



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 {

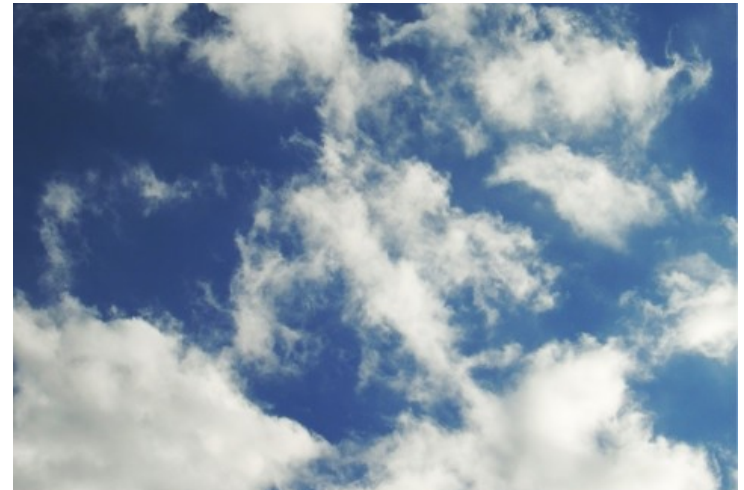
  public void transfer(...) { }

}
```

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

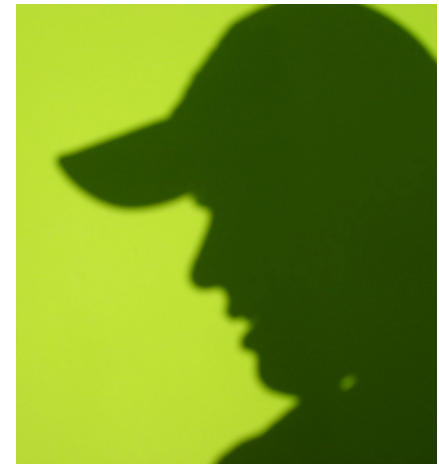
```
public class TransferService {  
    @Transactional  
    public void transfer(...) { }  
}
```

Exécution, où ?



Profils

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



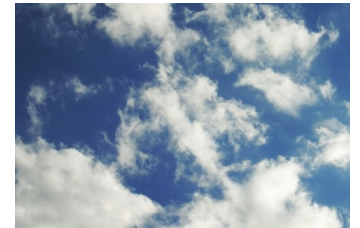
Profils

```
<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"
```

- Programmatically

```
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 namespace~~



@notations

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>
```

@notations

```
@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

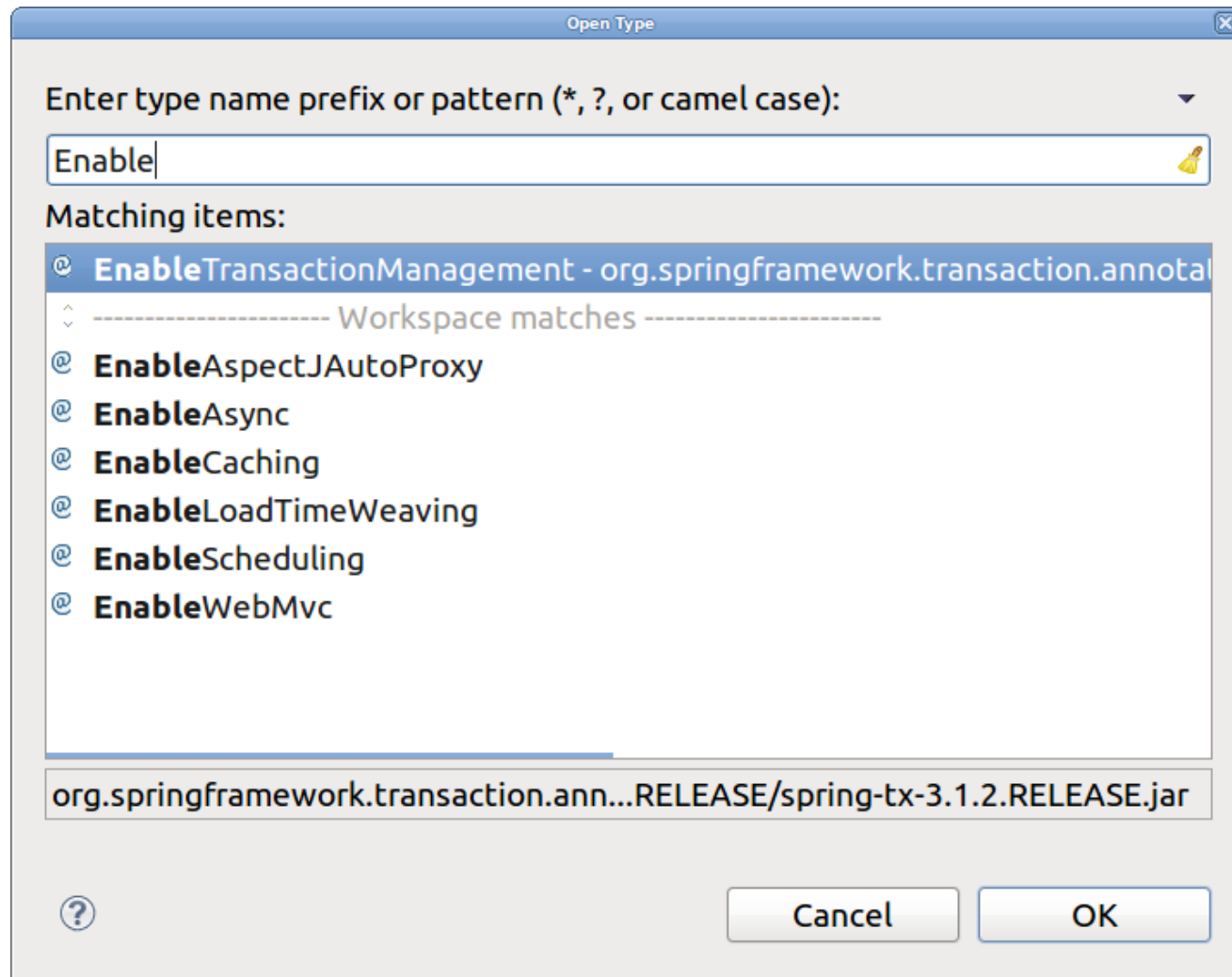
```
@Configuration
@EnableTransactionManagement
public class AppConfig {

    @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

Protocole applicatif

Alternative

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;  
  
    @RequestMapping(value="/contacts/{id}",method=RequestMethod.GET)  
    @ResponseBody  
    public Contact contact(@PathVariable Long id) {  
        return contactRepository.findOne(id);  
    }  
  
}
```



REST

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

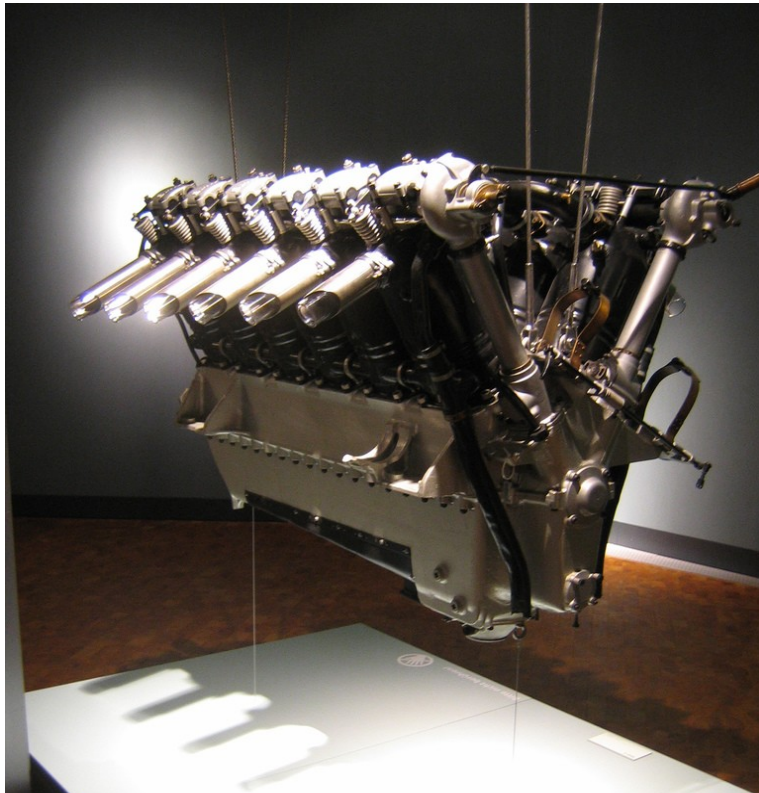


```
HTTP/1.1 200 OK  
Content-Type: application/json  
{ "id": 1, "firstname": "Joe", "lastname": "Dalton", "age": 37 }
```

```
@Controller  
public class ContactController {  
  
    @Autowired ContactRepository contactRepository;  
  
    @RequestMapping(value="/contacts/{id}", method=RequestMethod.GET)  
    @ResponseBody  
    public Contact contact(@PathVariable Long id) {  
        return contactRepository.findOne(id);  
    }  
  
}
```



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 displays the Eclipse IDE interface with the following components:

- Editor (Left):** Shows the XML file `spring-old-school.xml` with the following content:

```
<?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>
```
- JUnit Console (Middle):** Shows the test results for `com.zenika.SpringOldSchoolTest`. The test `transactional` passed successfully after 0.648 seconds. The overall run finished after 0.718 seconds with 1/1 runs, 0 errors, and 0 failures.

```
Finished after 0.718 seconds
Runs: 1/1 Errors: 0 Failures: 0
com.zenika.SpringOldSchoolTest [Runn
  transactional (0.648 s)
```
- Package Explorer (Bottom):** Lists the project dependencies, including various Spring and Commons libraries such as `spring-tx-3.1.3.RELEASE.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-asm-3.1.3.RELEASE.jar`, `spring-context-support-3.1.3.RELEASE.jar`, and `spring-expression-3.1.3.RELEASE.jar`.
- Editor (Right):** Shows a partial view of the XML file, including the `transferService` and `dataSource` bean definitions.

```
<?xml version="1.0" encoding="UTF-8"
<!DOCTYPE beans PUBLIC "-//SPRING//
"http://www.springframework.or
<beans>
  <bean id="transferServiceTarget
    <constructor-arg ref="accol
  </bean>
  <bean id="transferService"
    class="org.springframework.
    <property name="transactor
    <property name="target" ref
    <property name="transactor
      <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

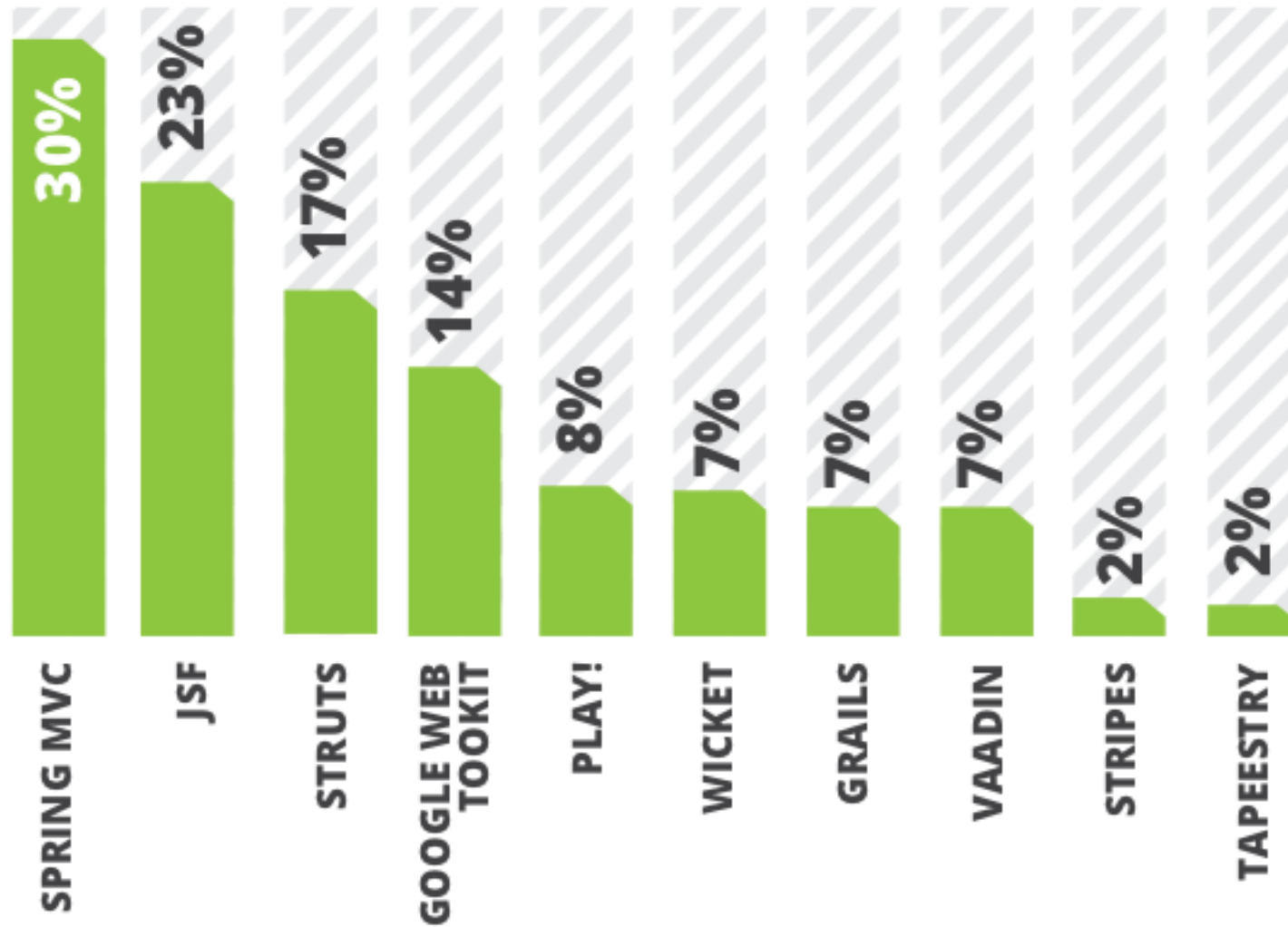
```
@Controller
public class ContactController {

    @Autowired ContactRepository contactRepository;

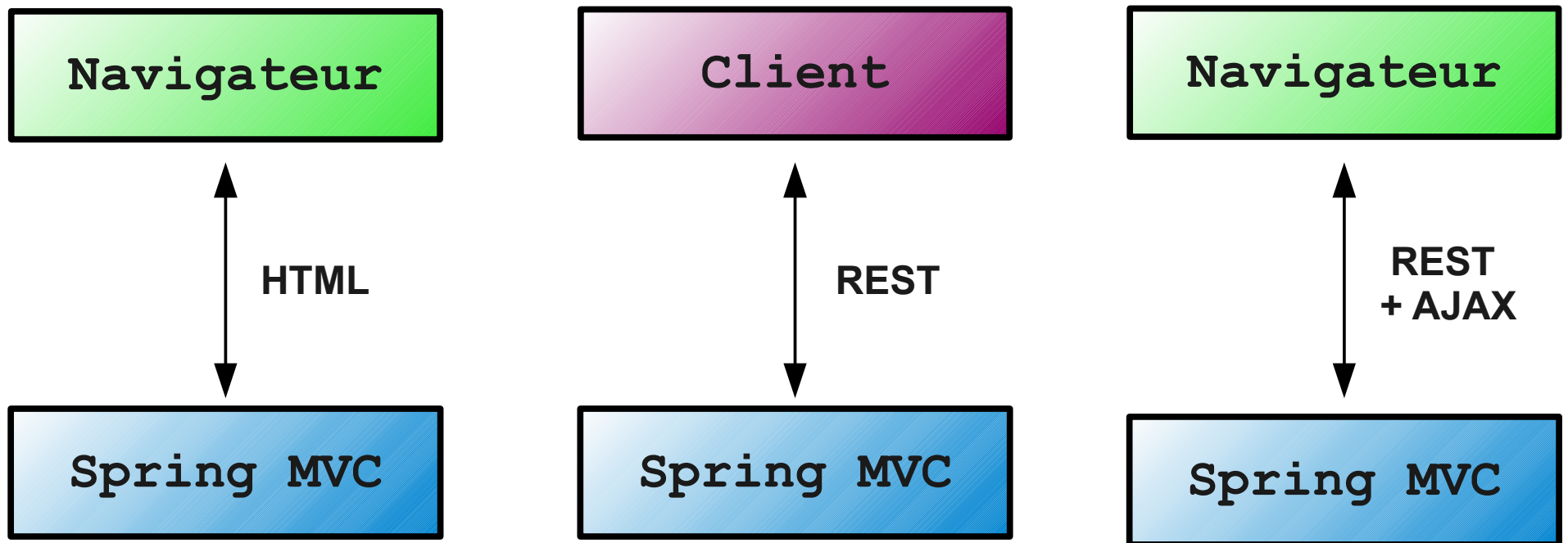
    @RequestMapping(value="/contacts/{id}",method=RequestMethod.GET)
    public ResponseEntity<Contact> contact(@Domain Contact contact) {
        ResponseEntity<Contact> response = new ResponseEntity<Contact>(
            contact,
            contact == null ? HttpStatus.NOT_FOUND : HttpStatus.OK
     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



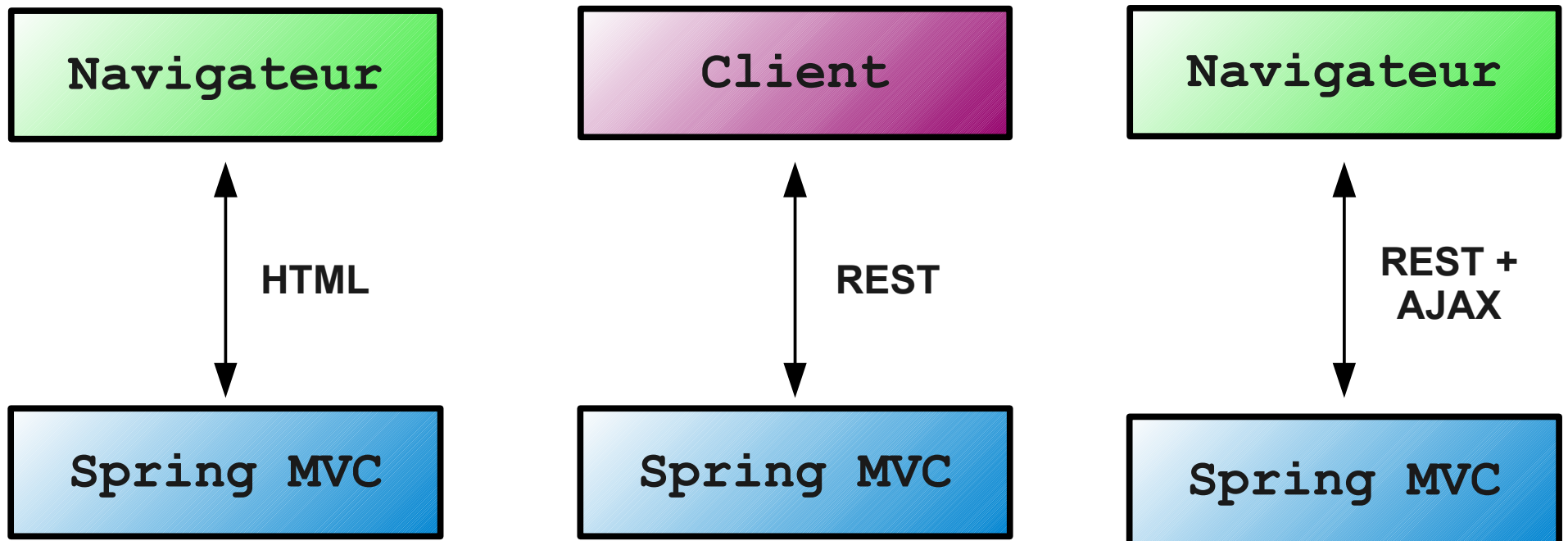
Spring MVC et popularité



Utilisations de Spring MVC



Utilisations de Spring MVC



**+ RestTemplate
pour la partie cliente**

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)
@Configuration
public class AccountRepositoryTest {

    @Autowired private AccountRepository repo;

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

    @Configuration
    public static class SpringJpaConfiguration {

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

    }

}
```

3.1



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("$[0].name").value("NAME1"))  
            .andExpect(jsonPath("$[1].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")  
            );  
    }  
}
```

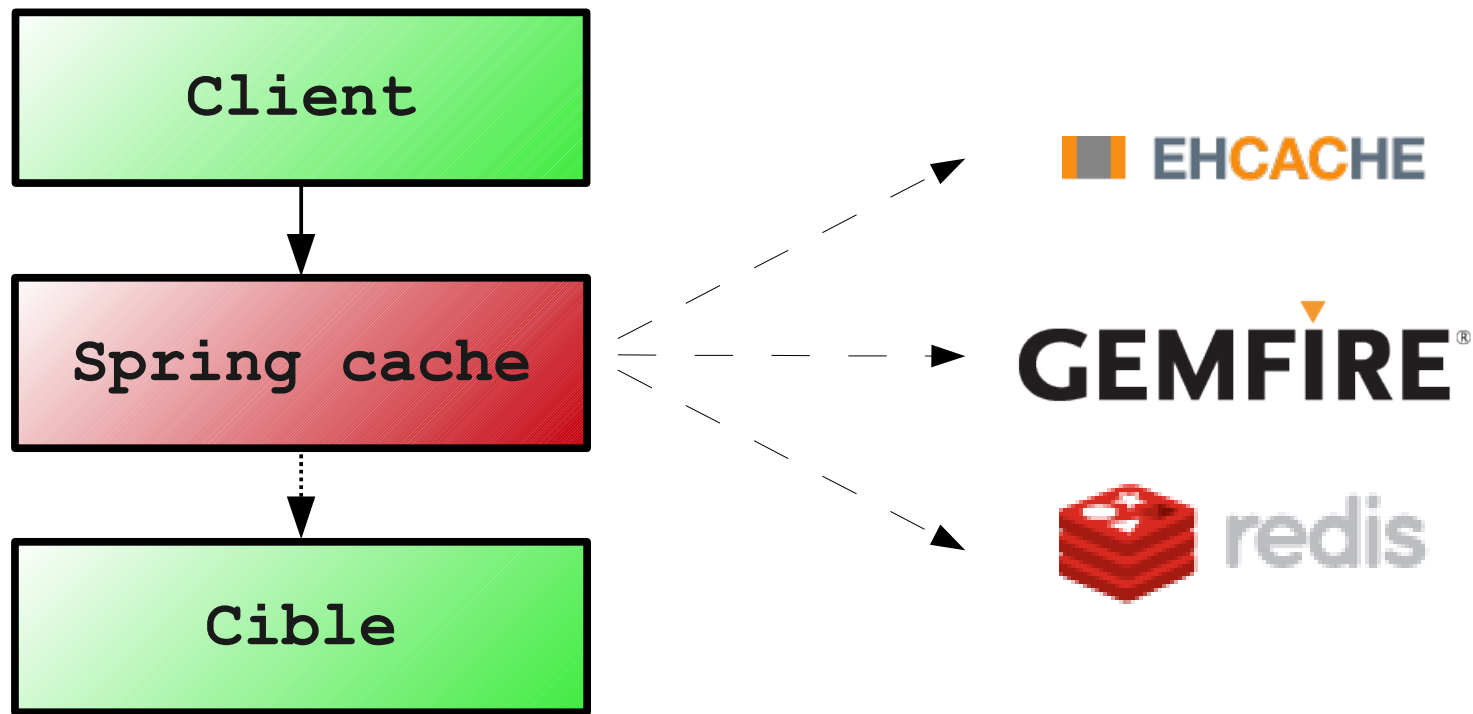


Cache

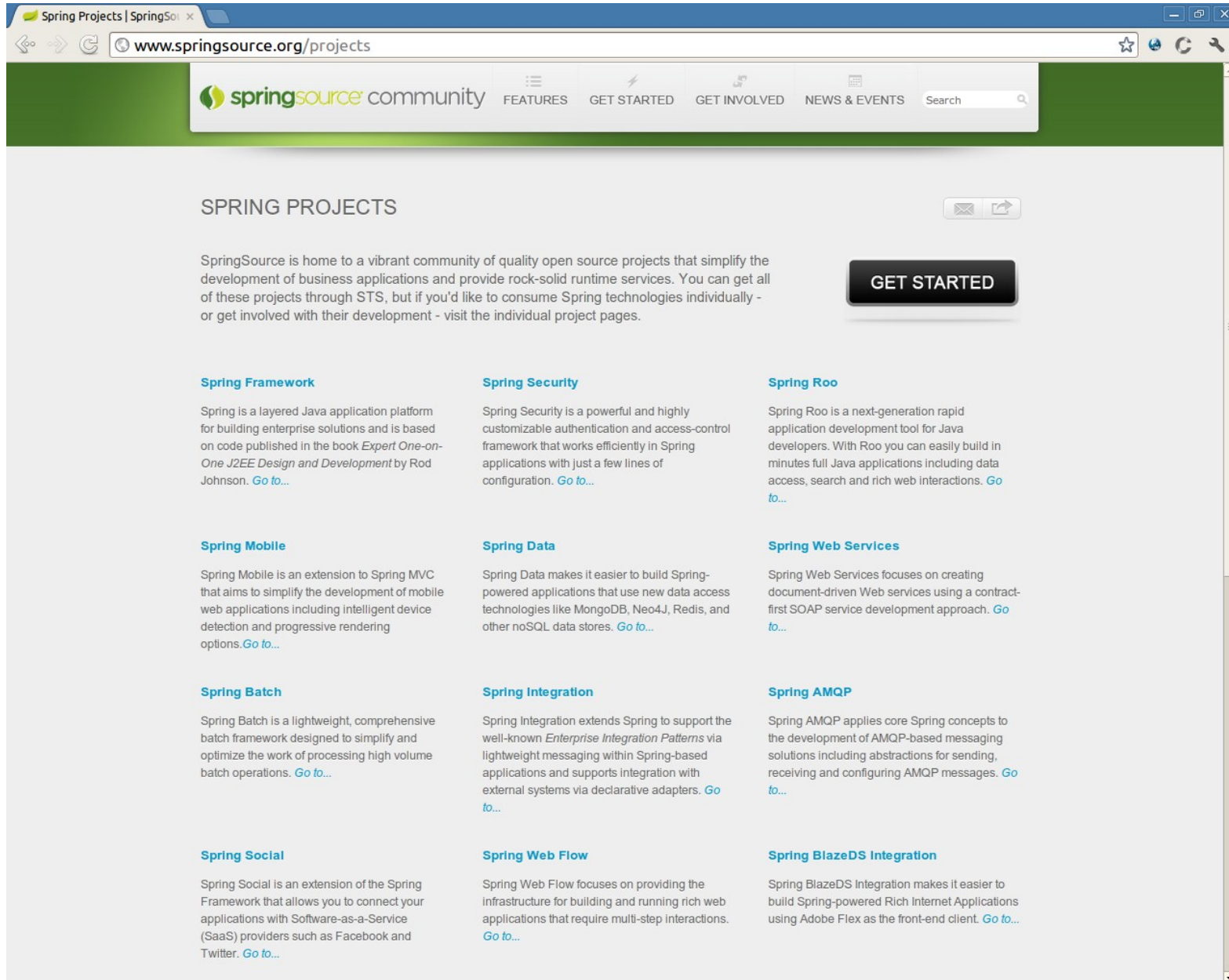
```
public class ContactRepository {  
  
    @Cacheable("contacts")  
    Contact findOne(Long id) { ... }  
  
}
```

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

Cache



Portfolio Spring



The screenshot shows a web browser window with the address bar displaying "www.springsource.org/projects". The page features a green header with the "springsource community" logo and navigation links for "FEATURES", "GET STARTED", "GET INVOLVED", and "NEWS & EVENTS". A search bar is also present. The main content area is titled "SPRING PROJECTS" and includes a "GET STARTED" button. Below this, there is a grid of project descriptions, each with a title, a brief overview, and a "Go to..." link.

SPRING PROJECTS

SpringSource is home to a vibrant community of quality open source projects that simplify the development of business applications and provide rock-solid runtime services. You can get all of these projects through STS, but if you'd like to consume Spring technologies individually - or get involved with their development - visit the individual project pages.

GET STARTED

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

DAO dynamiques, intelligents

Moins de code répétitif

Accès données

Testabilité

Conventions

JPA

Pagination & tri

Approche DDD

Spring Data JPA

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

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

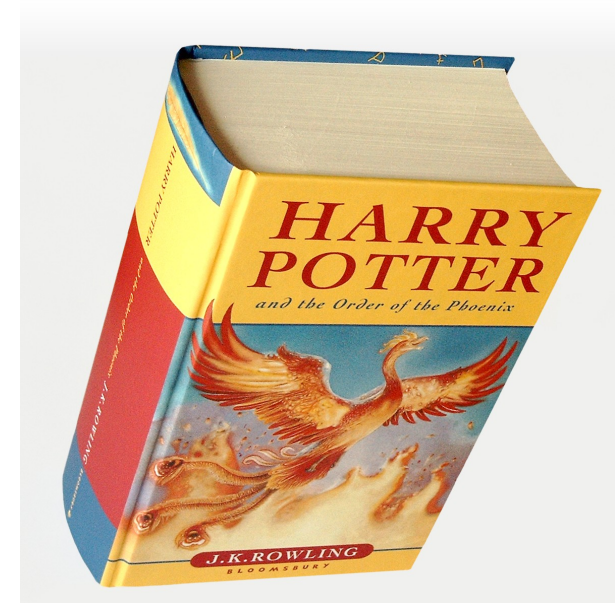
```
List<Contact> cts = repo.findAll();
```



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 o.s.data.mongodb.core.query.Criteria.*;  
import static o.s.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 o.s.data.mongodb.core.query.Criteria.*;  
import static o.s.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);
```

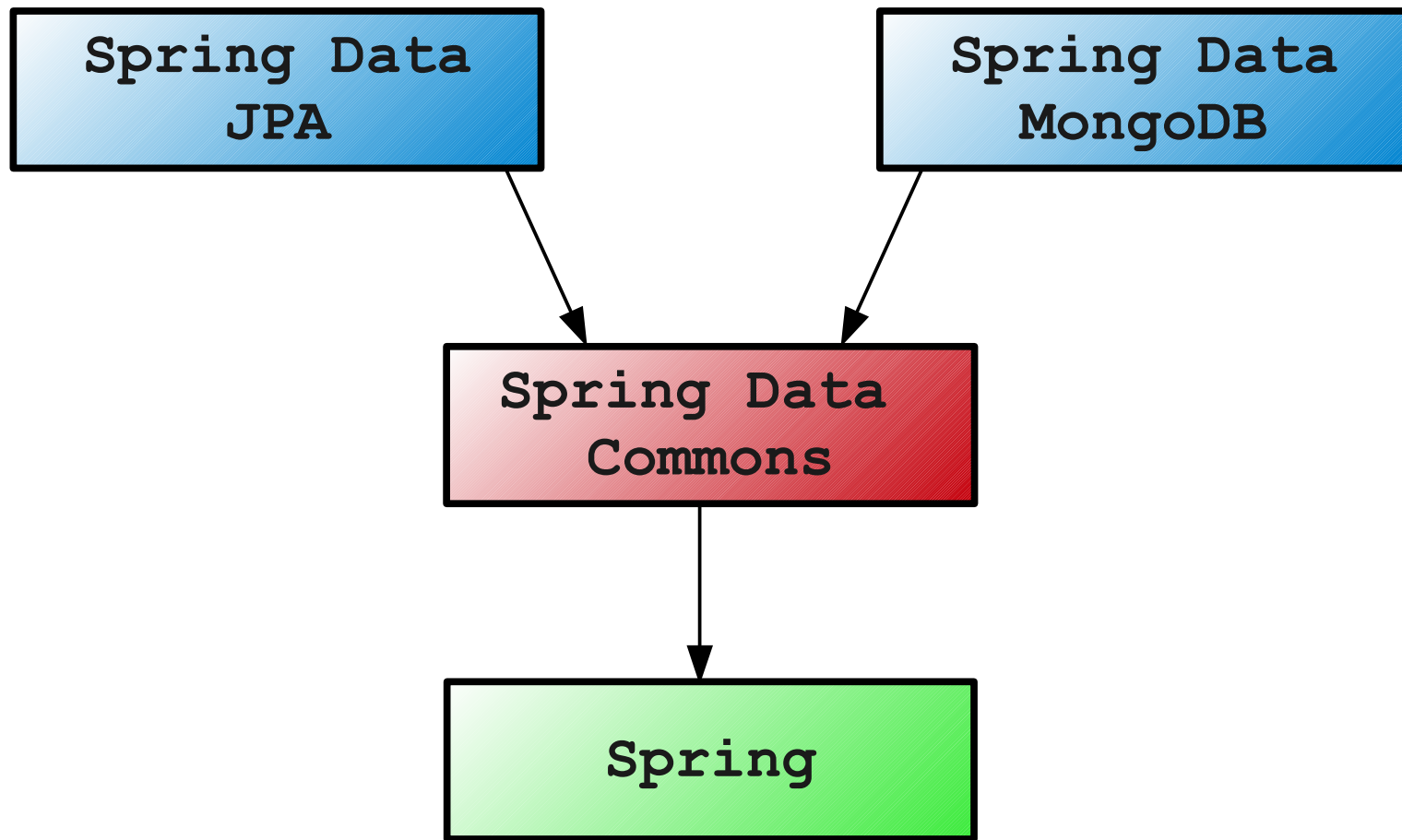
- in(Collection<?> c) : Criteria - Criteria
- in(Object... o) : Criteria - Criteria
- is(Object o) : Criteria - Criteria
- lt(Object o) : Criteria - Criteria
- lte(Object o) : Criteria - Criteria
- maxDistance(double maxDistance) : Criteria - Criteria
- mod(Number value, Number remainder) : Criteria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

```
Assert.assertEquals(1, entries.size());
```

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

Press 'Ctrl+Space' to show Template Proposals

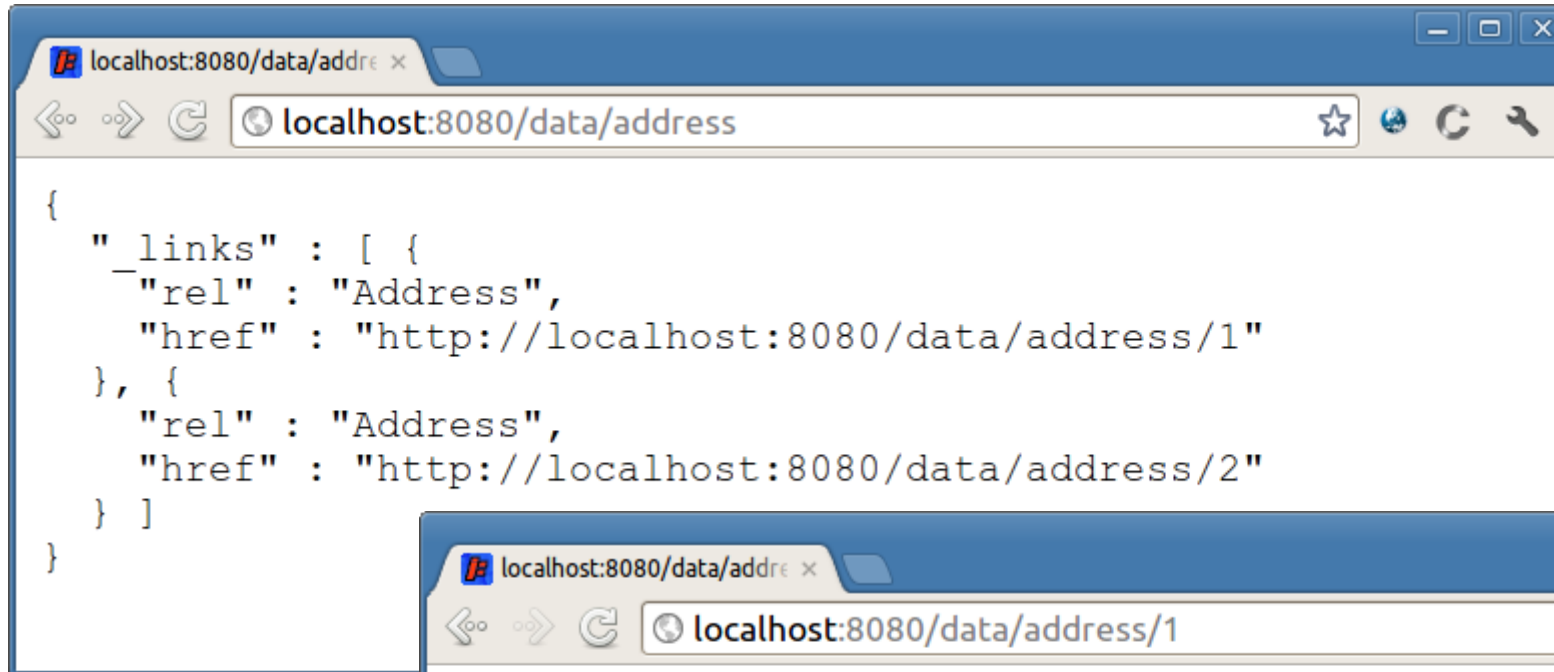
Spring Data JPA/MongoDB



Spring Data REST

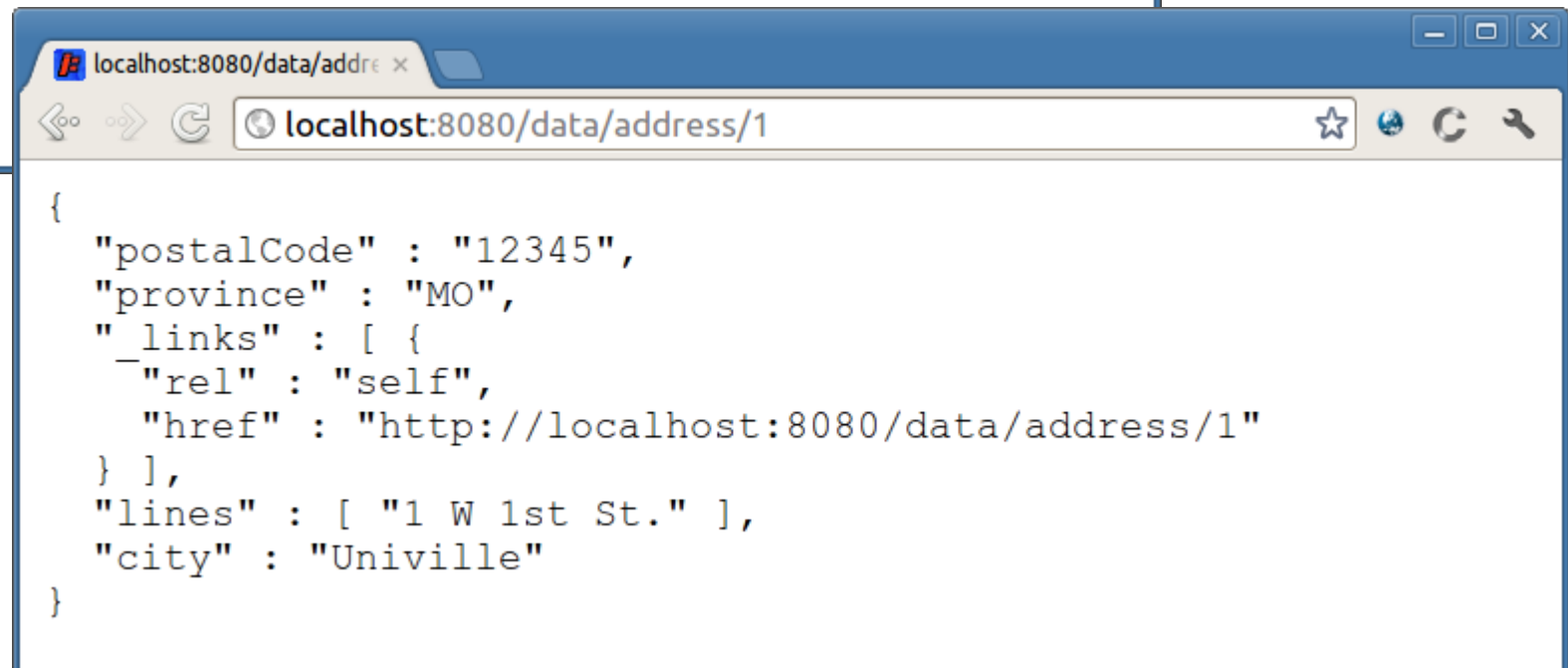
- Spring MVC et Spring Data JPA
 - Support MongoDB en cours de développement
- Opérations CRUD suivant la sémantique REST
- Gestion des relations
- Navigation entre entités
 - « HATEOAS »

Spring Data REST



A screenshot of a web browser window with the address bar showing `localhost:8080/data/address`. The main content area displays a JSON response representing a list of addresses. The response includes a `_links` array with two entries, each containing a `rel` of "Address" and a `href` pointing to `localhost:8080/data/address/1` and `localhost:8080/data/address/2` respectively.

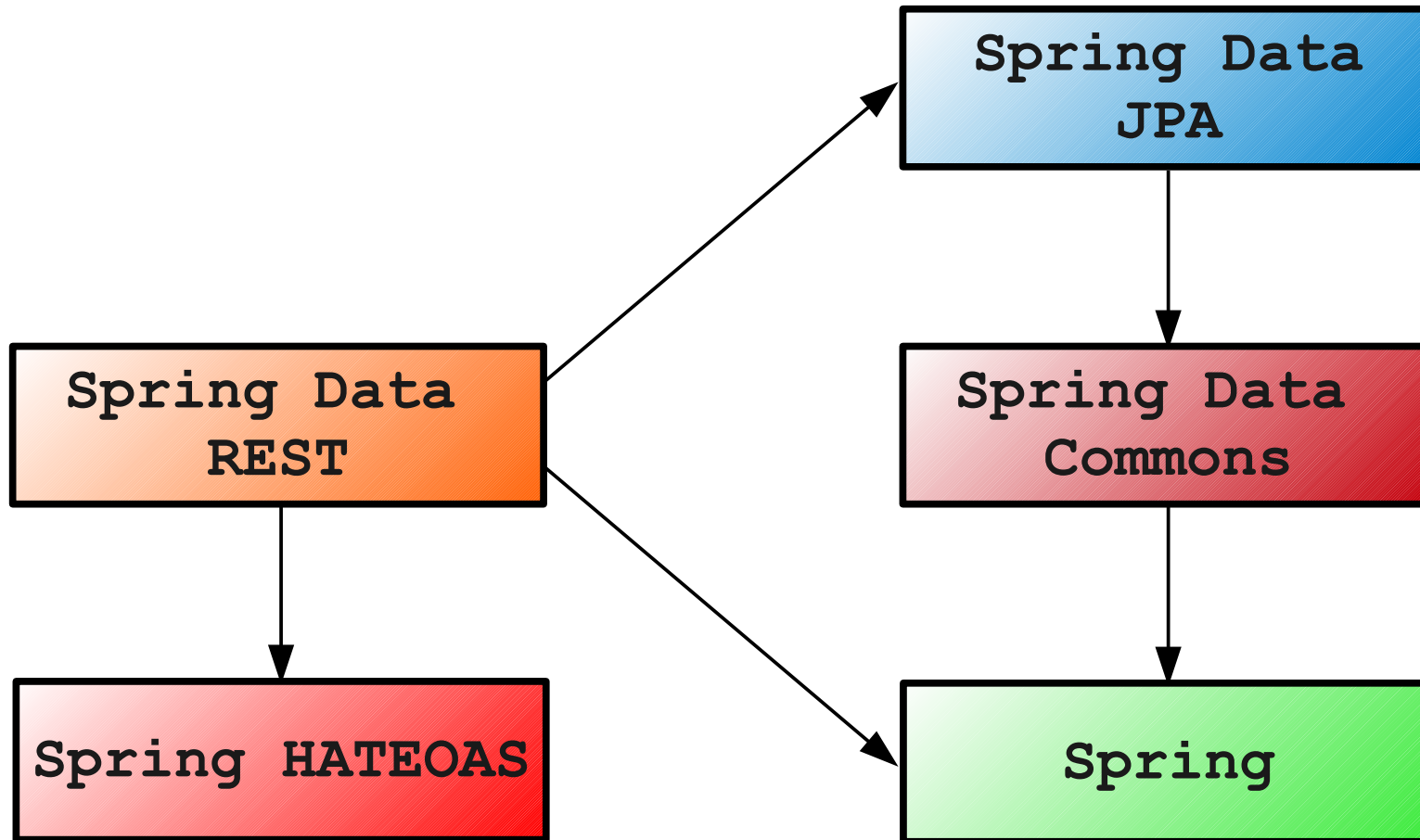
```
{
  "_links" : [ {
    "rel" : "Address",
    "href" : "http://localhost:8080/data/address/1"
  }, {
    "rel" : "Address",
    "href" : "http://localhost:8080/data/address/2"
  } ]
}
```



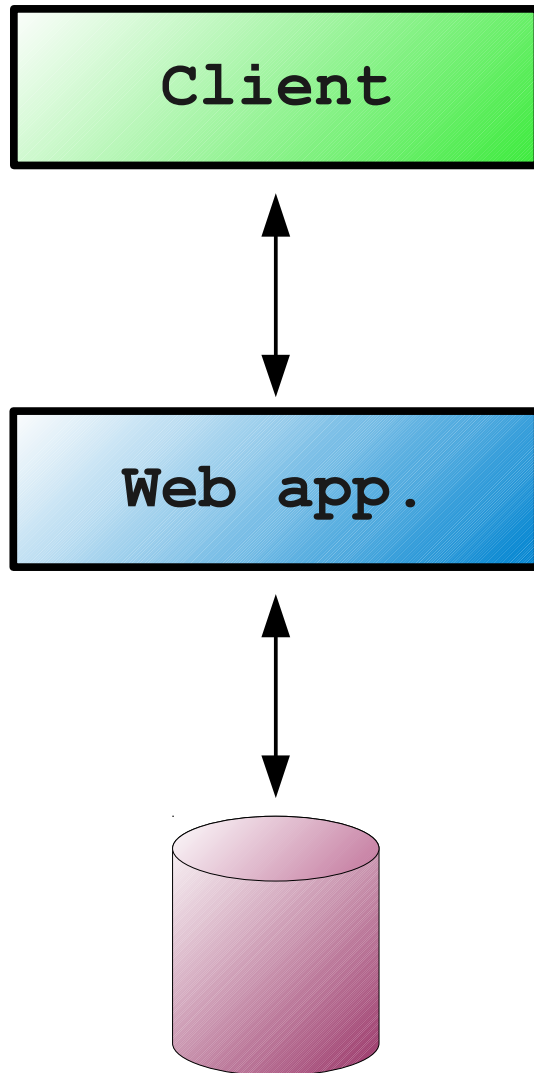
A screenshot of a web browser window with the address bar showing `localhost:8080/data/address/1`. The main content area displays a JSON response representing a single address resource. The response includes fields for `postalCode` ("12345"), `province` ("MO"), `lines` (["1 W 1st St."]), and `city` ("Univille"). It also includes a `_links` array with a `self` link pointing to `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"
}
```

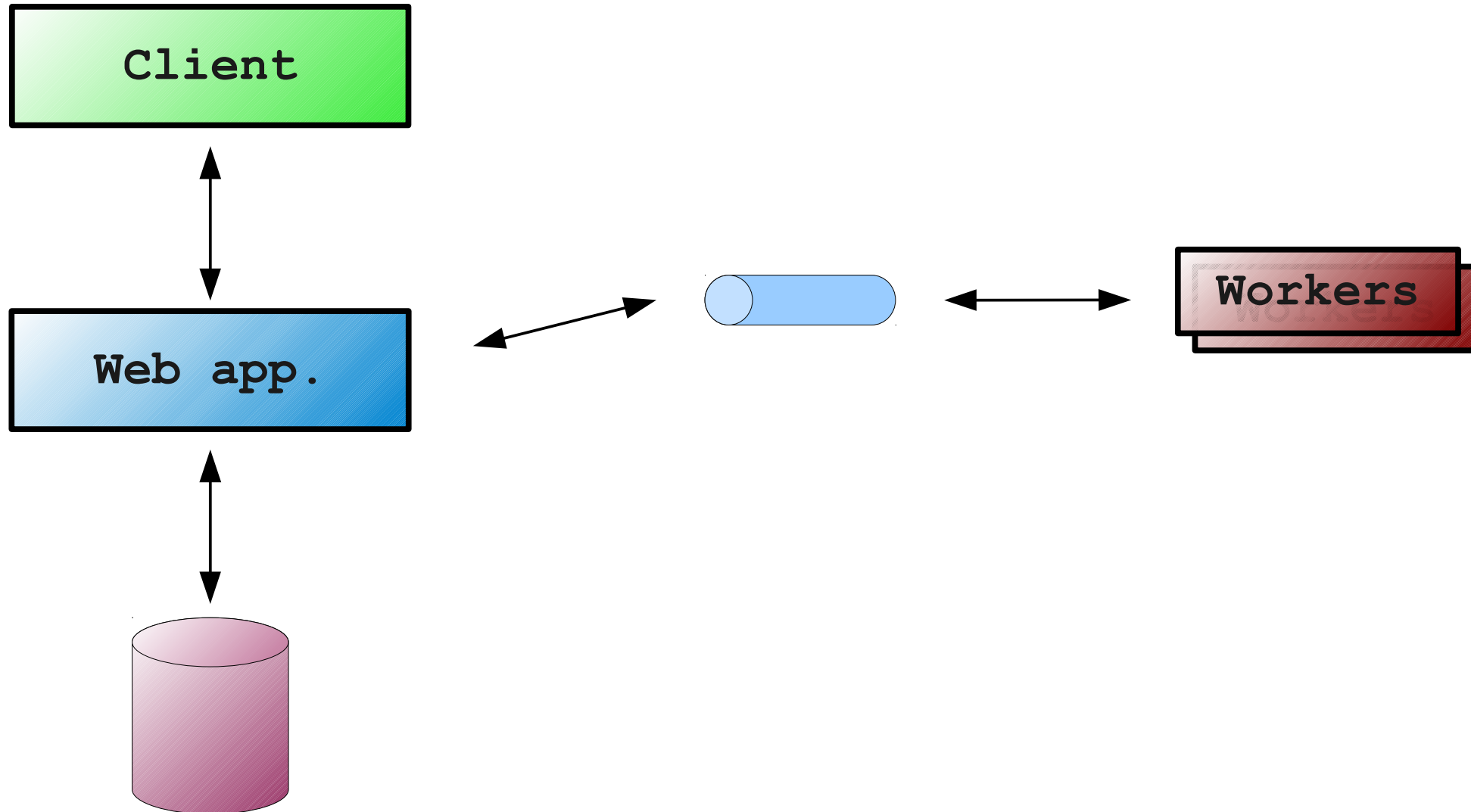
Spring Data REST



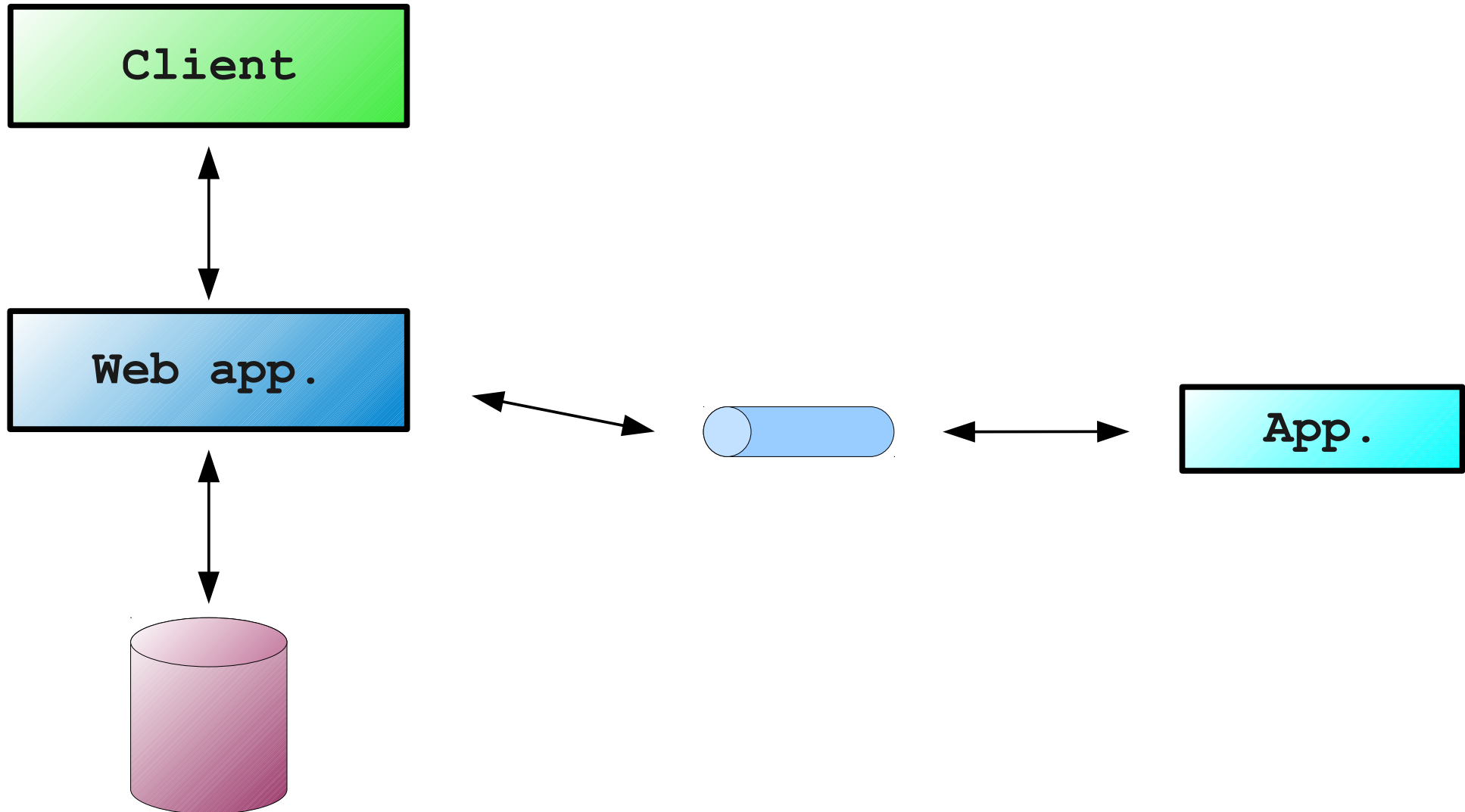
Intégration



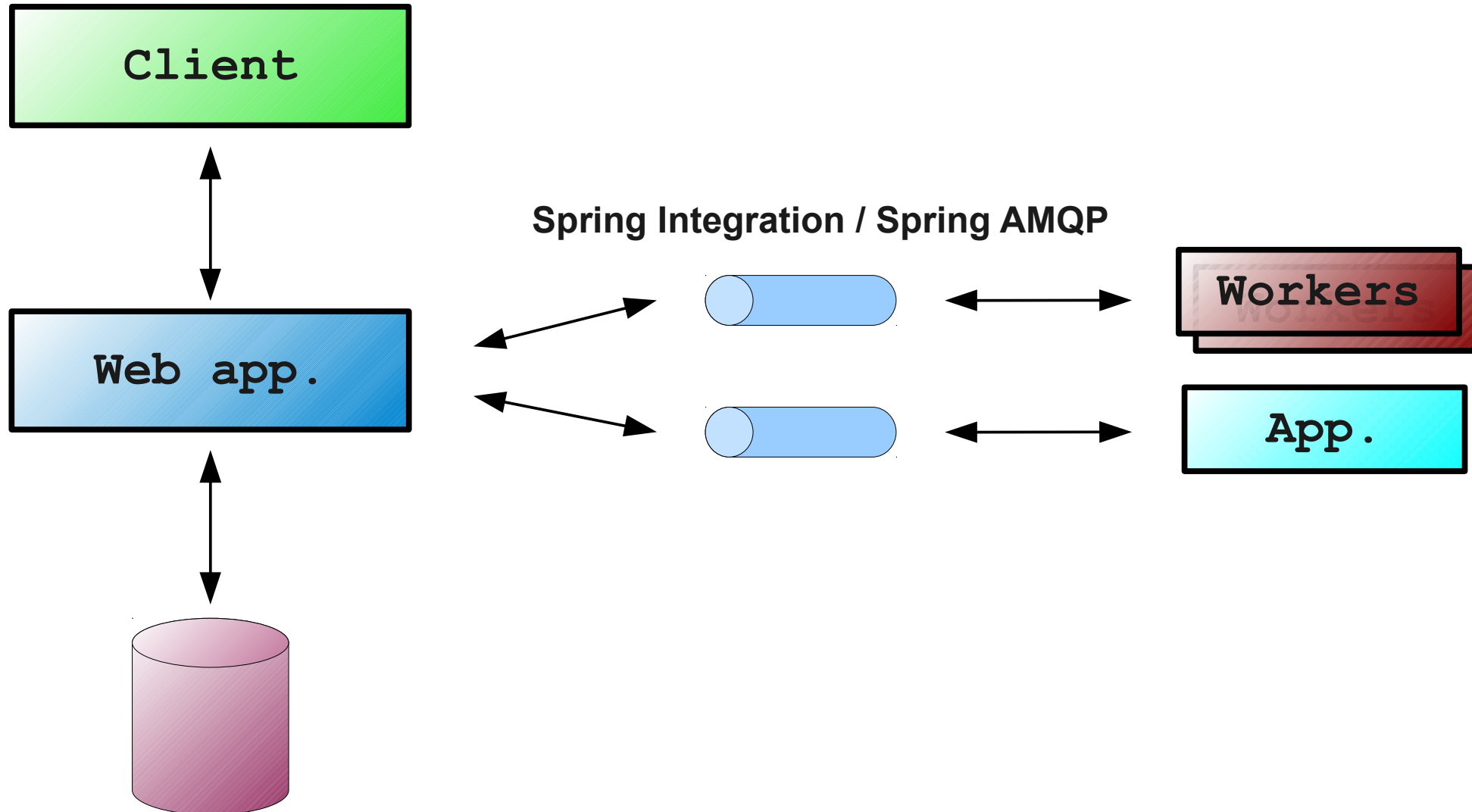
Intégration



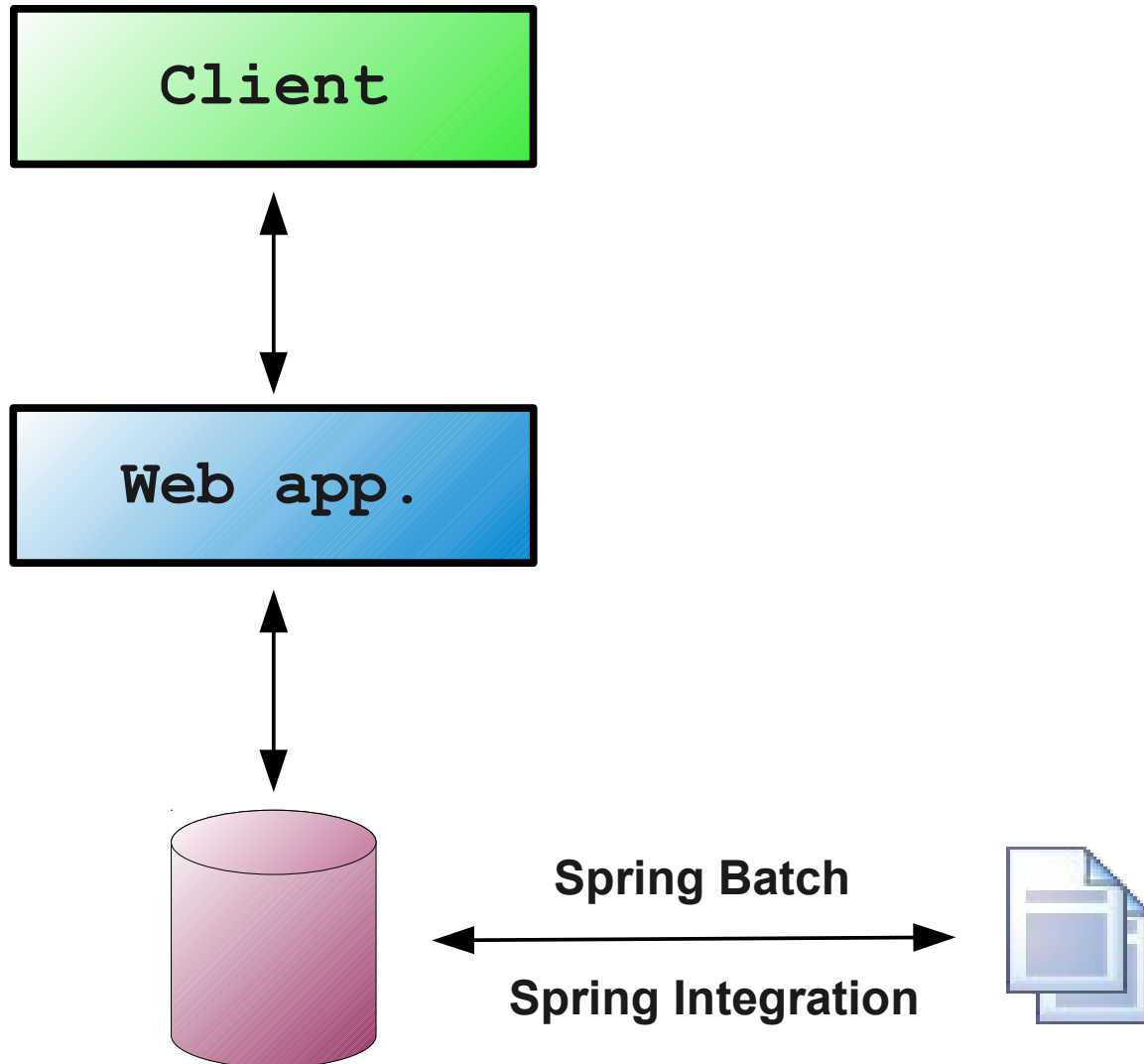
Intégration



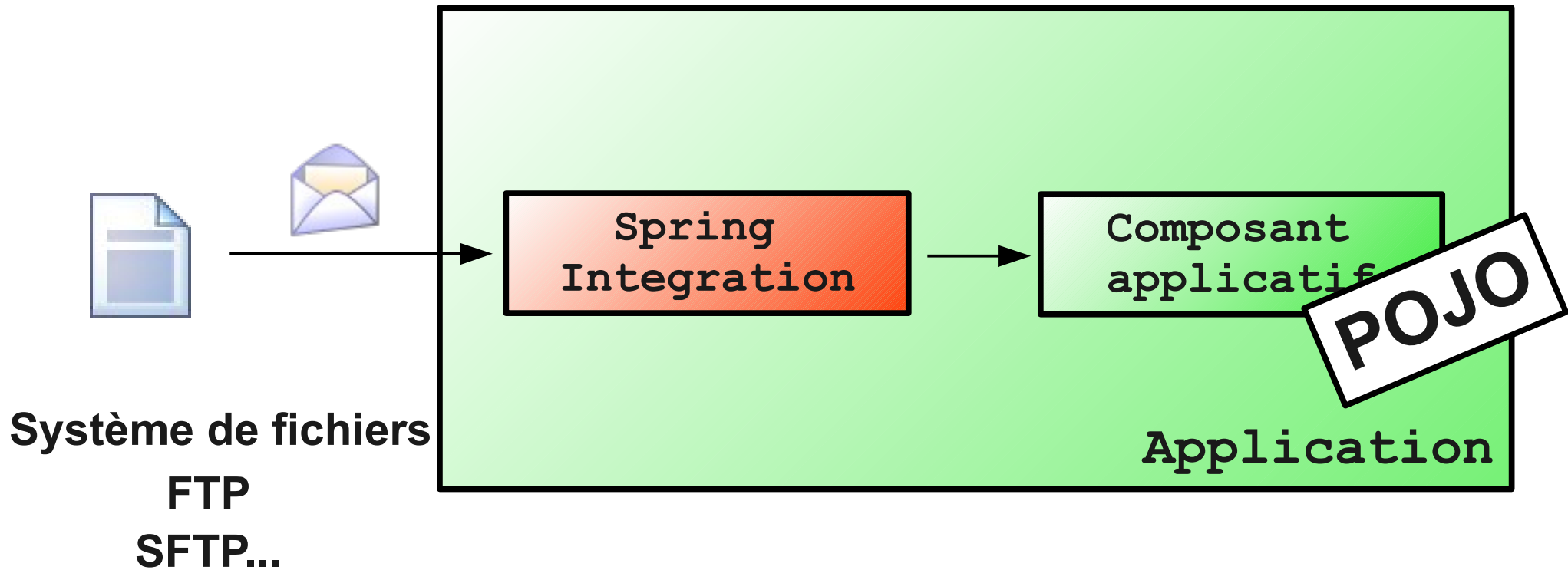
Intégration



Intégration



Spring Integration





















Spring Integration

Merge pull request [#664](#) from garyrussell/INT-2803 ...

 Mark Fisher authored 2 hours ago

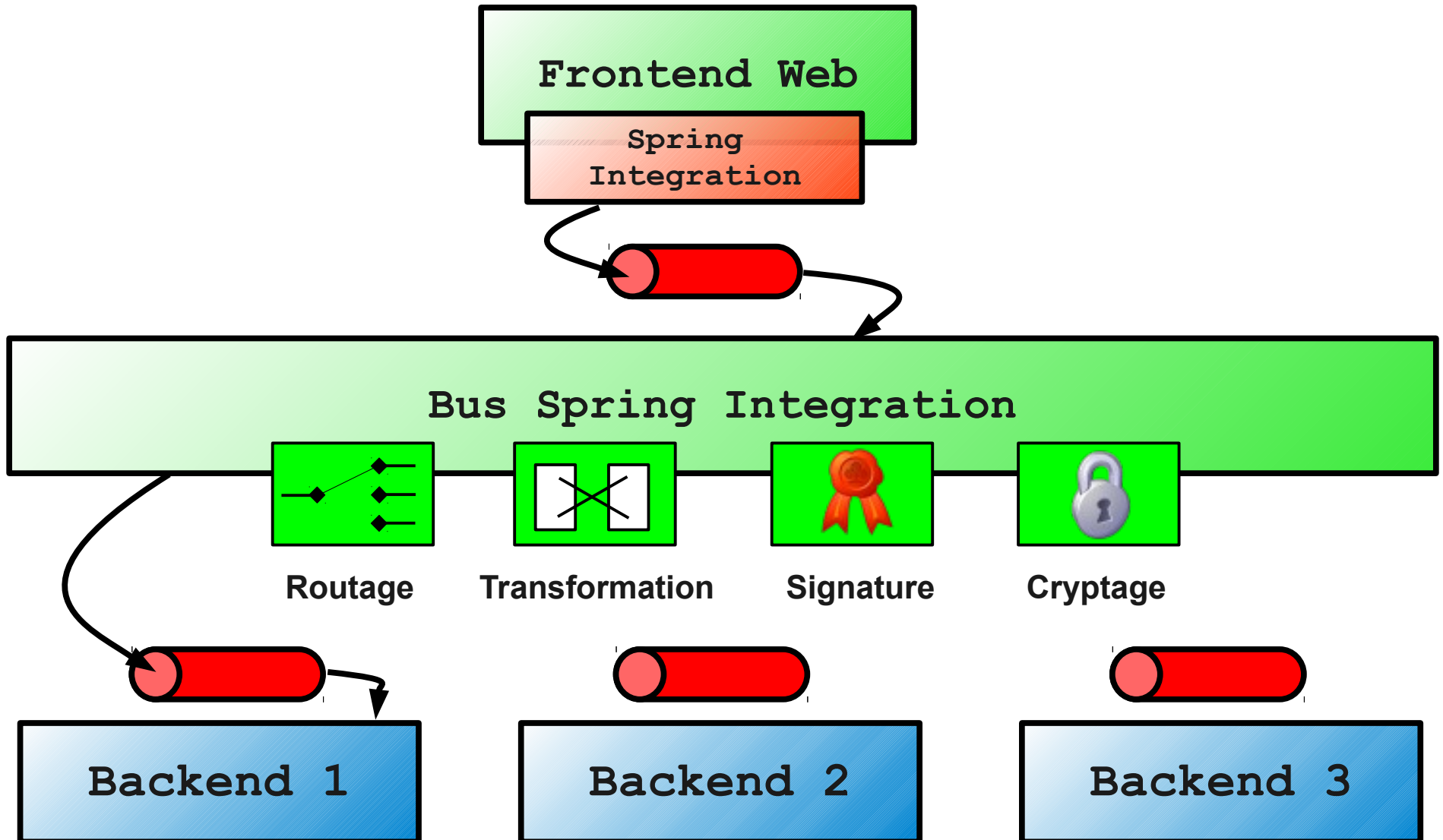
 latest commit [1c8b849af7](#)

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

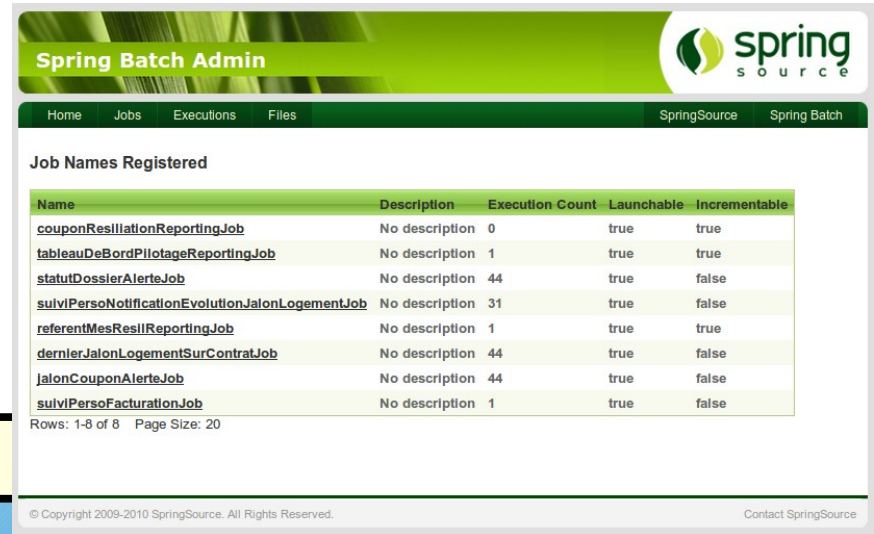
Spring Integration

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

Spring Integration <=> bus



Spring Integration & Spring Batch



Spring Batch Admin

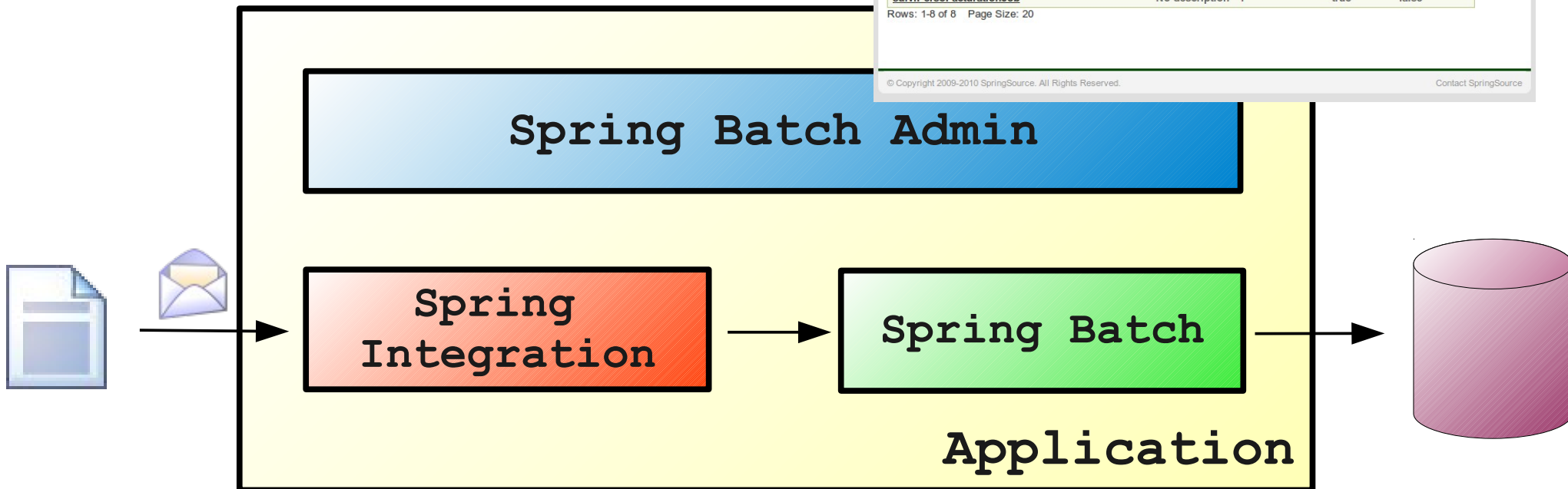
Home Jobs Executions Files SpringSource Spring Batch

Job Names Registered

Name	Description	Execution Count	Launchable	Incrementable
couponResiliationReportingJob	No description	0	true	true
tableauDeBordPilotageReportingJob	No description	1	true	true
statutDossierAlerteJob	No description	44	true	false
sulviPersoNotificationEvolutionJalonLogementJob	No description	31	true	false
referentMesResilReportingJob	No description	1	true	true
dernierJalonLogementSurContratJob	No description	44	true	false
JalonCouponAlerteJob	No description	44	true	false
sulviPersoFacturationJob	No description	1	true	false

Rows: 1-8 of 8 Page Size: 20

© Copyright 2009-2010 SpringSource. All Rights Reserved. Contact SpringSource

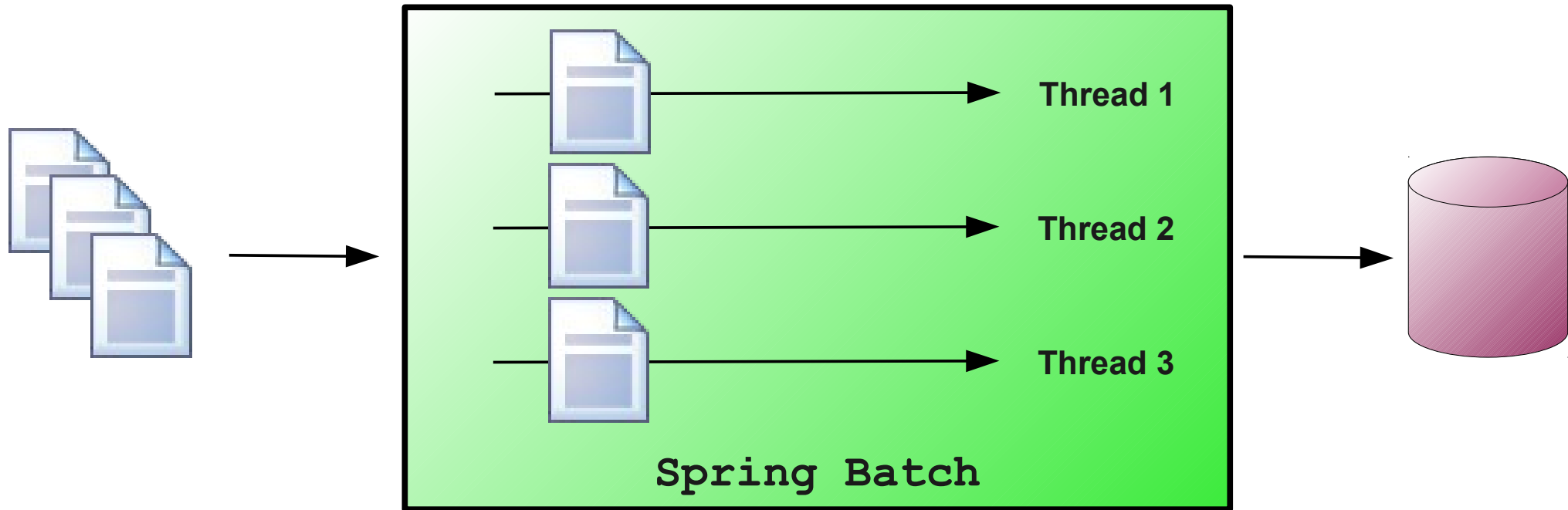


Spring Batch

- 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

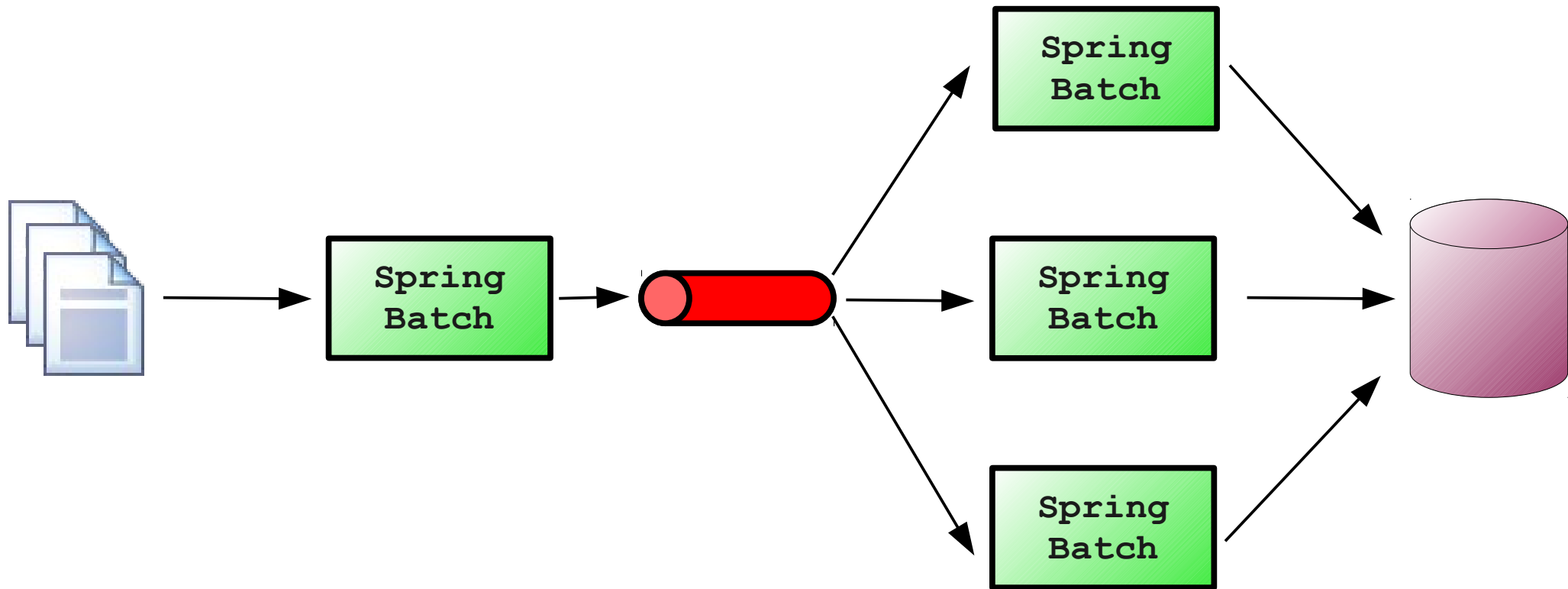
Spring Batch & scaling

- Local partitioning

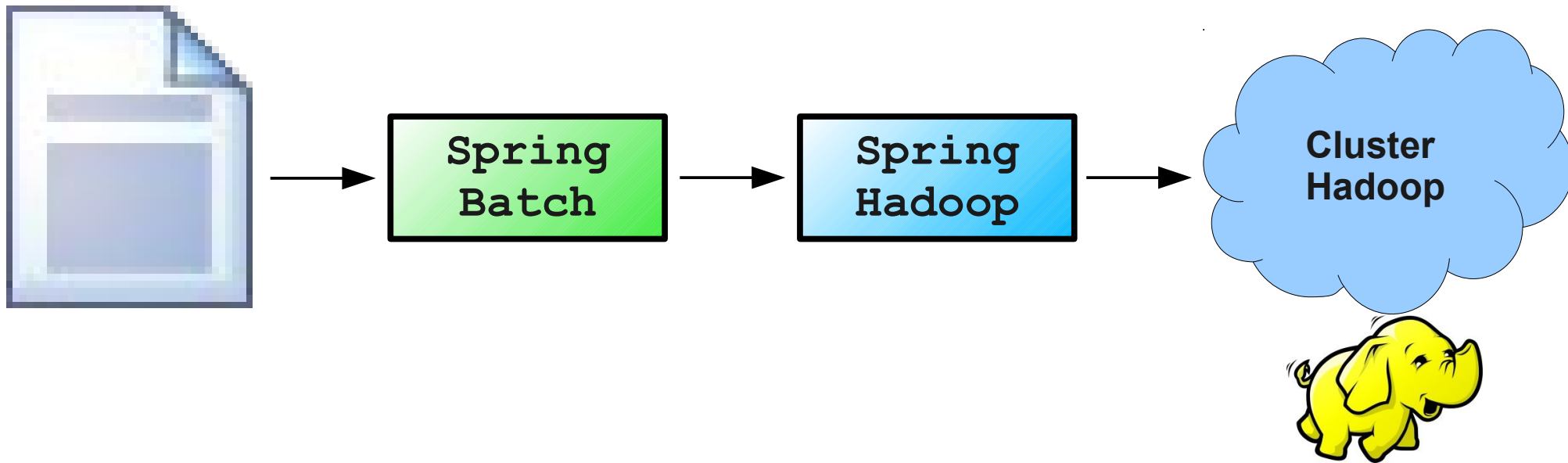


Spring Batch & scaling

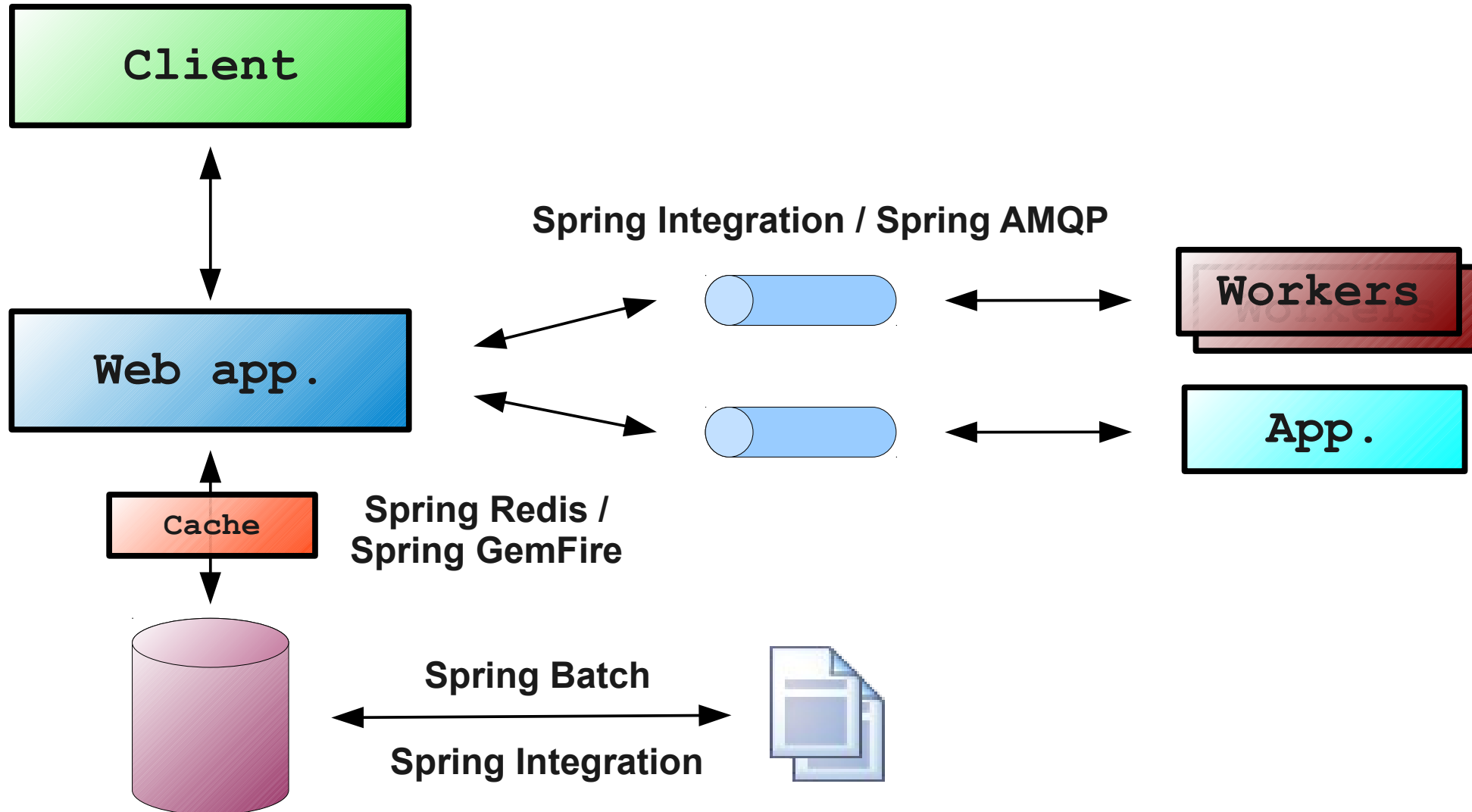
- Remote partitioning



Spring Batch & Spring Hadoop



Intégration



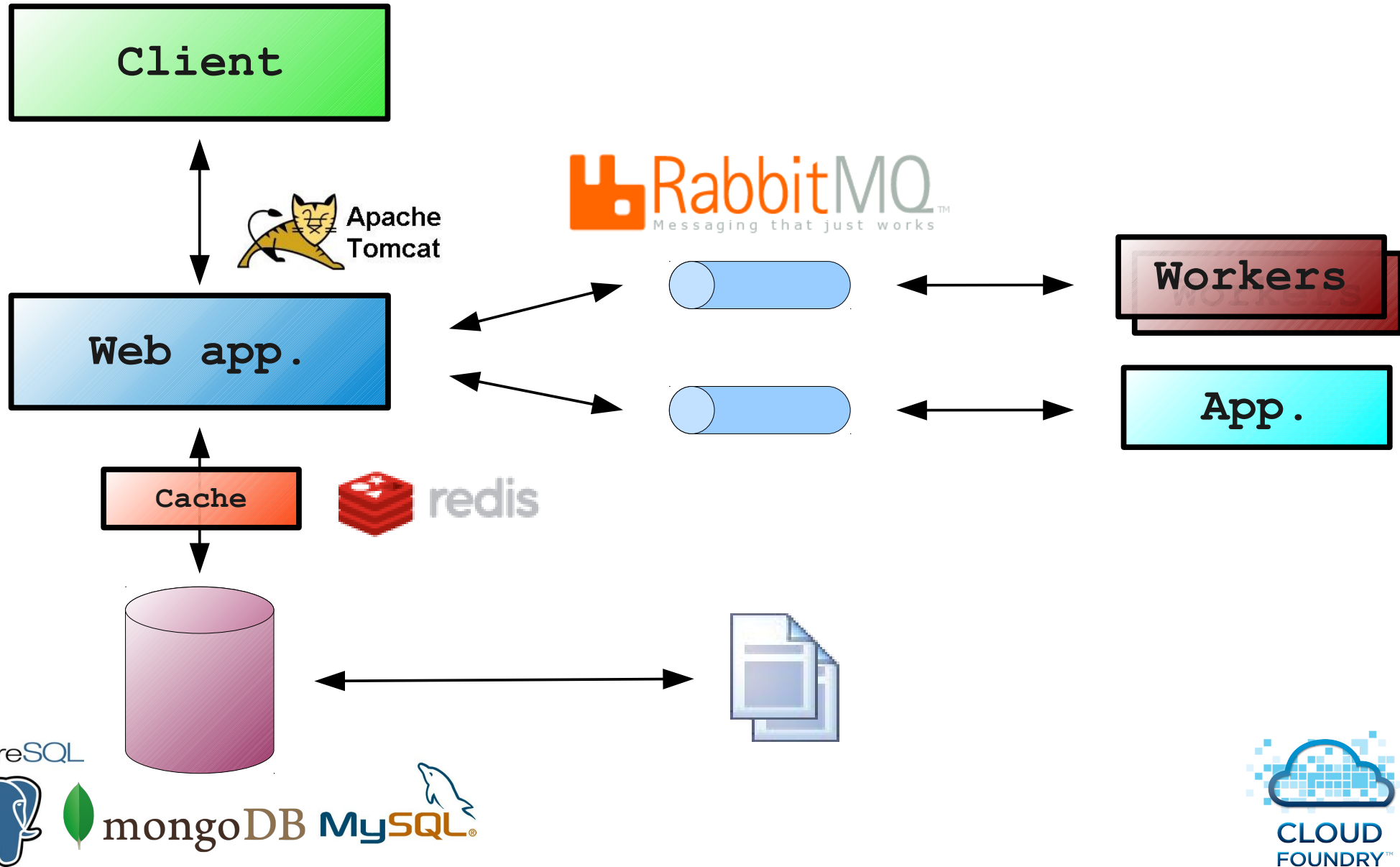
Intégration et déploiement



PostgreSQL



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 photograph of a chipmunk standing on a large, reddish-brown rock. The chipmunk is facing right, looking slightly upwards. It has a brown and tan striped pattern on its back and a lighter brown body. The background consists of more rocks and some small green plants. The text "Questions ?" is overlaid in white, bold font across the middle of the chipmunk's body.

Questions ?

Merci !