Spring : tendances, nouveautés et perspectives

Arnaud Cogoluègnes

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(...) { }

}
```

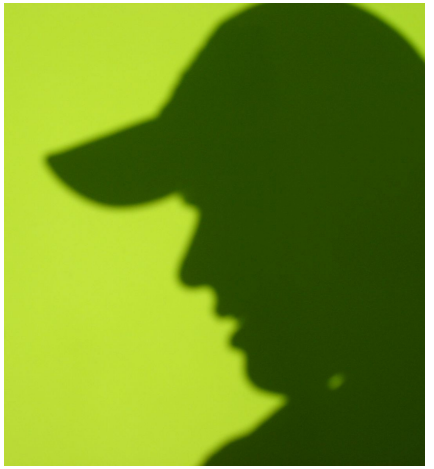
```
<tx:annotation-driven />
```

```
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 namespac~~

@nnotations

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>
```

@nnotations

```
@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" />
```

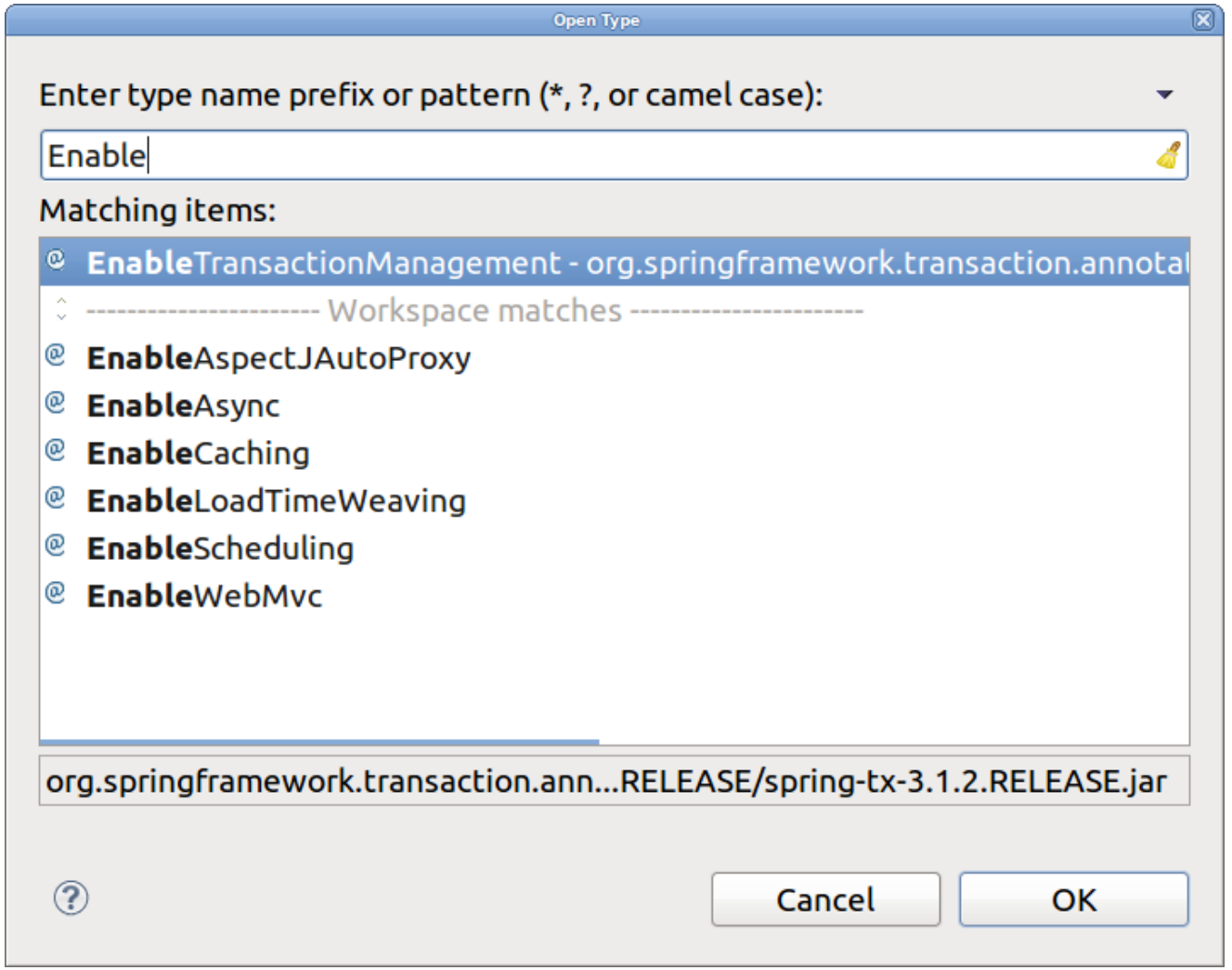
Configuration Java

```
@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

JSON

Simple

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);
  }

}
```

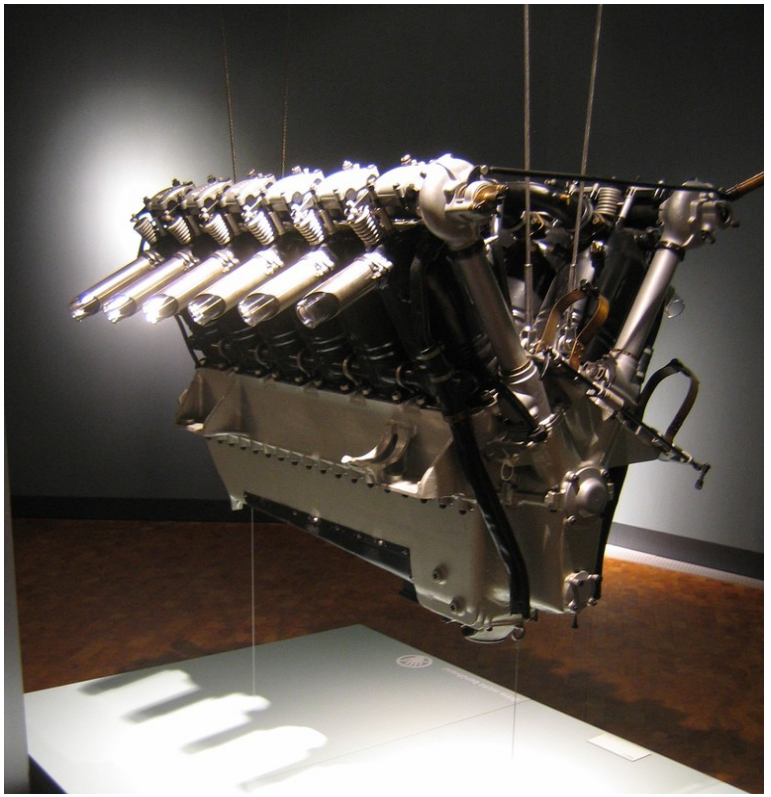
REST

```
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 ?



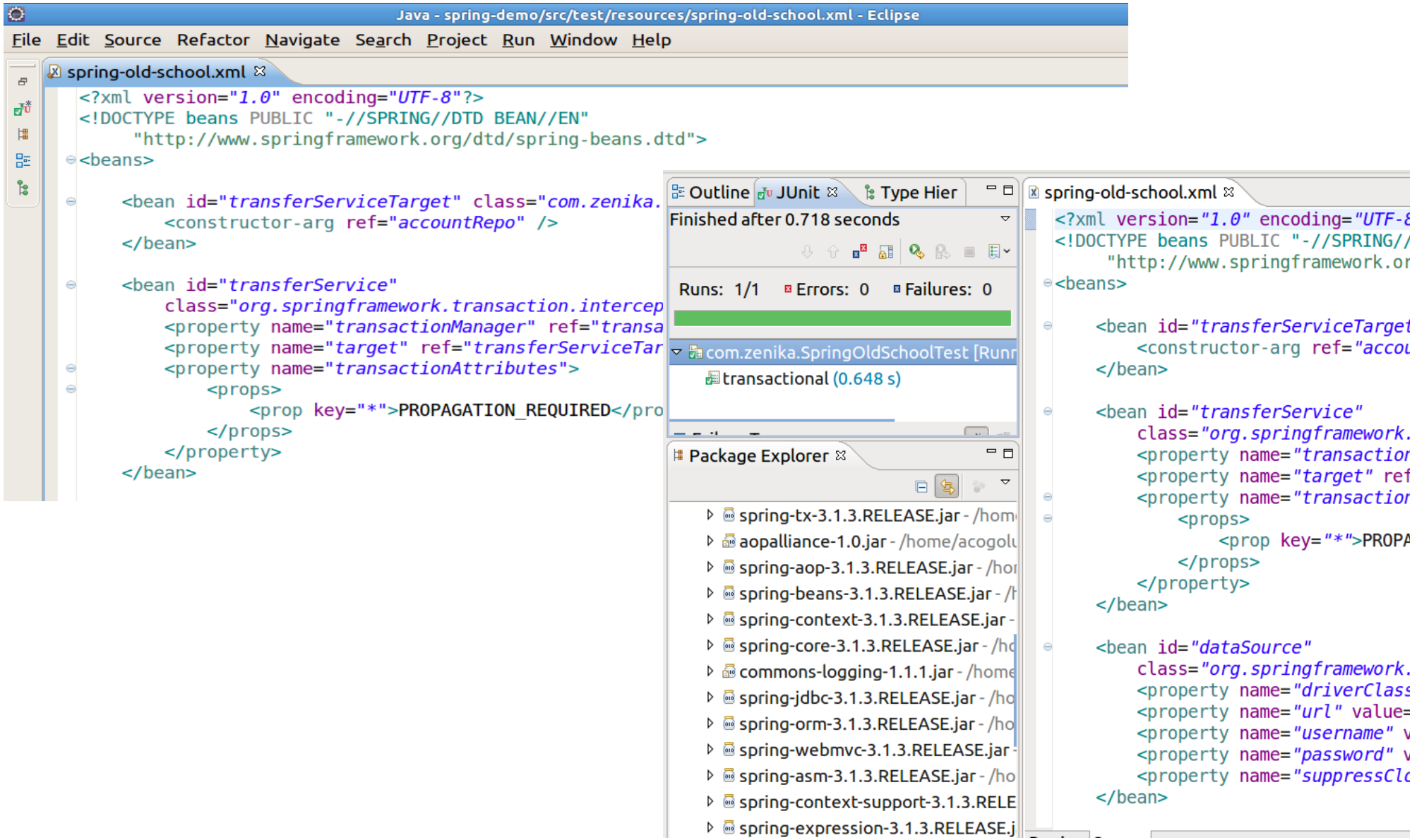
Refactoring complet entre 3.0 et 3.1

Traitement des méthodes
des contrôleurs

100% compatible

Nouveaux points d'extension

Spring : stabilité, compatibilité



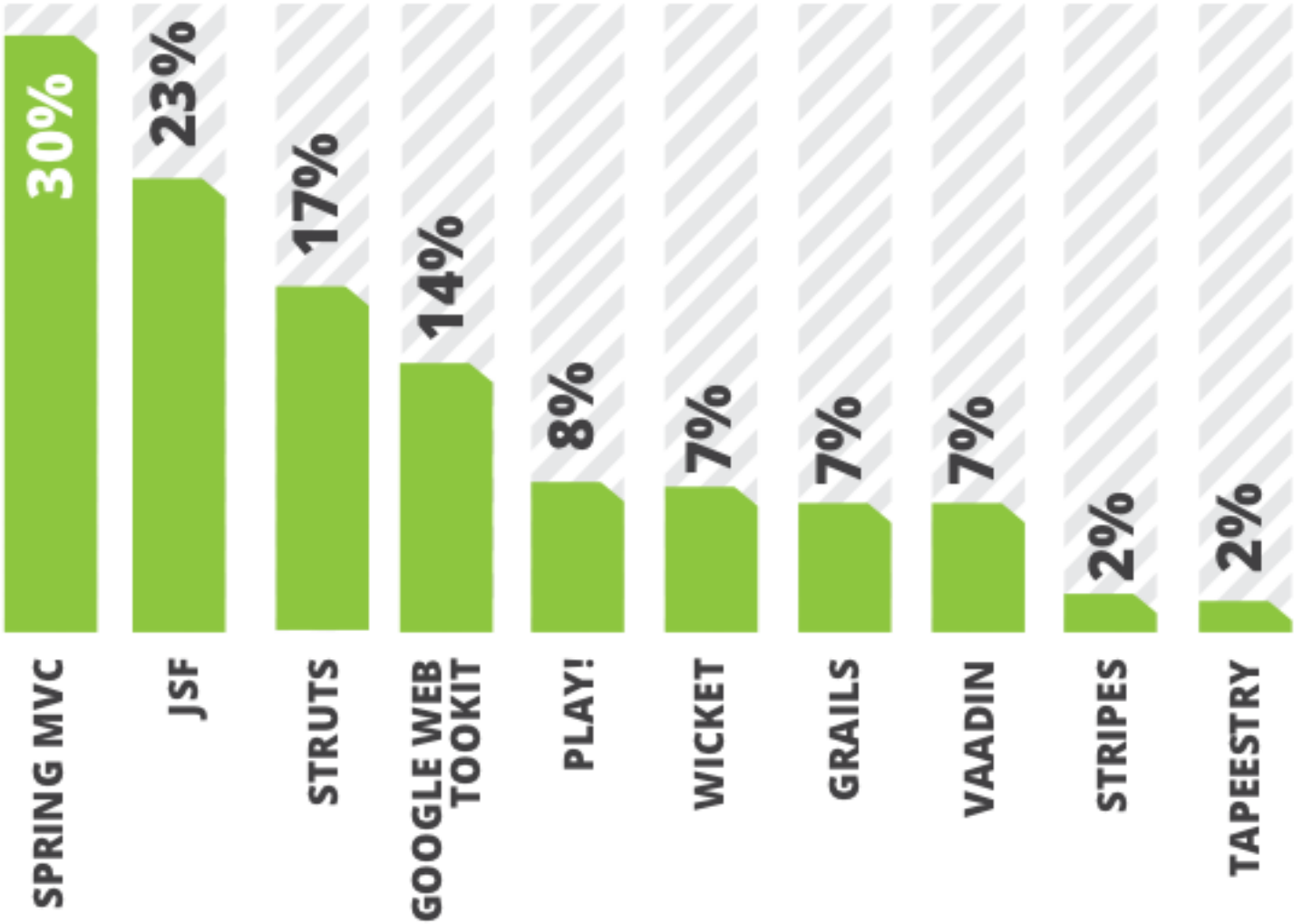
Spring MVC : extensibilité

```
@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

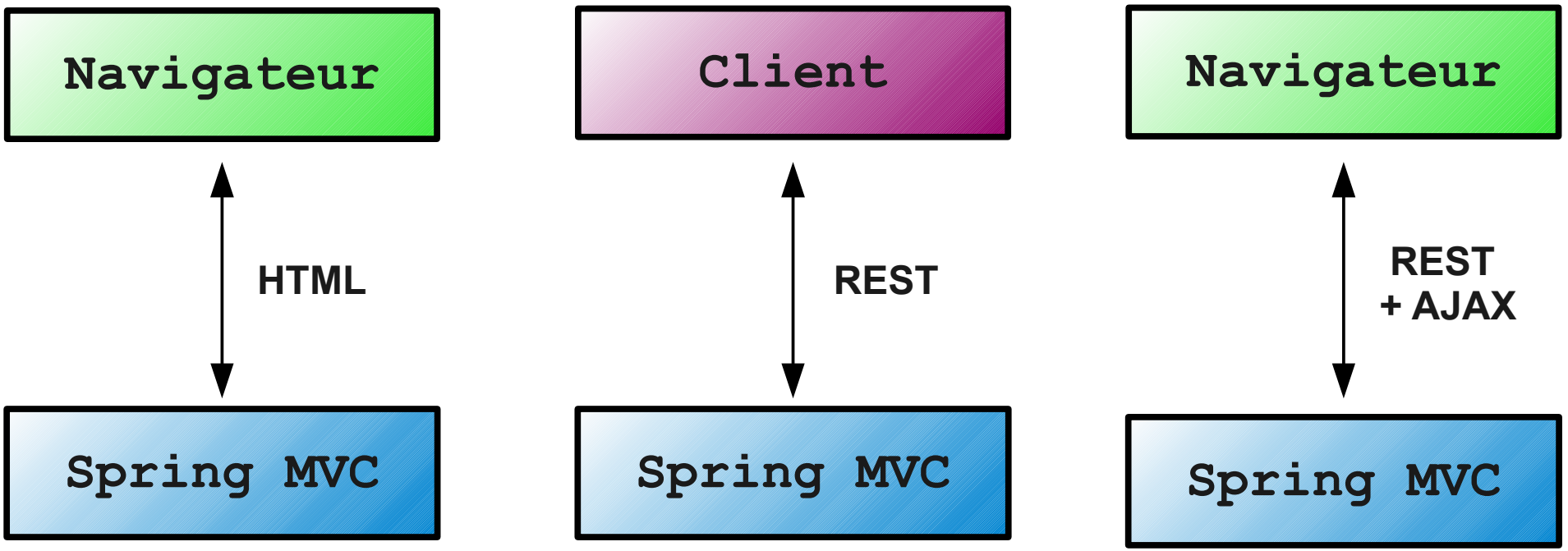
```
@Controller
public class ContactController {

  @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

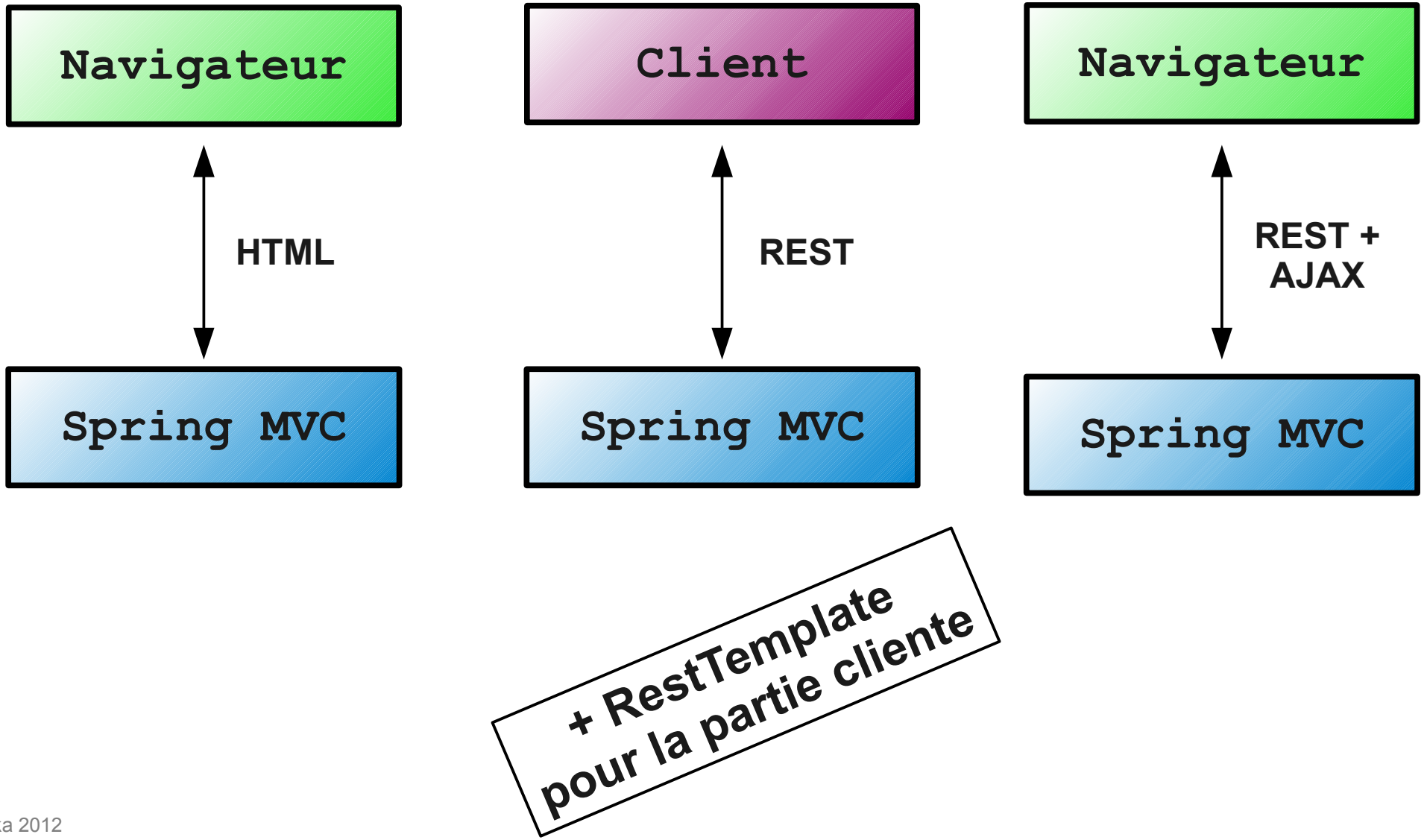


Source : http://zeroturnaround.com/blog/developer-productivity-report-2012-java-tools-tech-devs-and-data/

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("$[0].name").value("NAME1"))
      .andExpect(jsonPath("$[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 »

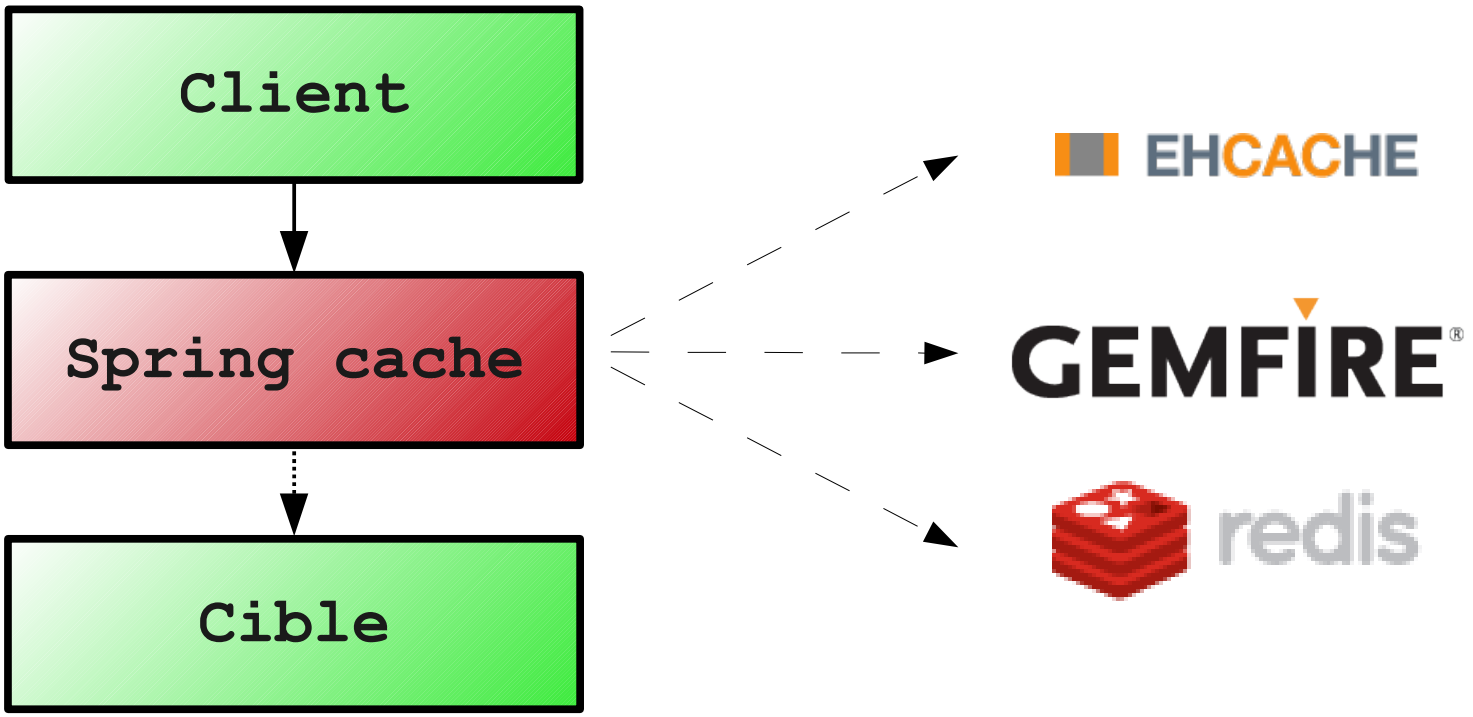
```
(...)
public class MetaModelControllerTest {
  (...)
  @Test public void list() throws Exception {
    mockMvc.perform(get("/metamodel").accept(MediaType.APPLICATION_JSON))
      .andExpect(status().isOk())
      .andExpect(content().contentType("application/json;charset=UTF-8"))
      .andExpect(jsonPath("$[0].name").value("NAME1"))
      .andExpect(jsonPath("$[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")
      );
  }
}
```

Cache

```
public class ContactRepository {

  @Cacheable("contacts")
  Contact findOne(Long id) { ... }

}
```

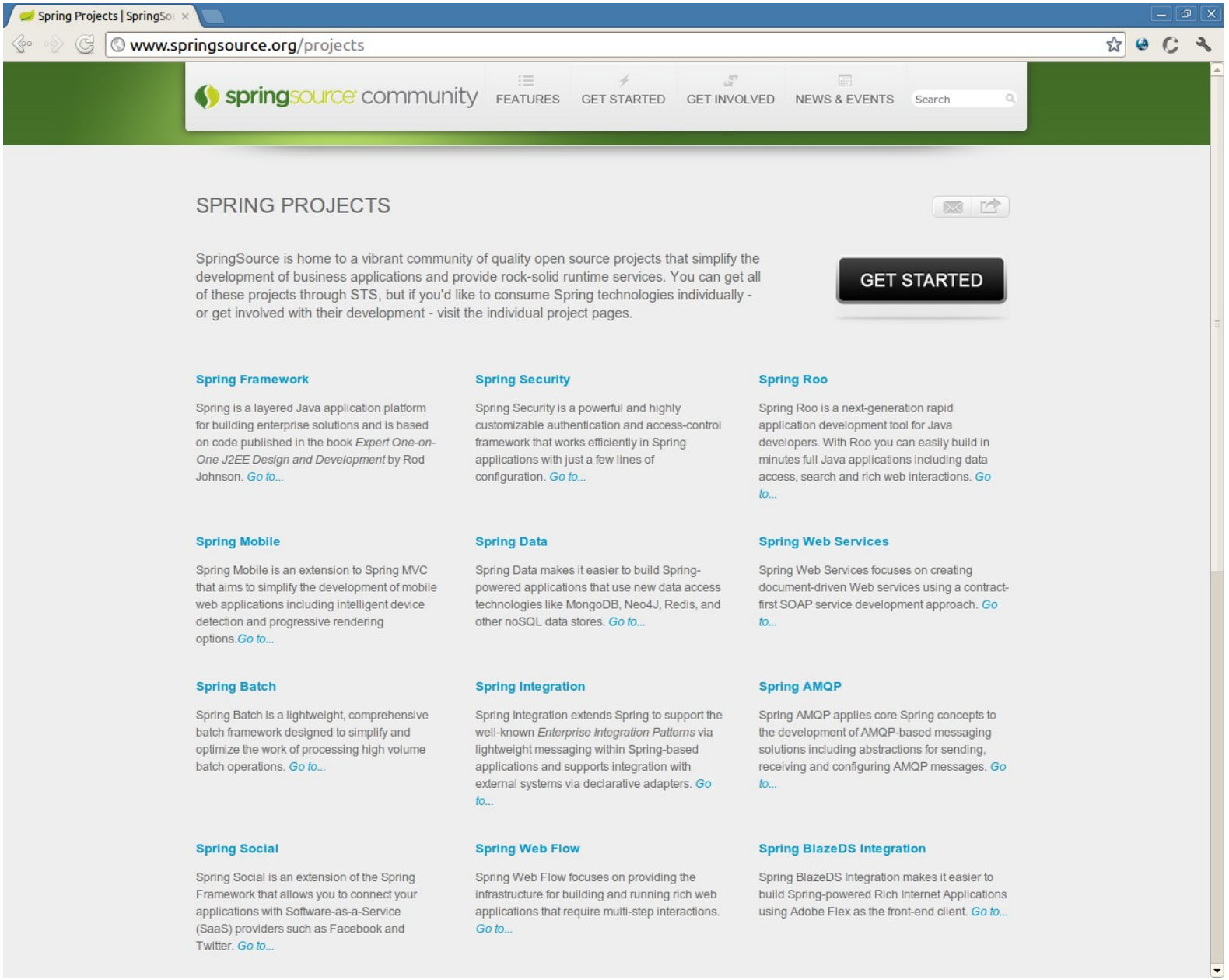
```
<cache:annotation-driven />

<bean id="cacheManager"
      class="...">
  ...
</bean>
```

Cache



3.1 spring

Portfolio Spring



Portfolio Spring

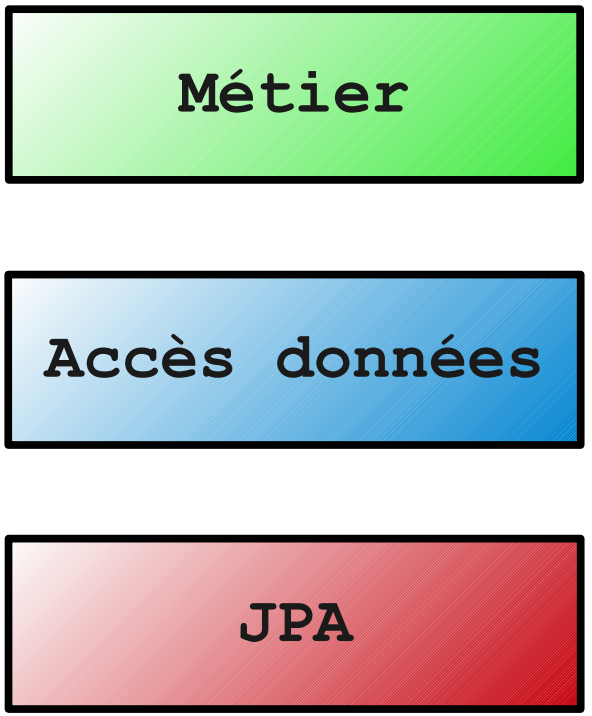
Spring Mobile / Spring Android

Spring Social

Spring Data


Neo4j
the graph database
mongoDB
hadoop
GEMFIRE
redis
riak

Spring Data JPA



DAO dynamiques, intelligents

Moins de code répétitif

Testabilité

Conventions

Pagination & tri

Approche DDD

Spring Data JPA

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

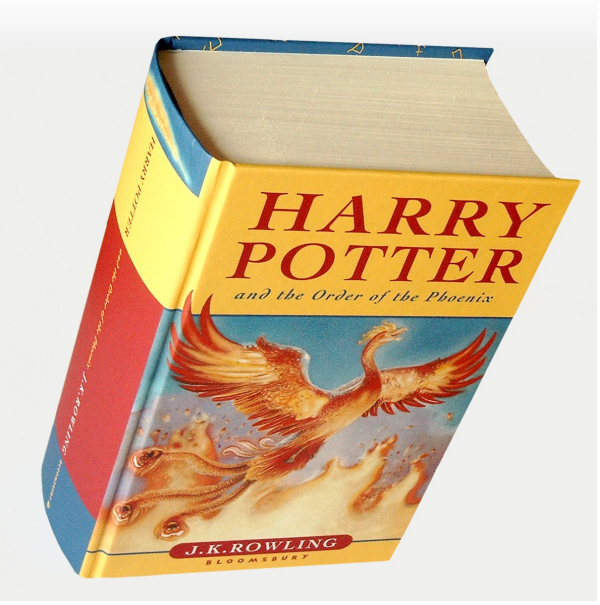
```
<jpa:repositories base-package="com.zenika.repository" />
```

```
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");
```

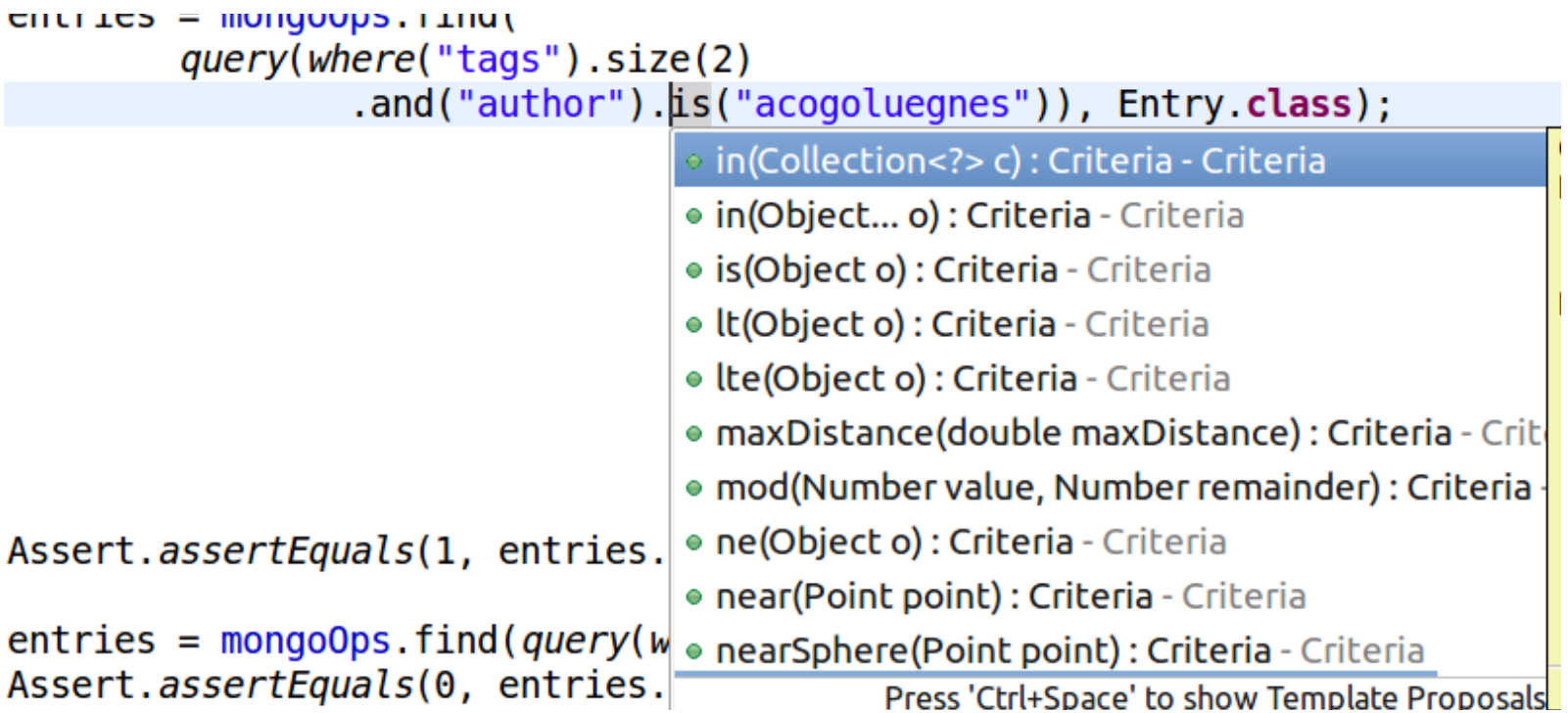
Spring Data MongoDB, API query

```
import static 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
);
```

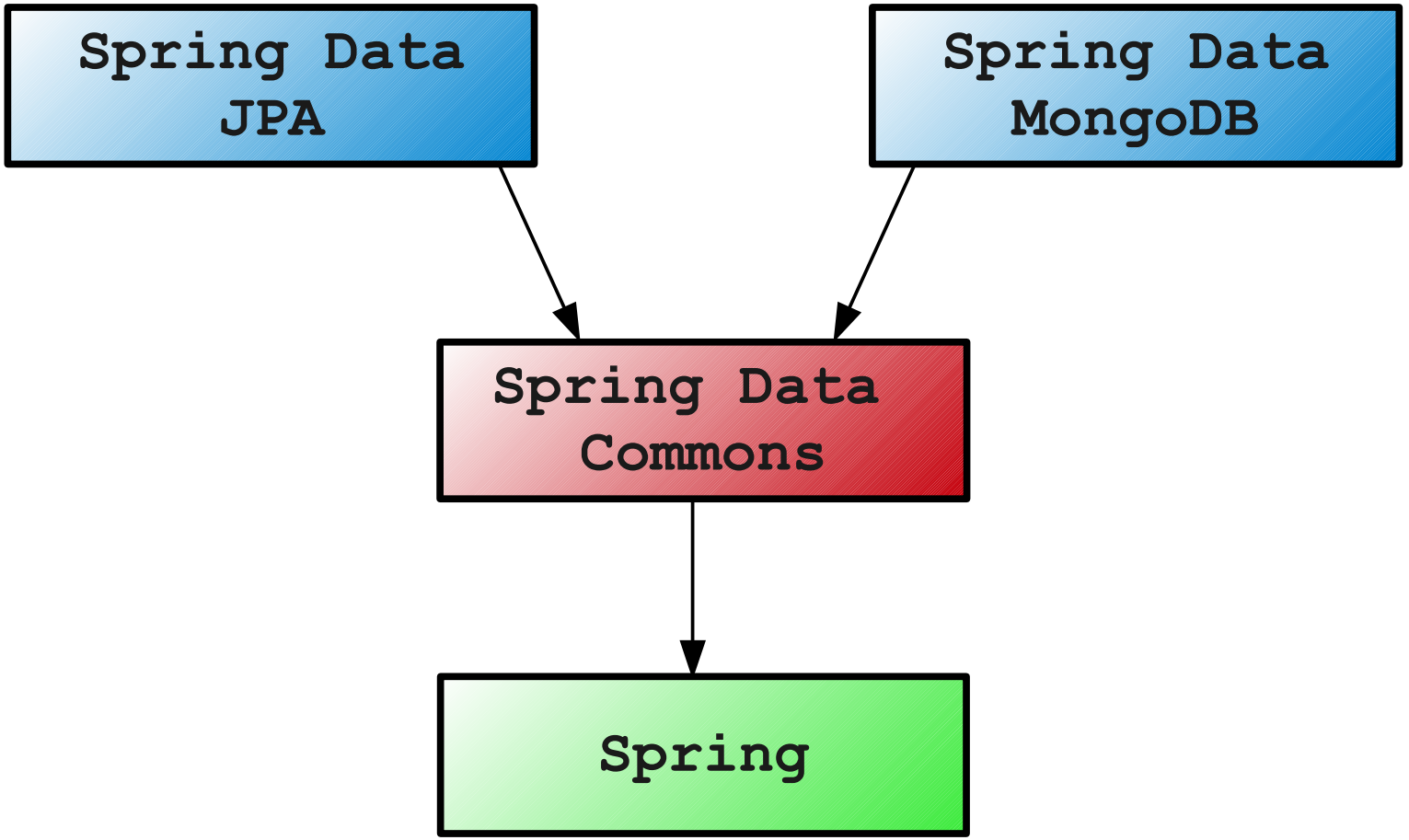
Spring Data MongoDB, API query

```
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



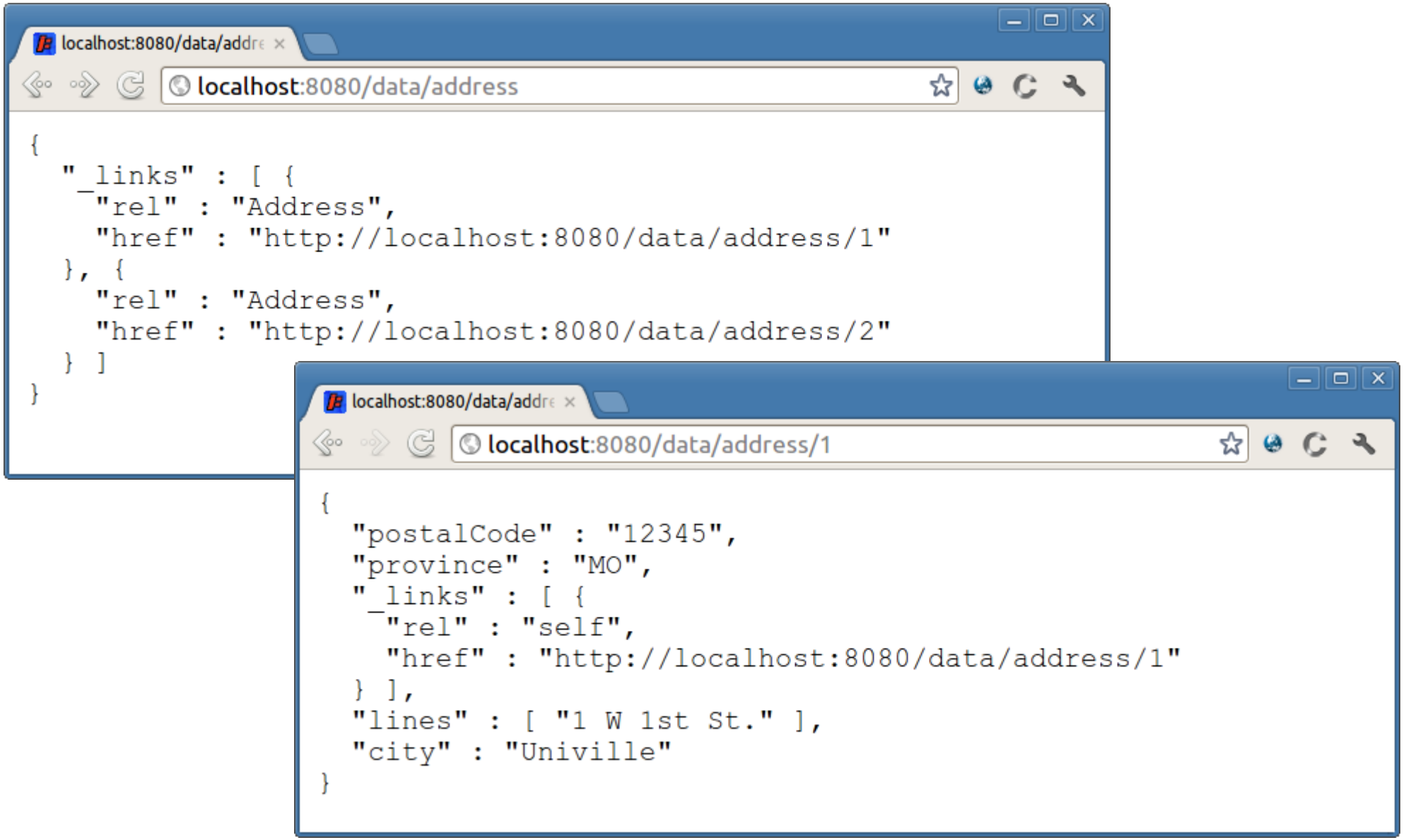
Spring Data JPA/MongoDB



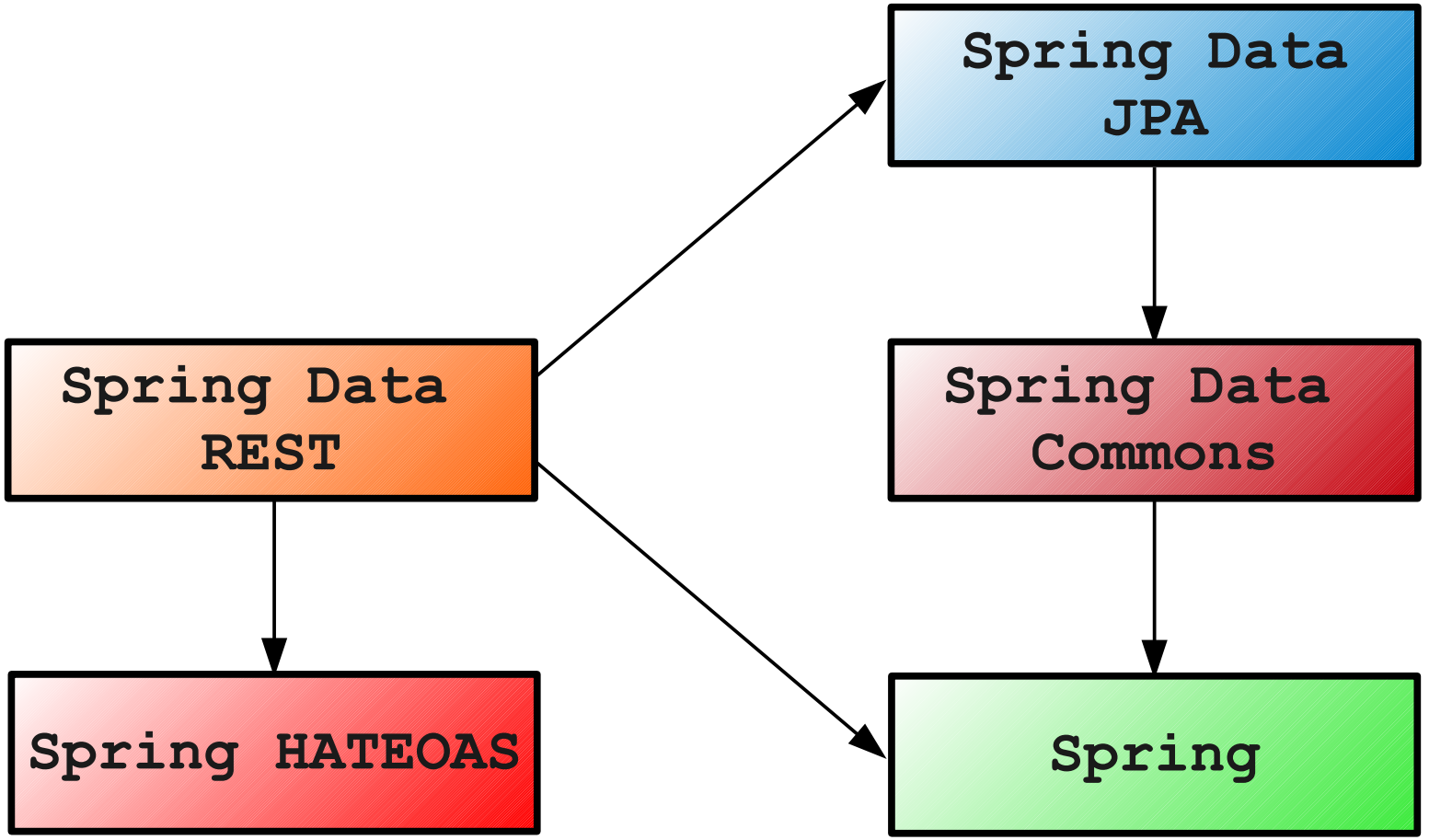
Spring Data REST

- 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 »

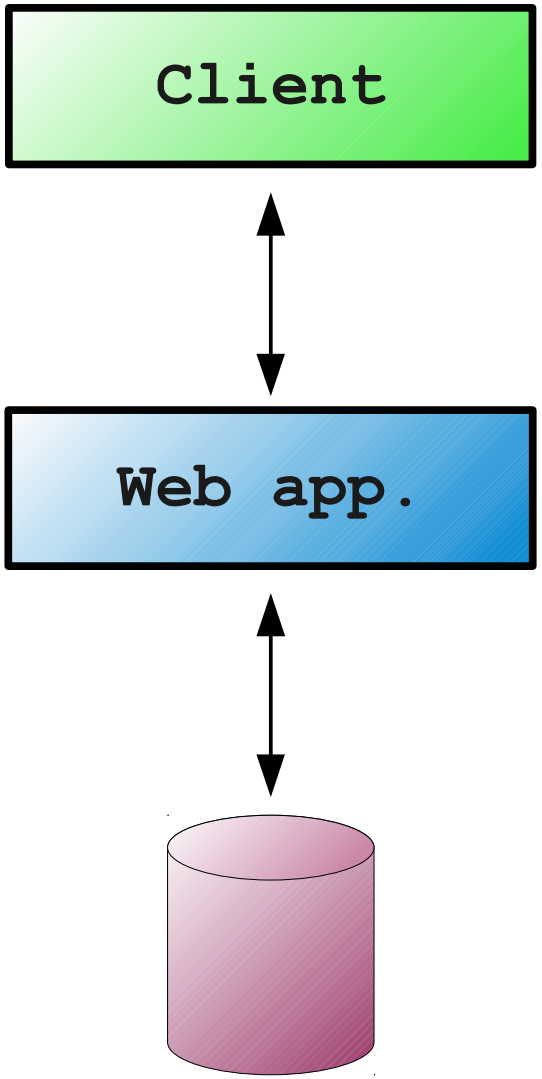
Spring Data REST



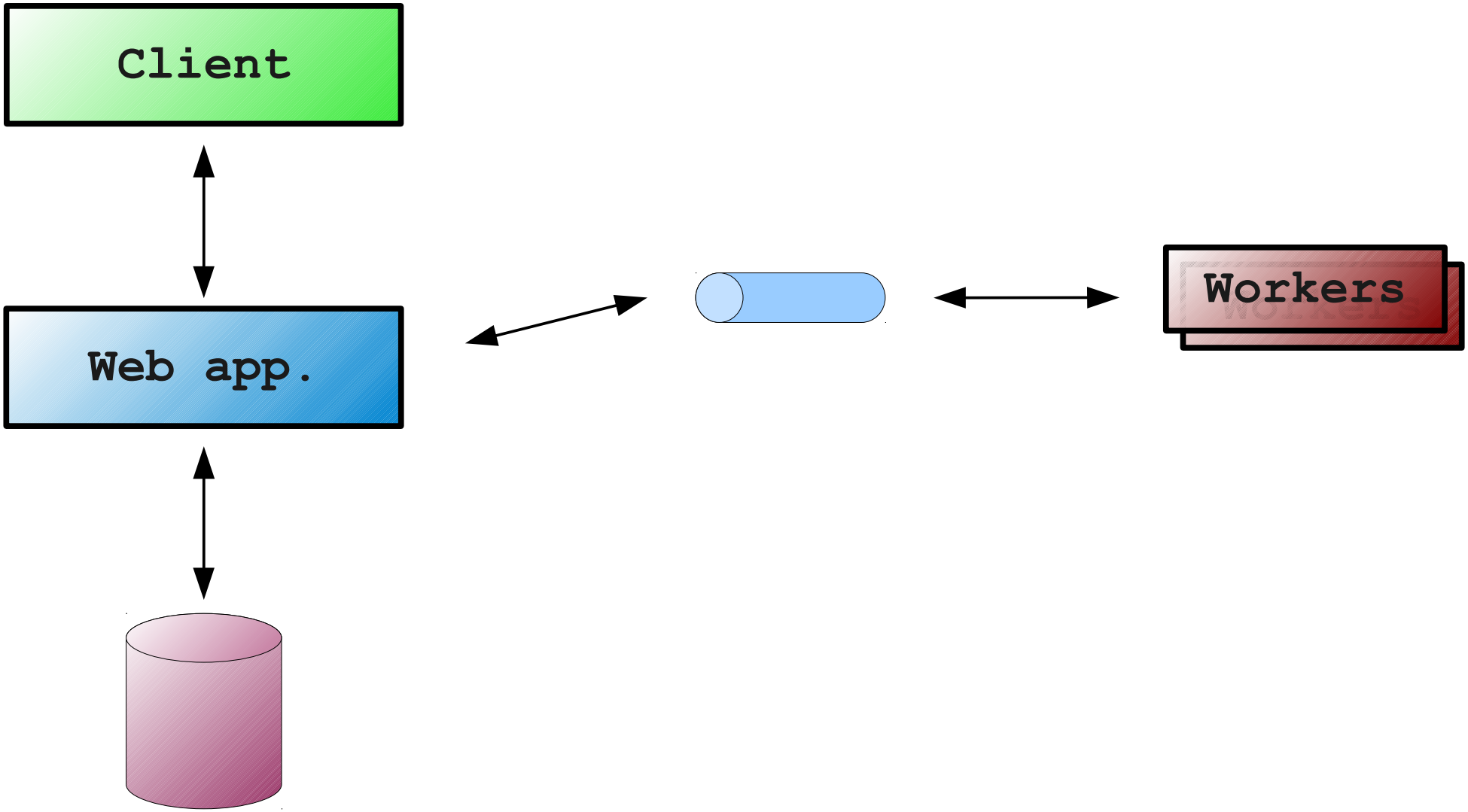
Spring Data REST



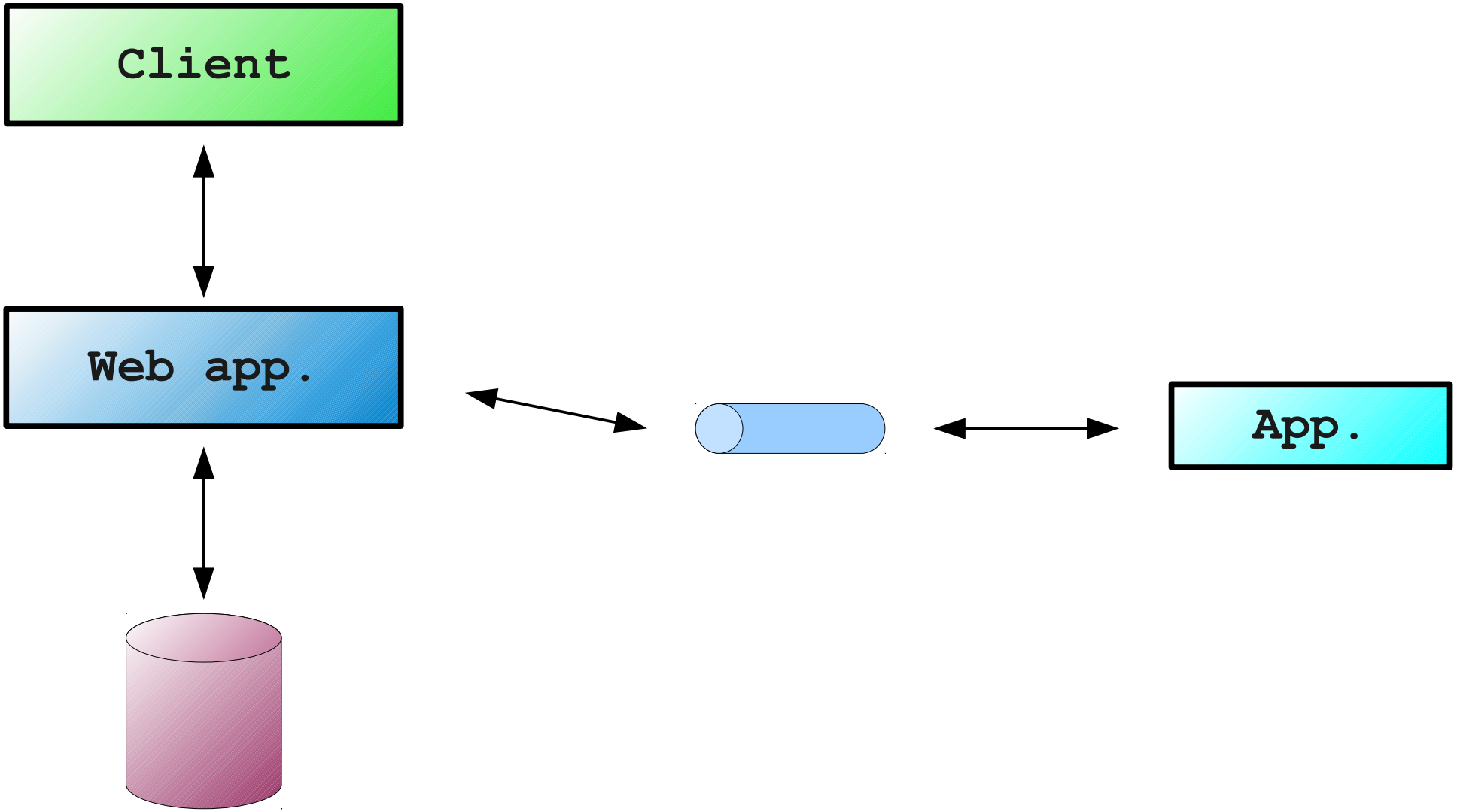
Intégration



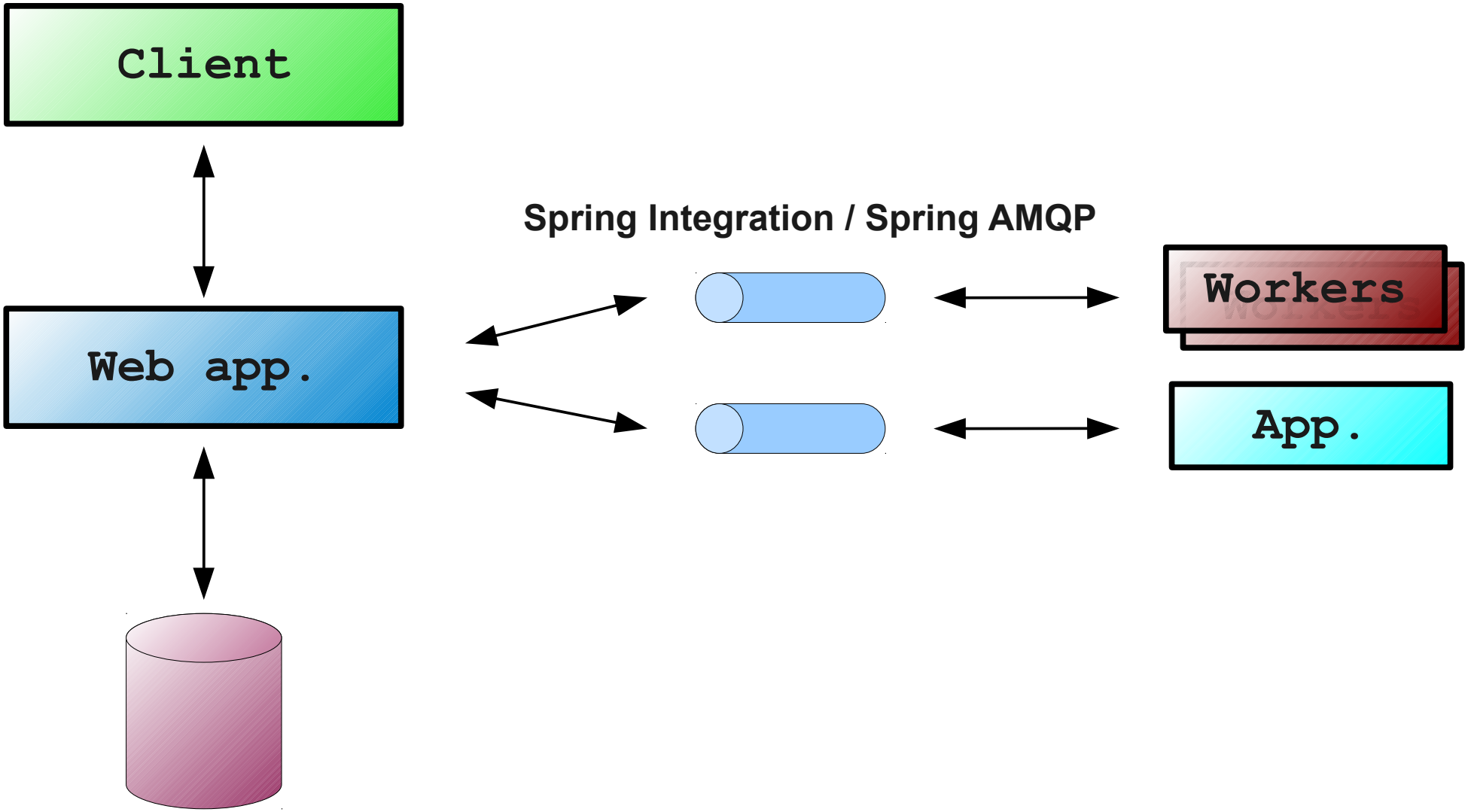
Intégration



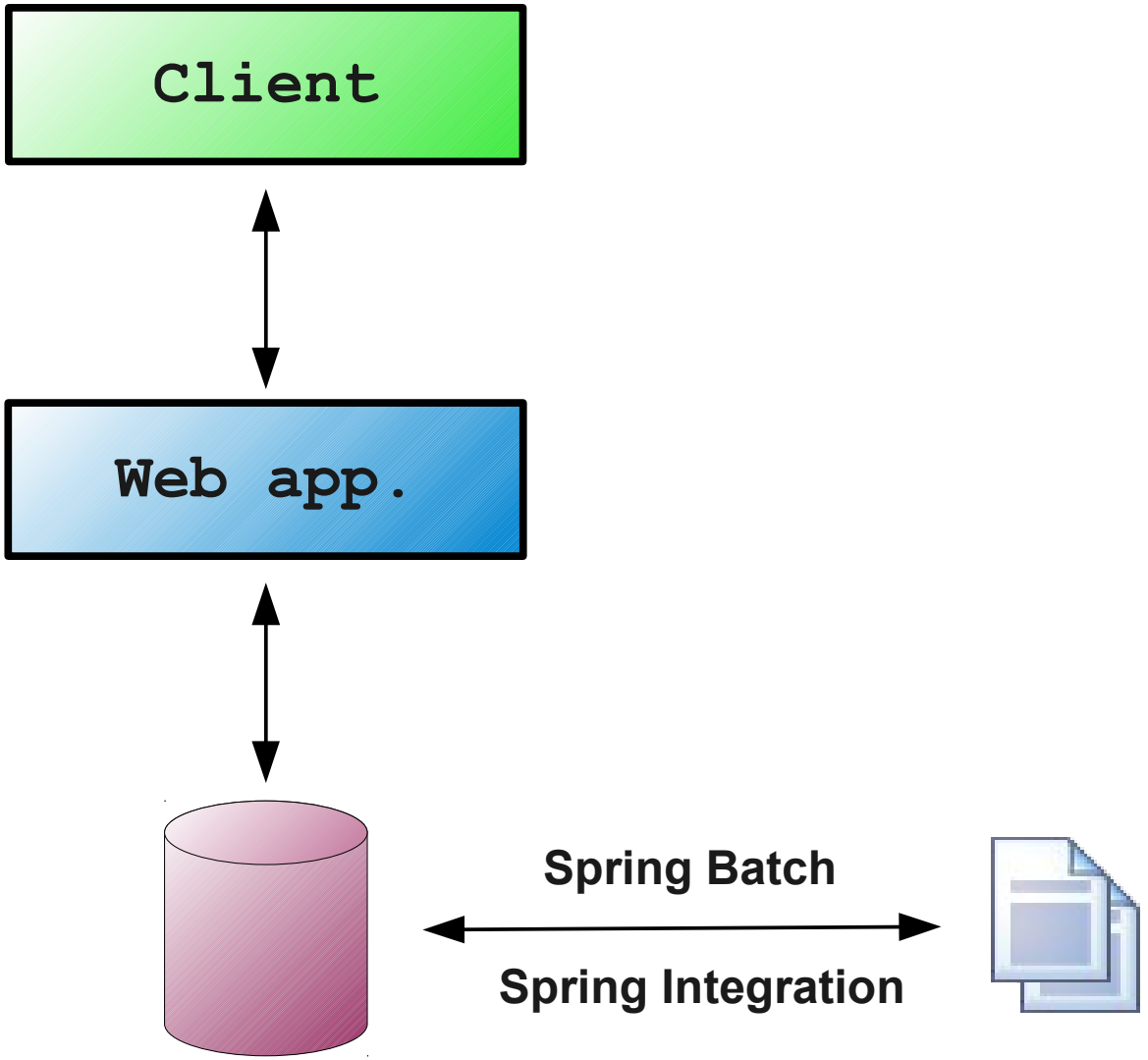
Intégration



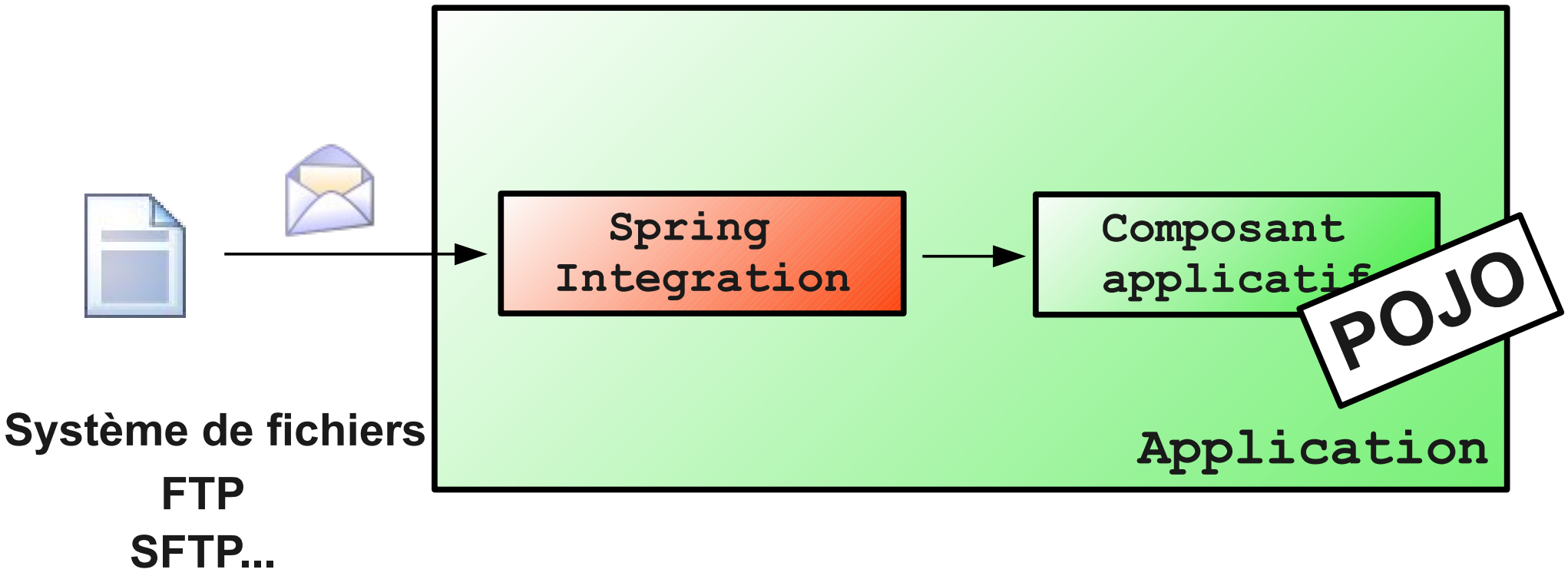
Intégration



Intégration



Spring Integration



Spring Integration

Merge pull request **#664** from garyrussell/INT-2803 ···

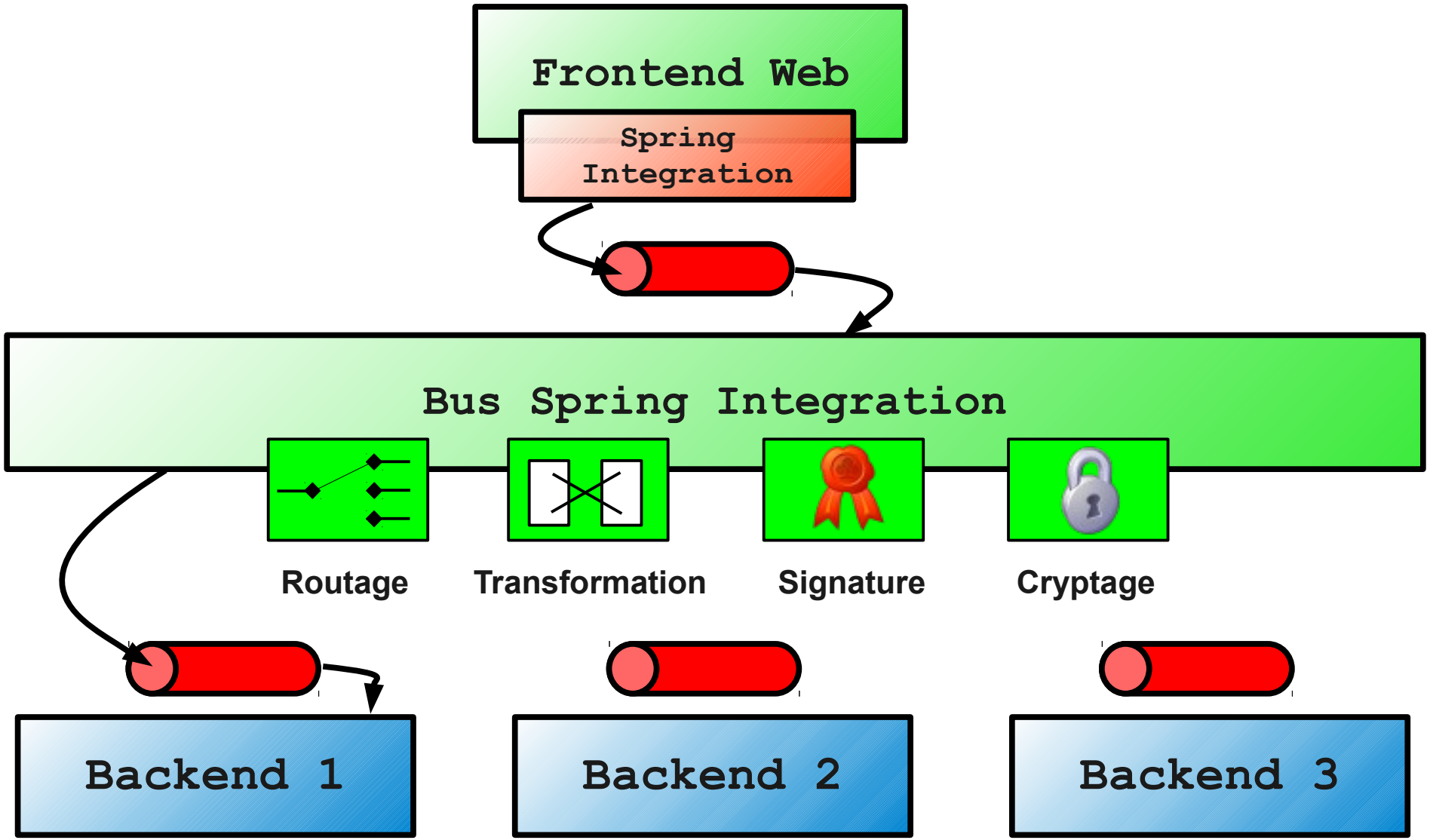
Mark Fisher authored 2 hours ago — latest commit `1c8b849af7`

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 TxSynchFactory: 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]

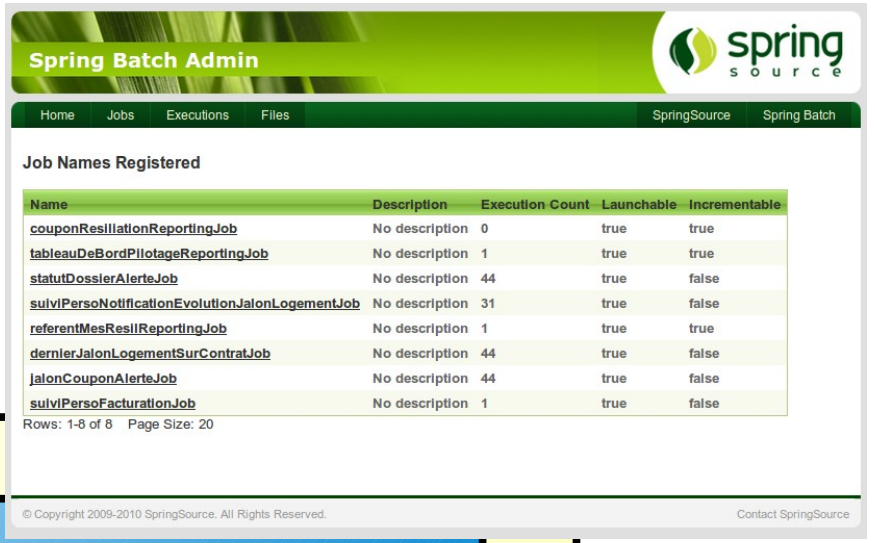
Spring Integration

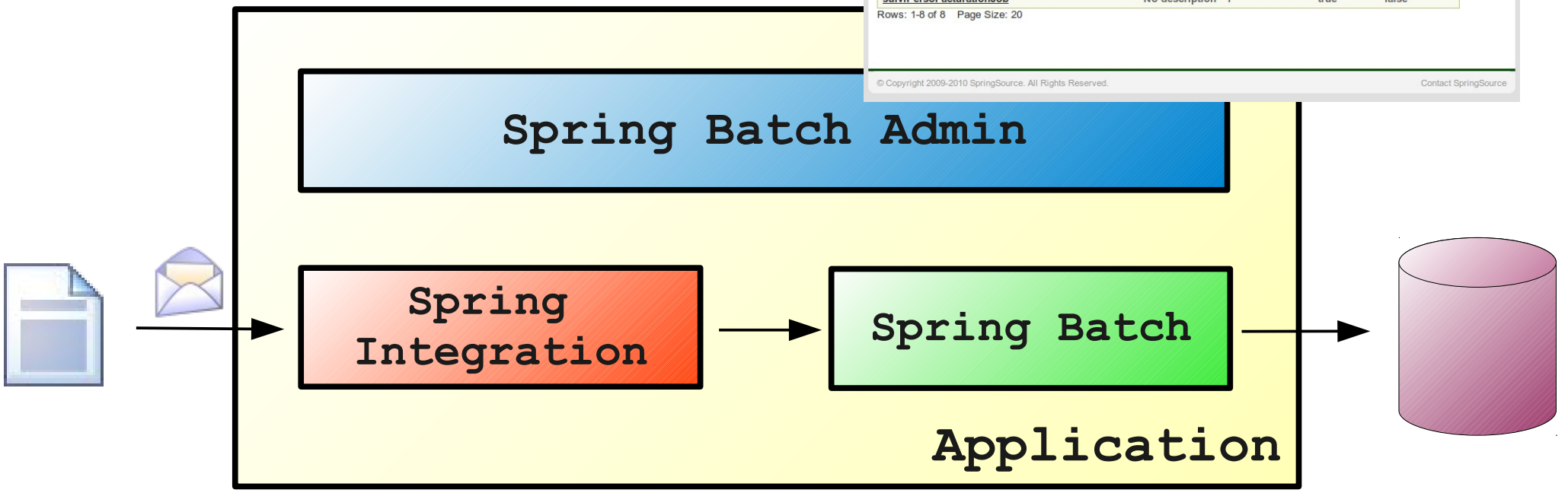
- Lightweight messaging
- Enterprise Integration Patterns
  - Routage, transformation, etc.
- Configuration XML, @ + DSLs Scala, Groovy

Spring Integration <=> bus



Spring Integration & Spring Batch



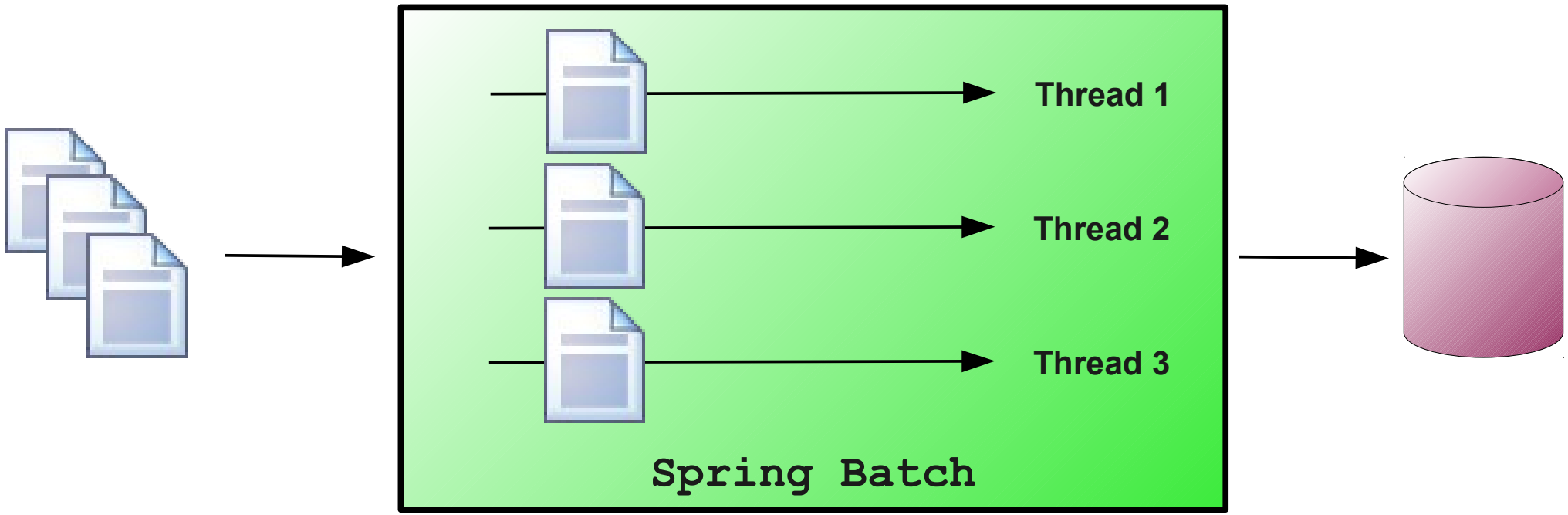


Spring Batch

- 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

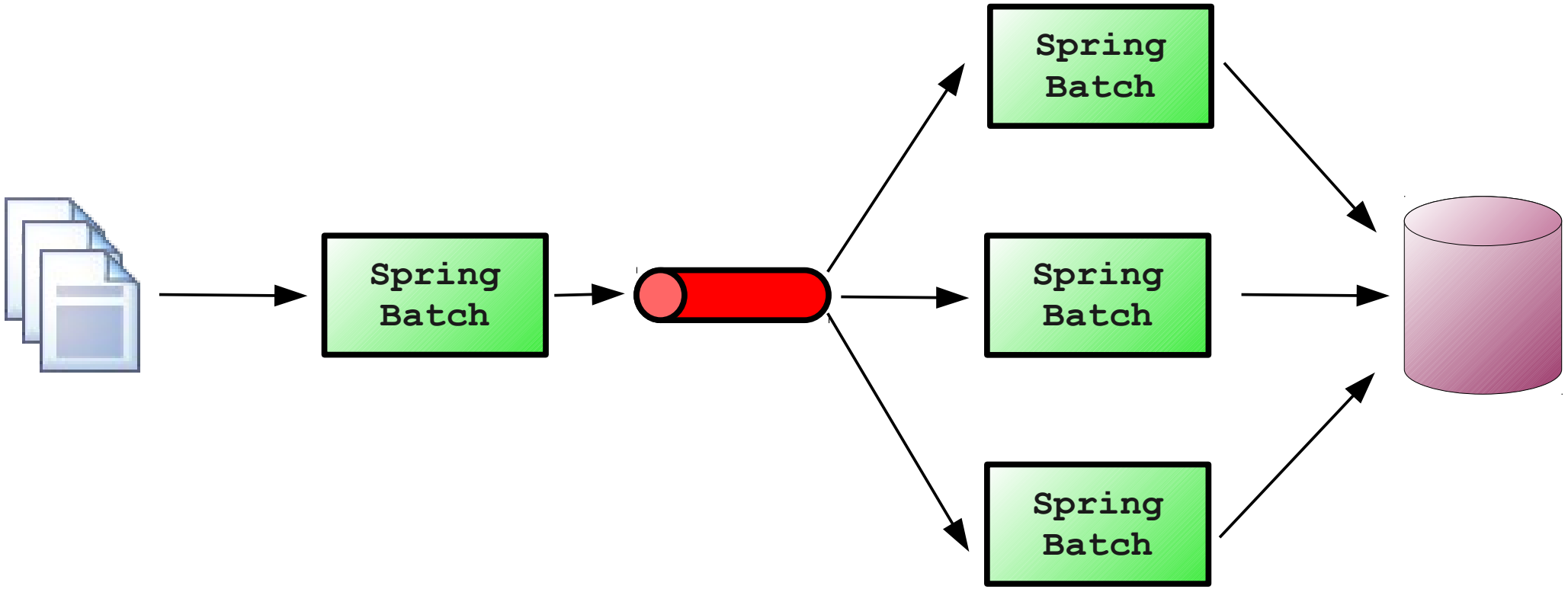
Spring Batch & scaling

- Local partitioning

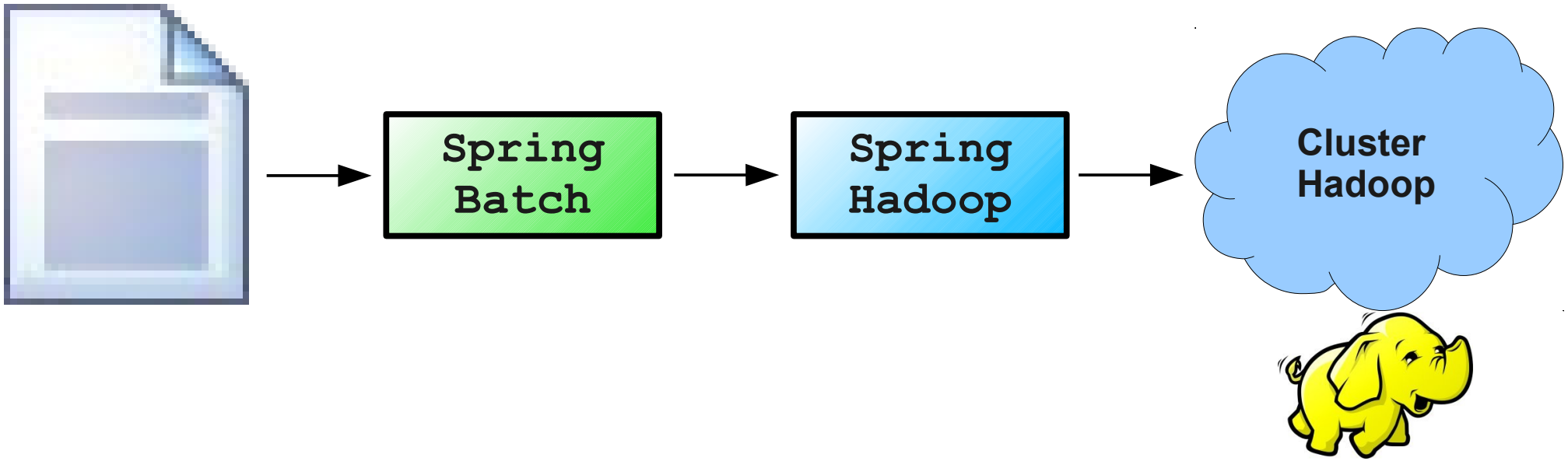


Spring Batch & scaling

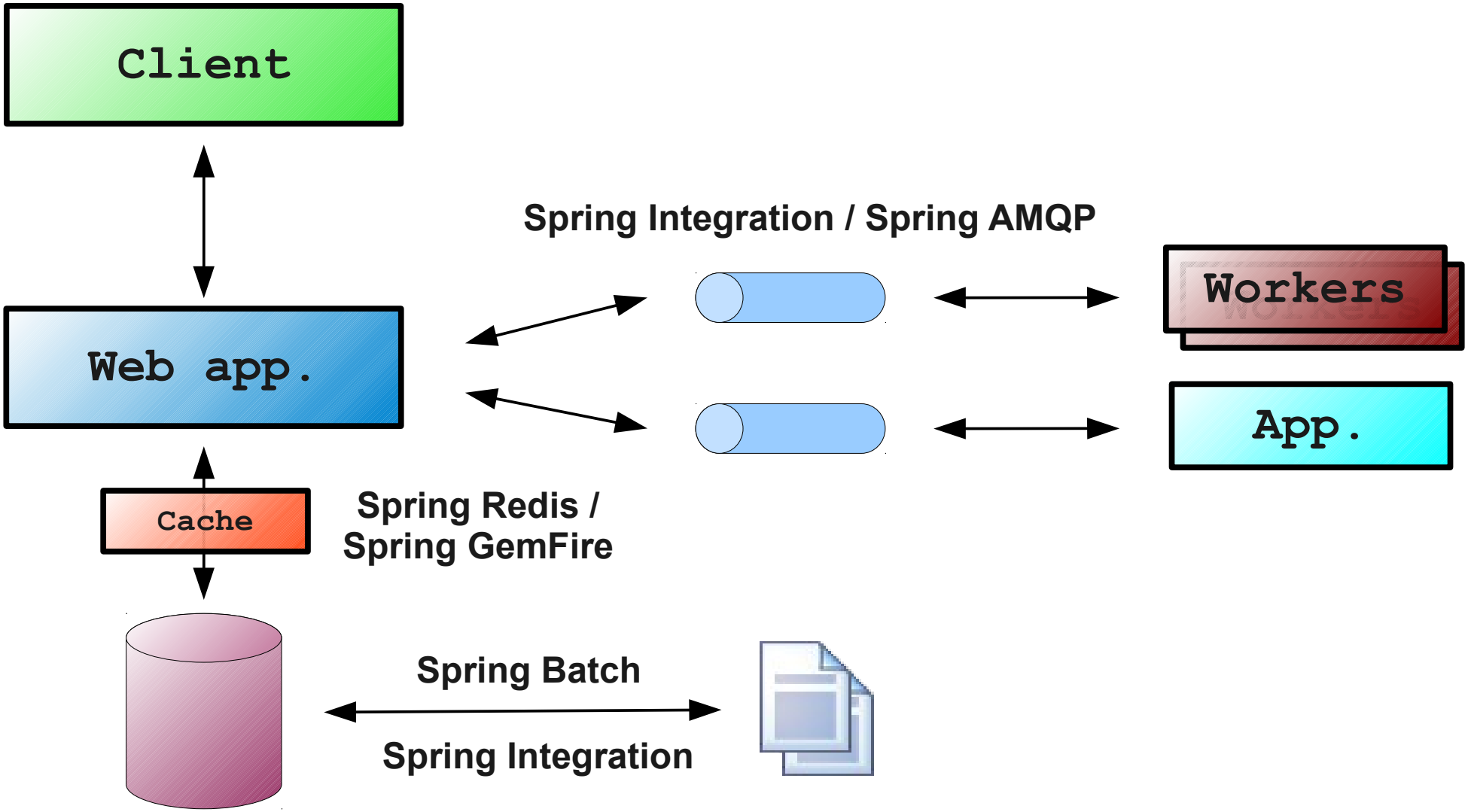
- Remote partitioning



Spring Batch & Spring Hadoop



Intégration



Intégration et déploiement

CLOUD FOUNDRY™

PostgreSQL

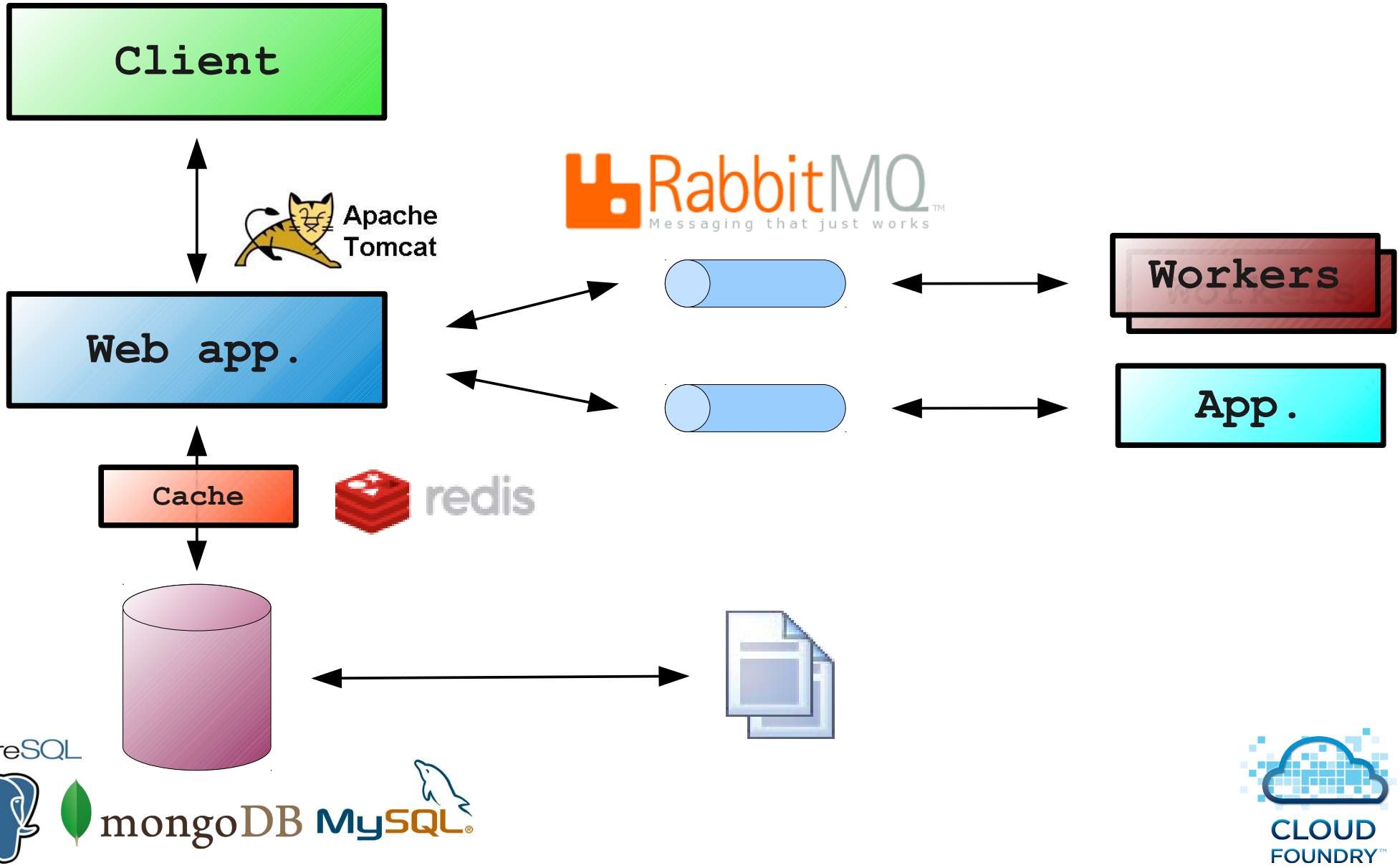
RabbitMQ™
Messaging that just works

mongoDB

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


Questions ?

Merci !