

Spring Framework 3.0



Arjen Poutsma & Alef Arendsen
SpringSource

Agenda

- Configuration options in 1.0, 2.0, 2.5 and 3.0
- Introduction of REST support in Spring MVC
- Introduction of expression language support
- Other features and considerations for Spring 3.0

Agenda

- **Configuration options in 1.0, 2.0, 2.5 and 3.0**
- Introduction of REST support in Spring MVC
- Introduction of expression language support
- Other features and considerations for Spring 3.0

Spring 1.0 <beans/> schema

```
<?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="...">

  <bean id="clinic"
    class="org.springframework.samples.petclinic.JdbcClinic"/>

</beans>
```

Spring 2.0 schema support

```
<?xml version="1.0" encoding="UTF-8"?>  
<beans xmlns="http://www.springframework.org/schema/beans"  
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
  xmlns:tx="http://www.springframework.org/schema/tx"  
  xsi:schemaLocation="...">  
  
  <tx:annotation-driven  
    base-package="org.springframework.samples.petclinic"/>  
  
</beans>
```

Spring 2.5 annotations

```
@Transactional @Repository
public class HibernateClinic implements Clinic {

    private SessionFactory sessionFactory;

    @Autowired public HibernateClinic(
        SessionFactory sessionFactory) {
        this.sessionFactory = sessionFactory;
    }
}
```

Component scanning

```
<?xml version="1.0" encoding="UTF-8"?>  
<beans xmlns="http://www.springframework.org/schema/beans"  
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
  xmlns:context="http://www.springframework.org/schema/context"  
  xsi:schemaLocation="...">  
  
  <context:component-scan  
    base-package="org.springframework.samples.petclinic"/>  
  
</beans>
```

@Controller for Spring MVC

```
@Controller
public class ClinicController {

    private final Clinic clinic;

    @Autowired public ClinicController(Clinic clinic) {
        this.clinic = clinic;
    }

    ...
}
```

@RequestMapping methods

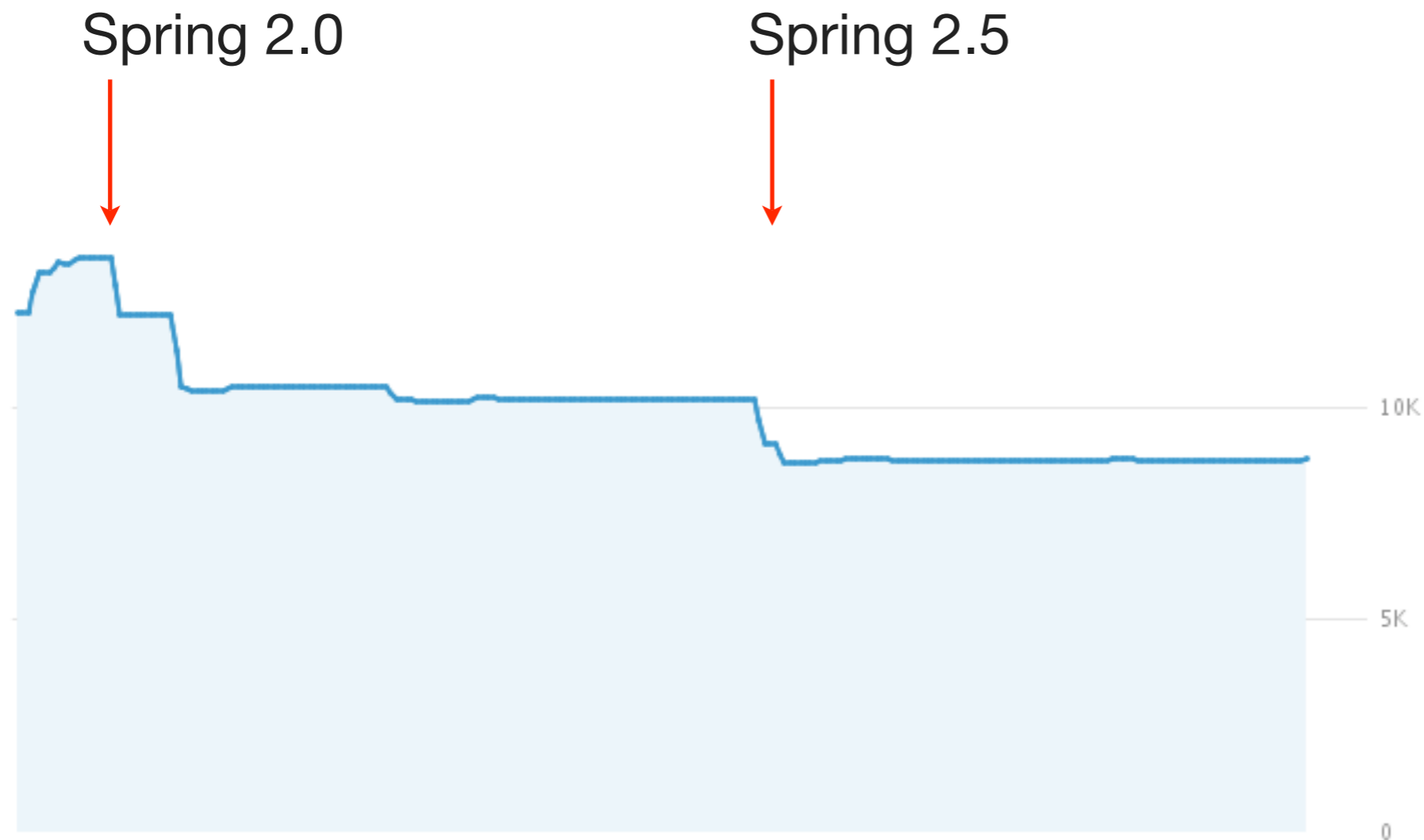
```
@Controller
public class ClinicController {

    ...

    @RequestMapping("/vets")
    public List<Vet> vets() {
        return clinic.getVets();
    }
}
```

Constantly simplifying

- LoC for sample application PetClinic over time



Spring as a basis for

- Other technologies building
 - Spring Batch 2.0 (e.g. with `@BatchComponent`)
 - Spring Integration 1.0 (e.g. with `@MessageEndpoint`)
 - Spring Web Services (e.g. with `@Endpoint`)
 - ...
- Both annotation-based options as well as XML

@MessageEndpoint

```
@MessageEndpoint
public class OrderSplitter {

    @Splitter(inputChannel="orders",
              outputChannel="drinks")
    public List<OrderItem> split(Order order) {
        return order.getItems();
    }
}
```

- Spring Integration 1.0 was released last week at SpringOne

Agenda

- Configuration options in 1.0, 2.0, 2.5 and 3.0
- **Introduction of REST support in Spring MVC**
- Introduction of expression language support
- Other features and considerations for Spring 3.0

Example URI templates

URI Template	Request	Variables
<code>/pets/{petId}</code>	<code>/hotels/tinkerbell</code>	<code>petId=tinkerbell</code>
<code>/pets/{petId}/visits</code>	<code>/pets/tinkerbell/visits</code>	<code>petId=tinkerbell</code>
<code>/pets/{petId}/visits/{date}</code>	<code>/pets/tinkerbell/visits/2008-09-10</code>	<code>petId=tinkerbell</code> <code>date=2008-09-10</code>

@PathVariable support

```
@RequestMapping("/pets/{petName}")  
public Pet pet(@PathVariable("petName") String petId) {  
    ...  
}
```

```
@RequestMapping("/pets/{petId}/visits/{date}")  
public Visit visit(  
    @PathVariable long petId,  
    @PathVariable Date date) {  
    ...  
}
```

Views

Mime Type	View	When
application/xml	MarshallingView	3.0 M2/SWS 1.5
application/atom+xml	AtomFeedView	3.0 M1
application/rss+xml	RssFeedView	3.0 M1
application/json	JsonView	Spring-JS

HTTP methods

- HTML only supports GET and POST
- What about PUT, DELETE and friends?
- Possible workarounds:
 - Javascript
 - POST with hidden method parameter
 - `HiddenHttpMethodFilter`

ShallowEtagHeaderFilter

- Introduced in Spring 3.0 M1
- Creates ETag header based on MD5 of rendered view
- Saves bandwidth only
- Deep ETag support comes in M2
 - Through @RequestHeader

DEMO

URI template support
with Spring

Atom support with
Spring



Agenda

- Recap of Spring 2.5 configuration options and @MVC
- Introduction of REST support in Spring MVC
- **Introduction of expression language support**
- Other features and considerations for Spring 3.0

Property externalization in 2.5

■ Traditionally using PropertyPlaceholderConfigurer

```
<bean id="dataSource"  
      class="org.apache.commons.dbcp.BasicDataSource"  
      destroy-method="close">  
  <property name="driverClassName" value="{driver}"/>  
  <property name="url" value="{url}"/>  
  <property name="username" value="{username}"/>  
  <property name="password" value="{password}"/>  
</bean>  
  
<context:property-placeholder  
  location="/WEB-INF/jdbc.properties"/>
```

Drawbacks of this approach

- It only supports properties
- The replacement is done at initialization time
 - Not at bean creation time
- It doesn't support conditionals or other constructs
- It's not very extensible for other frameworks

Introducing expressions

- Spring 3.0 will include full support for expressions
- The replacement is done at configuration time
 - Not at bean construction time

Expressions in <beans/>

```
<bean class="o.s.samples.petclinic.ClinicService">  
  <property name="timeout"  
    value="#{systemProperties.serviceTimeout}"/>  
</bean>
```

Expressions in <beans/>

```
<bean class="o.s.samples.petclinic.ClinicService"  
      scope="request">  
  <property name="timeout"  
    value="#{systemProperties.serviceTimeout}"/>  
</bean>
```

Expressions in <beans/>

```
<bean class="o.s.samples.petclinic.ClinicService">  
  <property name="cache"  
    value="#{otherBean.defaultCache}"/>  
</bean>
```

Other uses

```
@PreAuthorize(
    "hasRole('ADMIN') or (#pet.owner == currentUser)")
public void registerVisit(Pet pet, Visit v) {
    // ...
}
```

Agenda

- Recap of Spring 2.5 configuration options and @MVC
- Introduction of REST support in Spring MVC
- Introduction of expression language support
- **Other features and considerations for Spring 3.0**

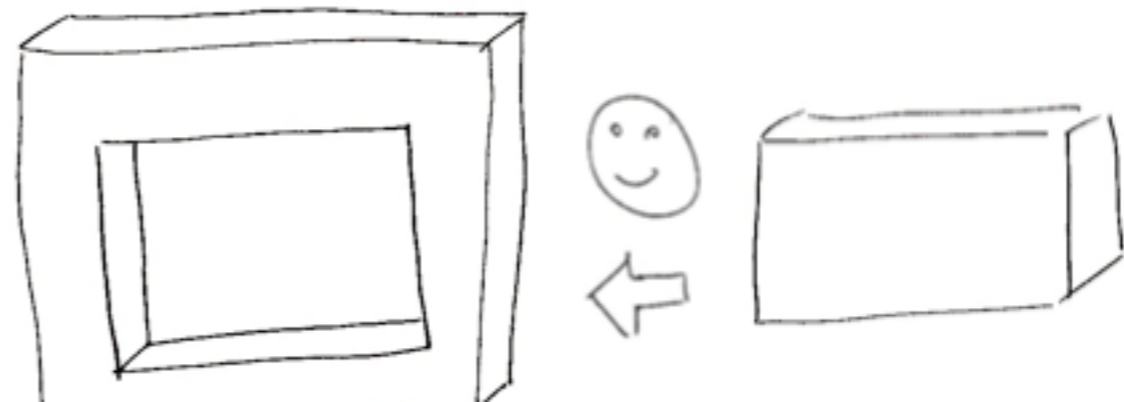
Other features

- Java 5+ foundation
 - Compatible with J2EE 1.4 and Java EE 5
- Support for Portlet 2.0

- Depending on specs finalizing
 - Java EE 6 support
 - Servlet 3.0, JSF 2.0, JAX-RS, JPA 2.0
 - Web Beans annotations

Backwards compatibility

- Spring 3.0 will deprecate / remove various things
 - Traditional Spring MVC controller hierarchy
 - Commons Attributes support
 - Traditional TopLink support
 - Traditional JUnit 3.8 class hierarchy
- 95% backwards compatible with regards to APIs
- 99% backwards compatible in the programming model



Roadmap (est.)

- Spring 3.0 M1 released last week
- Spring 3.0 Milestones January / February 2009
- Spring 3.0 Release Candidates March / April 2009

- More information on what features are included in which milestones: <http://jira.springframework.org>

Questions?

<http://jira.springframework.org>
<http://www.springframework.org>