



Spring : tendances, nouveautés et perspectives

Speaker

- Arnaud Cogoluègnes
- Consultant chez Zenika
- Formateur certifié SpringSource
- Co-auteur
 - Spring Batch in Action
 - Spring Dynamic Modules in Action
 - Spring par la pratique, 2nde édition

Un peu d'histoire...



- v. 0.9 : juin 2003
- v. 1.0 : mars 2004
- v. 1.1 : septembre 2004
- v. 1.2 : mai 2005

Un peu d'histoire...



- v. 2.0 : octobre 2006
 - namespaces XML
- v. 2.5 : novembre 2007
 - annotations
- v. 3.0 : décembre 2009
 - Java config, SpEL, REST
- v. 3.1 : décembre 2011
 - profils, cache, Java config++

Une vision, des idées



Simplification

Injection de dépendances

Programmation orientée aspect

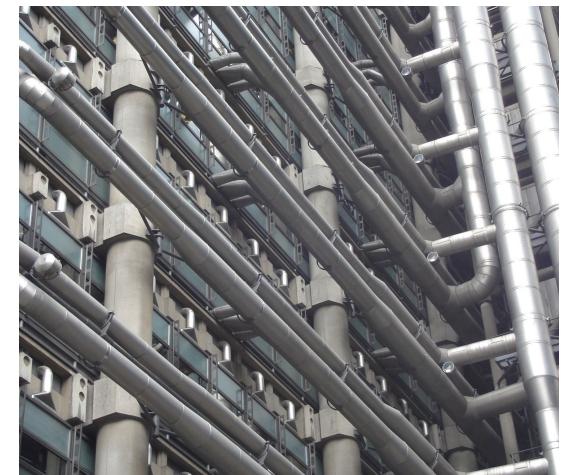
Une vision ?



« Gérer la plomberie »



```
public class TransferService {  
  
    public void transfer(...) {  
        openConnection();  
        beginTransaction();  
  
        // code applicatif  
  
        commitTransaction();  
        closeConnection();  
    }  
  
}
```



```
public class TransferService {  
    public void transfer(...) {  
        // code applicatif  
    }  
}
```

+ AOP



Simplicité

Réutilisation

Testabilité

Changement => adaptation



```
<aop:config>
    <aop:advisor advice-ref="txAdvice"
                  pointcut="execution(* com.zenika.service.*.*(..))"/>
</aop:config>

<tx:advice id="txAdvice">
    <tx:attributes>
        <tx:method name="update*" />
    </tx:attributes>
</tx:advice>
```

```
public class TransferService {

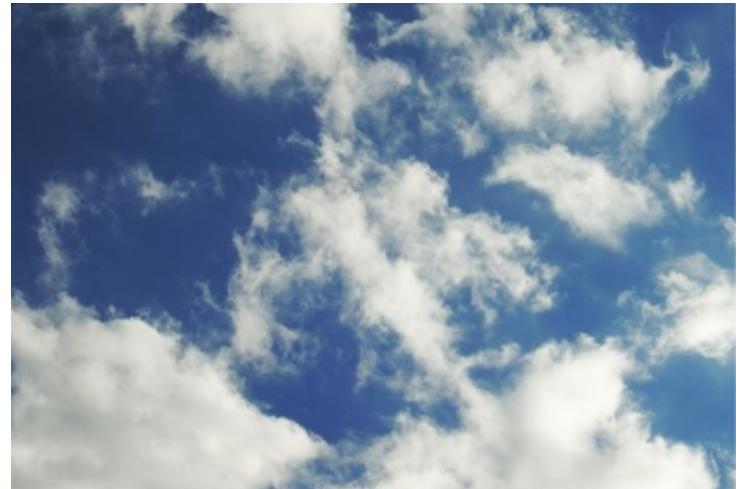
    public void transfer(...) { }

}
```

```
<tx:annotation-driven />
```

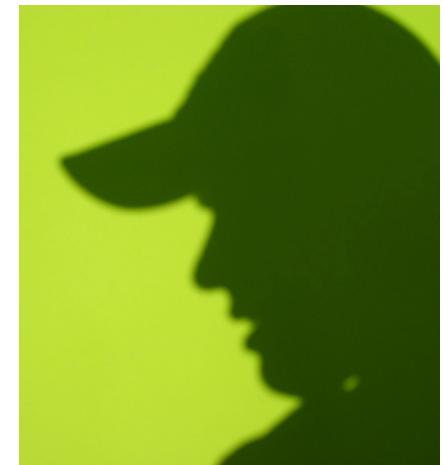
```
public class TransferService {  
    @Transactional  
    public void transfer(...) { }  
}
```

Exécution, où ?



Profils

```
<beans profile="test">  
  ...  
</beans>  
  
<beans profile="prod">  
  ...  
</beans>  
  
<beans profile="cloud">  
  ...  
</beans>
```



Profils

```
<beans profile="test">
  <jdbc:embedded-database id="dataSource" type="H2" />
</beans>
```



```
<beans profile="prod">
  <jee:jndi-lookup id="ds"
    jndi-name="java:comp/env/jdbc/ds" />
</beans>
```

```
<beans profile="cloud">
  <cloud:data-source id="dataSource"
    service-name="contactDs"/>
</beans>
```



Activation des profils

- Ligne de commande

```
-Dspring.profiles.active="prod"
```

- Programmatiquement

```
GenericXmlApplicationContext ctx =  
    new GenericXmlApplicationContext();  
ctx.getEnvironment(). setActiveProfiles("test");  
ctx.load("classpath:/application-context.xml");  
ctx.refresh();
```

Où suis-je ?

```
public void initialize(ConfigurableApplicationContext ctx) {  
    CloudEnvironment cloud = new CloudEnvironment();  
    if(cloud.getInstanceInfo() == null) {  
        ctx.getEnvironment().setActiveProfiles("default");  
    } else {  
        ctx.getEnvironment().setActiveProfiles("cloud");  
    }  
}
```



Solutions de configuration

<XML />

Solution « historique »

Externe Centralisée

Non-intrusive

Limitée Namespaces

Java

Externe

Puissante

Java !

~~-Pas de namespaces~~



@annotations

Rapide Type-safe

Intrusive

Décentralisée

<XML />

```
<bean id="transferService"
      class="com.zenika.service.TransferServiceImpl">
    <constructor-arg ref="accountRepo" />
</bean>

<bean id="accountRepo"
      class="com.zenika.repo.JdbcAccountRepository">
    <constructor-arg ref="dataSource" />
</bean>
```

@annotations

```
@Service
public class TransferServiceImpl {

    @Autowired private AccountRepository accountRepository;

}

@Repository
public class JdbcAccountRepository implements AccountRepository {

    @Autowired private DataSource ds;

}

<context:component-scan base-package="com.zenika" />
```

Configuration Java

```
@Configuration
public class AppConfig {

    @Autowired private DataSource ds;

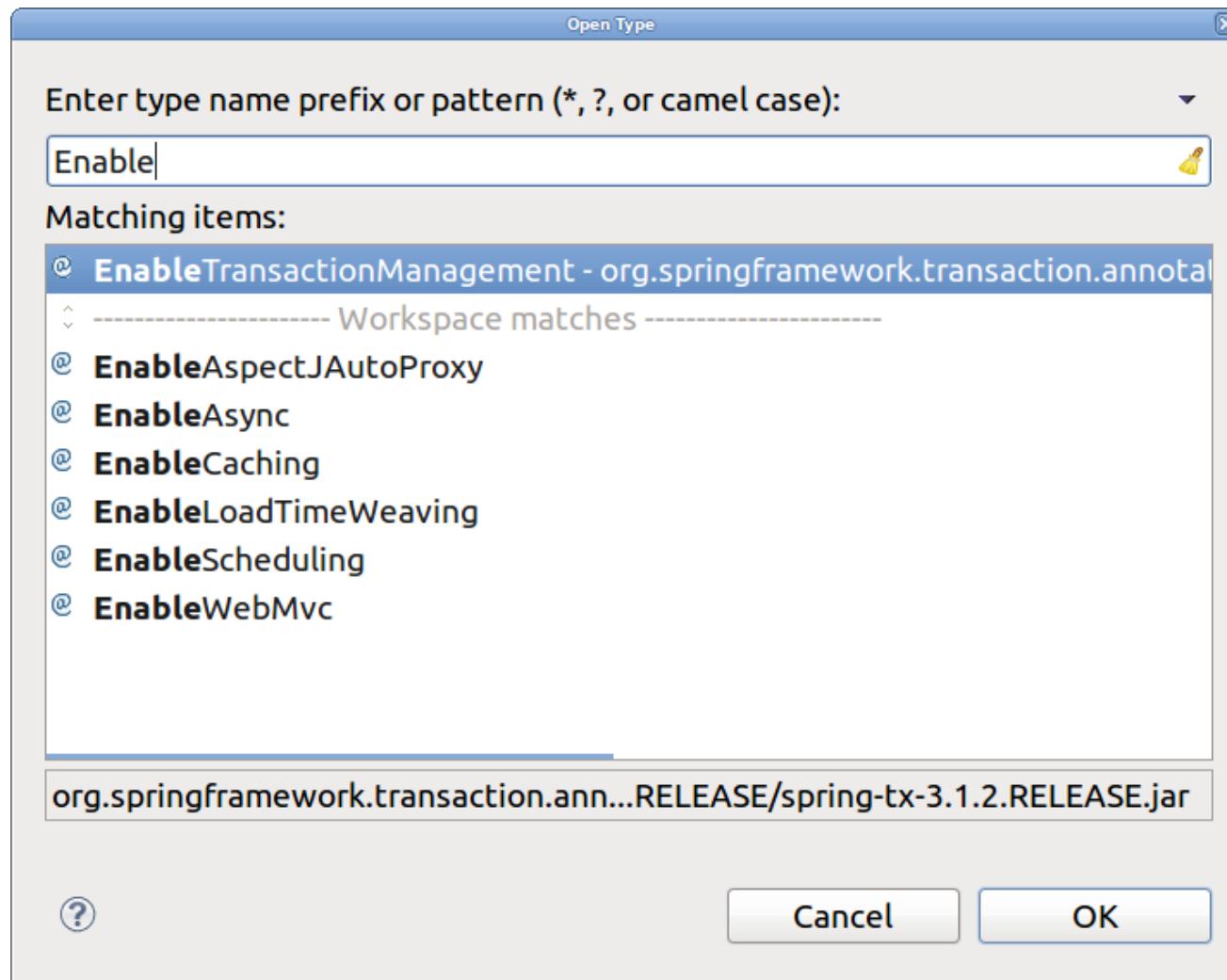
    @Bean public TransferService transferService() {
        return new TransferServiceImpl(accountRepo());
    }

    @Bean public AccountRepository accountRepo() {
        return new JdbcAccountRepository(ds);
    }
}
```

@Enable* <=> namespaces

```
@Configuration  
@EnableTransactionManagement  
public class AppConfig {  
  
    @Autowired private DataSource ds;  
  
    @Bean public TransferService transferService() {  
        return new TransferServiceImpl(accountRepo());  
    }  
  
    @Bean public AccountRepository accountRepo() {  
        return new JdbcAccountRepository(ds);  
    }  
}
```

@Enable* <=> namespaces



Solutions de configuration



Solutions de configuration

- Chacune avec ses avantages/inconvénients
- Iso-fonctionnelles
- Prendre la plus adaptée à ses besoins
- Bonne cohabitation dans une même application

REST

SOA

ROA

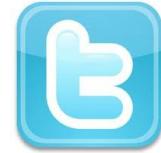
HTTP

Alternative

Protocole applicatif

Simple

JSON



elasticsearch.

REST

```
GET /crud-rest/zen-contact/contacts/1 HTTP/1.1  
Accept: application/json  
Host: localhost:8080
```



```
HTTP/1.1 200 OK  
Content-Type: application/json  
{"id":1,"firstname":"Joe","lastname":"Dalton","age":37}
```

```
@Controller  
public class ContactController {  
  
    @Autowired ContactRepository contactRepository;  
  

```



REST

```
GET /crud-rest/zen-contact/contacts/1 HTTP/1.1  
Accept: application/json  
Host: localhost:8080
```



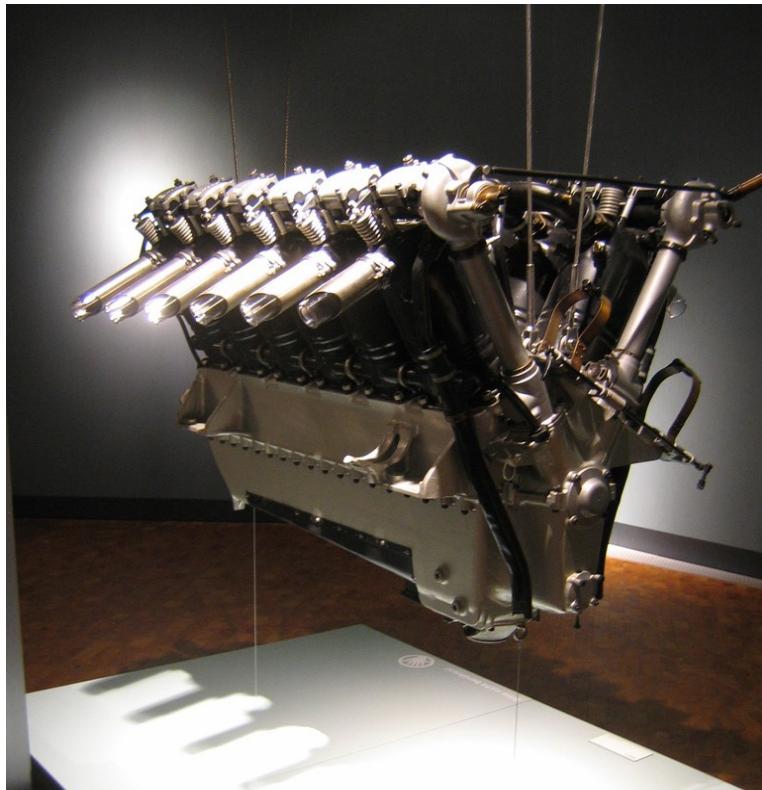
```
HTTP/1.1 200 OK  
Content-Type: application/json  
{"id":1,"firstname":"Joe","lastname":"Dalton","age":37}
```

```
@Controller  
public class ContactController {  
  
    @Autowired ContactRepository contactRepository;  
  

```



Changement ?



Refactoring complet entre 3.0 et 3.1

Traitement des méthodes
des contrôleurs

100% compatible

Nouveaux points d'extension

Spring : stabilité, compatibilité

The screenshot shows the Eclipse IDE interface with several windows open:

- Java - spring-demo/src/test/resources/spring-old-school.xml - Eclipse**: The main editor window containing the XML configuration file.
- Outline**: Shows the structure of the XML file with nodes like `<beans>`, `<bean id="transferServiceTarget"`, and `<bean id="transferService"`.
- JUnit**: A results window showing "Runs: 1/1", "Errors: 0", and "Failures: 0". It lists a single run named "com.zenika.SpringOldSchoolTest [Run]" with a duration of "transactional (0.648 s)".
- Type Hier**: A tree view showing the class hierarchy or type structure.
- Package Explorer**: Shows the project structure with various Spring jars listed: spring-tx-3.1.3.RELEASE.jar, aopalliance-1.0.jar, spring-aop-3.1.3.RELEASE.jar, spring-beans-3.1.3.RELEASE.jar, spring-context-3.1.3.RELEASE.jar, spring-core-3.1.3.RELEASE.jar, commons-logging-1.1.1.jar, spring-jdbc-3.1.3.RELEASE.jar, spring-orm-3.1.3.RELEASE.jar, spring-webmvc-3.1.3.RELEASE.jar, spring-asn-3.1.3.RELEASE.jar, spring-context-support-3.1.3.RELEASE.jar, and spring-expression-3.1.3.RELEASE.jar.
- spring-old-school.xml**: A secondary editor window showing the same XML configuration file.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE beans PUBLIC "-//SPRING//DTD BEAN//EN"
  "http://www.springframework.org/dtd/spring-beans.dtd">
<beans>

  <bean id="transferServiceTarget" class="com.zenika."
    <constructor-arg ref="accountRepo" />
  </bean>

  <bean id="transferService"
    class="org.springframework.transaction.intercep
    <property name="transactionManager" ref="transa
    <property name="target" ref="transferServiceTar
    <property name="transactionAttributes">
      <props>
        <prop key="*">>PROPAGATION_REQUIRED</pro
      </props>
    </property>
  </bean>

```

```
<?xml version="1.0" encoding="UTF-8"
<!DOCTYPE beans PUBLIC "-//SPRING//"
  "http://www.springframework.or
<beans>

  <bean id="transferServiceTarget"
    <constructor-arg ref="acco
  </bean>

  <bean id="transferService"
    class="org.springframework.
    <property name="transaction
    <property name="target" ref=
    <property name="transaction
      <props>
        <prop key="*">>PROPA
      </props>
    </property>
  </bean>

  <bean id="dataSource"
    class="org.springframework.
    <property name="driverClass
    <property name="url" value=
    <property name="username" v
    <property name="password" v
    <property name="suppressClc
  </bean>

```

Spring MVC : extensibilité

```
@Controller
public class ContactController {

    @Autowired ContactRepository contactRepository;

    @RequestMapping(value="/contacts/{id}",method=RequestMethod.GET)
    public ResponseEntity<Contact> contact(@Domain Contact contact) {
        ResponseEntity<Contact> response = new ResponseEntity<Contact>(
            contact,
            contact == null ? HttpStatus.NOT_FOUND : HttpStatus.OK
        );
        return response;
    }
}
```

Spring MVC : extensibilité

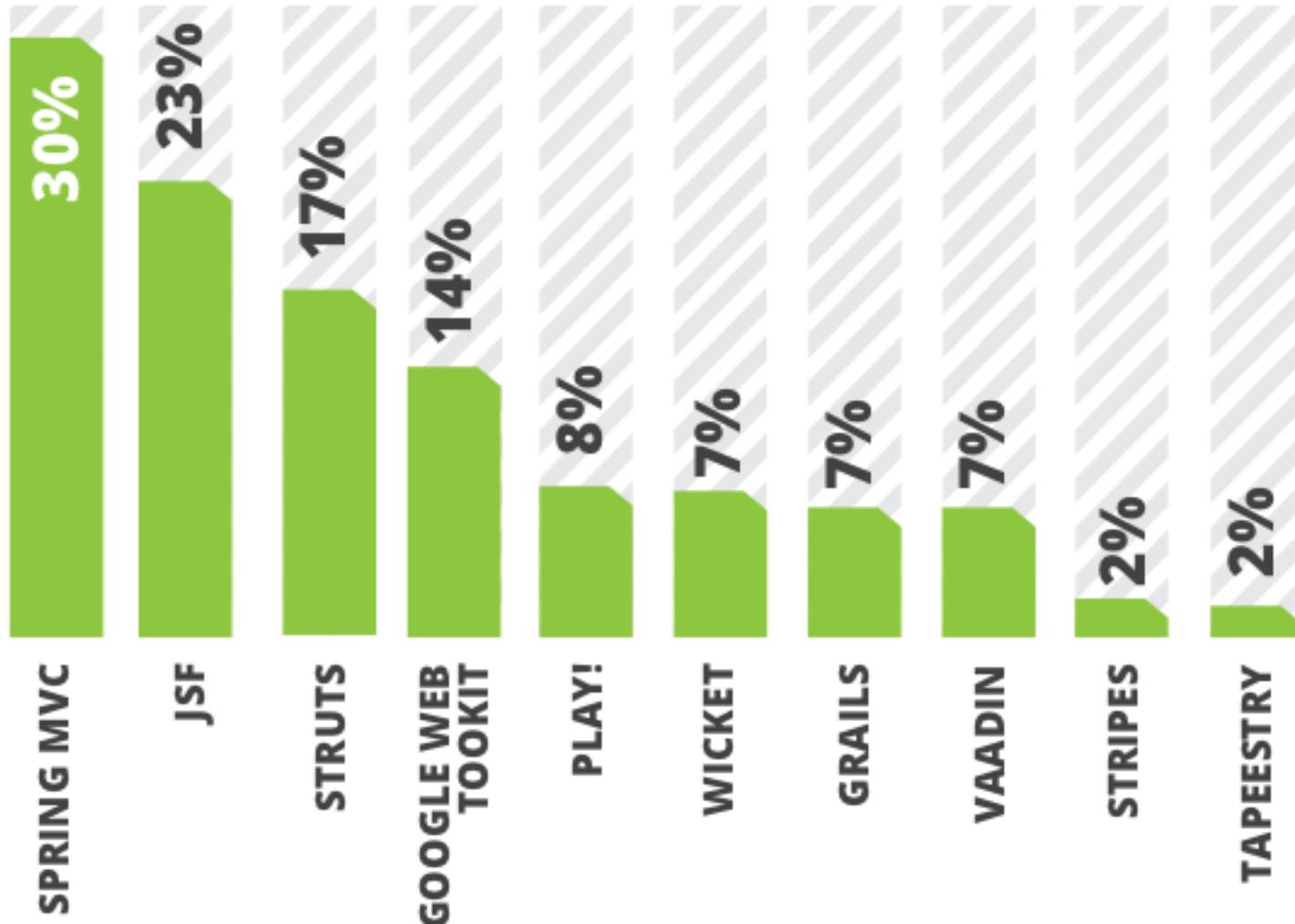
```
@Controller
public class ContactController {

    @Autowired ContactRepository contactRepository;

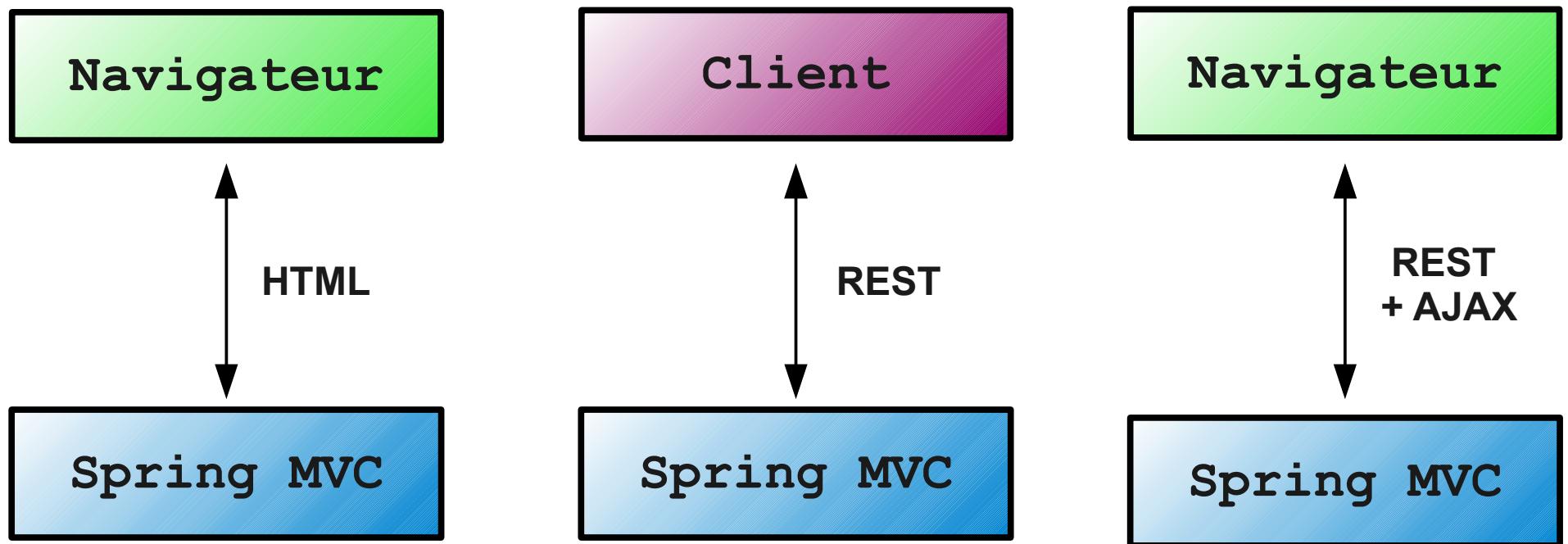
    @RequestMapping(value="/contacts/{id}",method=RequestMethod.GET)
    public ResponseEntity<Contact> contact(@Domain Contact contact) {
        ResponseEntity<Contact> response = new ResponseEntity<Contact>(
            contact,
            contact == null ? HttpStatus.NOT_FOUND : HttpStatus.OK
        );
        return response;
    }
}
```



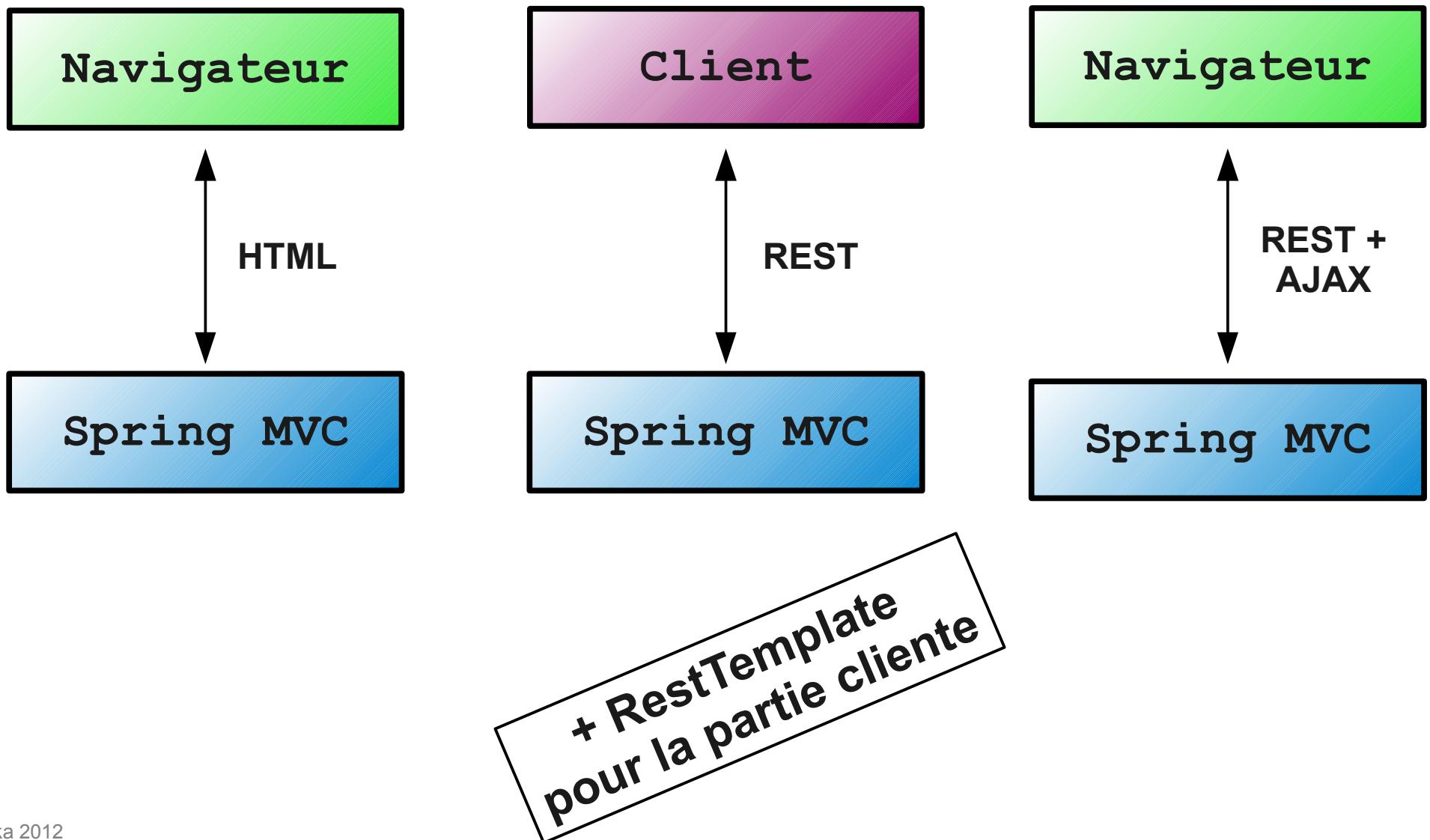
Spring MVC et popularité



Utilisations de Spring MVC



Utilisations de Spring MVC



Test

```
@RunWith(SpringJUnit4ClassRunner.class)
@ContextConfiguration("/spring-jpa.xml")
public class AccountRepositoryTest {

    @Autowired private AccountRepository repo;

    @Test public void save() {
        // test...
    }

}
```

Test

```
@RunWith(SpringJUnit4ClassRunner.class)
@ContextConfiguration("/spring-jpa.xml")
public class AccountRepositoryTest {

    @Autowired private AccountRepository repo;

    @Test public void save() {
        // test...
    }

}
```

Test + profils

```
@RunWith(SpringJUnit4ClassRunner.class)
@ContextConfiguration("/spring-jpa.xml")
@ActiveProfiles("test")
public class AccountRepositoryTest {

    @Autowired private AccountRepository repo;

    @Test public void save() {
        // test...
    }
}
```



Test + Java config

```
@RunWith(SpringJUnit4ClassRunner.class)
@ContextConfiguration
public class AccountRepositoryTest {

    @Autowired private AccountRepository repo;

    @Test public void save() {
        // test...
    }

    @Configuration
    public static class SpringJpaConfiguration {

        @Bean public DataSource ds() { ... }

    }
}
```



Spring MVC test « no container »

```
@RunWith(SpringJUnit4ClassRunner.class)
@WebAppConfiguration
@ContextConfiguration
public class MetaModelControllerTest {

    @Autowired
    private WebApplicationContext wac;

    private MockMvc mockMvc;

    @Before
    public void setUp() {
        this.mockMvc = MockMvcBuilders.webAppContextSetup(this.wac).build();
    }

    @Test public void list() throws Exception {
        ....
    }
}
```

Spring MVC test « no container »

```
@RunWith(SpringJUnit4ClassRunner.class)
@WebAppConfiguration
@ContextConfiguration
public class MetaModelControllerTest {

    @Autowired
    private WebApplicationContext wac;

    private MockMvc mockMvc;

    @Before
    public void setUp() {
        this.mockMvc = MockMvcBuilders.webAppContextSetup(this.wac).build();
    }

    @Test public void list() throws Exception {
        ....
    }
}
```

Spring MVC test « no container »

```
(...)
public class MetaModelControllerTest {
    (...)

    @Test public void list() throws Exception {
        mockMvc.perform(get("/metamodel")).accept(MediaType.APPLICATION_JSON)
            .andExpect(status().isOk())
            .andExpect(content().contentType("application/json; charset=UTF-8"))
            .andExpect(jsonPath("$.name").value("NAME1"))
            .andExpect(jsonPath("$.name").value("NAME2"));
    }

    @Test public void createOk() throws Exception {
        mockMvc.perform(post("/metamodel")
            .content(format(object(field("name", string("NEW")))))
            .contentType(MediaType.APPLICATION_JSON))
            .andExpect(status().isCreated())
            .andExpect(
                header().string("Location", "http://localhost/metamodel/NEW")
            );
    }
}
```

Spring MVC test « no container »

```
(...)
public class MetaModelControllerTest {
    (...)

    @Test public void list() throws Exception {
        mockMvc.perform(get("/metamodel")).accept(MediaType.APPLICATION_JSON)
            .andExpect(status().isOk())
            .andExpect(content().contentType("application/json; charset=UTF-8"))
            .andExpect(jsonPath("$.name").value("NAME1"))
            .andExpect(jsonPath("$.name").value("NAME2"));
    }

    @Test public void createOk() throws Exception {
        mockMvc.perform(post("/metamodel")
            .content(format(object(field("name", string("NEW")))))
            .contentType(MediaType.APPLICATION_JSON))
            .andExpect(status().isCreated())
            .andExpect(
                header().string("Location", "http://localhost/metamodel/NEW")
            );
    }
}
```

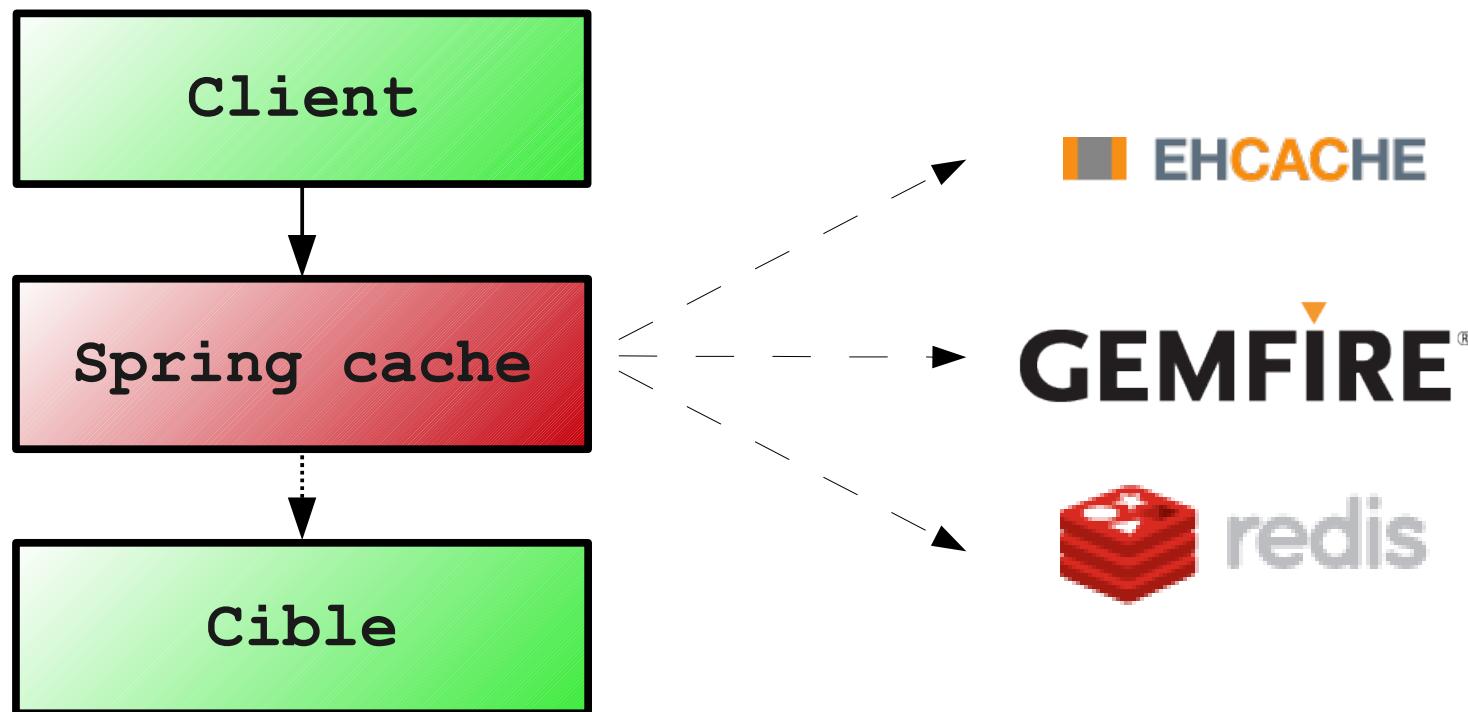


Cache

```
public class ContactRepository {  
  
    @Cacheable("contacts")  
    Contact findOne(Long id) { ... }  
  
}
```

```
<cache:annotation-driven />  
  
<bean id="cacheManager"  
      class="... ">  
  ...  
</bean>
```

Cache



Portfolio Spring

The screenshot shows a web browser displaying the SpringSource.org projects page. The page has a green header with the SpringSource logo and navigation links for FEATURES, GET STARTED, GET INVOLVED, NEWS & EVENTS, and a search bar. Below the header, there's a section titled "SPRING PROJECTS" with a "GET STARTED" button. The main content area lists several Spring projects:

- Spring Framework**: Spring is a layered Java application platform for building enterprise solutions and is based on code published in the book *Expert One-on-One J2EE Design and Development* by Rod Johnson. [Go to...](#)
- Spring Security**: Spring Security is a powerful and highly customizable authentication and access-control framework that works efficiently in Spring applications with just a few lines of configuration. [Go to...](#)
- Spring Roo**: Spring Roo is a next-generation rapid application development tool for Java developers. With Roo you can easily build in minutes full Java applications including data access, search and rich web interactions. [Go to...](#)
- Spring Mobile**: Spring Mobile is an extension to Spring MVC that aims to simplify the development of mobile web applications including intelligent device detection and progressive rendering options. [Go to...](#)
- Spring Data**: Spring Data makes it easier to build Spring-powered applications that use new data access technologies like MongoDB, Neo4J, Redis, and other noSQL data stores. [Go to...](#)
- Spring Web Services**: Spring Web Services focuses on creating document-driven Web services using a contract-first SOAP service development approach. [Go to...](#)
- Spring Batch**: Spring Batch is a lightweight, comprehensive batch framework designed to simplify and optimize the work of processing high volume batch operations. [Go to...](#)
- Spring Integration**: Spring Integration extends Spring to support the well-known *Enterprise Integration Patterns* via lightweight messaging within Spring-based applications and supports integration with external systems via declarative adapters. [Go to...](#)
- Spring AMQP**: Spring AMQP applies core Spring concepts to the development of AMQP-based messaging solutions including abstractions for sending, receiving and configuring AMQP messages. [Go to...](#)
- Spring Social**: Spring Social is an extension of the Spring Framework that allows you to connect your applications with Software-as-a-Service (SaaS) providers such as Facebook and Twitter. [Go to...](#)
- Spring Web Flow**: Spring Web Flow focuses on providing the infrastructure for building and running rich web applications that require multi-step interactions. [Go to...](#)
- Spring BlazeDS Integration**: Spring BlazeDS Integration makes it easier to build Spring-powered Rich Internet Applications using Adobe Flex as the front-end client. [Go to...](#)

Portfolio Spring

Spring Mobile / Spring Android



Spring Social

Spring Data



Spring Data JPA

Métier

Accès données

JPA

DAO dynamiques, intelligents

Moins de code répétitif

Testabilité

Conventions

Pagination & tri

Approche DDD

Spring Data JPA

```
public interface ContactRepository  
    extends JpaRepository<Contact, Long>{ }
```

```
<jpa:repositories base-package="com.zenika.repository" />
```

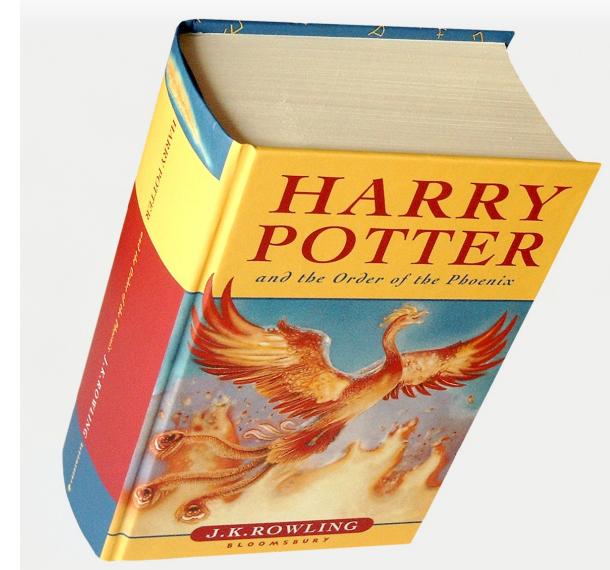
```
List<Contact> cts = repo.findAll();
```



Spring Data JPA

```
public interface ContactRepository  
    extends JpaRepository<Contact, Long>{  
  
    List<Contact> findByLastname(String lastname);  
}
```

```
List<Contact> cts = repo.findByLastname("Potter");
```



Spring Data MongoDB

- DAO dynamiques
- MongoTemplate
- Mapping objet-document
- DSL requête et mise à jour



Spring Data MongoDB

```
public interface ContactRepository  
    extends MongoRepository<Contact, String> {  
  
    List<Contact> findByFirstname(String firstname);  
  
}
```

```
<mongo:repositories base-package="com.zenika.repository" />
```

```
List<Contact> cts = repo.findAll();
```

```
List<Contact> cts = repo.findByLastname("Potter");
```

Spring Data MongoDB, API query

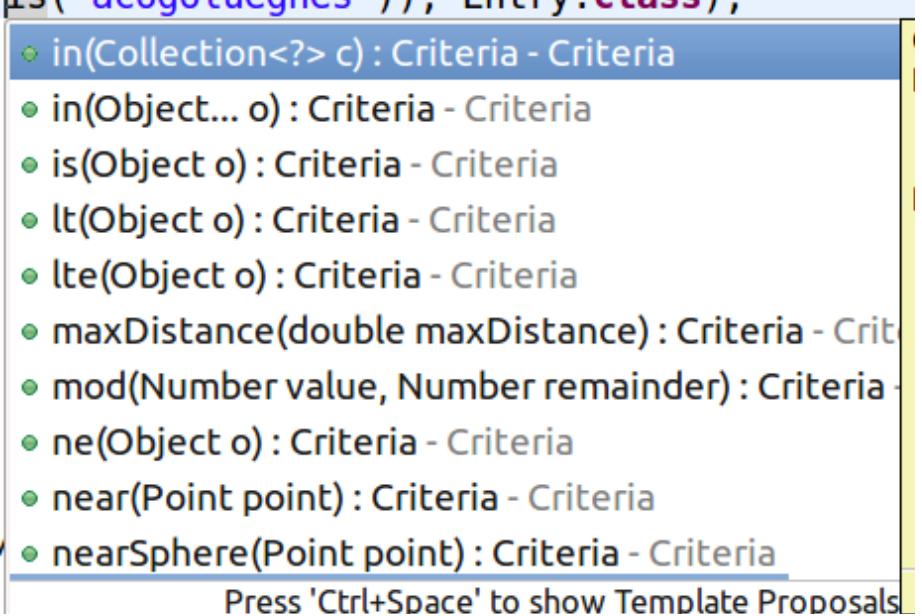
```
import static org.springframework.data.mongodb.core.query.Criteria.*;  
import static org.springframework.data.mongodb.core.query.Query.query;  
  
List<Entry> entries = mongoOps.find(  
    query(where("tags").size(2)), Entry.class  
);
```

Spring Data MongoDB, API query

```
import static org.springframework.data.mongodb.core.query.Criteria.*;  
import static org.springframework.data.mongodb.core.query.Query.query;  
  
List<Entry> entries = mongoOps.find(  
    query(where("tags").size(2)  
          .and("author").is("acogoluegnes")) ,  
    Entry.class  
);
```

Spring Data MongoDB, API query

```
entries = mongoOps.find(
    query(where("tags").size(2)
        .and("author").is("acogoluegnes"))), Entry.class);
Assert.assertEquals(1, entries.size());
entries = mongoOps.find(query(
    .and("author").is("acogoluegnes"))), Entry.class);
Assert.assertEquals(0, entries.size());
```

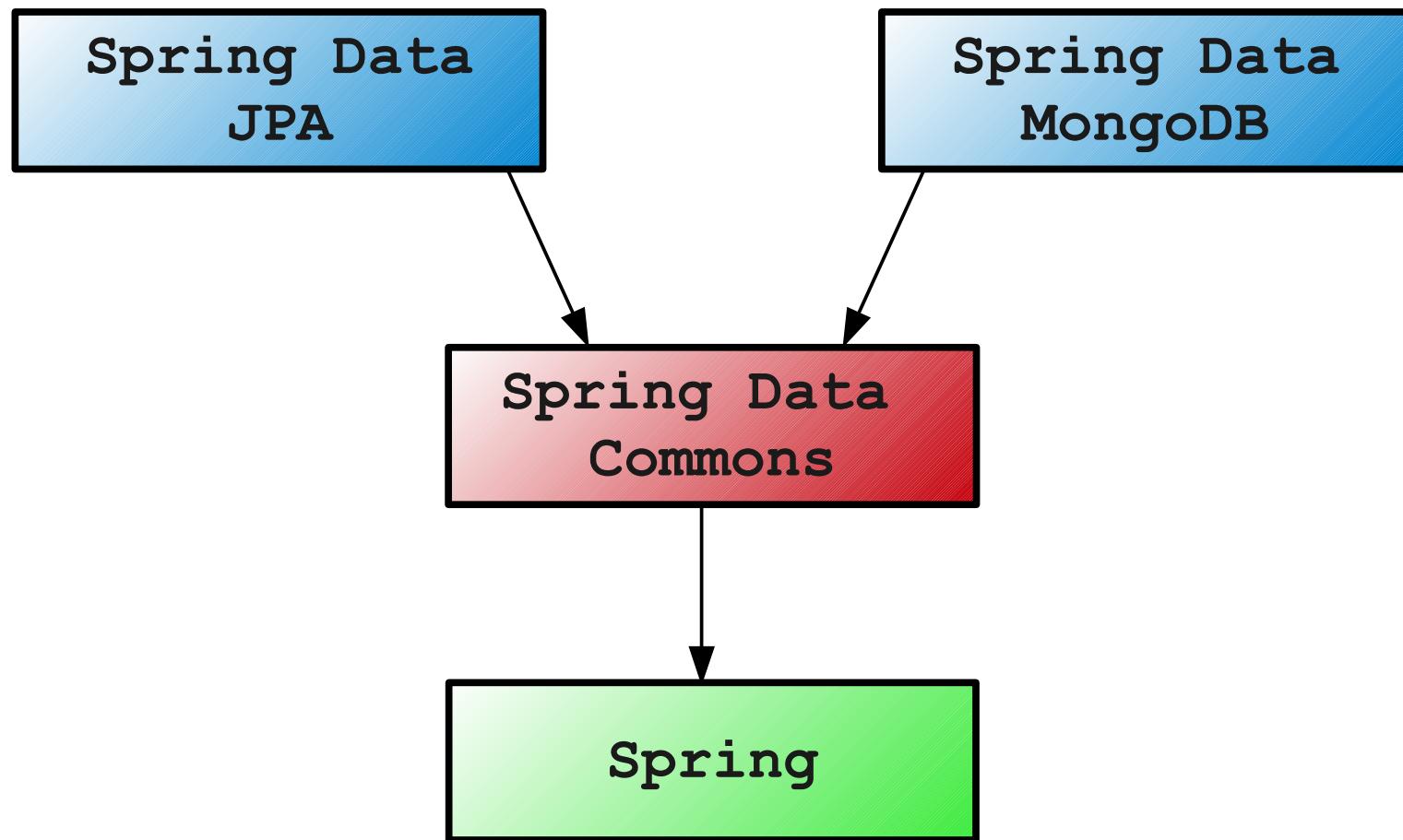


The screenshot shows an IntelliJ IDEA code editor with a code completion dropdown open over the line ".is("acogoluegnes"))". The dropdown lists several methods from the Criteria class, all starting with 'is':

- in(Collection<?> c) : Criteria - Criteria
- in(Object... o) : Criteria - Criteria
- is(Object o) : Criteria - Criteria
- lt(Object o) : Criteria - Criteria
- lte(Object o) : Criteria - Criteria
- maxDistance(double maxDistance) : Criteria - Criteria
- mod(Number value, Number remainder) : Criteria - Criteria
- ne(Object o) : Criteria - Criteria
- near(Point point) : Criteria - Criteria
- nearSphere(Point point) : Criteria - Criteria

At the bottom right of the dropdown, there is a message: "Press 'Ctrl+Space' to show Template Proposals".

Spring Data JPA/MongoDB



Spring Data REST

- Spring MVC et Spring Data JPA
 - Support MongoDB en cours de développement
- Opérations CRUD suivant la sémantique REST
- Gestion des relations
- Navigation entre entités
 - « HATEOAS »

Spring Data REST

The image displays two separate browser windows, each showing a JSON response from a Spring Data REST endpoint. The top window shows a collection of addresses, and the bottom window shows a single address entity.

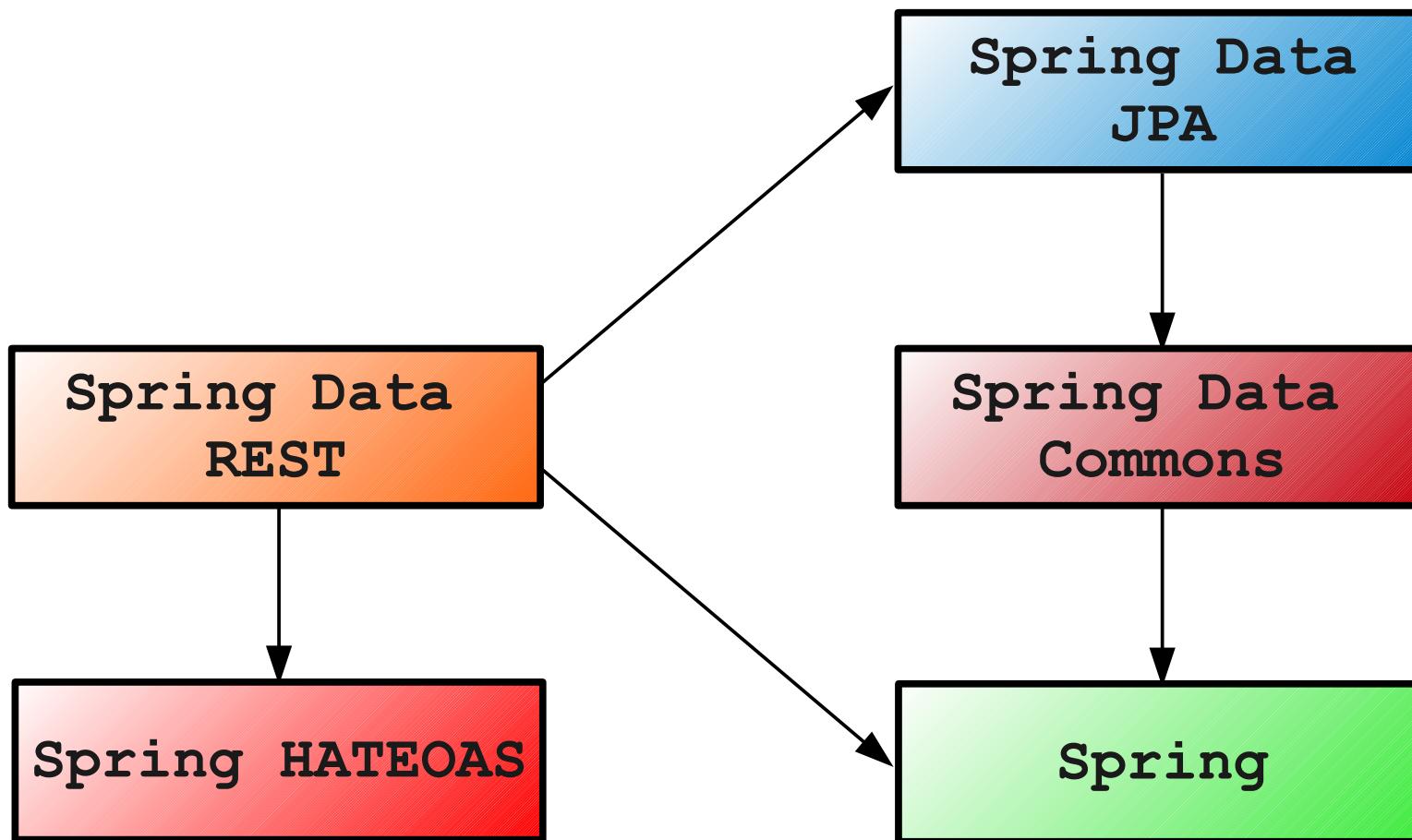
Top Window (localhost:8080/data/address):

```
{
  "_links": [
    {
      "rel": "Address",
      "href": "http://localhost:8080/data/address/1"
    },
    {
      "rel": "Address",
      "href": "http://localhost:8080/data/address/2"
    }
  ]
}
```

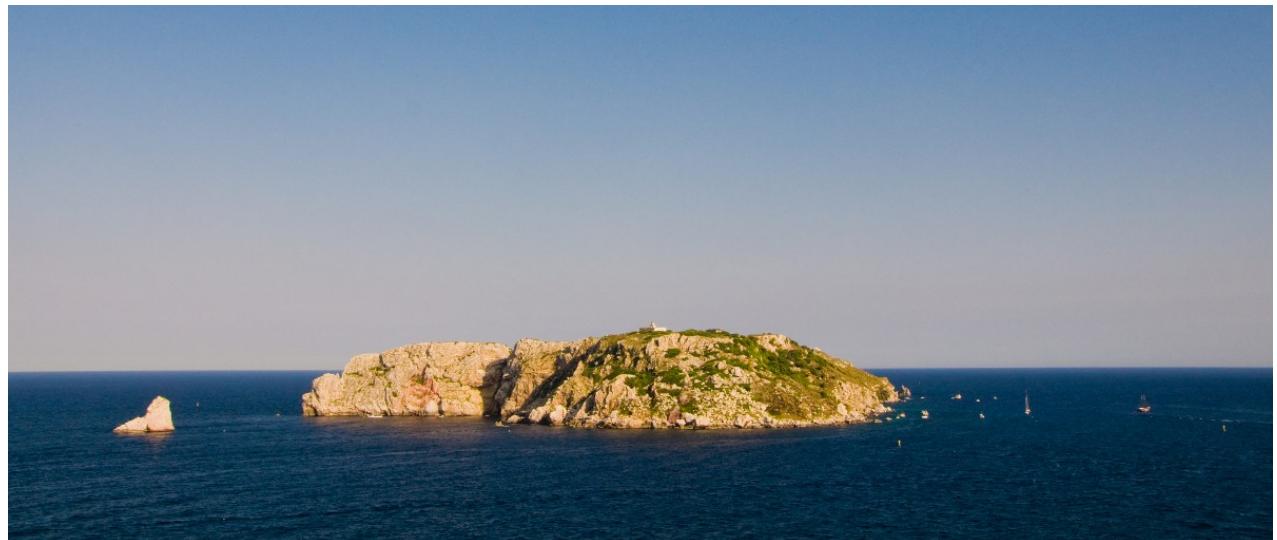
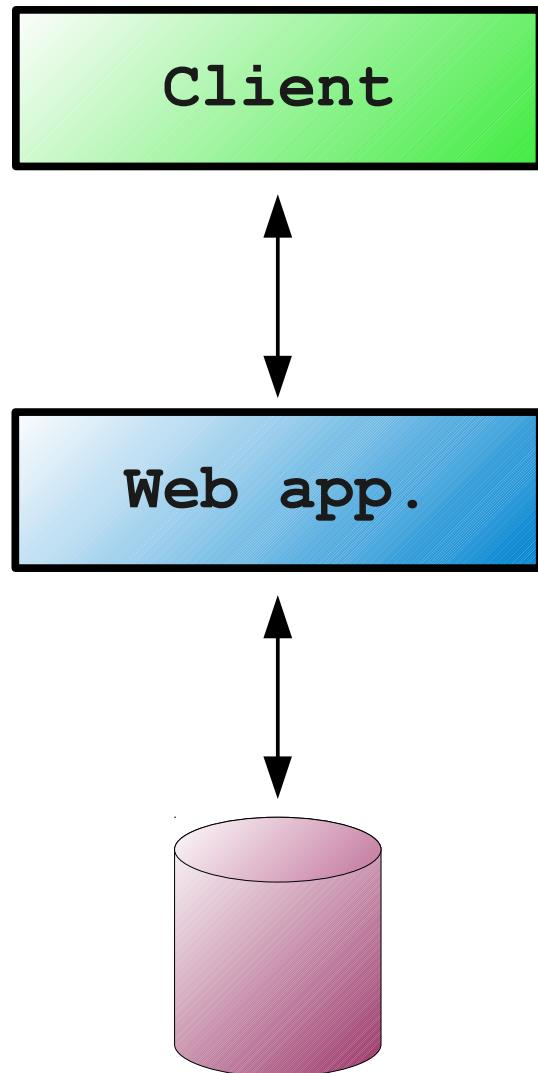
Bottom Window (localhost:8080/data/address/1):

```
{
  "postalCode": "12345",
  "province": "MO",
  "_links": [
    {
      "rel": "self",
      "href": "http://localhost:8080/data/address/1"
    }
  ],
  "lines": [ "1 W 1st St." ],
  "city": "Univille"
}
```

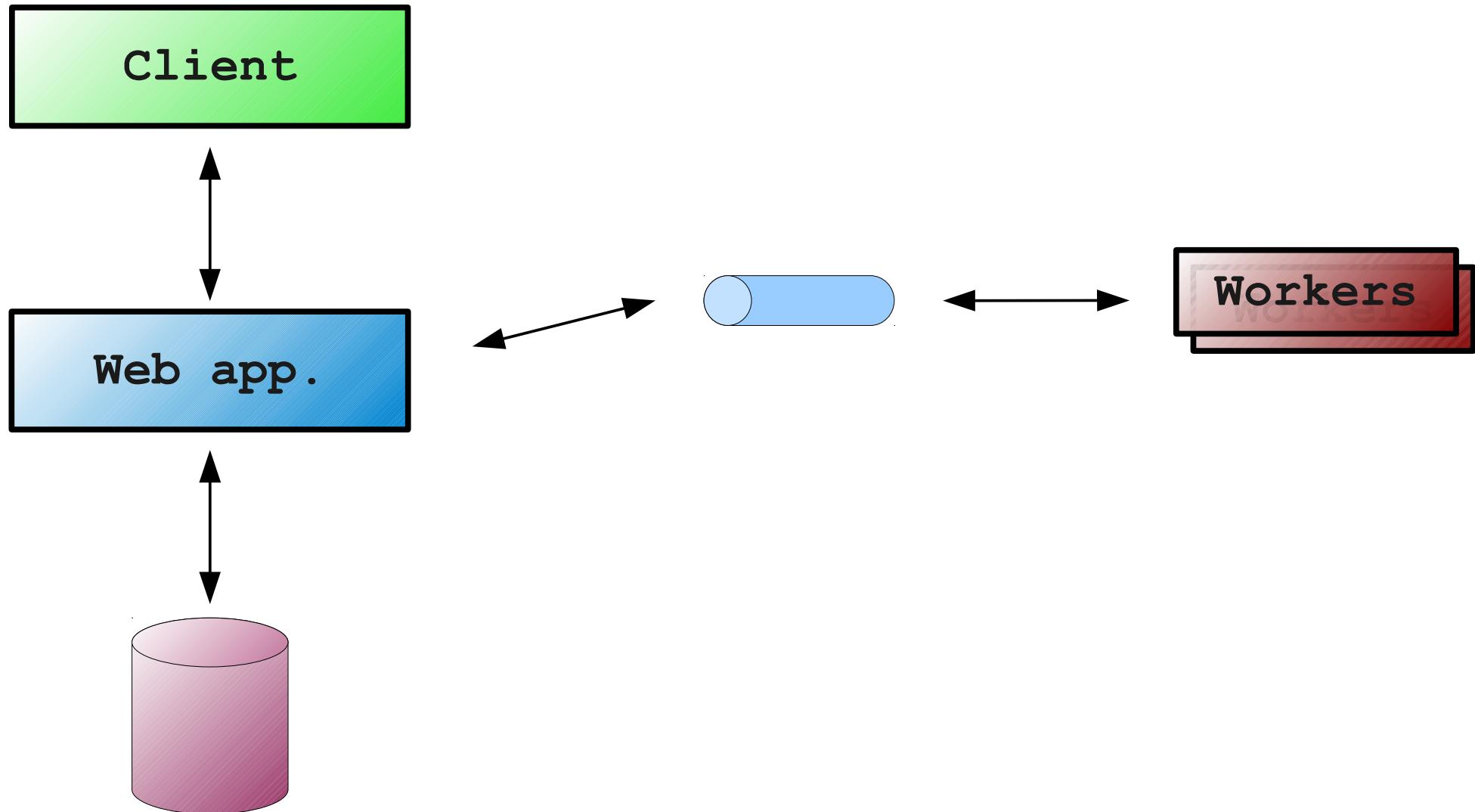
Spring Data REST



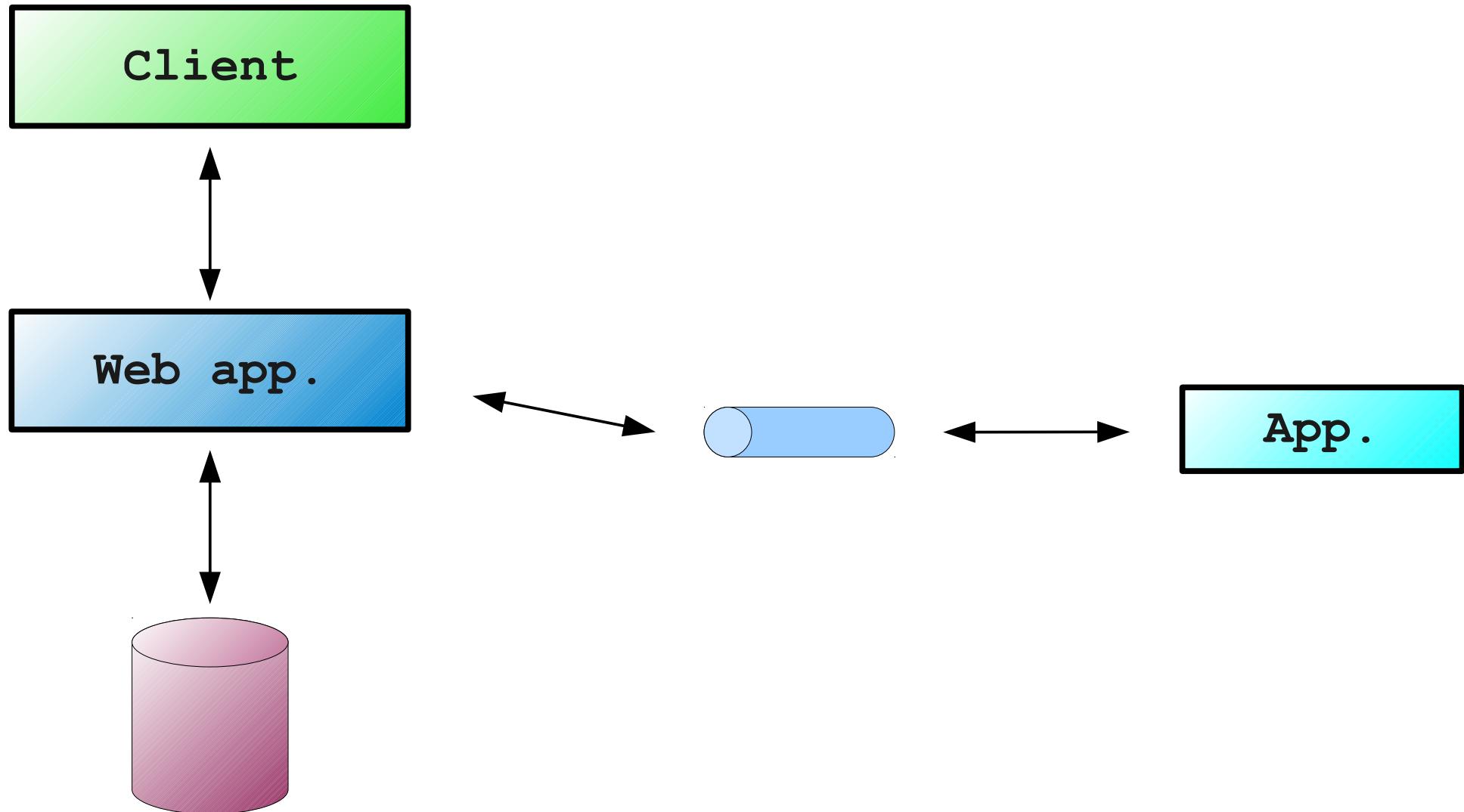
Intégration



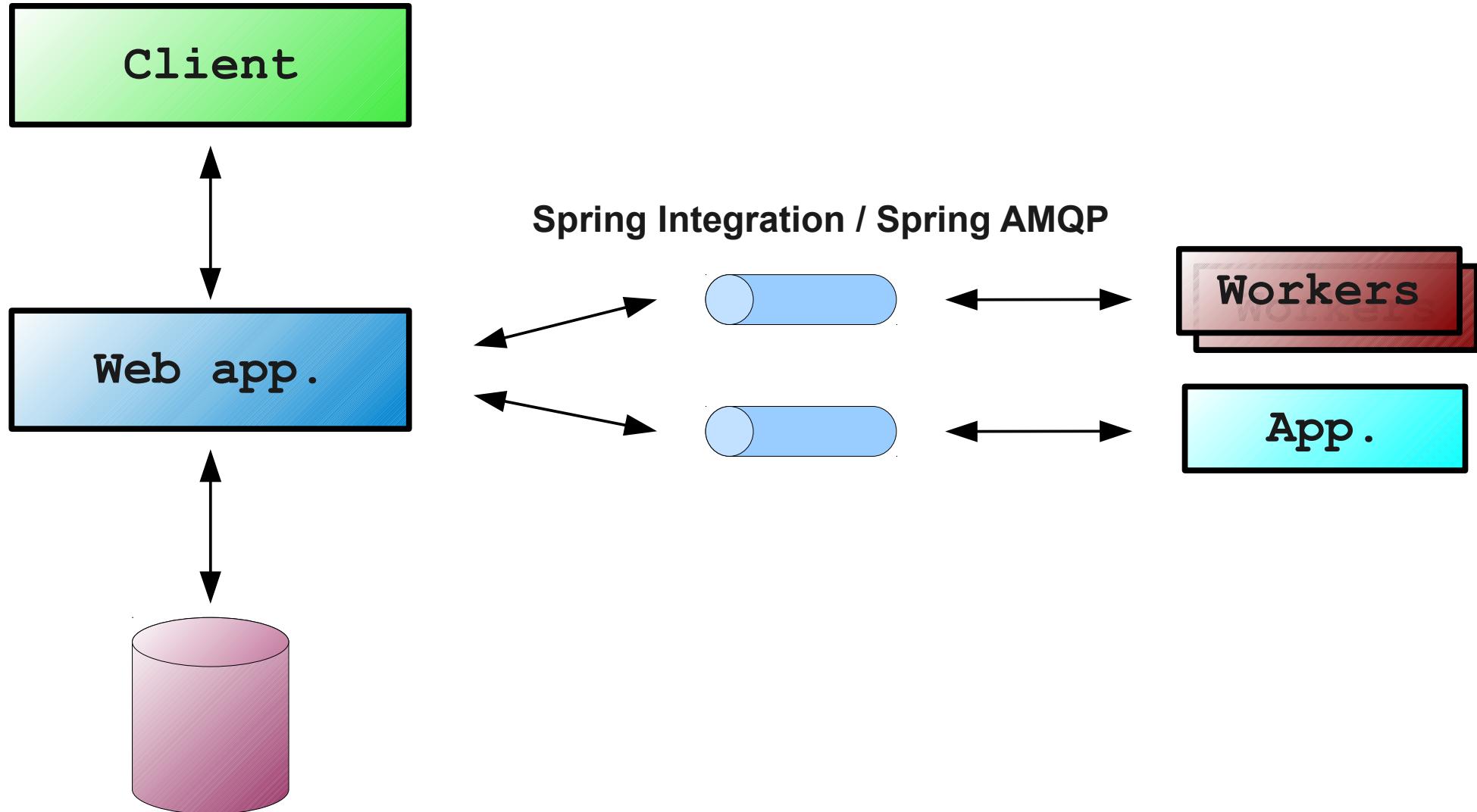
Intégration



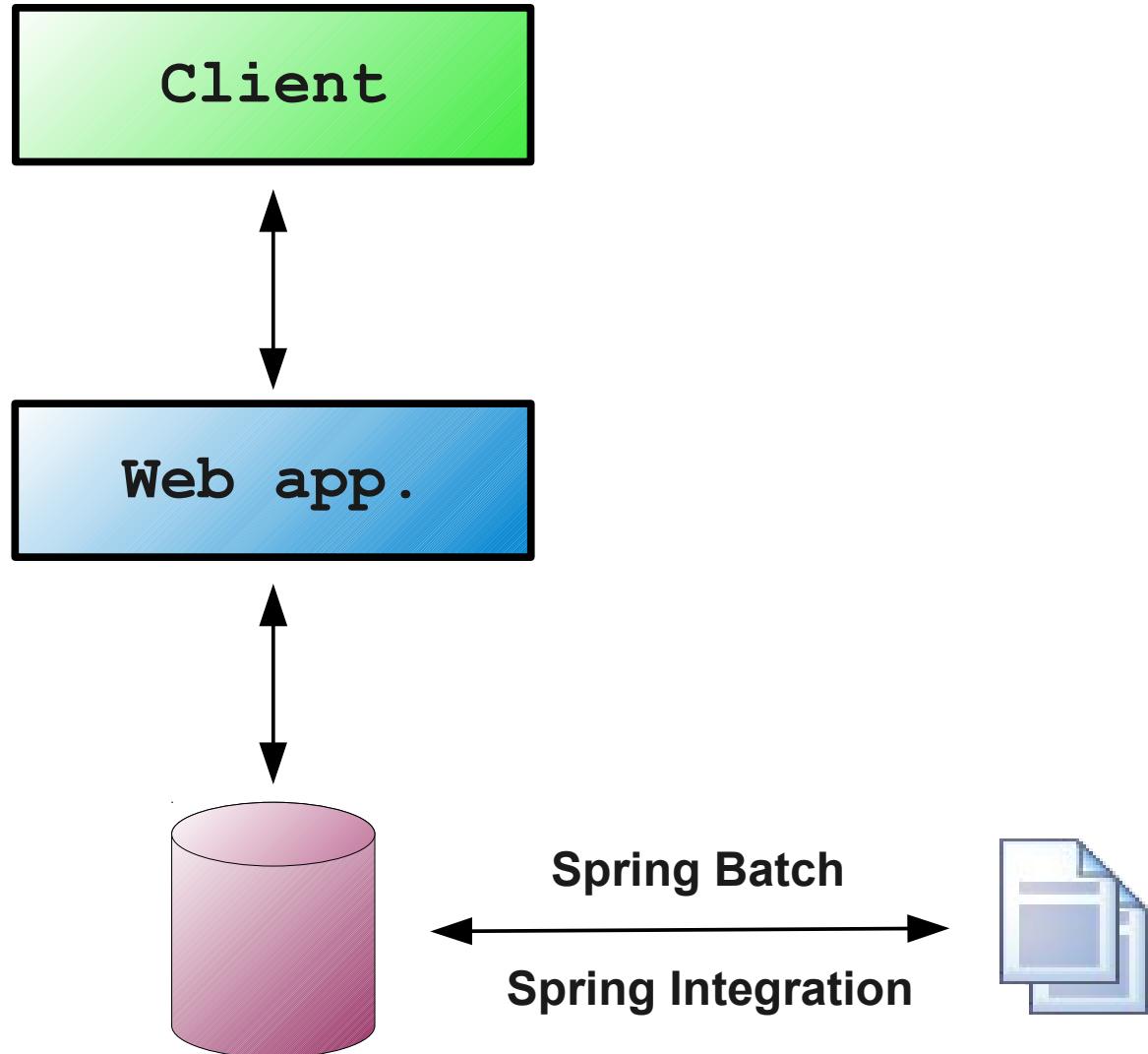
Intégration



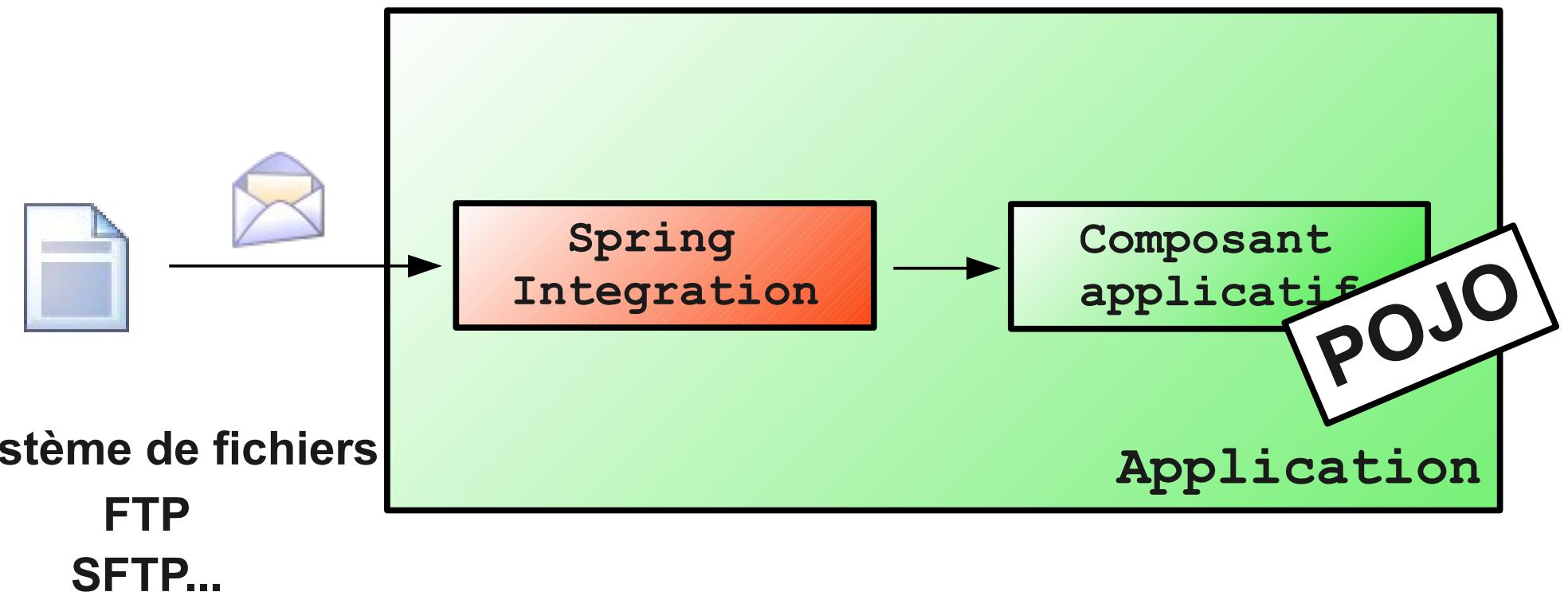
Intégration



Intégration



Spring Integration



Spring Integration

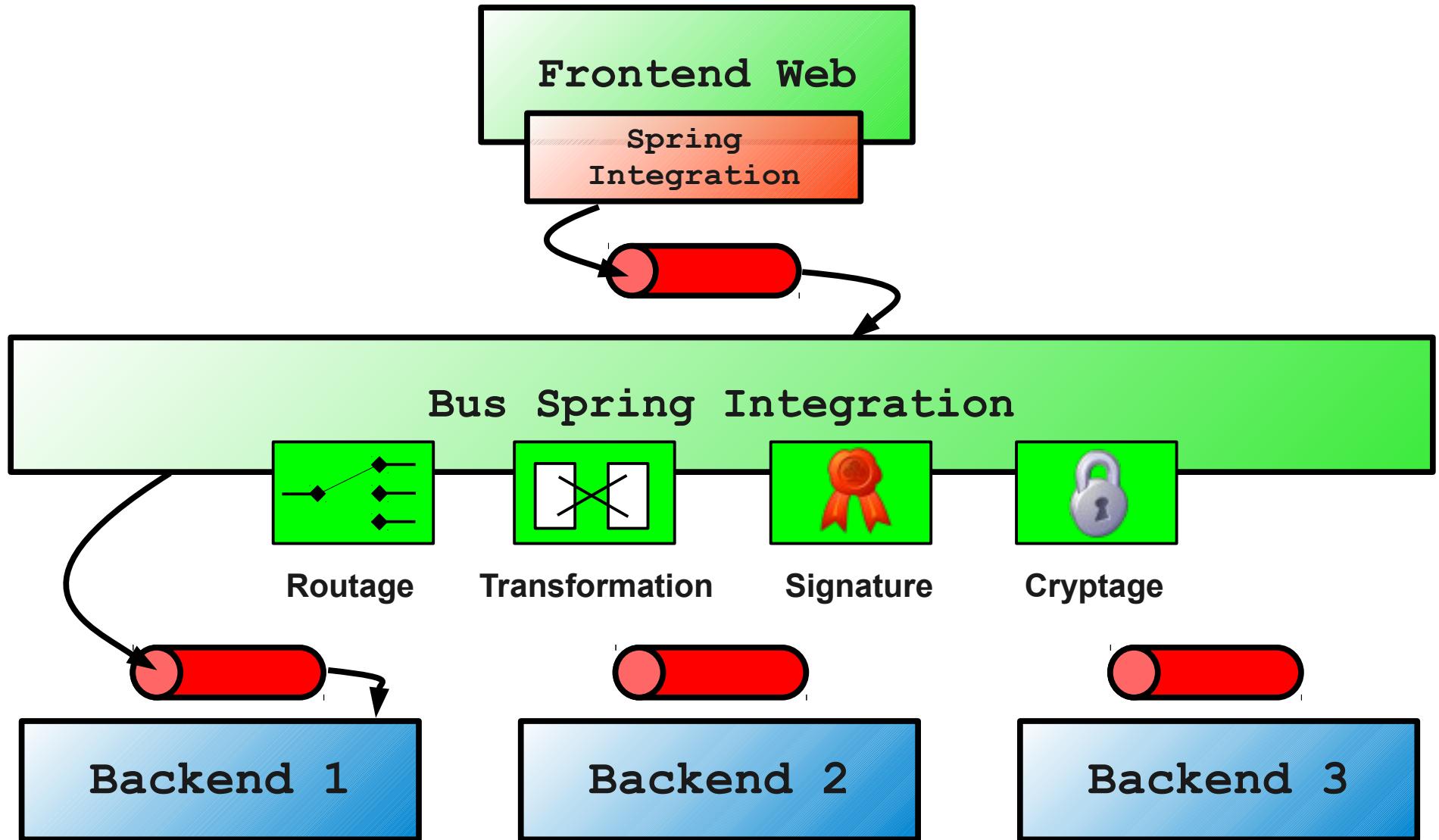
Merge pull request #664 from garyrussell/INT-2803 ...

 Mark Fisher authored 2 hours ago	 latest commit 1c8b849af7
 gradle	5 days ago
 spring-integration-amqp	4 days ago
 spring-integration-core	17 days ago
 spring-integration-event	3 months ago
 spring-integration-feed	6 months ago
 spring-integration-file	a month ago
 spring-integration-ftp	2 months ago
 spring-integration-gemfire	2 months ago
 spring-integration-groovy	2 months ago
 spring-integration-http	a month ago
 spring-integration-ip	25 days ago
 spring-integration-jdbc	25 days ago
 spring-integration-jms	2 months ago
 spring-integration-jmx	2 months ago
 spring-integration-jpa	a month ago
 spring-integration-mail	2 hours ago
 spring-integration-mongodb	17 days ago
 spring-integration-redis	11 days ago
	INT-2802 - Upgrade build.gradle to use Gradle 1.2 [Gunnar Hillert]
	INT-2788 AMQP return-channel Doc. Improvements [garyrussell]
	INT-2777 TxSynchFactory: tests for bound resource [artembilan]
	INT-2214, INT-343, INT-2250 MessageHandler Advice [garyrussell]
	INT-2536 Create 2.2 Schemas [garyrussell]
	INT-2218 - Chain Parser Validation Improvements [Gunnar Hillert]
	INT-2718 & INT-2721: fixes for <chain> & fix bugs [artembilan]
	INT-2710 - Remove hard-coded schema references [Gunnar Hillert]
	INT-2718 & INT-2721: fixes for <chain> & fix bugs [artembilan]
	INT-2218 - Chain Parser Validation Improvements [Gunnar Hillert]
	INT-2776 Add Sender's UDP Port To Headers [garyrussell]
	INT-2280 fixed broken JDBC test [Oleg Zhurakousky]
	INT-2683 Add Reply Listener Container Option [garyrussell]
	INT-2487 Object Name Patterns for Notifications [garyrussell]
	INT-2770 & INT-2771 fixes: [artembilan]
	IntegrationMimeType wrapper class is now private [Mark Fisher]
	INT-2779 spring-data-mongo upgrade [Oleg Zhurakousky]
	INT-2790 - extractZsetIncrementHeader should return 'true' by default [Gunnar Hillert]

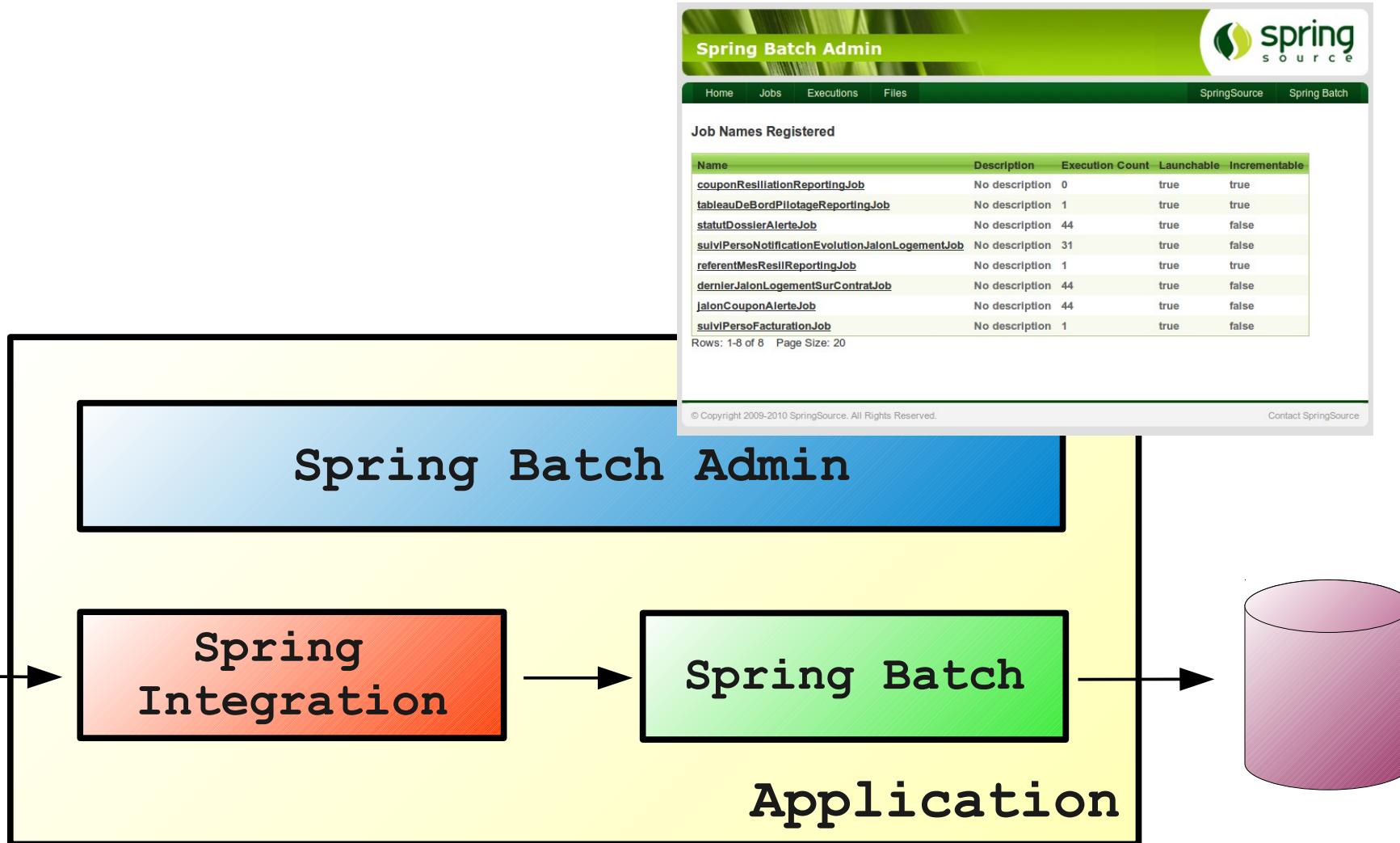
Spring Integration

- Lightweight messaging
- Enterprise Integration Patterns
 - Routage, transformation, etc.
- Configuration XML, @ + DSLs Scala, Groovy

Spring Integration <=> bus



Spring Integration & Spring Batch

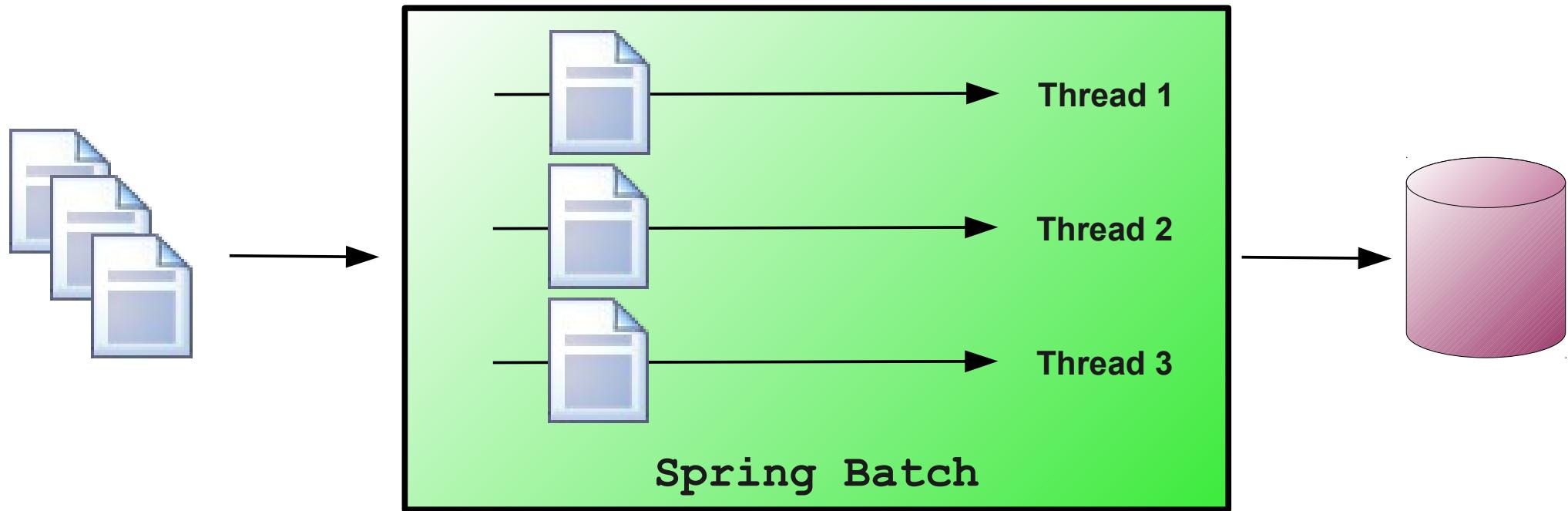


Spring Batch

- Traitements par lot
- Saut suite à erreur, retry
- Reprise sur erreur
- Fichiers plats/XML, JDBC/Hibernate, JPA, JMS, email
- Spring Batch Admin : console de monitoring web

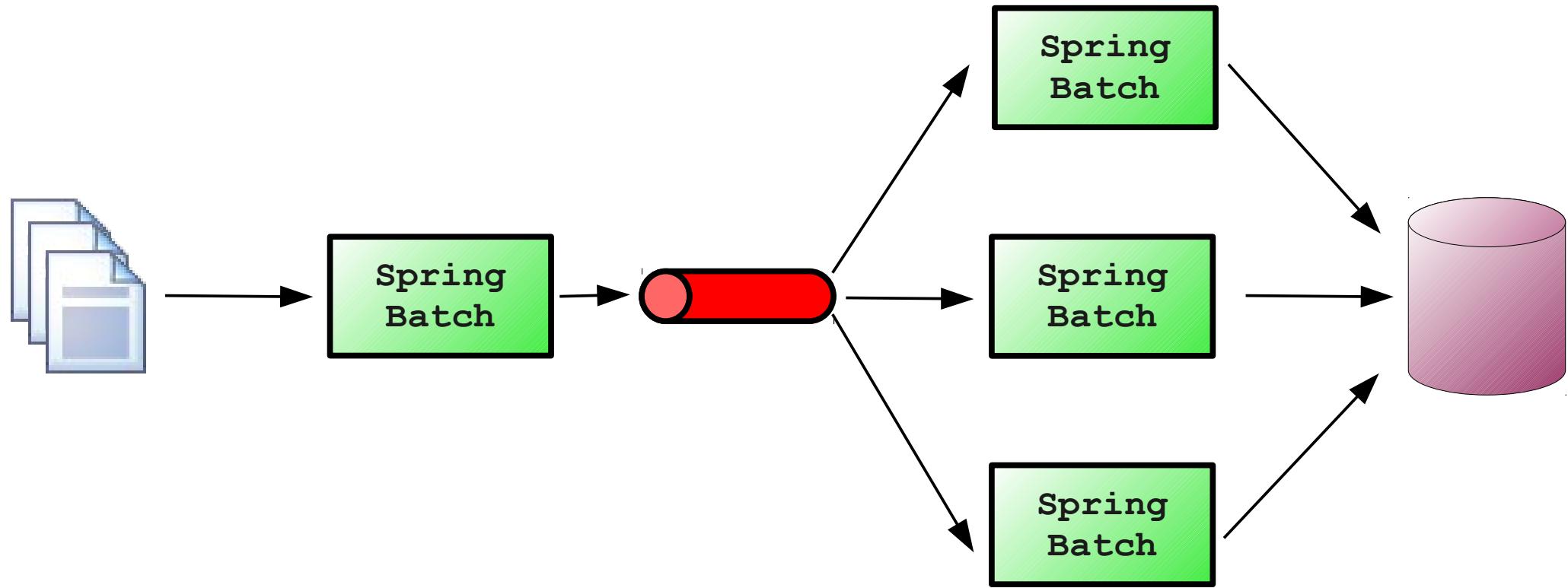
Spring Batch & scaling

- Local partitioning

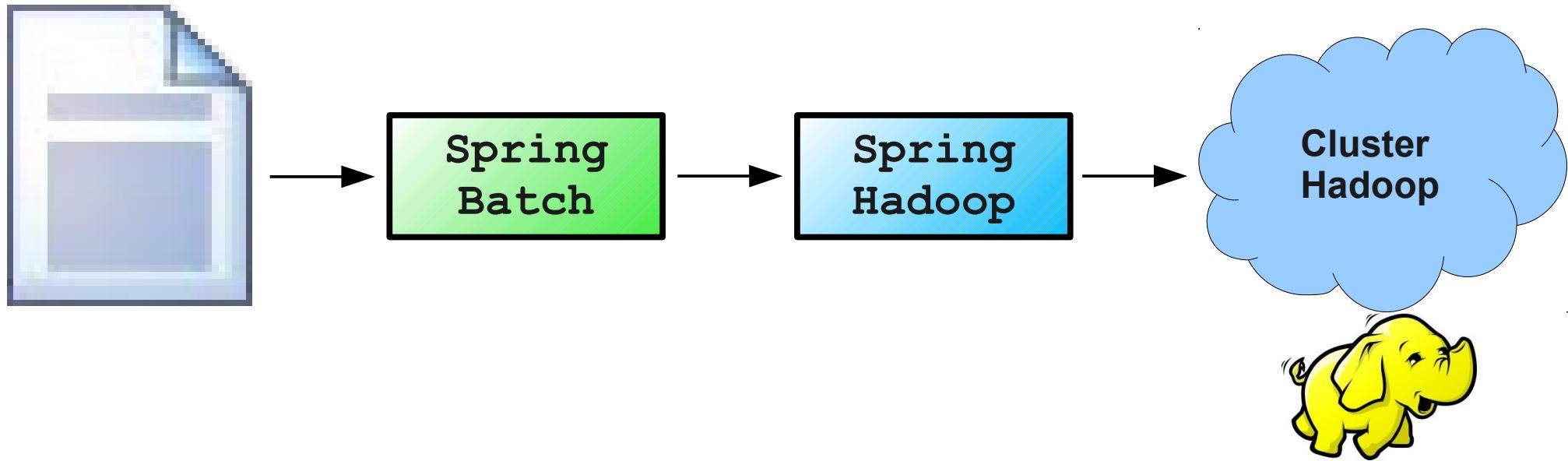


Spring Batch & scaling

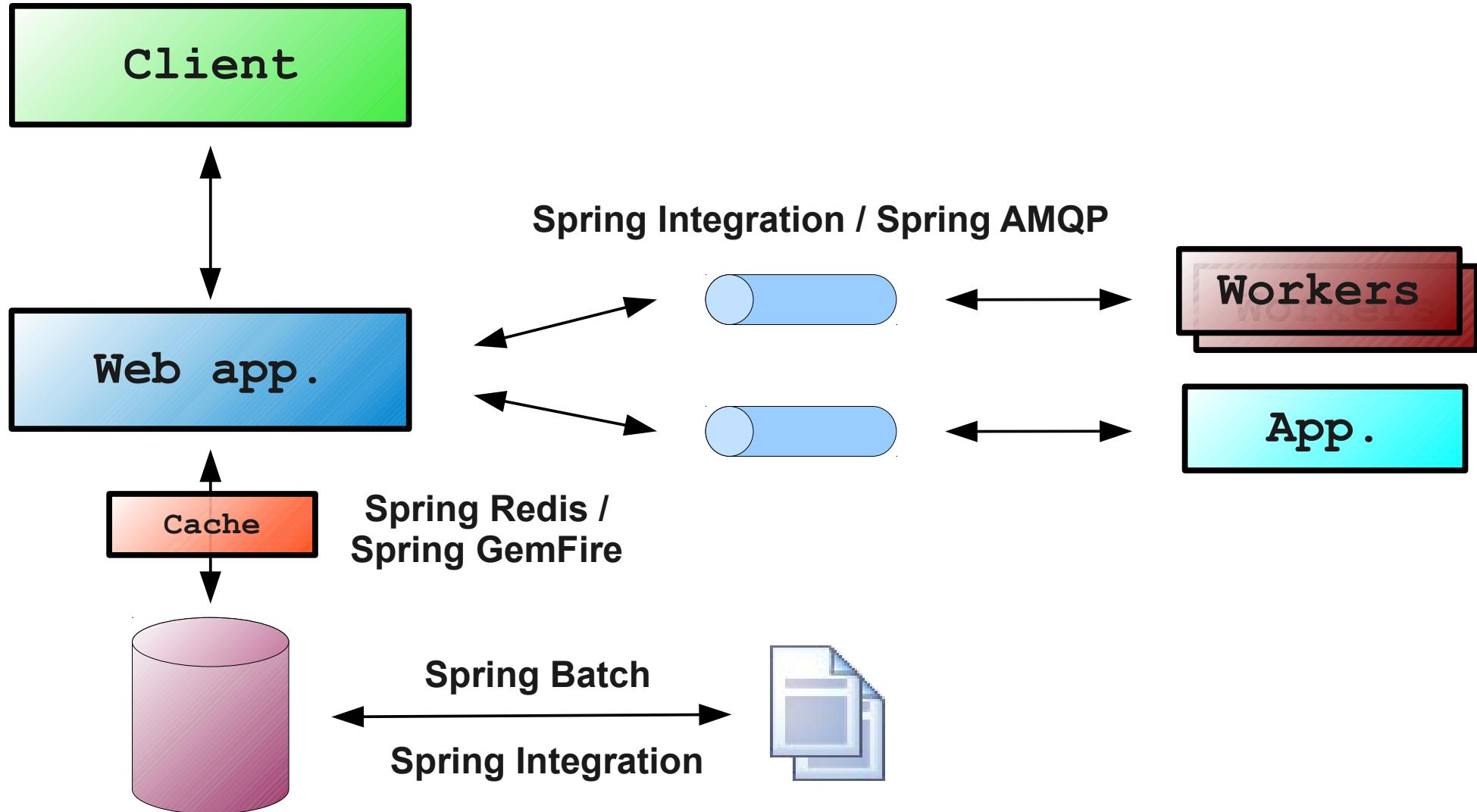
- Remote partitioning



Spring Batch & Spring Hadoop



Intégration



Intégration et déploiement



PostgreSQL



 mongoDB

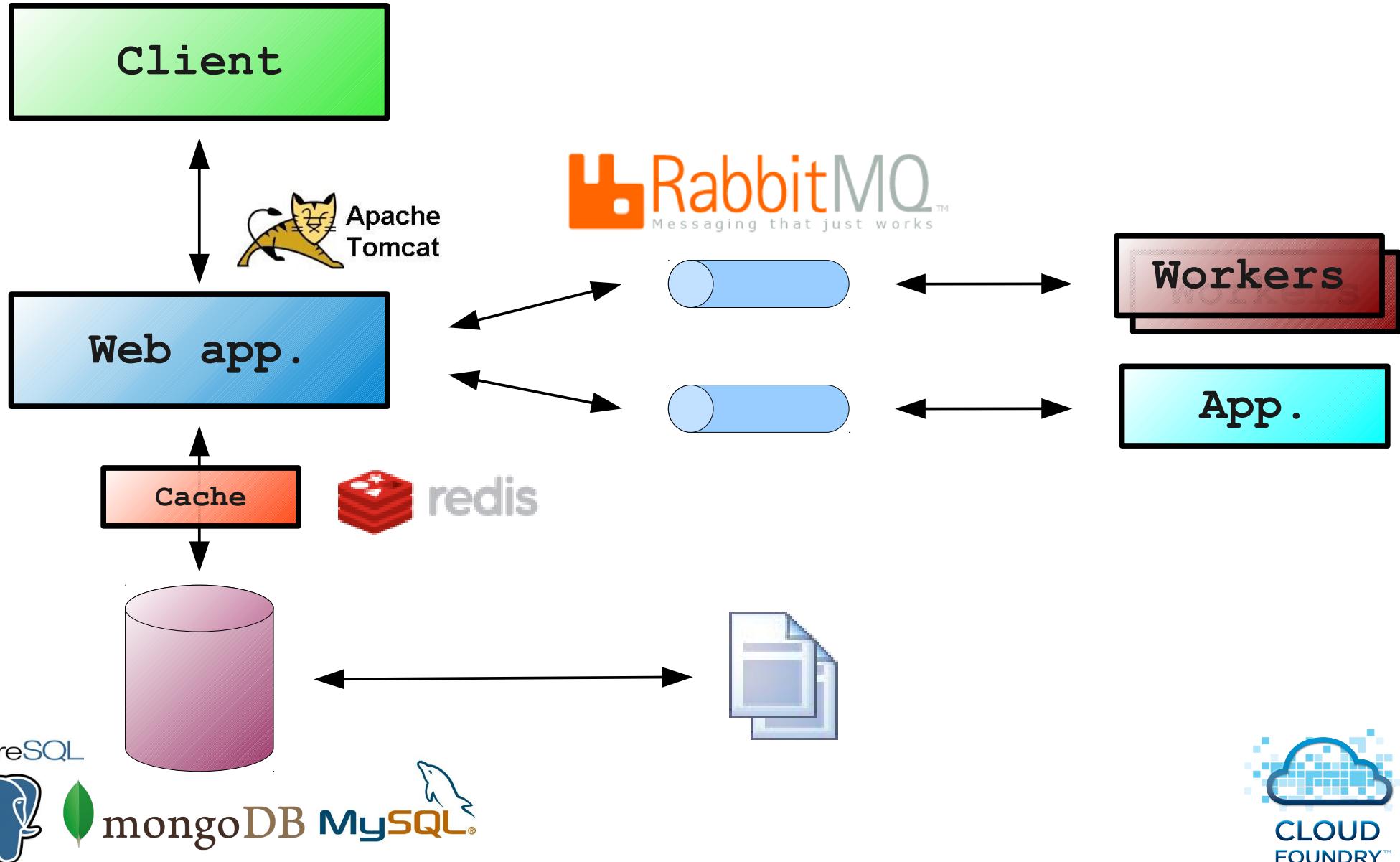


 RabbitMQ™
Messaging that just works

 Apache
Tomcat

 MySQL®

Cloud Foundry



Conclusion

- Technologies Spring
 - Simplicité et souplesse
 - Stabilité et pérennité
 - Un framework pour chaque besoin en entreprise
 - Une grande cohérence dans le portfolio

A close-up photograph of a squirrel with a light brown and tan coat, standing on a rocky surface. The squirrel is looking upwards and slightly to its right. Its bushy tail is visible on the left. The background consists of large, light-colored rocks and some small green plants.

Questions ?

Merci !