Experts in delivering business-driven technology solutions.

Spring 3

Spring without XML

Perficient

Agenda

- Industry Forces
- What's New
  - Spring 2.0
  - Spring 2.5
- Spring Annotations
- Reducing Spring XML
- Spring Without XML!

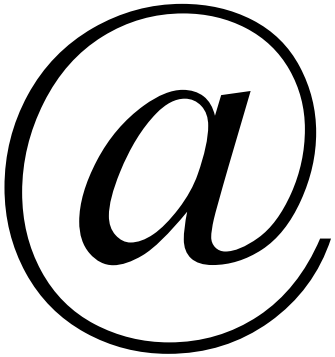
Industry Forces

- Annotations
  - EJB 3.X
  - JSR-250 Common Annotations
  - JSR-299 Web Beans
- Guice / SEAM

Spring Frustrations

- Autowiring
  - Qualifiers
- Long XML Files
- Required Injections

Tired of XML?



Industry Move to Annotations

Commons Annotation

@Resource

@PreDestroy

@PostConstruct

EJB Annotation

Commons

@Resource

@PreDestroy

@PostConstruct

EJB Annotation

@Home

Commons

@SecurityRoles

@Init

@EJB

@Resource

@PreDestroy

@PostConstruct

@MessageDriven

@MethodPermissions

@Stateful

@Interceptor

@Inject

@RunAs

@Stateless

@TransactionManagement

@TransactionAttribute

JPA Annotation

Commons

@Resource

@PreDestroy

@PostConstruct

EJB

@Home

@EJB

@MessageDriven

@Stateful

@Inject

@Stateless

@SecurityRoles

@RunAs

@MethodPermissions

@Init

@Interceptor

@TransactionManagement

@TransactionAttribute

JPA Annotation

@Column

@JoinColumn

@GeneratedValue

@Entity

Commons

@Resource

@PreDestroy

@PostConstruct

@Table

@Version

@Serialized

EJB

@Home

@RunAs

@OneToMany

@EJB

@MethodPermissions

@MessageDriven

@Init

@OneToOne

@Id

@ManyToMany

@Stateful

@Interceptor

@ManyToOne

@TransactionManagement

@Inject

@TransactionAttribute

@AssociationTable

@Stateless

@SecurityRoles

@DisciminatorColumn

@EmbeddedId

@Transient

JPA Annotation

Perficient

Commons

@Resource
@PreDestroy
@PostConstruct

EJB

@Home
@EJB
@MessageDriven
@Stateful
@Inject
@Stateless
@SecurityRoles

JPA

@Entity
@GeneratedValue
@JoinColumn
@Version
@Serialized
@RunAs
@Table
@Column
@OneToMany
@OneToOne
@Id
@ManyToOne
@ManyToMany
@DisciminatorColumn
@AssociationTable
@EmbeddedId
@Transient

@MethodPermissions
@Init
@Interceptor
@TransactionManagement
@TransactionAttribute

Web Services

@Resource

@PreDestroy

@PostConstruct

@Home

@MethodPermissions

@Init

@EJB

@MessageDriven

@TransactionManagement

@Stateful

@TransactionAttribute

@Inject

@Stateless

@SecurityRoles

@WebServiceRef

@WebServiceRefs

@ManyToMany

@Table

@DiscriminatorColu

@Column

@AssociationTable

@OneToMany

@EmbeddedId

@OneToOne

@Transient

@Entity

@ManyToOne

@GeneratedVa

@Id

@JoinColumn

@Version

@Serialized

@RunAs

Web Beans

@Resource

@PreDestroy

@PostConstruct

@WebServiceRefs

@Home

@MethodPermissions

@Init

@EJB

@MessageDriven

@TransactionManagement

@Stateful

@TransactionAttribute

@Inject

@Stateless

@SecurityRoles

@New

@Out

@In

@Model

@SessionScoped

@Current

@LoggedIn

@Produces

@Interceptor

@Secure

@Decorator

@Synchronous

@Asynchronous

@ManyToMany

@Table

@DiscriminatorColu

@WebServiceRef

@Column

@AssociationTable

@OneToMany

@EmbeddedId

@OneToOne

@Transient

@Entity

@ManyToOne

@GeneratedVa

@Id

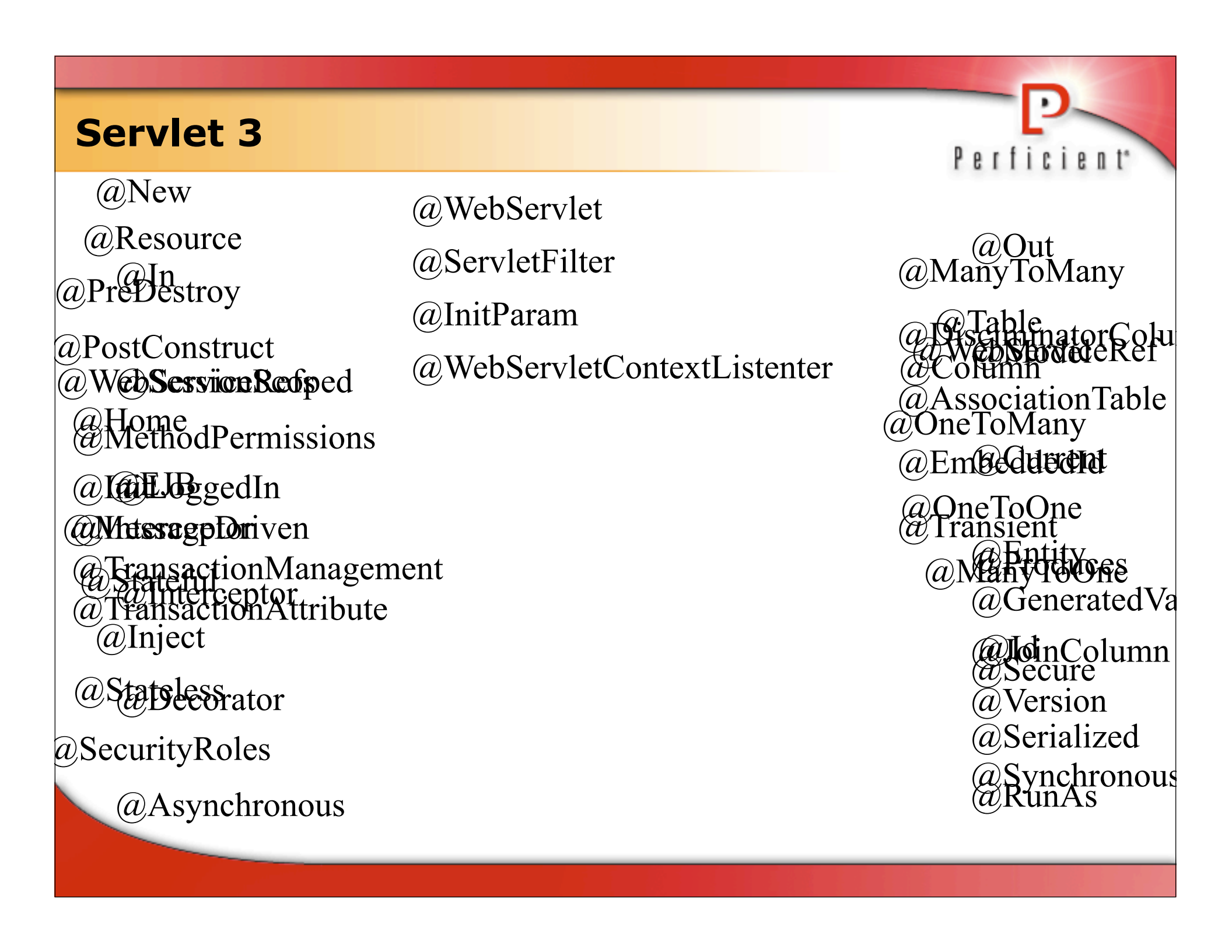
@JoinColumn

@Version

@Serialized

@RunAs

Servlet 3



@New
@Resource
@In
@PreDestroy
@PostConstruct
@WebServlet
@ServletFilter
@InitParam
@WebServletContextListenter
@Out
@ManyToMany
@Table
@Column
@AssociationTable
@OneToMany
@Current
@OneToOne
@Transient
@Entity
@ManyToOne
@GeneratedVa
@Id
@JoinColumn
@Secure
@Version
@Serialized
@Synchronous
@RunAs
@Asynchronous

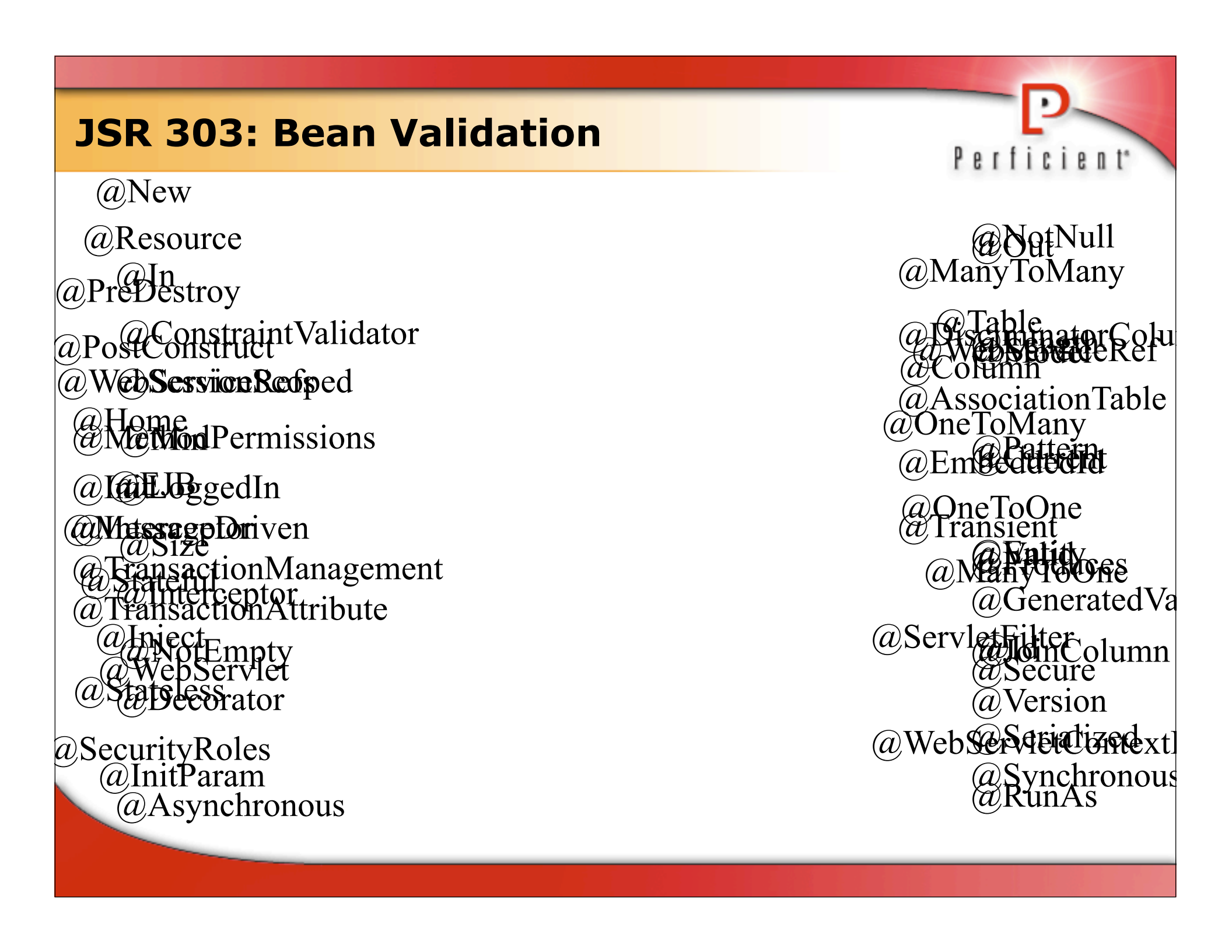
JSR 303: Bean Validation

@New
@Resource
@In
@PreDestroy
@PostConstruct
@WebServiceRef
@SessionScoped
@Home
@MethodPermissions
@EJB
@LoggedIn
@MessageDriven
@TransactionManagement
@Stateful
@Interceptor
@TransactionAttribute
@Inject
@WebServlet
@Stateless
@Decorator
@SecurityRoles
@InitParam
@Asynchronous

@NotNull
@ConstraintValidator
@Length
@Min
@Pattern
@Size
@Valid
@NotEmpty

@Out
@ManyToMany
@Table
@DiscriminatorColumn
@EmbeddedId
@Column
@AssociationTable
@OneToMany
@Current
@Embedded
@OneToOne
@Transient
@Entity
@ManyToOne
@GeneratedValue
@ServletFilter
@JoinColumn
@Secure
@Version
@Serialized
@WebServiceContext
@Synchronous
@RunAs

JSR 303: Bean Validation



JSR-299 Context and DI for Java

- @NonBinding
- @Named
- @Stereotype
- @Interceptor
- @InterceptorBindingType
- @Decorator
- @Decorates
- @ScopeType
- @ApplicationScoped
- @RequestScoped
- @SessionScoped
- @ConverationScoped
- @Dependent
- @BindingType
- @DeploymentType
- @Produces
- @Disposes
- @Specializes
- @Realizes
- @Initializer
- @New
- @Current
- @Production
- @Standard
- @Obtains
- @Initialized
- @Deployed
- @Observes
- @IfExists
- @Asynchronously
- @AfterTransactionCompletion
- @AfterTransactionFailure
- @AfterTransactionSuccess
- @BeforeTransactionCompletion
- @Fires
- @Model

Annotation Frustrations

- Not Mentioned
  - JMX 2.0
  - JAX-RS
  - JUnit 4 / TestNG
  - AOP frameworks
  - Spring

Spring 3.0 – What's New

Spring 2.0

- New Bean Scopes
  - httpSession
- Easier XML Configuration
  - <jee:jndi-lookup id="**dataSource**" jndi-name="jdbc/MyDataSource"/>
- Task Execution framework
- Portlet MVC
- Dynamic Language Support
  - Groovy
  - JRuby
- Message-driven POJOs
- AOP Enhancements
  - Includes @Aspect
- JMX Annotation Support

Spring 2.0 Benefits

- Easier XML
- Easier AOP
- Movement toward Annotations and Java 5 Support

Spring 2.5 – What's New

- More Reduction of XML
  - <context:* />
  - <jms:* />
- Significant Annotation Support
  - JSR 250 - @PostConstruct, @Resource…
  - JAX-WS 2.0's - @WebServiceRef
  - EJB 3.0 - @EJB
  - MVC annotations - @RequestParam, @RequestMapping…
  - Test Enhancements - Junit 4.4 and TestNG
  - Stereotypes - @Component, @Controller…
  - Spring enhancements - @Autowired,
  - AOP - @Configurable

Spring 2.5 What's New (Cont.)

- AOP Point Cut
  - Bean Point Cut
  - AspectJ load-time weaving **
- Auto-Detection of Components on Classpath
  - In combination with Annotations…
- OpenJPA 1.0 support with SavePoints
- Web Standards Updates
  - Tiles 2
  - JSF 1.2
- Autowiring of JRuby
- JMX Configuration Enhancements
  - <context:mbean-export />

Spring 2.5 Notes

Perficient

- Removed from Spring
  - JDK 1.3 Support
  - Hibernate 2.1 and 3.0 Support (Must be Hibernate 3.1+)
  - IBATIS SQL Maps 1.3
  - Apache OJB
  - JDO 1.0 Support
- Jar Changes
  - Spring-webmvc.jar and spring-webmvc-portlet.jar
    - MVC must use these jars for DispatcherServlet
      - No longer in spring.jar
- WebSphere and BEA Support
  - JTA Detection

Biggest Spring 2.5 Benefits

- Easier Configuration
  - Easier XML
  - Auto-detection
  - Annotations
  - Fine Grain Autowiring
- Amazing AOP Support

Spring 3.0 What's New

- Java 5
- Spring EL
- @MVC enhancements
- @REST
- @Portal 2.0 (JSR-286 support)
- Validation
  - JSR-303
- Early Support
  - JSF 2.0, JPA 2.0, JMX 2.0

Biggest Spring 3.0 Benefits

- Easier Configuration
  - Easier XML
  - Auto-detection
  - Annotations
  - Fine Grain Autowiring
- Amazing AOP Support

Spring Annotations

Spring Annotations

- Spring 2.x Data Access Annotations
- Spring 2.x Aspects
- Spring 2.5 Context Annotations
- Spring 2.5 Stereotypes
- Spring 2.5 Factory Annotations
- JSR-250 javax.annotations
- Spring 2.5 MVC Annotations
- Spring 3.0 MVC Additions
- Spring 3.0 Annotations

Spring 2.x Data Access Annotations

- @Transactional
  - Provides annotation driven demarcation for transactions
- @Repository
  - Indicates that a class functions as a repository or a data access object (DAO)
  - Exceptions are transparently translated
    - Springs DataAccessException Hierarchy

Spring 2.x Aspects

Perficient

```
@Aspect
public class TraceLogger {
  private static final Logger LOG = Logger.getLogger(TraceLogger.class);

  @Pointcut("execution(* com.cmentor.*Service(..))")
  public void serviceInvocation() {
  }

  @Before(" serviceInvocation()")
  public void log(JoinPoint joinPoint) {
        LOG.info("Before calling " + joinPoint.getSignature().getName()
                        + " with argument " + joinPoint.getArgs()[0]);
  }
}
```

- Requires:

```
<aop:aspectj-autoproxy />
```

Spring 2.5 New PointCut

- New 2.5 Pointcut Designator
  - Bean
  - For Spring only (not AspectJ)

@Pointcut("bean(nameofBean*)")

- Allows for stack selection

accountController -> accountService -> accountDAO

@Pointcut("bean(account*)") – matches vertical stack

@Pointcut("bean(*DAO)") – matches horizontal (all DAOs)

Spring 2.5 Context Annotations

- @Scope
  - Indicates the scope to use for annotated class instances
  - Default == “singleton”
  - Options:
    - Singleton
    - Prototype
  - Web Options:
    - Request
    - Session
    - Global session

Spring 2.5 Stereotypes

- @Component **
  - Indicates that a class is a component
  - Class is a candidate for auto-detection
  - Custom component extensions
- @Controller
  - Specialized Component
  - Typically used with RequestMapping annotation
  - Discussed in section on web mvc
- @Repository
  - 2.0 stereotype… previously mentioned
  - Now an extension of @Component
- @Service
  - Intended to be a business service facade

Spring 2.5 Factory Annotations

- @Autowired
  - Marks a constructor, field, setter or config method for injection.
  - Fields are injected
    - After construction
    - Before config methods
  - @Autowired(required=false)
  - Config:
    - AutowiredAnnotationBeanPostProcessor
- @Configurable
  - Marks class as being eligible for Spring-driven configuration
  - Used with AspectJ
- @Qualifier
  - Qualifies a bean for autowiring
  - May be customized
- @Required
  - Marks a method as being injection required

Types of Injections

- Constructor

```
public class AccountService {

    private AccountDAO dao;

     @Autowired
    public AccountService(AccountDAO dao) {
        this.dao = dao;
    }
```

- Setter

```
public class AccountService {

    private AccountDAO dao;

     @Autowired
    public void setDao(AccountDAO dao) {
        this.dao = dao;
    }
```

- Field

```
public class AccountService {

    @Autowired
    private AccountDAO dao;
```

New Injection Type

- configuration method

```
public class AccountService {

    private AccountDAO dao;

     @Autowired
    public void init(AccountDAO dao) {
        this.dao = dao;
    }
```

- with any number of arguments

```
public class AccountService {

    private AccountDAO dao;

     @Autowired
    public void init( AccountDAO dao, LogLevel level) {
        this.dao = dao;
        level.getLabel();
    }
```

@Autowired Notes

- Spring resources can be injected
- No Loaders or *LoaderAware

```
@Autowired
Private ApplicationContext appContext;
```

Configuring Autowired

<bean class="…AutowiredAnnotationBeanPostProcessor"/>

- Or…

<context:annotation-config/>

Let me Qualify that


```
@Autowired
Private DataSource dataSource
```

Let me Qualify that


```
@Autowired
@Qualifier("myDSName")
Private DataSource dataSource
```

Let me Qualify that


```
@Autowired
@Qualifier("myDSName")
Private DataSource dataSource
```

- Or

```
@Autowired
public void init(@Qualifier("srcName") DataSource
                 dataSource)
{…}
```

Spring 2.5 JSR-250 Support

- JSR-250 javax.annotations
  - Requires
    - CommonAnnotationBeanPostProcessor bean or
    - <context:annotation-config />
  - @Resource
    - Injects a named resource
      - Spring name not JNDI
        - » Unless configured :) to use JNDI
  - @PostConstruct
    - Method invocated after construction
    - No XML
    - Multiple methods possible
  - @PreDestroy
    - Method invocated when application context hosting object is closed

@Resource Options


```
@Resource(name="dataSourceName")
Public void setDataSource(DataSource source) {

@Resource
private DataSource dataSource; // name = dataSource
```

@Resource Options


```
<bean
 class="org.springframework.context.annotations.Common
 AttributeBeanPostProcessor">
  <property name="alwaysUseJndiLookup" value="true" />
```

Or

```
<jee:jndi-lookup id="dataSource"
        jdni-name="java:comp/env/jdbc/peopleDS" />
```

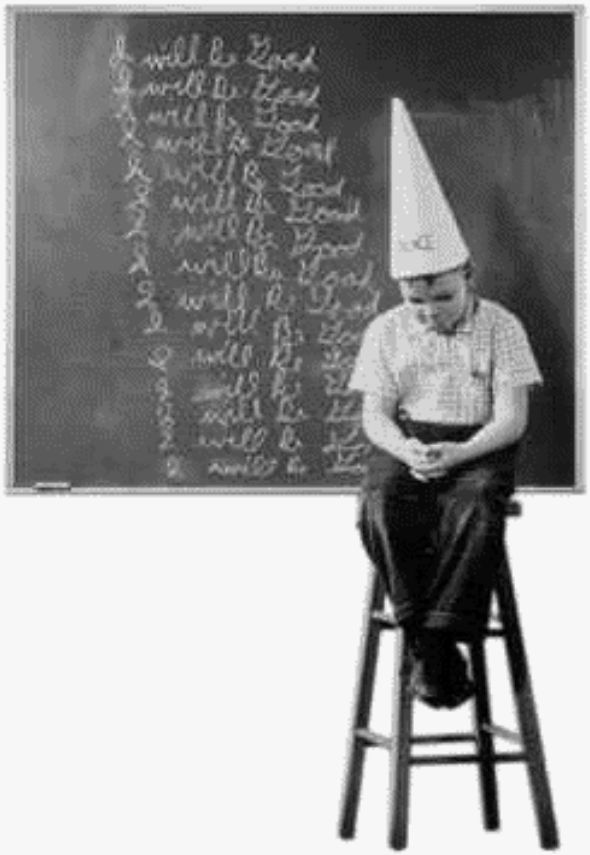
@Resource Notes

- No Spring managed resource for the default name
  - Injects object of type if there is only one

- Can be Turned Off

```
<bean class="…CommonAnnotationBeanPostProcessor">
        <property name="fallbackToDefaultTypeMatch"
                value="false" />
</bean>
```

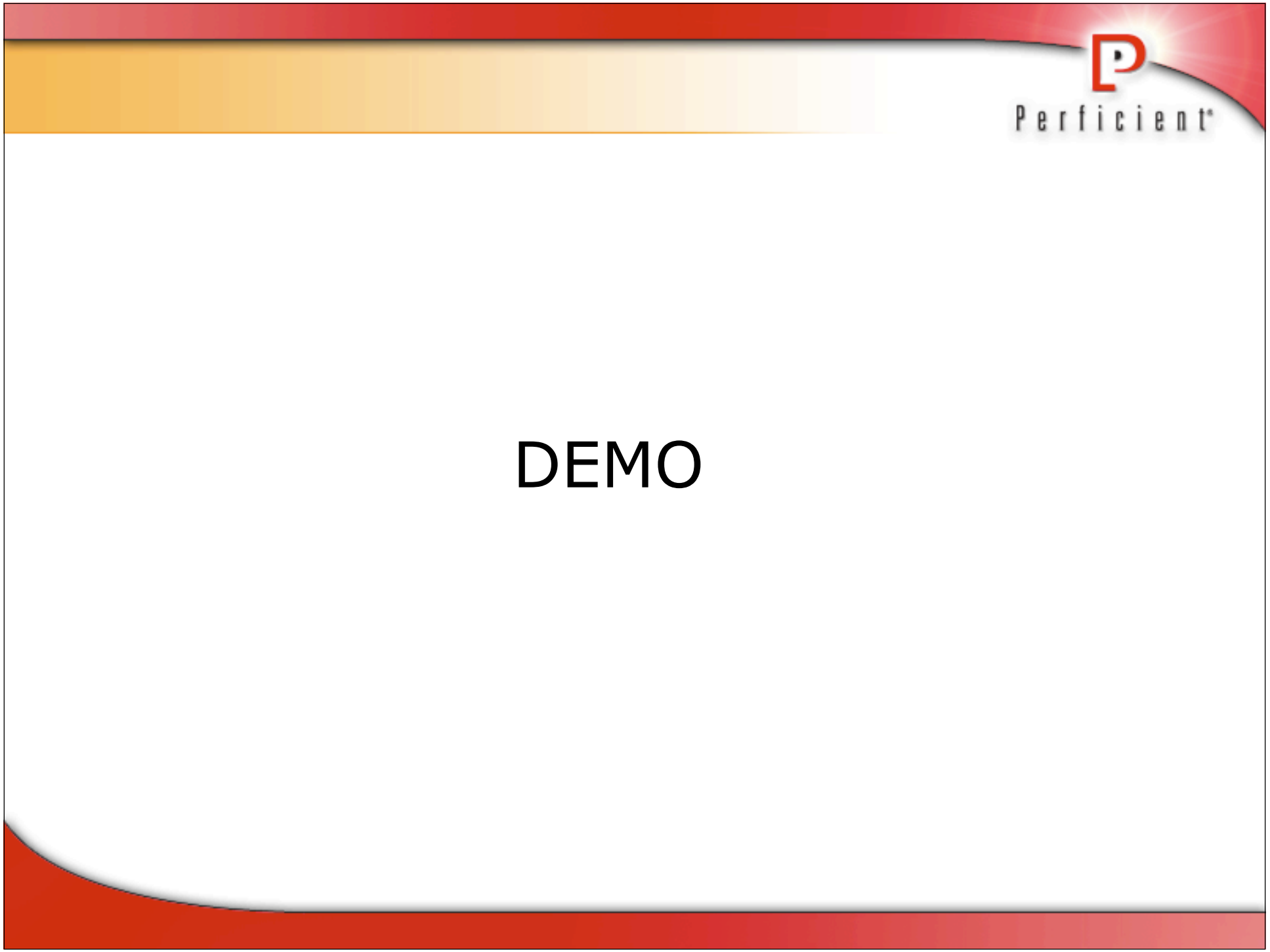
The Spring Debate



@Resource
private DataSource dataSource;

- Or

@Autowired
@Qualifier("dataSourceName")
Private DataSource dataSource


Perficient
DEMO

@MVC

Spring 2.5 MVC Annotations

- @Controller
  - Stereotype used to "Controller" of MVC
  - Scanned for RequestMappings
- @RequestMapping
  - Annotates a handler method for a request
  - Very flexible
- @RequestParam
  - Annotates that a method parameter should be bound to a web request parameter
- @SessionAttributes
  - Marks session attributes that a handler uses

New Controller Issues

- Doesn’t implement an Interface
- Multiple request mappings
- High degree of flexibility

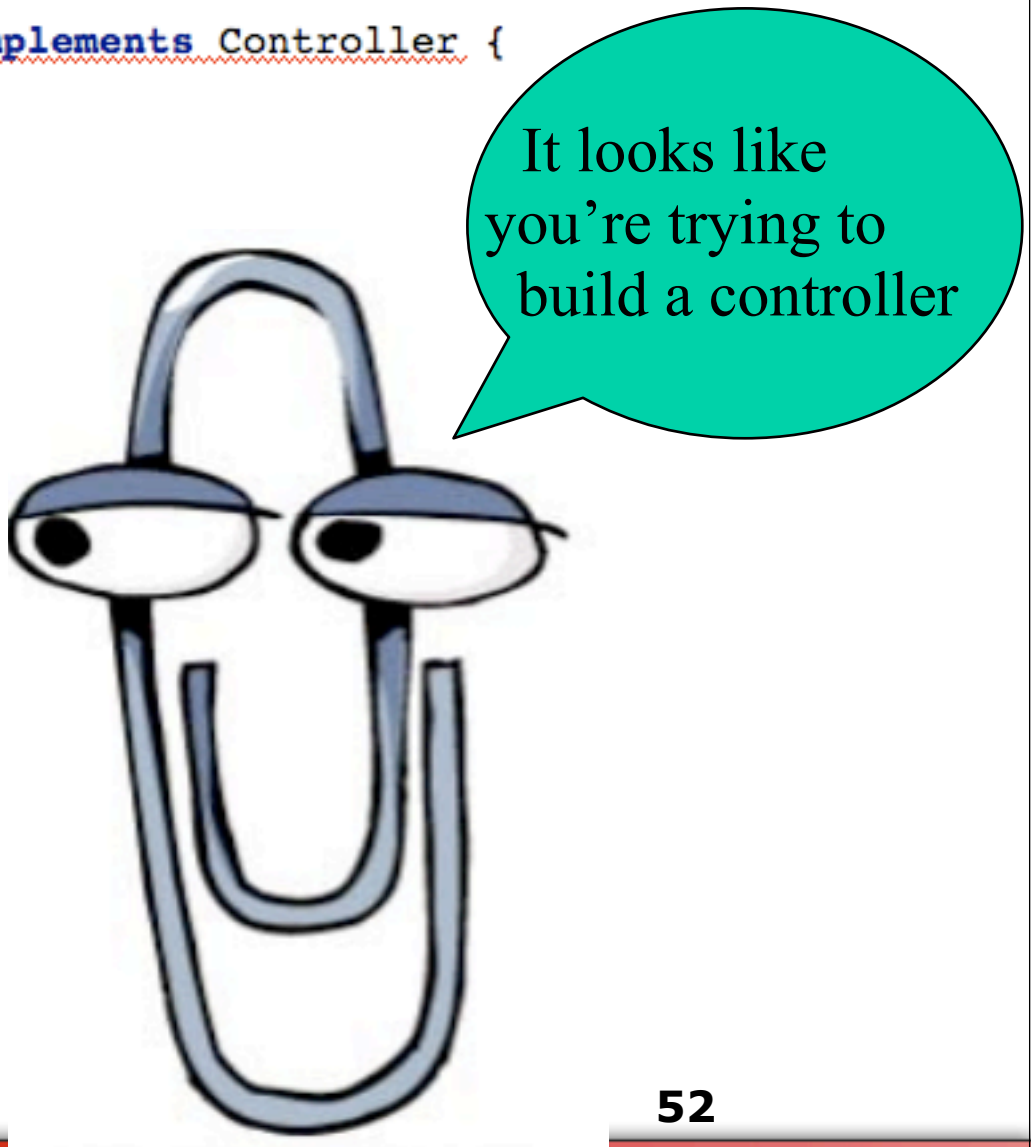
Advantages of Controller Interfaces


```
public class HelloWorldController implements Controller {

}
```

Advantages of Controller Interfaces

Perficient

```
public class HelloWorldController implements Controller {

}
```



Advantages of Controller Interfaces


```
public class HelloWorldController implements Controller {


    public ModelAndView handleRequest(HttpServletRequest httpServletRequest,
                                      HttpServletResponse httpServletResponse)
        return null;  //todo: implement me
    }
}
```

A World Without Rules

Perficient

- Return Type?
- Parameters?

@RequestMapping - Extreme Flexibility

- Parameters can be
  - Request / response / session
  - WebRequest
  - InputStream
  - OutputStream
  - @RequestParam
  - +++
- Return types
  - ModelAndView Object
  - Model Object
  - Map for exposing model
  - View Object
  - String which is a view name
  - Void… if method wrote the response content directly

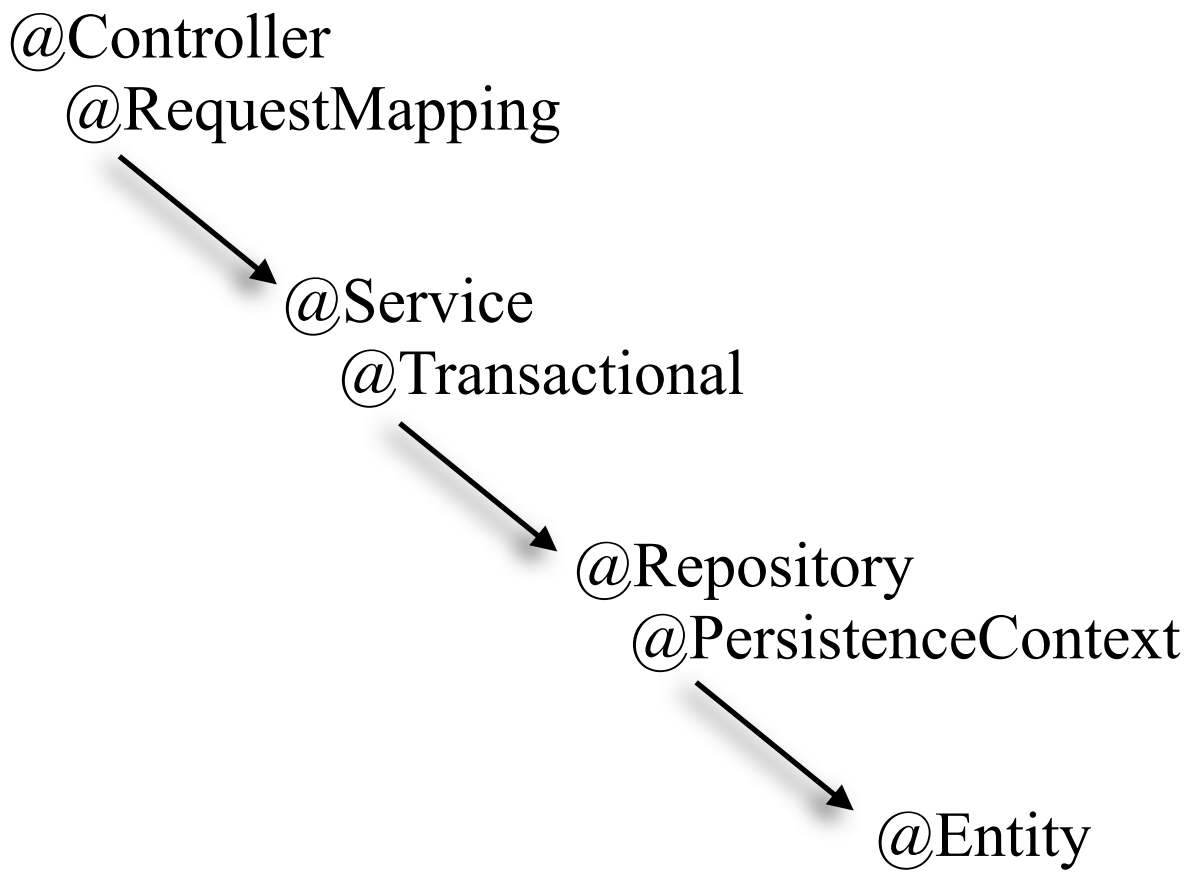
Spring 2.5 Controller Example

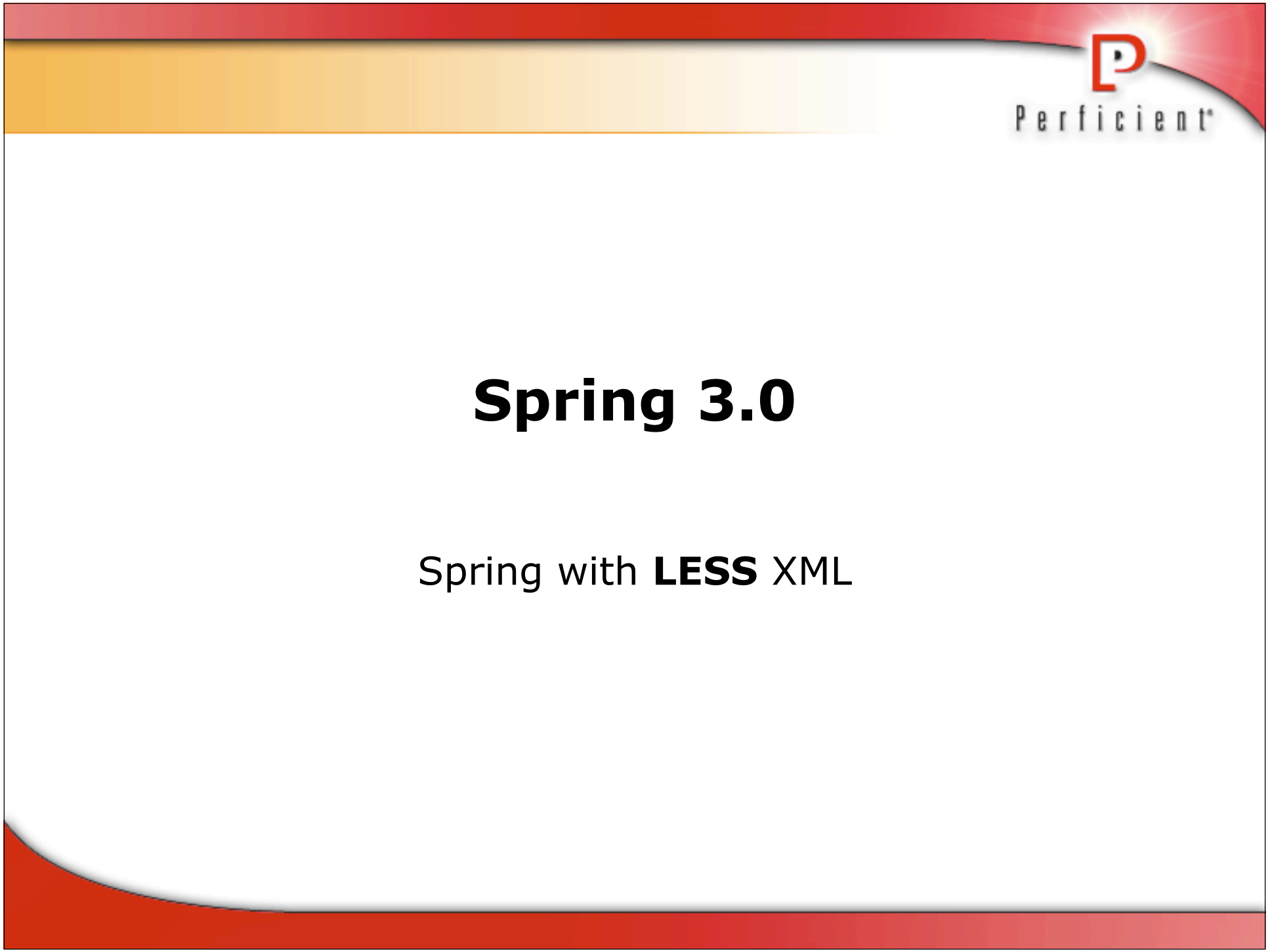

```
@Controller
public class ConfController {

    @Autowired
    private confDB confDB;

    @RequestMapping("/sessionList")
    public String showSessionList(ModelMap model)    {
        model.addAttribute("sessions", this.confDB.getSessions());
        return "sessionList";
    }

    @RequestMapping("speakerImage")
    public void streamSpeakerImage(@RequestParam("name") String name,
                                   OutputStream outputStream) throws IOException   {
        this.confDB.getSpeakerImage(name,outputStream);
    }

    @RequestMapping("/clearDatabase")
    public String clearDB() {
        this.confDB.clear();
        return "redirect:sessionList";
    }
}
```

Spring 3 Annotations (New Only)

- @Value
- @RequestHeader
- @CookieValue
- @PathVariable
- @Async
- @Scheduled
- @Model
- @Bound
- @ActionMapping
- @EventMapping
- @RenderMapping
- @ResourceMapping
- @ExceptionHandler
- @Mapping
- @RequestBody
- @ResponseBody
- @ResponseStatus

Annotation Call Stack





Spring 3.0

Spring with **LESS** XML

Spring with LESS XML

Perficient

- Annotations
  - Aspects
  - Component
  - Autowiring
- Name spaces XML
- Auto Discovery

Less XML – Namespace options

- Namespace options XML
- Move some environment properties out of XML
  - <context:property-placeholder
    location="classpath:jdbc.properties />

Scanning For Discovery

- XML Configure Scan
  - Disable default filters
    - No auto-detect of @Component, @Repository, etc.
  - Exclude or Include filters
  - Default Behavior
    - AutowiredAnnotationBeanPostProcessor
    - CommonAnnotationBeanPostProcessor

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

Filter your Scan

- Filter Types
  - Annotation
  - Assignable
  - Regex
  - Aspectj
- Example:

```
<context:component-scan base-package="org.example">
        <context:include-filter type="regex"

          expression=".*Stub.*Repository"/>
        <context:exclude-filter type="annotation"

          expression="org.springframework.stereotype.Repository"/>
</context:component-scan>
```

Spring 3.0

Spring with **NO** XML

Spring without XML

- Java Configuration
- Groovy
- Java Configuration II
  - Auto Discovery

Spring Configuration Points

- Spring at the core:
  - Java metadata
- Separate from the metadata parsing

Java Configuration

- http://www.springframework.org/javaconfig

```
@Configuration
public class NoXMLConfig {

    @Bean
    public GreetingService greetingService()    {
        GreetingServiceImpl service =  new GreetingServiceImpl();
        service.setMessageRepository(messageRepository());
        return service;
    }

    @Bean
    public MessageRepository messageRepository() {
        StubMessageRepository repository = new StubMessageRepository();
        repository.initialize();
        return repository;
    }
}
```

Java Auto-Discovery


```
GenericApplicationContext context =
        new GenericApplicationContext();

ClassPathBeanDefinitionScanner scanner =
            new ClassPathBeanDefinitionScanner(context);

scanner.scan("com.cmentor");

context.refresh();
```

Java and XML


```
GenericApplicationContext context =
   new GenericApplicationContext();
ClassPathBeanDefinitionScanner scanner =
        new ClassPathBeanDefinitionScanner(context);

scanner.scan("com.cmentor");

BeanDefinitionReader reader =
        new XmlBeanDefinitionReader(context);
reader.loadBeanDefinitions("classpath:dataSource.xml");

context.refresh();
```

XML and Java


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

Suggestions

Where to Use XML

- 3rd Party Classes
- Environment Centric Configurations

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

    <bean id="urlMapping" class="org.springframework.web.servlet.mvc.support.ControllerClassNameHa

    <bean class="org.springframework.web.servlet.view.InternalResourceViewResolver"
          p:prefix="/WEB-INF/jsp/"
          p:suffix=".jsp"/>

    <bean id="entityManagerFactory" class="org.springframework.orm.jpa.LocalEntityManagerFactoryBe

    <bean class="org.springframework.orm.jpa.support.PersistenceAnnotationBeanPostProcessor"  />

    <bean id="transactionManager" class="org.springframework.orm.jpa.JpaTransactionManager">
        <property name="entityManagerFactory" ref="entityManagerFactory" />
    </bean>

    <tx:annotation-driven />

</beans>
```

Ask Yourself

Ask Yourself

Perficient

Should an Admin be able to change this?

Where NOT to use XML

Where NOT to use XML

Perficient

Every where else!

Summary

- Spring 3.0 Annotations
- Spring MVC
- Spring 2.x Reduces XML
- Spring 3.0 Provides Auto-discovery
- Reality Check:
  - Most configurations will have XML and will provide a combination of annotations and XML

Questions

kensipe@gmail.com

twitter: @kensipe

blog: kensipe.blogspot.com