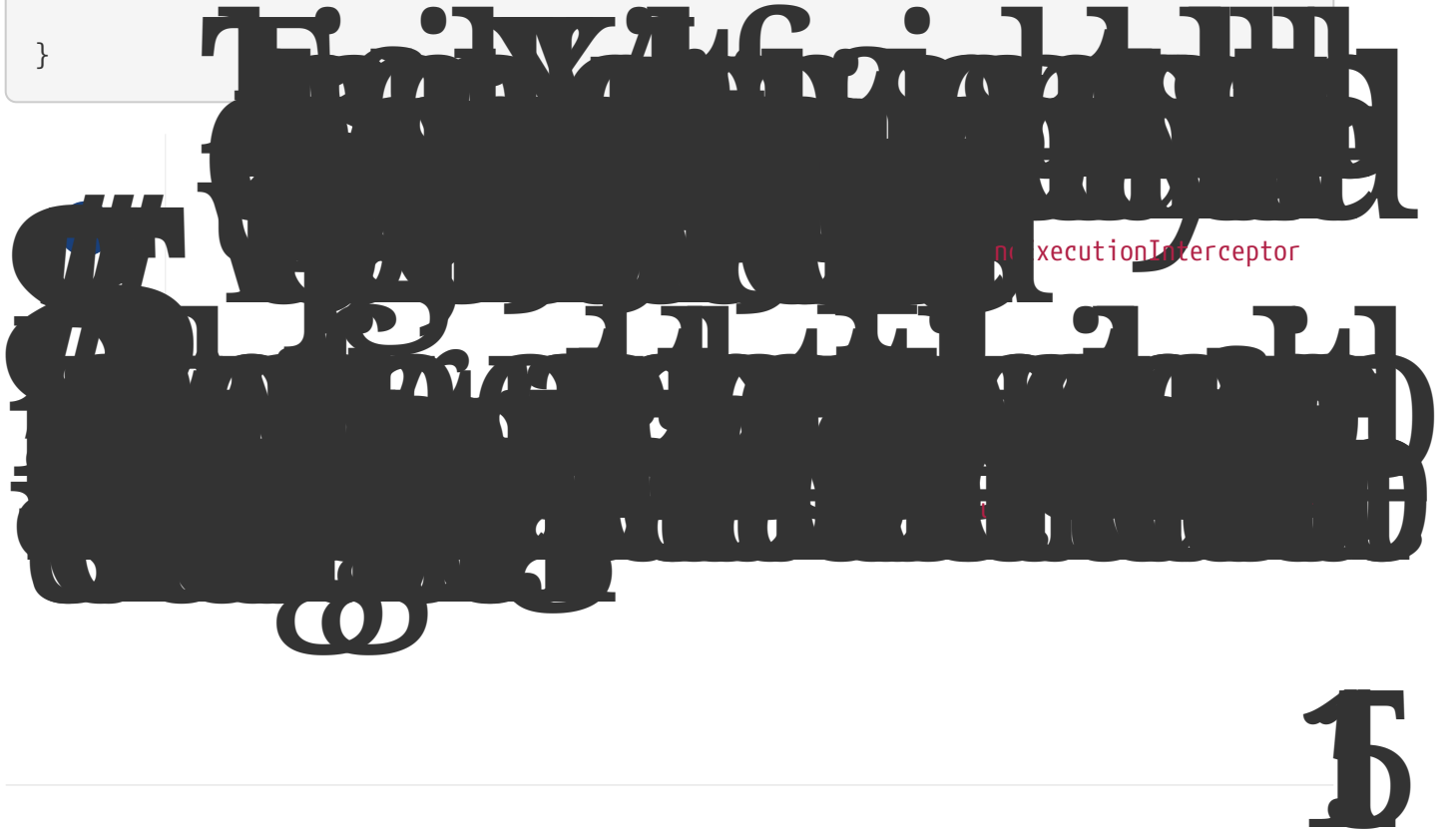


```
@Async
void doSomething(String s) {
    // this will be executed asynchronously
}
```



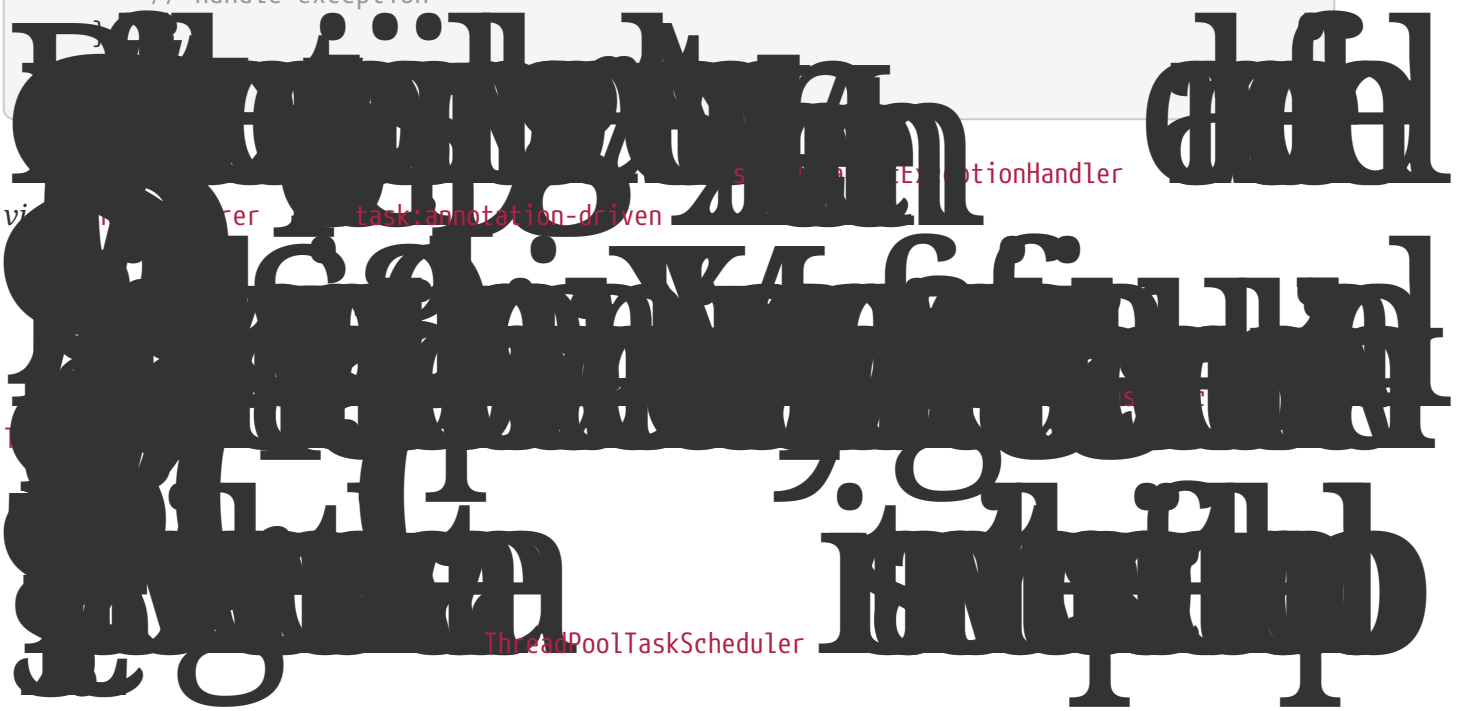
```
@Async
Future<String> returnSomething(int i) {
    // this will be executed asynchronously
}
```

```
public class SampleBeanImpl implements SampleBean {

    @Async
    void doSomething() {
        // ...
    }

}

public class SampleBeanInitializer {

    private final SampleBean bean;

    public SampleBeanInitializer(SampleBean bean) {
        this.bean = bean;
    }

    @PostConstruct
    public void initialize() {
        bean.doSomething();
    }

}
```



```
@Async("otherExecutor")
void doSomething(String s) {
```

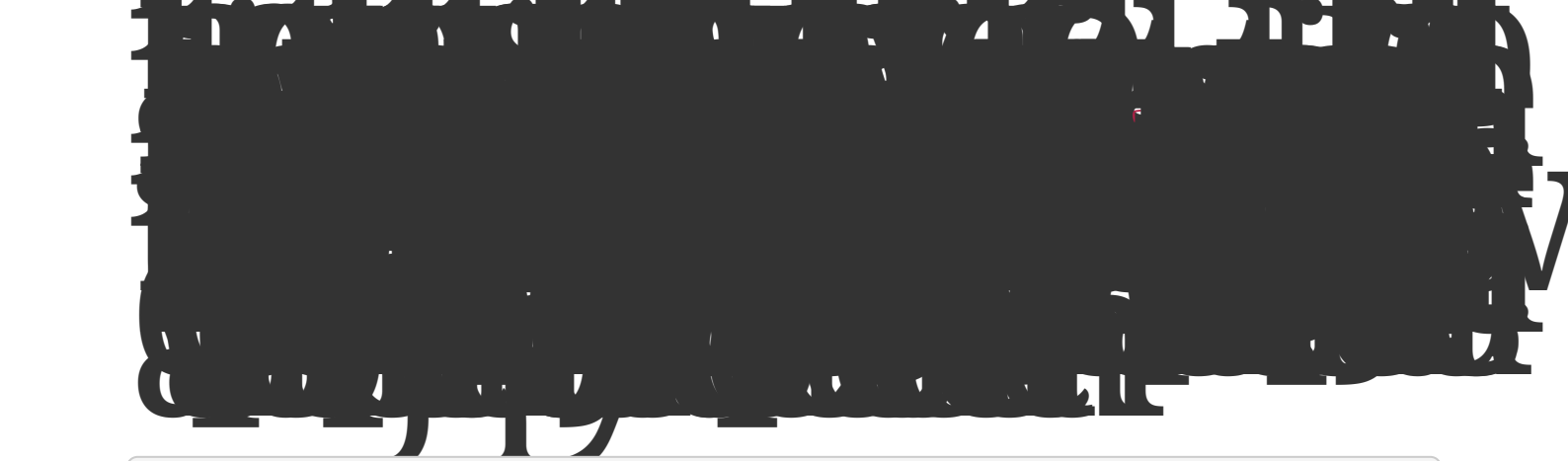
```
public class MyAsyncUncaughtExceptionHandler implements AsyncUncaughtExceptionHandler
{

    @Override
    public void handleUncaughtException(Throwable ex, Method method, Object... params)
{
        // handle exception
```



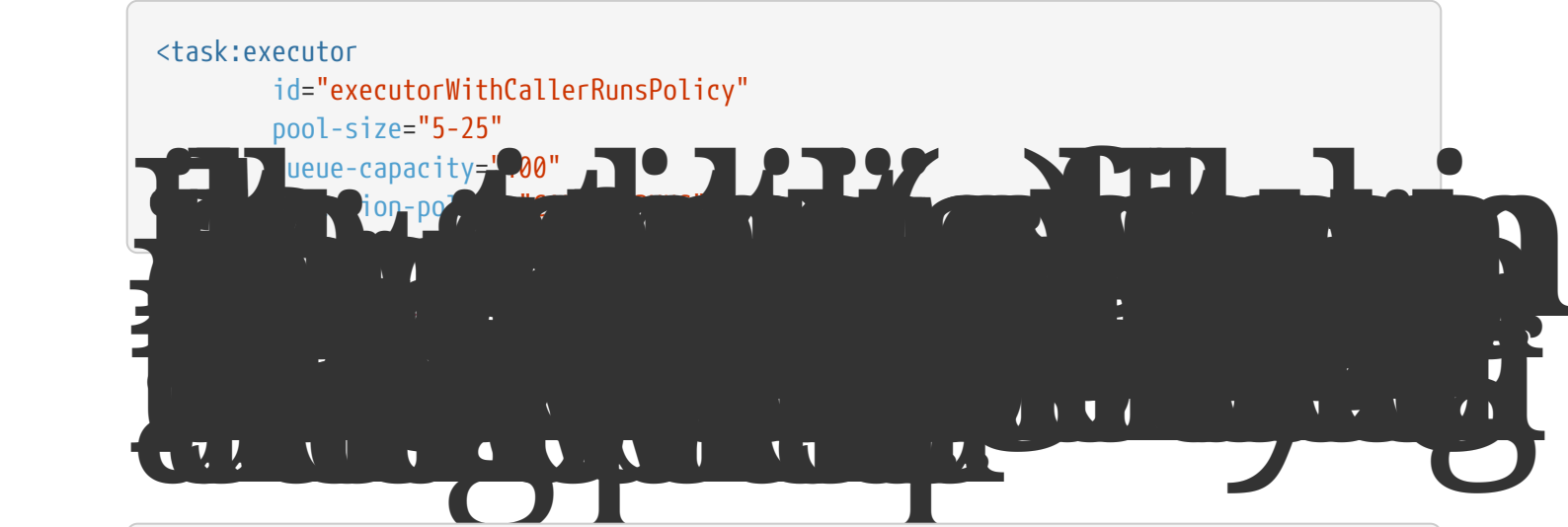
ThreadPoolTaskExecutor

```
<task:executor
        id="executorWithPoolSizeRange"
        pool-size="5-25"
```

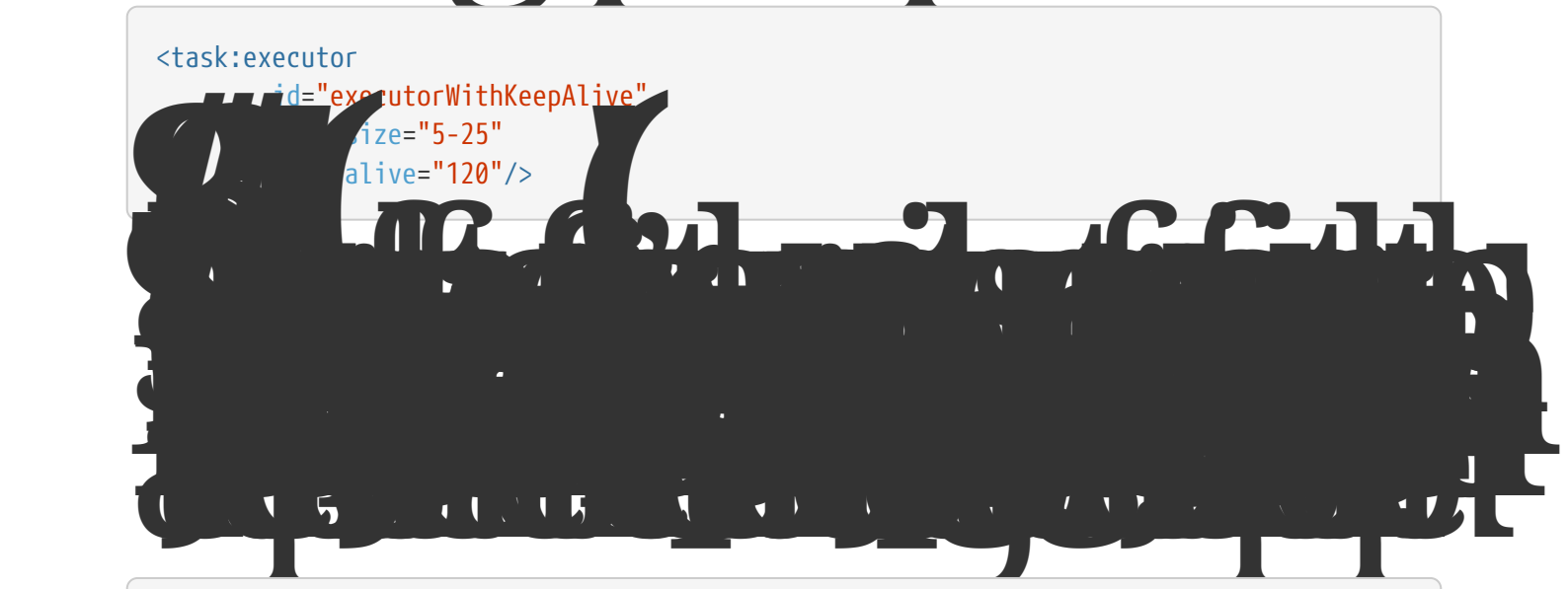
DiscardOldestPolicy



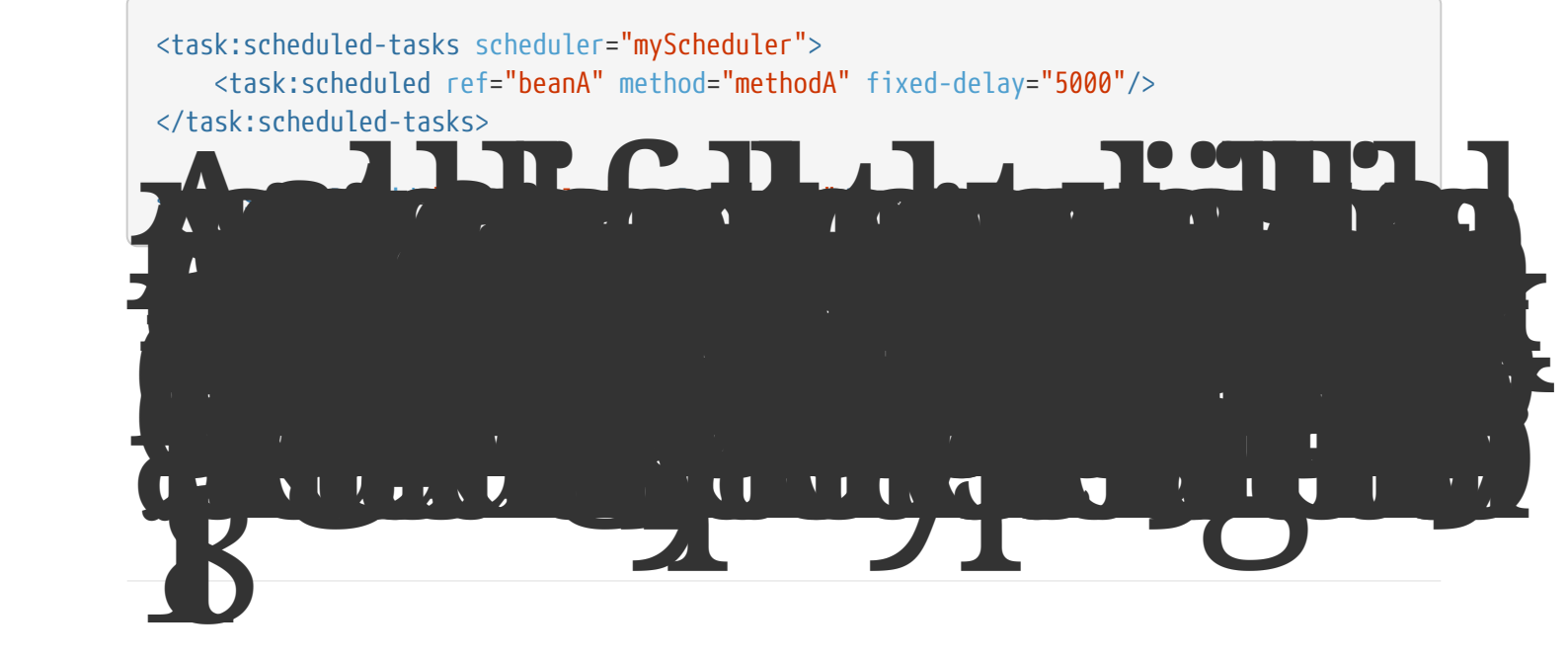
```
<task:executor
        id="executorWithCallerRunsPolicy"
        pool-size="5-25"
        queue-capacity="100"
        rejection-policy="CALLER_RUNS"/>
```



```
<task:executor
        id="executorWithKeepAlive"
        pool-size="5-25"
        keep-alive="120"/>
```



```
<task:scheduled-tasks scheduler="myScheduler">
    <task:scheduled ref="beanA" method="methodA" fixed-delay="5000"/>
</task:scheduled-tasks>
```



```
<task:scheduled-tasks scheduler="myScheduler">
    <task:scheduled ref="beanA" method="methodA" fixed-delay="5000" initial-delay=
"1000"/>
    <task:scheduled ref="beanB" method="methodB" fixed-rate="5000"/>
    <task:scheduled ref="beanC" method="methodC" cron="*/5 * * * * MON-FRI"/>
</task:scheduled-tasks>

<task:scheduler id="myScheduler" pool-size="10"/>
```

```
<bean name="exampleJob" class=
"org.springframework.scheduling.quartz.JobDetailFactoryBean">
    <property name="jobClass" value="example.ExampleJob"/>
    <property name="jobDataAsMap">
        <map>
            <entry key="timeout" value="5"/>
        </map>
    </property>
</bean>
```

```
package example;

public class ExampleJob extends QuartzJobBean {

    private int timeout;

    /**
     * Setter called after the ExampleJob is instantiated
     * with the value from the JobDetailFactoryBean (5)
     */
    public void setTimeout(int timeout) {
        this.timeout = timeout;
    }

    protected void executeInternal(JobExecutionContext ctx) throws
JobExecutionException {
        // do the actual work
    }
}
```

JobDetailFactoryBean exampleJob

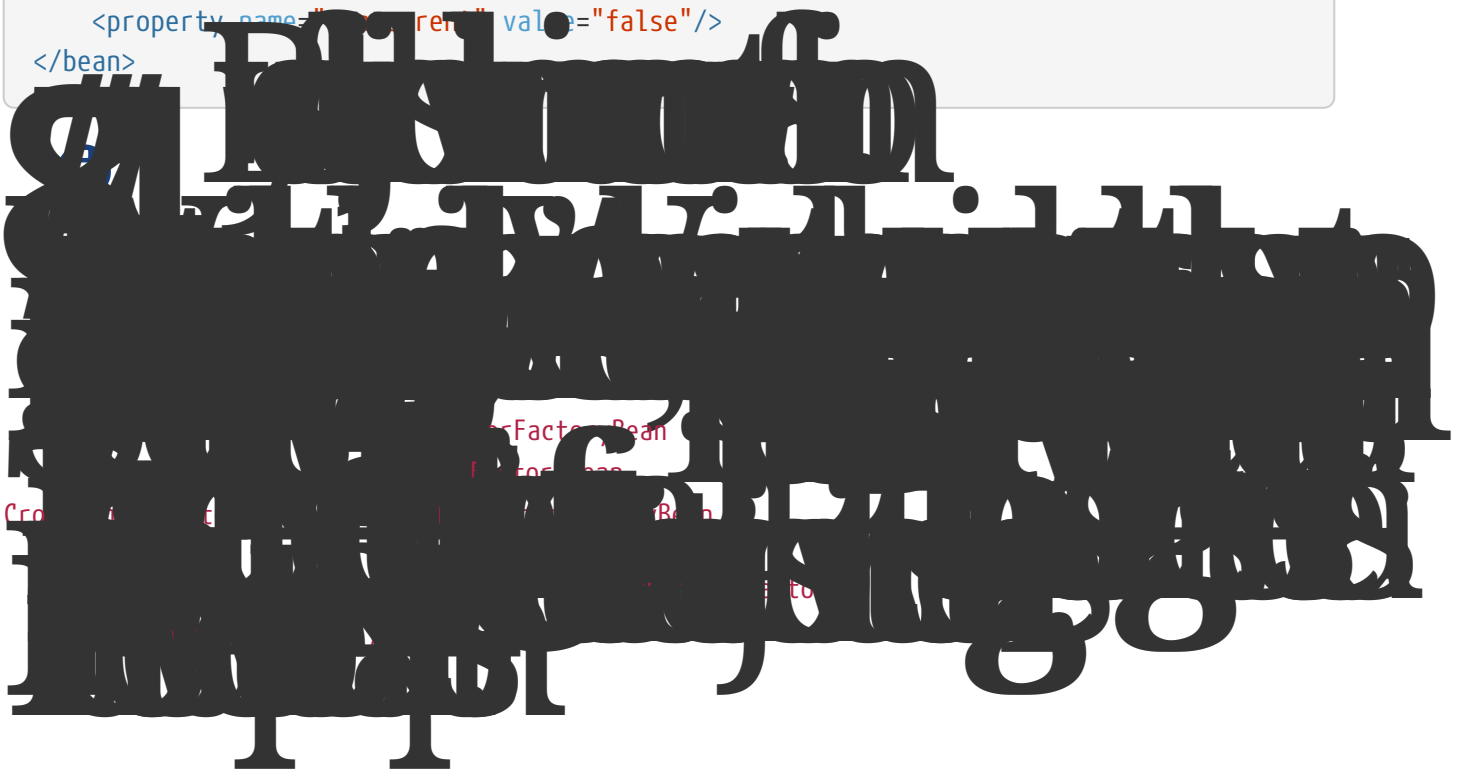
MethodInvokingJobDetailFactoryBean

```
<bean id="jobDetail" class=
"org.springframework.scheduling.quartz.MethodInvokingJobDetailFactoryBean">
    <property name="targetObject" ref="exampleBusinessObject"/>
    <property name="targetMethod" value="doIt" />
</bean>
```

doIt exampleBusinessObject

```
public class ExampleBusinessObject {

    // properties and collaborators

    public void doIt() {
        // do the actual work
    }
}
```

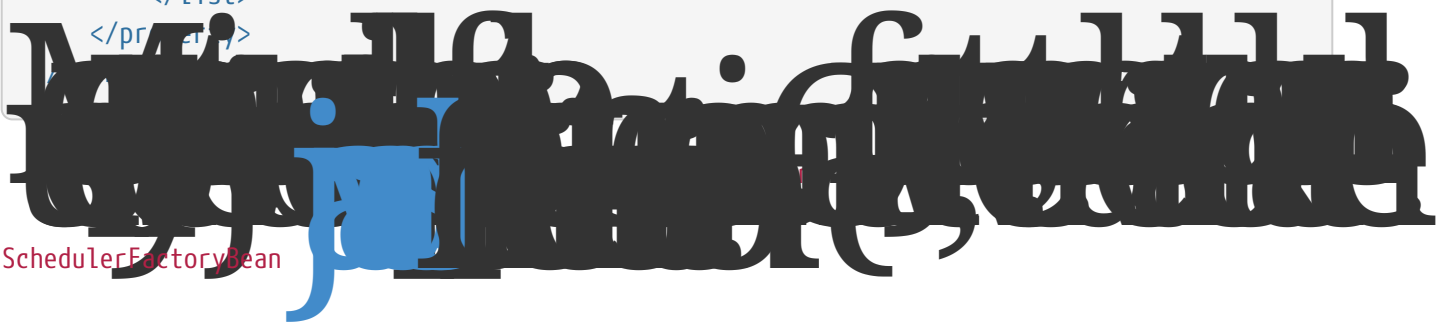
```
<bean id="jobDetail" class=
"org.springframework.scheduling.quartz.MethodInvokingJobDetailFactoryBean">
    <property name="targetObject" ref="exampleBusinessObject"/>
    <property name="targetMethod" value="doIt"/>
    <property name="concurrent" value="false"/>
</bean>
```



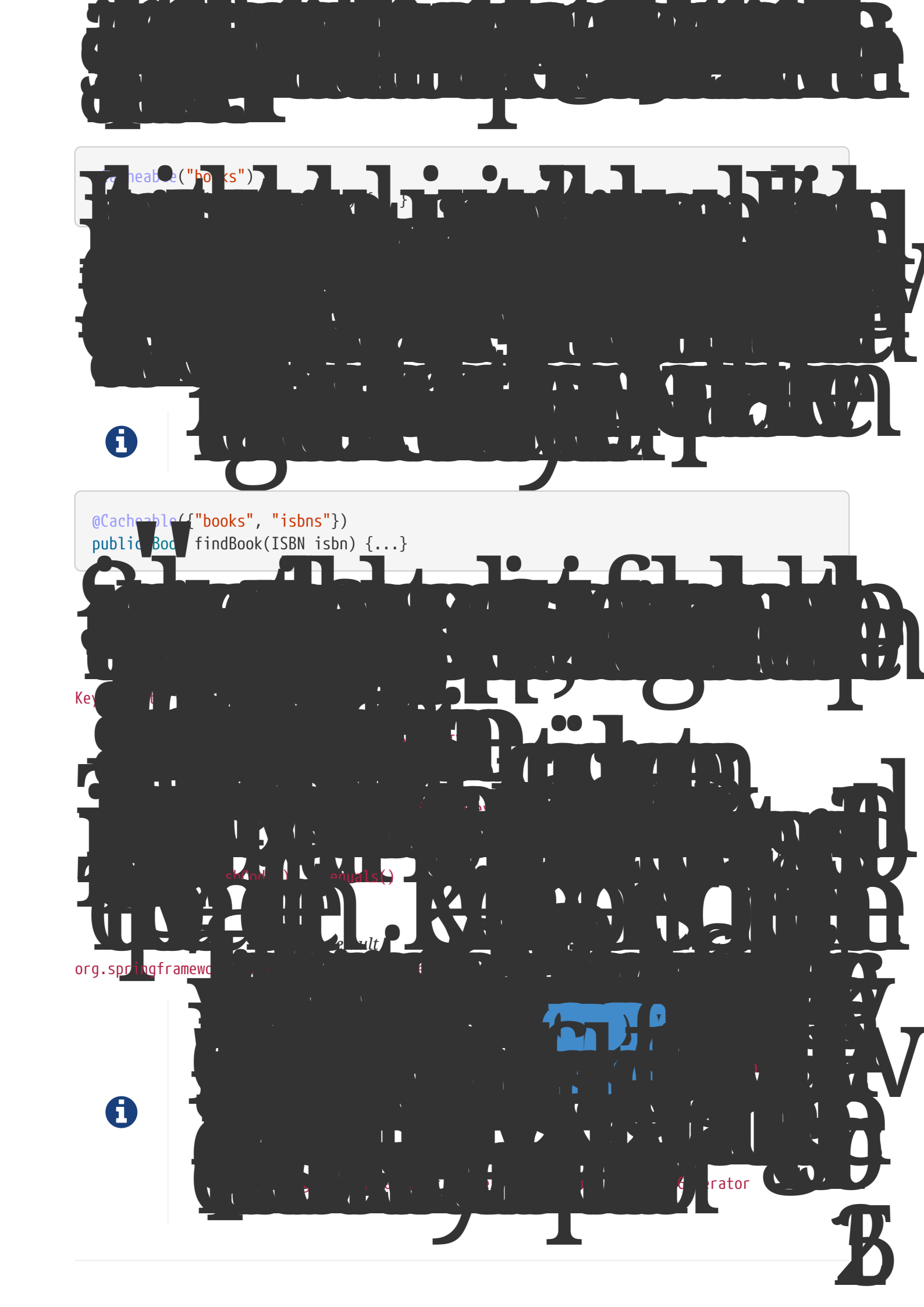
```
<bean id="simpleTrigger" class=
"org.springframework.scheduling.quartz.SimpleTriggerFactoryBean">
    <!-- see the example of method invoking job above -->
    <property name="jobDetail" ref="jobDetail"/>
    <!-- 10 seconds -->
    <property name="startDelay" value="10000"/>
    <!-- repeat every 50 seconds -->
    <property name="repeatInterval" value="50000"/>
</bean>

<bean id="cronTrigger" class=
"org.springframework.scheduling.quartz.CronTriggerFactoryBean">
    <property name="jobDetail" ref="exampleJob"/>
    <!-- run every morning at 6 AM -->
    <property name="cronExpression" value="0 0 6 * * ?"/>
```

SchedulerFactoryBean

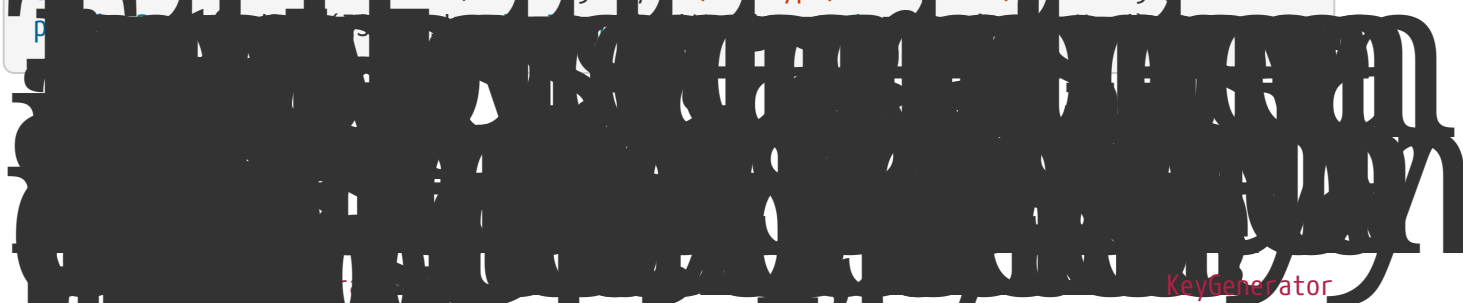
```
<bean class="org.springframework.scheduling.quartz.SchedulerFactoryBean">
    <property name="triggers">
        <list>
            <ref bean="cronTrigger"/>
            <ref bean="simpleTrigger"/>
        </list>
    </property>
```



SchedulerFactoryBean



```
@Cacheable(cacheNames="books", <strong>key="#isbn"</strong>)
public Book findBook(ISBN isbn, boolean checkWarehouse, boolean includeUsed)

@Cacheable(cacheNames="books", <strong>key="#isbn.rawNumber"</strong>)
public Book findBook(ISBN isbn, boolean checkWarehouse, boolean includeUsed)

@Cacheable(cacheNames="books", <strong>key="T(someType).hash(#isbn)"</strong>)
public Book findBook(ISBN isbn, boolean checkWarehouse, boolean includeUsed)
```



```
@Cacheable(cacheNames="books", <strong>keyGenerator="myKeyGenerator"</strong>)
public Book findBook(ISBN isbn, boolean checkWarehouse, boolean includeUsed)
```

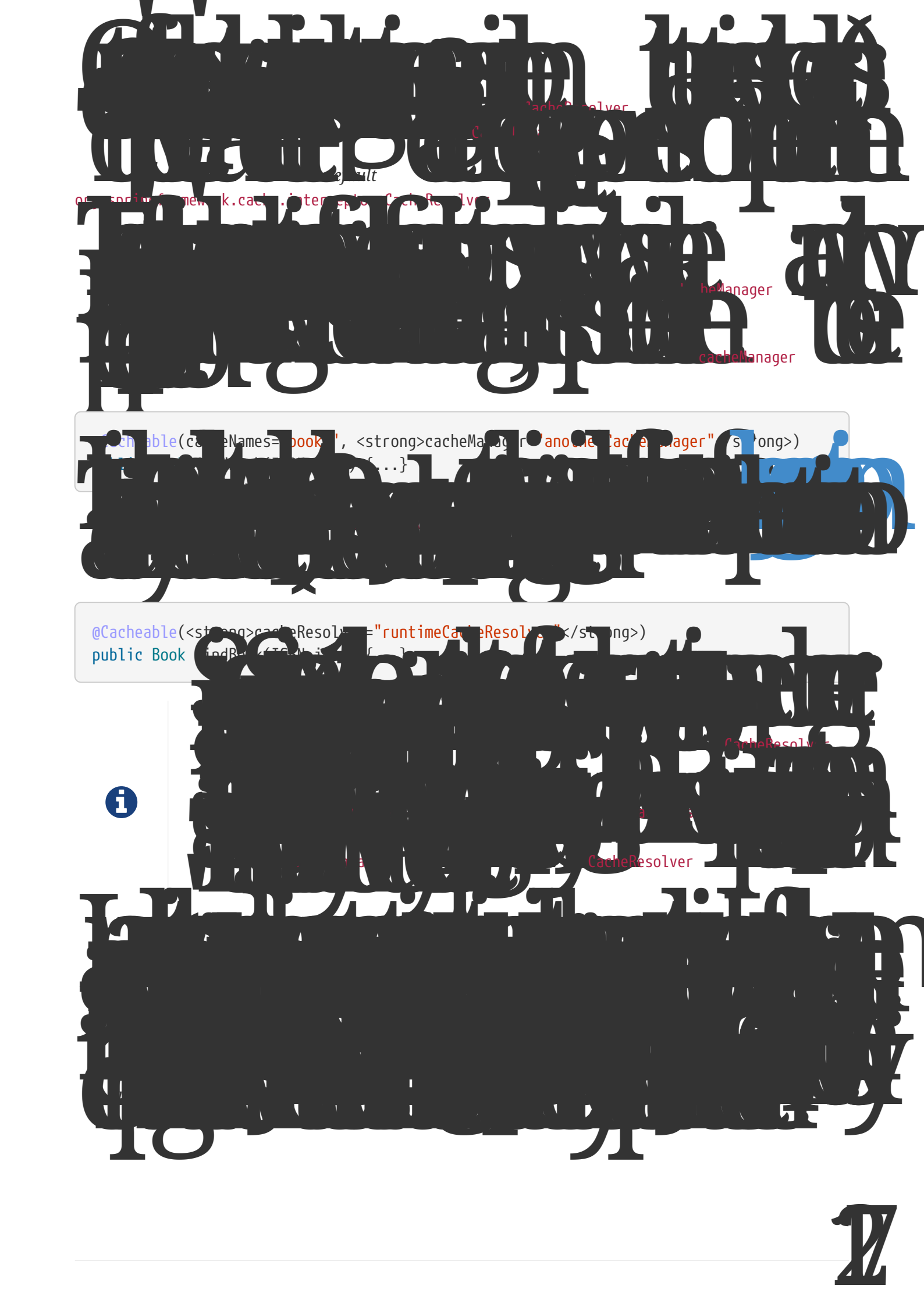
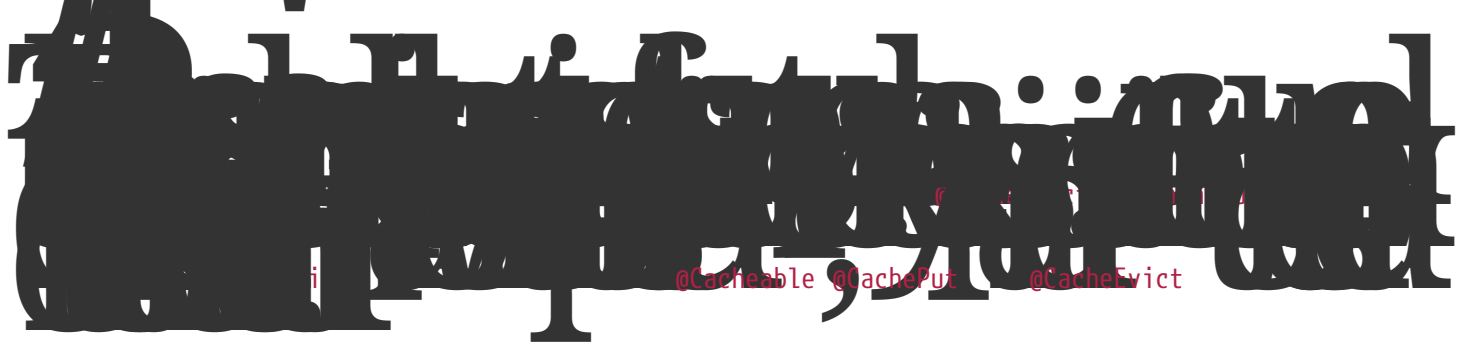
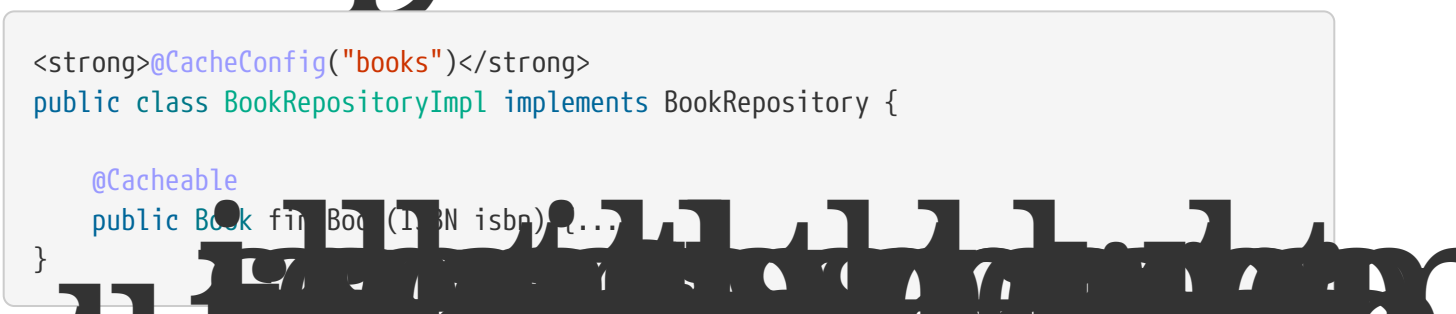


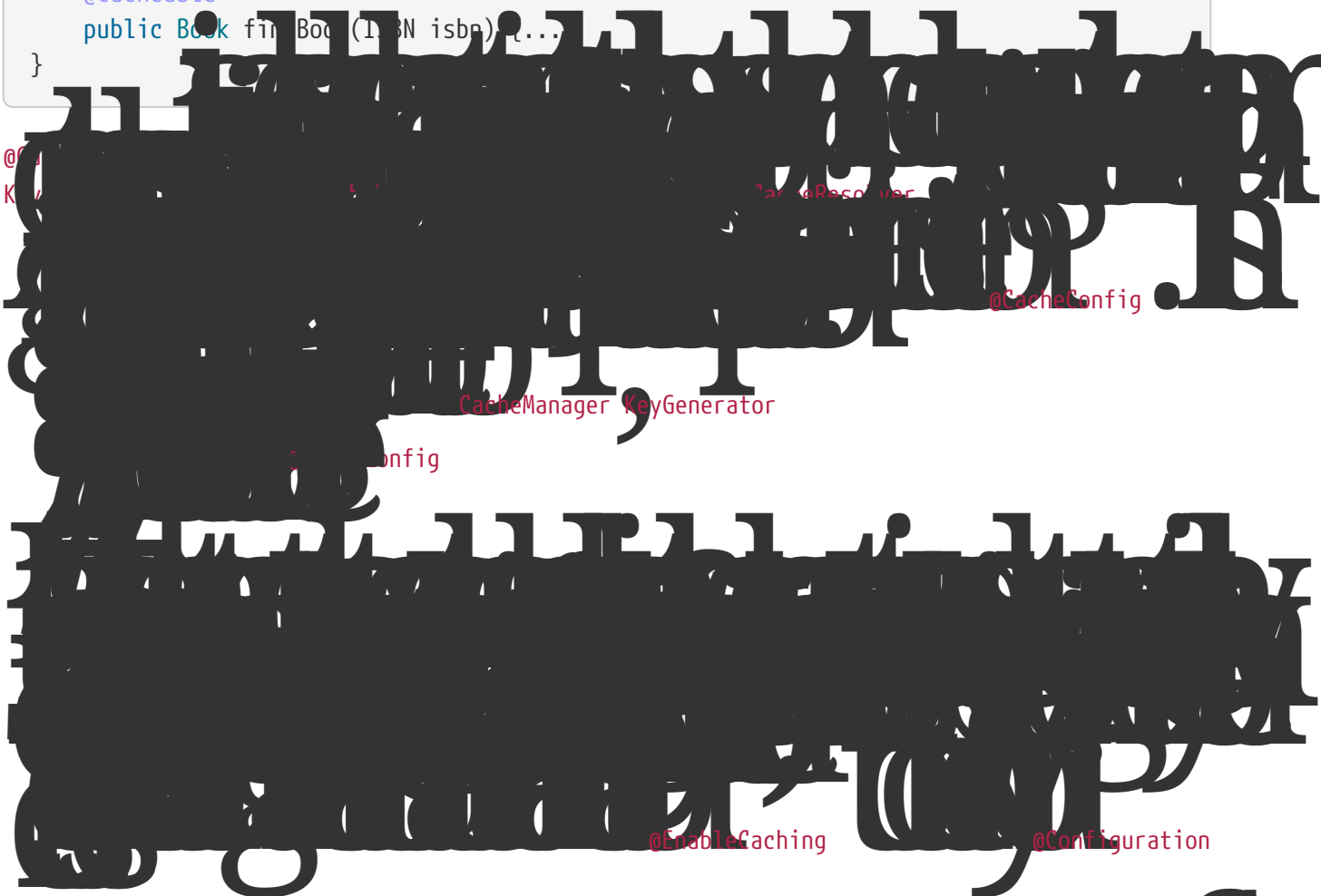
Table … available metadata

	`#root.methodName`
	`#root.method.name`
	`#root.target`
	`#root.targetClass`
	`#root.args[0]`
	`#root.caches[0].name`
argument name	`#`
	`#`
	`#result`
	`O`

`@CachePut`







```
@Configuration
@EnableCaching
public class [illegible] {
}
```

cache:annotation-driven

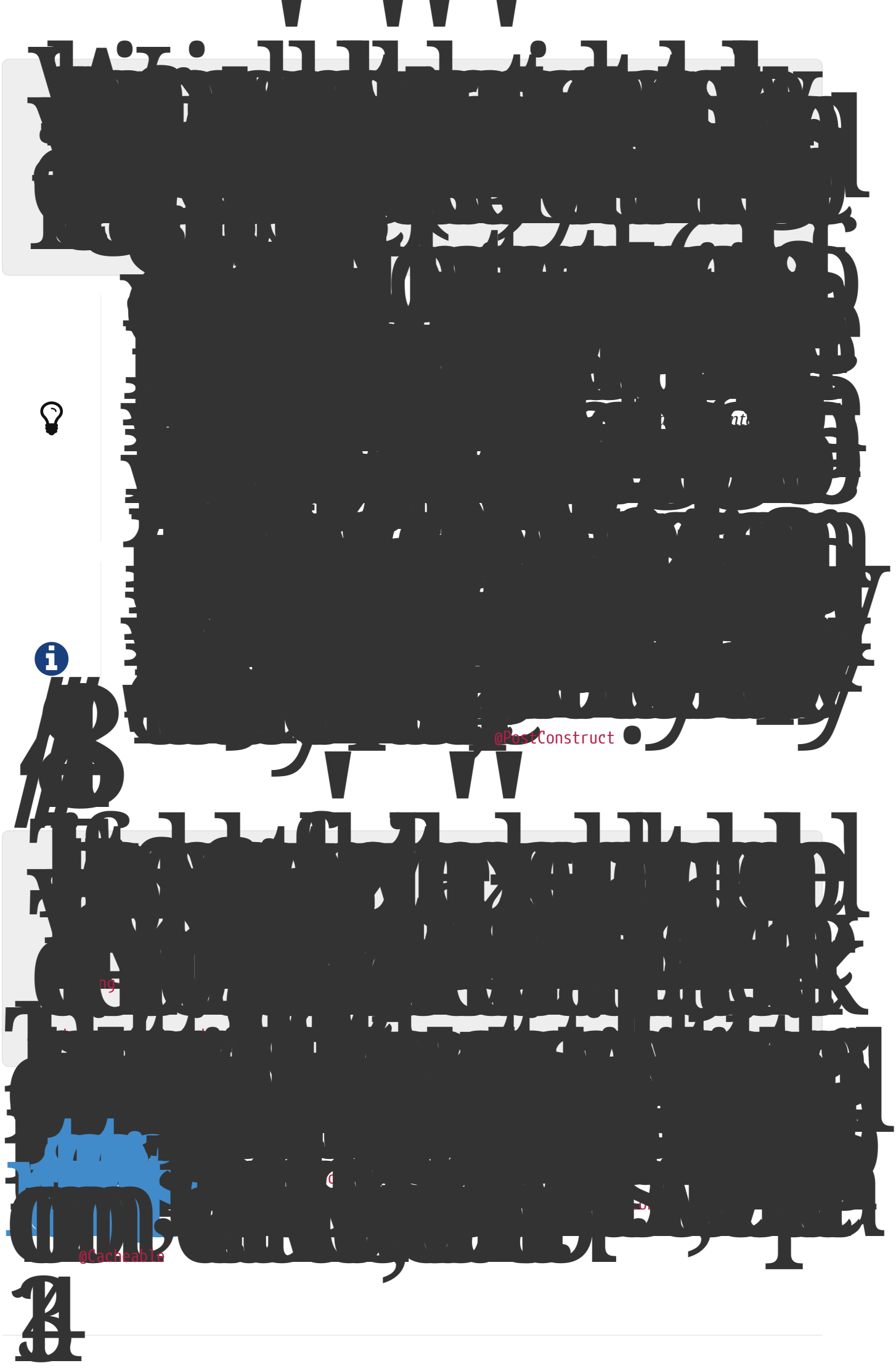
```
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:cache="http://www.springframework.org/schema/cache"
    xsi:schemaLocation="
        http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd
        http://www.springframework.org/schema/cache
http://www.springframework.org/schema/cache/spring-cache.xsd">

    <cache:annotation-driven />
```

CachingConfigurer

Table 12. Cache annotation settings

cache-manager	Cachi[illegible]rer		
cache-resolver	Cachi[illegible]rer	S[illegible]	cacheManager

key-generator	Ca h re	SimpleKeyGener ator	
error-handler	Cachi rer	SimpleCacheErr	
mode	mode		
proxy-target-class	proxyTargetCla ss		
order	order		@


@PostConstruct
@Cacheable

```
@Retention(RetentionPolicy.RUNTIME)
@Target({ElementType.METHOD})
@Cacheable(cacheNames="books", key="#isbn")
public @interface CacheResult {
}
```

```
@Cacheable(cacheNames="books", key="#isbn")
Book findBook(ISBN isbn, boolean checkWarehouse, boolean includeUsed)
```

Table 13. Spring vs. JSR-107 caching annotations

@Cacheable	@CacheResult	

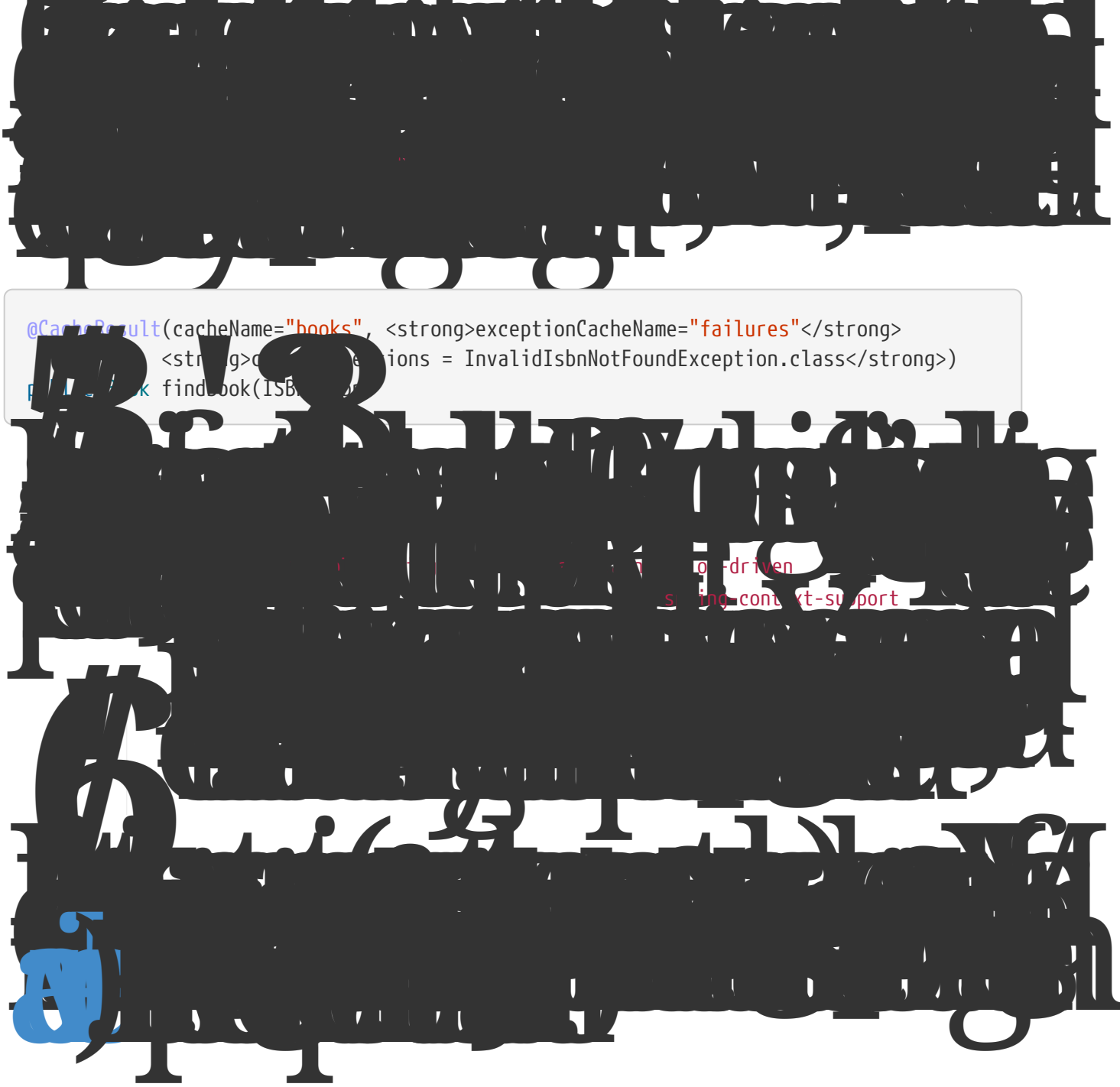
@CachePut	@CachePut	
@CacheEvict	@CacheRemove	
@CacheEvict(allEntries=true)	@CacheRemoveAll	
@CacheDefaults		

```
@CacheResult(cacheNames="books", cacheResolverFactory=MyCacheResolverFactory
.class)
public Book
```

```
@Cacheable(cacheNames="books", key="#isbn")
public Book findBook(ISBN isbn, boolean checkWarehouse, boolean includeUsed)

@CacheResult(cacheName="books")
public Book findBook(@CacheKey ISBN isbn, boolean checkWarehouse,
boolean includeUsed)
```

CacheKeyResolver

CacheResolverFactory



```
<!-- the service we want to make cacheable -->
<bean id="bookService" class="x.y.service.DefaultBookService"/>

<!-- cache definitions -->
<cache:advice id="cacheAdvice" cache-manager="cacheManager">
    <cache:caching cache="books">
        <cache:cacheable method="findBook" key="#isbn"/>
        <cache:cache-evict method="loadBooks" all-entries="true"/>
    </cache:caching>
</cache:advice>

<!-- apply the cacheable behavior to all BookService interfaces -->
<aop:config>
    <aop:advisor advice-ref="cacheAdvice" pointcut="execution(*
x.y.BookService.*(..))"/>
</aop:config>
```

ConcurrentHashMap Cache

```
<!-- simple cache manager -->
<bean id="cacheManager" class="org.springframework.cache.support.SimpleCacheManager">
    <property name="caches">
        <set>
            <bean class=
"org.springframework.cache.concurrent.ConcurrentMapCacheFactoryBean" p:name="default
"/>
            <bean class=
"org.springframework.cache.concurrent.ConcurrentMapCacheFactoryBean" p:name="books"/>
        </set>
    </property>
</
```

gframework.cache.ehcache
CacheManager

```
<bean id="cacheManager"
        class="org.springframework.cache.ehcache.EhCacheCacheManager" p:cache-manager-
ref="ehcache"/>

<!-- EhCache library setup -->
<bean id="ehcache"
      class="org.springframework.cache.ehcache.EhCacheManagerFactoryBean" p:config
```

CacheManager
ehcache.xml

org.springframework.cache.caffeine

CacheManager

```
<bean id="cacheManager" class="
org.springframework.cache.caffeine.CaffeineCacheManager">
    <property name="caches">
        <set>
            <value>default</value>
            <value>books</value>
        </set>
    </property>
</bean>
```

CacheManager

```
<bean id="cacheManager"
        class="org.springframework.cache.jcache.JCacheCacheManager"
        p:cache-manager-ref="jCacheManager"/>
```

```
<bean id="cacheManager" class="
org.springframework.cache.support.CompositeCacheManager">
    <property name="cacheManagers">
        <list>
            <ref bean="jdkCache"/>
            <ref bean="gemfireCache"/>
        </list>
    </property>
    <property name="fallbackToNoOpCache" value="true"/>
</bean>
```

jee

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    <em>xmlns:jee="http://www.springframework.org/schema/jee"</em>
xsi:schemaLocation="
        http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd
        <em>http://www.springframework.org/schema/jee
http://www.springframework.org/schema/jee/spring-jee.xsd"</em>> <!-- bean definitions
here -->

</beans>
```

```
<bean id="<strong>dataSource</strong>"
class="org.springframework.jndi.JndiObjectFactoryBean">
    <property name="jndiName" value="jdbc/MyDataSource"/>
</bean>
<bean id="userDao" class="com.foo.JdbcUserDao">
    <!-- Spring will do the cast automatically (as usual) -->
    <property name="dataSource" ref="<strong>dataSource</strong>"/>
</bean>
```

```
<jee:jndi-lookup id="<strong>dataSource</strong>" jndi-name="jdbc/MyDataSource"/>

<bean id="userDao" class="com.foo.JdbcUserDao">
    <!-- Spring will do the cast automatically (as usual) -->
    <property name="dataSource" ref="<strong>dataSource</strong>"/>
</bean>
```

```
<bean id="simple" class="org.springframework.jndi.JndiObjectFactoryBean">
    <property name="jndiName" value="jdbc/MyDataSource"/>
    <property name="jndiEnvironment">
        <props>
            <prop key="foo">bar</prop>
        </props>
    </property>
</bean>
```

```
<jee:jndi-lookup id="simple" jndi-name="jdbc/MyDataSource">
    <jee:environment>foo=bar</jee:environment>
</jee:jndi-lookup>
```

```
<bean id="simple" class="org.springframework.jndi.JndiObjectFactoryBean">
    <property name="jndiName" value="jdbc/MyDataSource"/>
    <property name="jndiEnvironment">
        <props>
            <prop key="foo">bar</prop>
            <prop key="ping">pong</prop>
        </props>
    </property>
</bean>
```

```
<jee:jndi-lookup id="simple" jndi-name="jdbc/MyDataSource">
    <!-- newline-separated, key-value pairs for the environment (standard Properties
format) -->
    <jee:environment>
        foo=bar
        ping=pong
    </jee:environment>
</jee:jndi-lookup>
```

```
<bean id="simple" class="org.springframework.jndi.JndiObjectFactoryBean">
    <property name="jndiName" value="jdbc/MyDataSource"/>
    <property name="cache" value="true"/>
    <property name="resourceRef" value="true"/>
    <property name="lookupOnStartup" value="false"/>
    <property name="expectedType" value="com.myapp.DefaultFoo"/>
    <property name="proxyInterface" value="com.myapp.Foo"/>
</bean>
```

```
<jee:jndi-lookup id="simple"
        jndi-name="jdbc/MyDataSource"
        cache="true"
        resource-ref="true"
        lookup-on-startup="false"
        expected-type="com.myapp.DefaultFoo"
        proxy-interface="com.myapp.Foo"/>
```

```
<bean id="simple"
        class="org.springframework.ejb.access.LocalStatelessSessionProxyFactoryBean">
    <property name="jndiName" value="ejb/RentalServiceBean"/>
    <property name="businessInterface" value="com.foo.service.RentalService"/>
</bean>
```

```
<jee:local-slsb id="simpleSlsb" jndi-name="ejb/RentalServiceBean"
        business-interface="com.foo.service.RentalService"/>
```

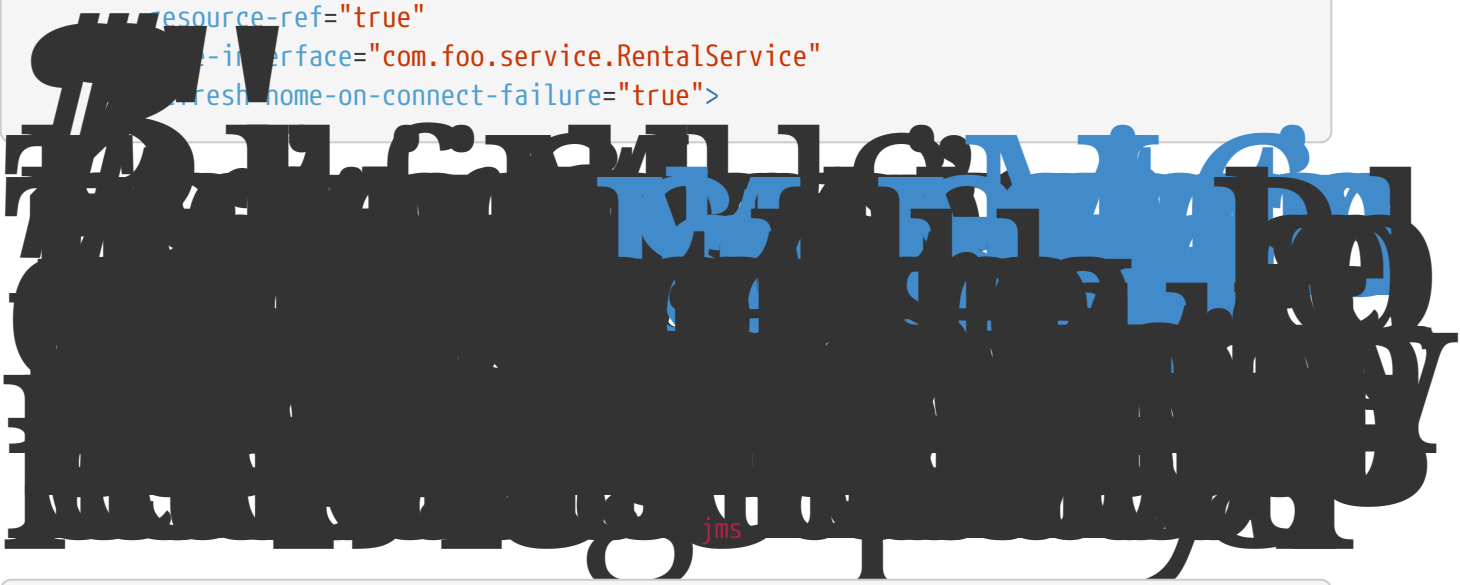
.9 -

```
<bean id="complexLocalEjb"
        class="org.springframework.ejb.access.LocalStatelessSessionProxyFactoryBean">
    <property name="jndiName" value="ejb/RentalServiceBean"/>
    <property name="businessInterface" value="com.foo.service.RentalService"/>
    <property name="cacheHome" value="true"/>
    <property name="lookupHomeOnStartup" value="true"/>
    <property name="resourceRef" value="true"/>
</bean>
```

```
<jee:local-slsb id="complexLocalEjb"
        jndi-name="ejb/RentalServiceBean"
        business-interface="com.foo.service.RentalService"
        cache-home="true"
        lookup-home-on-startup="true"
        resource-ref="true">
</jee:local-slsb>
```

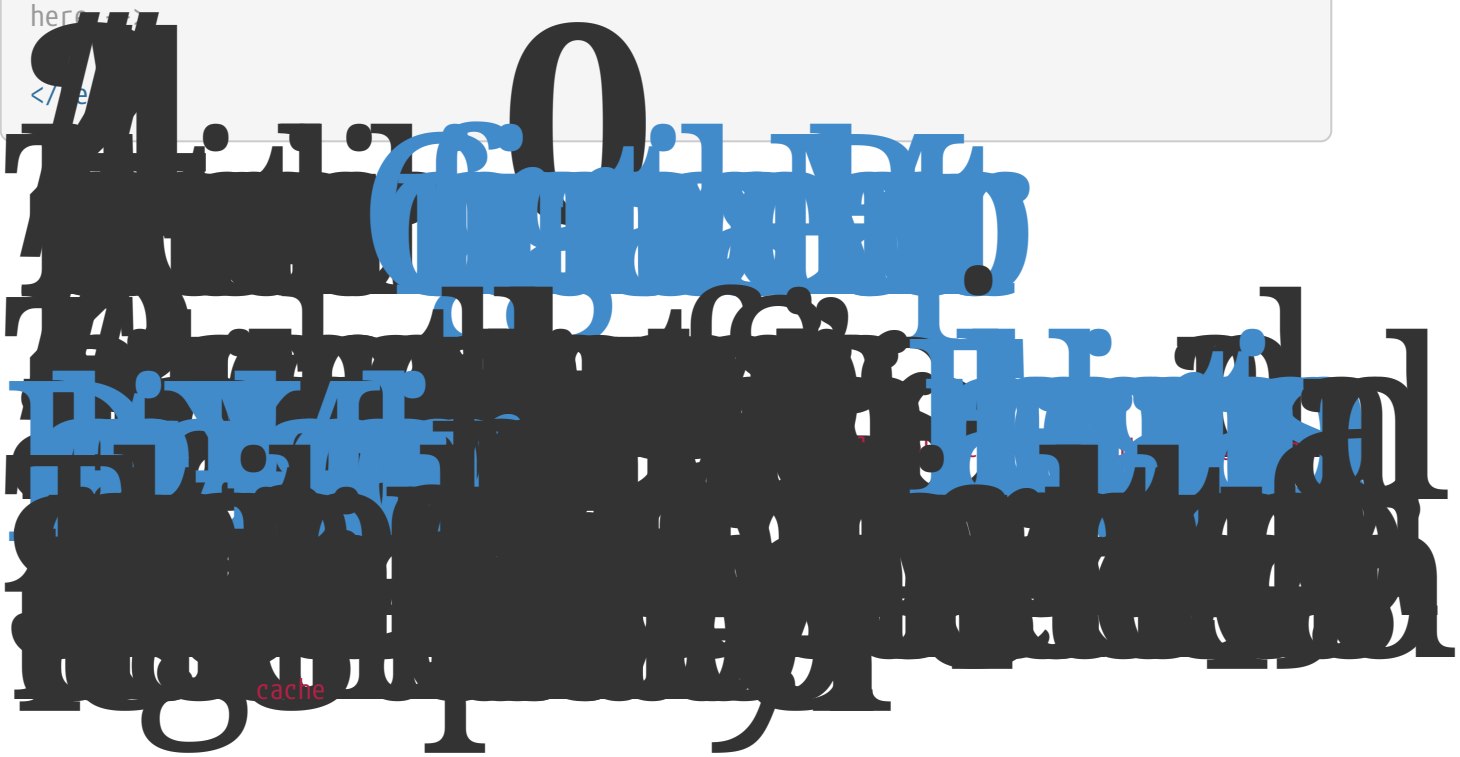
```
<bean id="complexRemoteEjb"
        class=
"org.springframework.ejb.access.SimpleRemoteStatelessSessionProxyFactoryBean">
    <property name="jndiName" value="ejb/MyRemoteBean"/>
    <property name="businessInterface" value="com.foo.service.RentalService"/>
    <property name="cacheHome" value="true"/>
    <property name="lookupHomeOnStartup" value="true"/>
    <property name="resourceRef" value="true"/>
    <property name="homeInterface" value="com.foo.service.RentalService"/>
    <property name="refreshHomeOnConnectFailure" value="true"/>
</bean>
```

```
<jee:remote-slsb id="complexRemoteEjb"
        jndi-name="ejb/MyRemoteBean"
        business-interface="com.foo.service.RentalService"
        cache-home="true"
        lookup-home-on-startup="true"
        resource-ref="true"
        -interface="com.foo.service.RentalService"
        refresh-home-on-connect-failure="true">
```



jms

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    <em>xmlns:jms="http://www.springframework.org/schema/jms"</em>
xsi:schemaLocation="
        http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd
        <em>http://www.springframework.org/schema/jms
http://www.springframework.org/schema/jms/spring-jms.xsd"</em>> <!-- bean definitions
here -->
</beans>
```



cache

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    <em>xmlns:cache="http://www.springframework.org/schema/cache"</em>
xsi:schemaLocation="
        http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd
        <em>http://www.springframework.org/schema/cache
http://www.springframework.org/schema/cache/spring-cache.xsd"</em>> <!-- bean
definitions here -->

</beans>
```