What's New in Spring 3.1

Matt Raible

matt@raibledesigns.com
@mraible

Photos by Trish McGinity
http://mcginityphoto.com

Introductions

- Your experience with Spring? Spring MVC?
- Your experience with Java EE 6?
- What do you hope to learn today?
- Open Source experience: Guice, Maven, Eclipse, Hibernate, Tomcat, JBoss?
- Favorite IDE? Favorite OS?

© 2012 Raible Designs

Who is **Matt Raible**?

Father, Skier, Cyclist

Web Framework Connoisseur

Founder of AppFuse

Blogger on raibledesigns.com

Clients and Open Source

LinkedIn

Evite

Time Warner Cable

overstock.com
shortcut: O.co

Taleo
Talent Drives Performance

Struts 2

displaytag

AppFuse

Roller

© 2012 Raible Designs

Agenda

- Spring History
- Spring 2.5 and 3.0
- What's New in 3.1
- Summary
- Q and A
- Beer

© 2012 Raible Designs

The Spring Framework

- Simplify J2EE
- Manage Dependencies
- Inversion of Control
- À la carte framework

© 2012 Raible Designs

Spring Mission Statement

J2EE should be easier to use.

It's best to program to interfaces, rather than classes. Spring reduces the complexity of using interfaces to zero.

JavaBeans offer a great way of configuring applications.

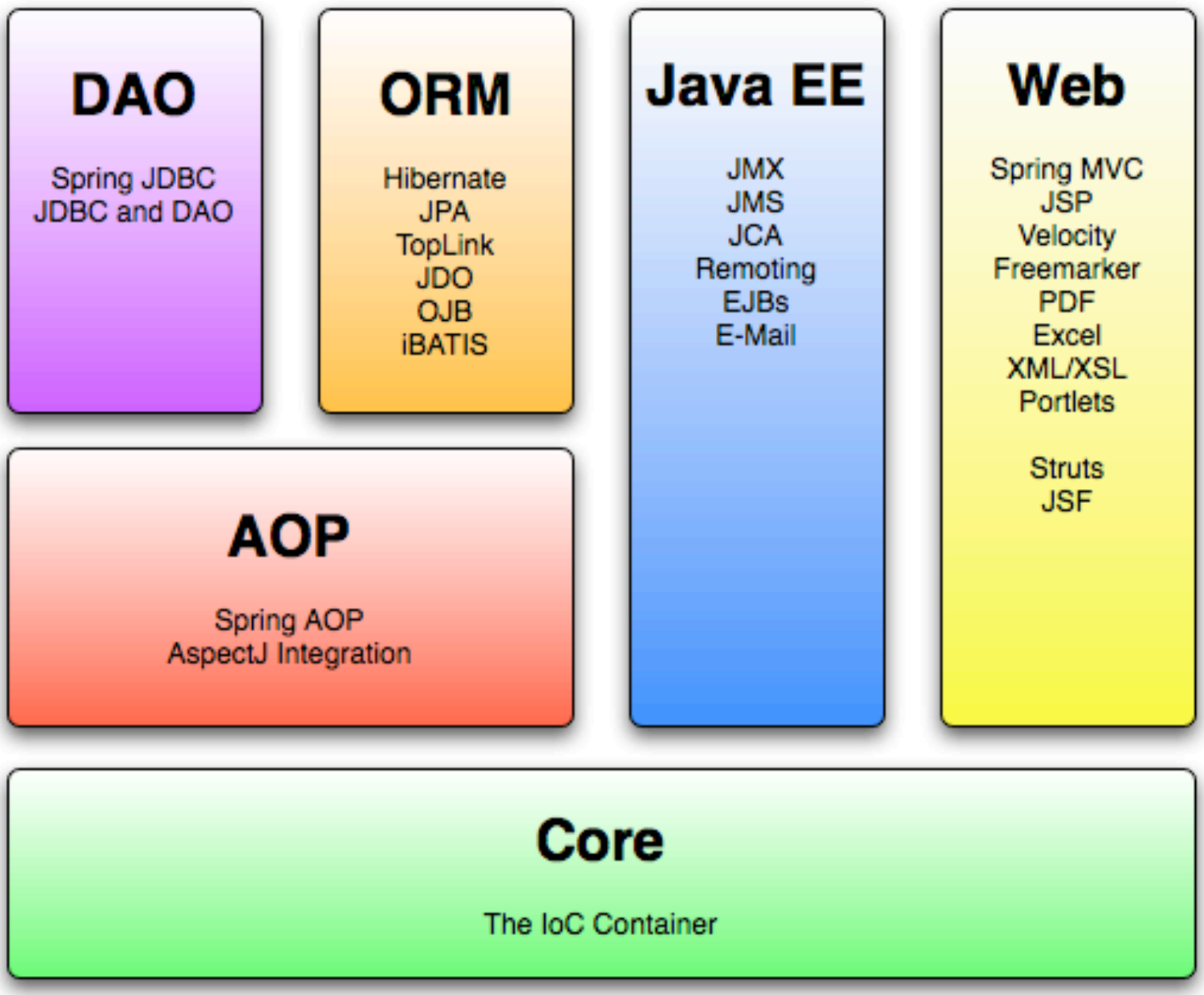
Checked exceptions are overused in Java. A framework shouldn't force you to catch exceptions you're unlikely to recover from.

OO Design is more important than any implementation technology, such as J2EE.

Testability is essential, and a framework such as Spring should help make your code easier to test.

© 2012 Raible Designs

Spring Modules



© 2012 Raible Designs

Spring Portfolio - Oct. 2009

SPRING.net
Application Framework

Acegi
SECURITY
SYSTEM
FOR SPRING

Spring
IDE

Spring 2.5

Spring Web Flow

© 2012 Raible Designs

Spring Portfolio - Today

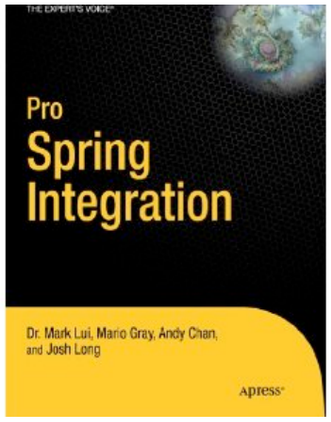
springsource TOOLSUITE

spring

CLOUD FOUNDRY™

springroo

GRAILS



eclipseRT Gemini Blueprint

Spring Web Flow

Other Spring Technologies

Spring Web Services

Spring Android

Spring Dynamic Modules

Spring Web Flow

Spring Social

Spring GemFire

Spring LDAP

Spring Security

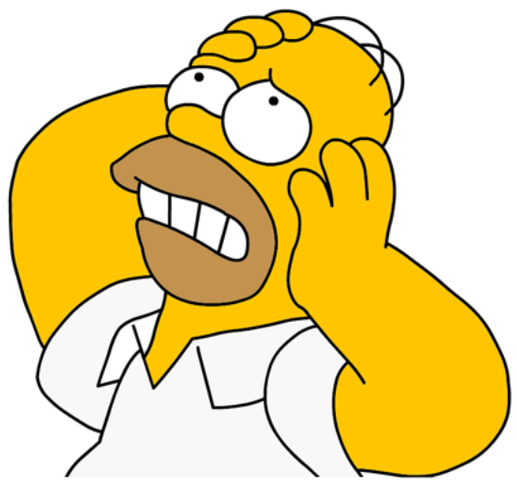
Spring GemFire for .NET

Spring Integration

Spring Batch

Spring IDE

Spring BlazeDS Integration



Spring.NET

Spring AMQP

Spring Roo

Spring Data

Spring AMQP.NET

Spring Python

SpringSource dm Server

http://www.springsource.org/download/community

© 2012 Raible Designs

Release Timeline

- March 24, 2004 - Spring 1.0
- October 5, 2006 - Spring 2.0
- November 19, 2007 - Spring 2.5
- December 16, 2009 - Spring 3.0
- December 13, 2011 - Spring 3.1

© 2012 Raible Designs

New in Spring 2.5

- Extended **Platform** Support
  - Java SE 6, Java EE 5 and OSGi
- Enhanced **AspectJ** support
- Comprehensive support for **Annotations**
  - Bean lifecycle
  - Autowiring
  - Spring MVC enhancements

© 2012 Raible Designs

Annotation Examples

```
@Repository(value = "userDao")
public class UserDaoHibernate implements UserDao {
    HibernateTemplate hibernateTemplate;

    @Autowired
    public UserDaoHibernate(SessionFactory sessionFactory) {
        this.hibernateTemplate = new HibernateTemplate(sessionFactory);
    }

    public List getUsers() {
        return hibernateTemplate.find("from User");
    }

    @Transactional
    public void saveUser(User user) {
        hibernateTemplate.saveOrUpdate(user);
    }
}
```

```
@Service(value = "userManager")
public class UserManagerImpl implements UserManager {
    @Autowired UserDao dao;

    public List getUsers() {
        return dao.getUsers();
    }
}
```

© 2012 Raible Designs

Annotation Examples

```
@Repository(value = "userDao")
public class UserDaoHibernate implements UserDao {
    HibernateTemplate hibernateTemplate;

    @Autowired
    public UserDaoHibernate(SessionFactory sessionFactory) {
        this.hibernateTemplate = new HibernateTemplate(sessionFactory);
    }

    public List getUsers() {
        return hibernateTemplate.find("from User");
    }

    @Transactional
    public void saveUser(User user) {
        hibernateTemplate.saveOrUpdate(user);
    }
}
```

```
@Service(value = "userManager")
public class UserManagerImpl implements UserManager {
    @Autowired UserDao dao;

    public List getUsers() {
        return dao.getUsers();
    }
}
```

© 2012 Raible Designs

XML Configuration

Annotation Scanning

```
<!-- Replaces ${...} placeholders with values from a properties file -->
<!-- (in this case, JDBC-related settings for the dataSource definition below) -->
<context:property-placeholder location="classpath:jdbc.properties"/>

<!-- Enable @Transactional support -->
<tx:annotation-driven/>

<!-- Enable @AspectJ support -->
<aop:aspectj-autoproxy/>

<!-- Scans for @Repository, @Service and @Component -->
<context:component-scan base-package="org.appfuse"/>
```

© 2012 Raible Designs

XML Configuration

Declarative Transactions

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

<tx:advice id="txAdvice">
    <tx:attributes>
        <tx:method name="get*" read-only="true"/>
        <tx:method name="*"/>
    </tx:attributes>
</tx:advice>
```

© 2012 Raible Designs

Testing and JUnit 4

```
package org.appfuse.service;

import org.appfuse.model.User;
import static org.junit.Assert.*;
import org.junit.Test;
import org.junit.runner.RunWith;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.test.context.ContextConfiguration;
import org.springframework.test.context.junit4.SpringJUnit4ClassRunner;
import org.springframework.transaction.annotation.Transactional;

@RunWith(SpringJUnit4ClassRunner.class)
@ContextConfiguration
public class UserManagerTest {
    @Autowired
    UserManager userManager;

    @Test
    @Transactional
    public void testGetUsers() {

    }
}
```

© 2012 Raible Designs

Spring MVC Improvements

```
@Controller
public class UserController {
    @Autowired
    UserManager userManager;

    @RequestMapping("/users")
    public String execute(ModelMap model) {
        model.addAttribute(userManager.getUsers());
        return "userList";
    }
}
```

```
<!-- Activates mapping of @Controller -->
<context:component-scan base-package="org.appfuse.web"/>

<!-- Activates @Autowired for Controllers -->
<context:annotation-config/>

<!-- View Resolver for JSPs -->
<bean id="viewResolver"
    class="org.springframework.web.servlet.view.InternalResourceViewResolver">
    <property name="viewClass"
              value="org.springframework.web.servlet.view.JstlView"/>
    <property name="prefix" value="/"/>
    <property name="suffix" value=".jsp"/>
</bean>
```

© 2012 Raible Designs

ModelMap Naming

- Automatically generates a key name
  - o.a.m.User ➟ ‘user’
  - java.util.HashMap ➟ ‘hashMap’
  - o.a.m.User[] ➟ ‘userList’
  - ArrayList, HashSet ➟ ‘userList’

© 2012 Raible Designs

@RequestMapping

- Methods with @RequestMapping are allowed very flexible signatures
- Arguments supported:
  - ServletRequest and ServletResponse
  - HttpSession
  - Locale
  - ModelMap

© 2012 Raible Designs

New in Spring 3.0

- Java 5+
- Spring Expression Language
- New Spring MVC Features
  - REST
  - Ajax
  - Declarative Validation
- Backwards compatible with Spring 2.5

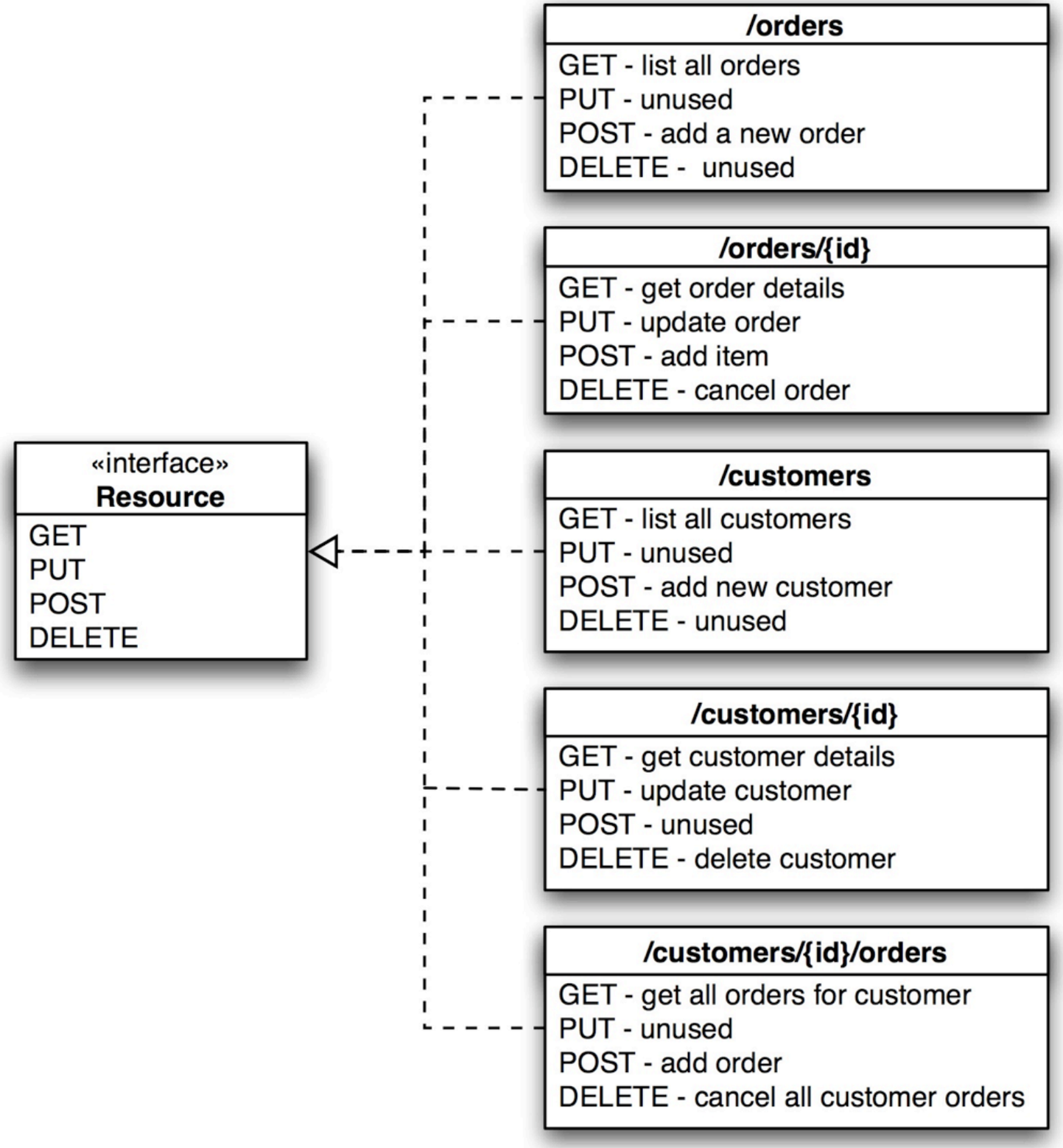
© 2012 Raible Designs

Spring Expression Language

- Unified EL++
  - Deferred evaluation of expressions
  - Support for expressions that can set values and invoke methods
  - Pluggable API for resolving Expressions
  - Unified Expression Language: http://is.gd/2xqF
  - Spring 3.0 allows usage in XML files and @Value annotations

© 2012 Raible Designs

REST Support



http://infoq.com/articles/rest-introduction

REST with @PathParam

http://myserver/myapp/show/123

```
@RequestMapping(method = RequestMethod.GET)
public User show(@PathParam Long id) {
    return userManager.getUser(id);
}
```

© 2012 Raible Designs

REST with RestController

```
@Controller
public class ResumesController implements RestController<Resume, Long> {
```

GET http://myserver/myapp/resumes

```
    public List<Resume> index() {}
```

POST http://myserver/myapp/resumes

```
    public void create(Resume resume) {}
```

GET http://myserver/myapp/resumes/1

```
    public Resume show(Long id) {}
```

DELETE http://myserver/myapp/resumes/1

```
    public void delete(Long id) {}
```

PUT http://myserver/myapp/resumes/1

```
    public void update(Resume resume) {}
}
```

© 2012 Raible Designs

Dropping Support For...

- Commons Attributes
- TopLink (EclipseLink instead)
- MVC Controller hierarchy
- JUnit 3.8 Test classes

```
import org.springframework.test.AbstractTransactionalDataSourceSpringContextTests;

public class UserManagerTest extends AbstractTransactionalDataSourceSpringContextTests {
    private UserManager userManager;

    public void setUserManager(UserManager userManager) {
        this.userManager = userManager;
    }

    protected String[] getConfigLocations() {
        setAutowireMode(AUTOWIRE_BY_NAME);
        return new String[] {"classpath*:/WEB-INF/applicationContext*.xml"};
    }

    protected void onSetUpInTransaction() throws Exception {
        deleteFromTables(new String[] {"app_user"});
    }
```

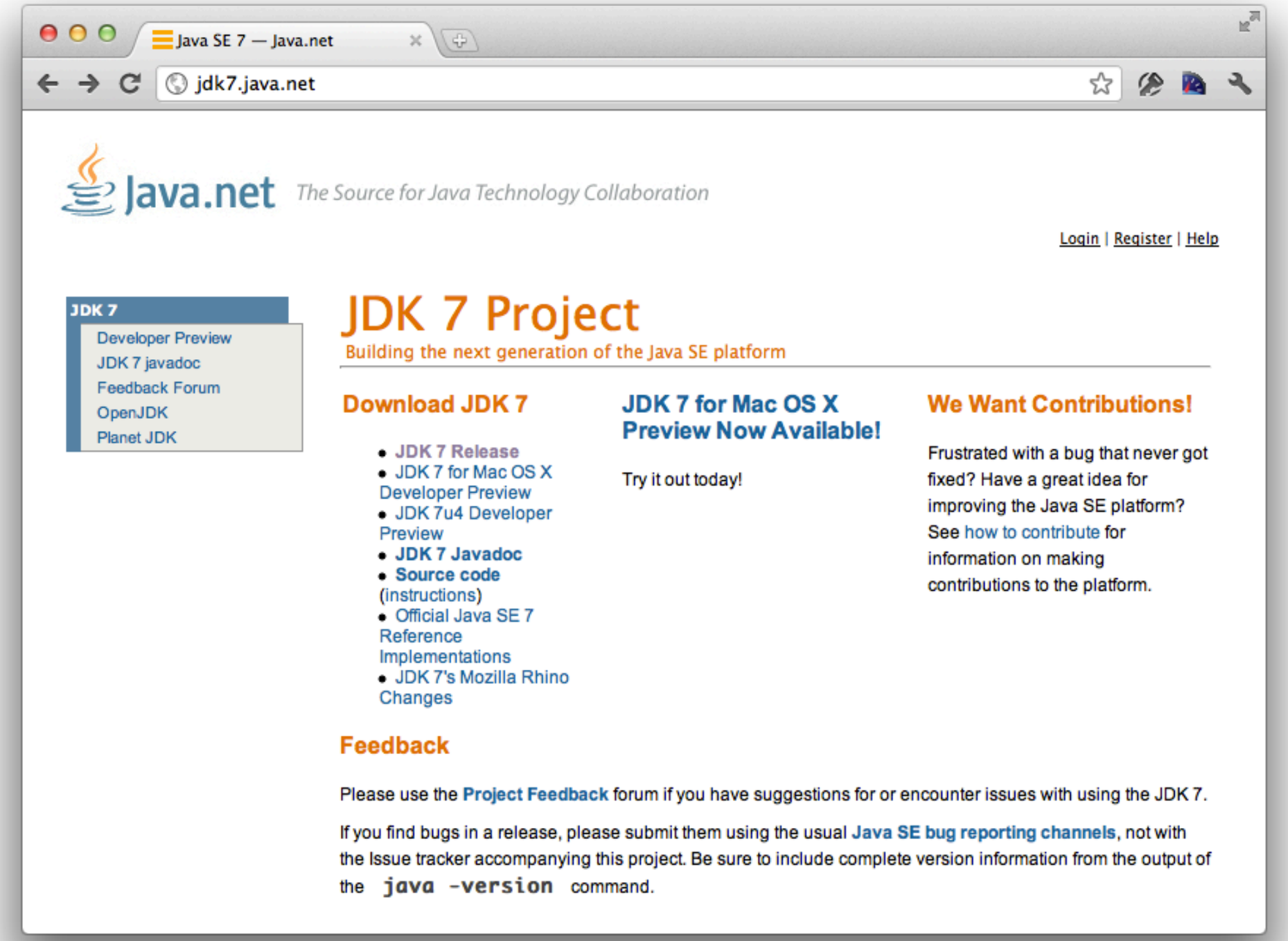
© 2012 Raible Designs

What's New in Spring 3.1

- Java 7 Support
- Hibernate 4 Support
- Servlet 3.0 Support
- Cache Abstraction
- Java Configuration
- Environments and Profiles
- Test Context Support for Configuration Classes and Profiles

© 2012 Raible Designs

Installing Java 7



© 2012 Raible Designs

Java 7 on OS X

© 2012 Raible Designs

Java 7 Support

- Fork/join support
- JDBC 4.1
  - Try-with-resources
  - RowSet 1.1
- Multicatch and final rethrow
- String in switch statement support

© 2012 Raible Designs

What's New in Hibernate 4?

- Move to Gradle for builds
- SessionFactory building
- Initial osgi-fication
- Java 6 / JDBC 4 as baseline
- Multi-tentant database support
- Migration to i18n logging framework

© 2012 Raible Designs

Hibernate 4

```
<!DOCTYPE hibernate-configuration PUBLIC "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
    "http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">

<hibernate-configuration>
    <session-factory>
        <mapping class="org.appfuse.model.User"/>
        <mapping class="org.appfuse.model.Role"/>
    </session-factory>
</hibernate-configuration>
```

```
[INFO] ------------------------------------------------------------------------
[INFO] BUILD FAILURE
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 1.269s
[INFO] Finished at: Tue Jan 31 09:25:26 MST 2012
[INFO] Final Memory: 6M/81M
[INFO] ------------------------------------------------------------------------
[ERROR] Failed to execute goal org.codehaus.mojo:hibernate3-maven-plugin:2.2:hbm2ddl (default) on project appfuse-h
bernate: Execution default of goal org.codehaus.mojo:hibernate3-maven-plugin:2.2:hbm2ddl failed: Could not parse co
figuration: file:/Users/mraible/dev/appfuse/data/hibernate/src/test/resources/hibernate.cfg.xml: www.hibernate.org
ested exception: www.hibernate.org -> [Help 1]
```

Changing to hibernate.sourceforge.net allows you to work offline

```
<!DOCTYPE hibernate-configuration PUBLIC "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
    "http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">
```

© 2012 Raible Designs

Hibernate 3

```
<bean class="org.springframework.dao.annotation.PersistenceExceptionTranslationPostProcessor"/>

<!-- Hibernate SessionFactory -->
<bean id="sessionFactory"
    class="org.springframework.orm.hibernate3.annotation.AnnotationSessionFactoryBean">
    <property name="dataSource" ref="dataSource"/>
    <property name="configLocation" value="classpath:hibernate.cfg.xml"/>
    <property name="hibernateProperties">
        <value>
            hibernate.dialect=${hibernate.dialect}
            hibernate.query.substitutions=true 'Y', false 'N'
            hibernate.cache.use_second_level_cache=true
            hibernate.cache.provider_class=org.hibernate.cache.EhCacheProvider
        </value>
        <!-- Turn batching off for better error messages under PostgreSQL -->
        <!-- hibernate.jdbc.batch_size=0 -->
    </property>
</bean>

<!-- Transaction manager for a single Hibernate SessionFactory (alternative to JTA) -->
<bean id="transactionManager" class="org.springframework.orm.hibernate3.HibernateTransactionManager">
    <property name="sessionFactory" ref="sessionFactory"/>
</bean>
```

© 2012 Raible Designs

Hibernate 4

```
<bean class="org.springframework.dao.annotation.PersistenceExceptionTranslationPostProcessor"/>

<bean id="hibernateExceptionTranslator"
      class="org.springframework.orm.hibernate4.HibernateExceptionTranslator"/>

<!-- Hibernate SessionFactory -->
<bean id="sessionFactory" class="org.springframework.orm.hibernate4.LocalSessionFactoryBean">
    <property name="dataSource" ref="dataSource"/>
    <property name="configLocation" value="classpath:hibernate.cfg.xml"/>
    <property name="hibernateProperties">
        <value>
            hibernate.dialect=${hibernate.dialect}
            hibernate.query.substitutions=true 'Y', false 'N'
            hibernate.cache.use_second_level_cache=true
            hibernate.cache.provider_class=org.hibernate.cache.EhCacheProvider
        </value>
        <!-- Turn batching off for better error messages under PostgreSQL -->
        <!-- hibernate.jdbc.batch_size=0 -->
    </property>
</bean>

<!-- Transaction manager for a single Hibernate SessionFactory (alternative to JTA) -->
<bean id="transactionManager" class="org.springframework.orm.hibernate4.HibernateTransactionManager">
    <property name="sessionFactory" ref="sessionFactory"/>
</bean>
```

© 2012 Raible Designs

Hibernate 4

```
<bean class="org.springframework.dao.annotation.PersistenceExceptionTranslationPostProcessor"/>

<bean id="hibernateExceptionTranslator"
      class="org.springframework.orm.hibernate4.HibernateExceptionTranslator"/>

<!-- Hibernate SessionFactory -->
<bean id="sessionFactory" class="org.springframework.orm.hibernate4.LocalSessionFactoryBean">
    <property name="dataSource" ref="dataSource"/>
    <property name="packagesToScan" value="org.appfuse.model"/>
    <property name="hibernateProperties">
        <value>
            hibernate.dialect=${hibernate.dialect}
            hibernate.query.substitutions=true 'Y', false 'N'
            hibernate.cache.use_second_level_cache=true
            hibernate.cache.provider_class=org.hibernate.cache.EhCacheProvider
        </value>
        <!-- Turn batching off for better error messages under PostgreSQL -->
        <!-- hibernate.jdbc.batch_size=0 -->
    </property>
</bean>

<!-- Transaction manager for a single Hibernate SessionFactory (alternative to JTA) -->
<bean id="transactionManager" class="org.springframework.orm.hibernate4.HibernateTransactionManager">
    <property name="sessionFactory" ref="sessionFactory"/>
</bean>
```

© 2012 Raible Designs

Spring + Hibernate 3

```
@Autowired
@Required
public void setSessionFactory(SessionFactory sessionFactory) {
    this.sessionFactory = sessionFactory;
    this.hibernateTemplate = new HibernateTemplate(sessionFactory);
}

public List<T> getAll() {
    return hibernateTemplate.loadAll(this.persistentClass);
}

public T get(PK id) {
    T entity = hibernateTemplate.get(this.persistentClass, id);
    if (entity == null) {
        log.warn("Uh oh, '" + this.persistentClass + "' object with id '" + id + "' not found...");
        throw new ObjectRetrievalFailureException(this.persistentClass, id);
    }

    return entity;
}

public T save(T object) {
    return hibernateTemplate.merge(object);
}

public void remove(PK id) {
    hibernateTemplate.delete(this.get(id));
}
```

© 2012 Raible Designs

Hibernate 4

```
@Autowired
@Required
public void setSessionFactory(SessionFactory sessionFactory) {
    this.sessionFactory = sessionFactory;
}

public List<T> getAll() {
    return sessionFactory.getCurrentSession().createQuery("from " + this.persistentClass).list();
}

public T get(PK id) {
    T entity = (T) sessionFactory.getCurrentSession().get(this.persistentClass, id);
    if (entity == null) {
        log.warn("Uh oh, '" + this.persistentClass + "' object with id '" + id + "' not found...");
        throw new ObjectRetrievalFailureException(this.persistentClass, id);
    }

    return entity;
}

public T save(T object) {
    return (T) sessionFactory.getCurrentSession().merge(object);
}

public void remove(PK id) {
    sessionFactory.getCurrentSession().delete(this.get(id));
}
```

© 2012 Raible Designs

JPA with Spring 2.0

```
<bean id="customerRepository" class="spring.kickstart.repository.CustomerRepositoryImpl"/>
<bean id="productRepository" class="spring.kickstart.repository.ProductRepositoryImpl"/>

<bean id="customerService" class="spring.kickstart.domain.CustomerServiceImpl" autowire="byName"/>

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

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

<tx:annotation-driven/>

<bean id="entityManagerFactory" depends-on="database"
      class="org.springframework.orm.jpa.LocalContainerEntityManagerFactoryBean">
    <property name="dataSource" ref="dataSource"/>
    <property name="persistenceUnitName" value="kickstart"/>
    <property name="jpaVendorAdapter">
        <bean class="org.springframework.orm.jpa.vendor.HibernateJpaVendorAdapter">
            <property name="showSql" value="true"/>
            <property name="generateDdl" value="false"/>
            <property name="database" value="HSQL"/>
        </bean>
    </property>
</bean>
```

© 2012 Raible Designs

JPA with Spring 3.1

```
<!-- Activates scanning of @Autowired -->
<context:annotation-config/>

<!-- Activates scanning of @Repository and @Service -->
<context:component-scan base-package="spring.kickstart"/>

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

<tx:annotation-driven/>

<bean id="entityManagerFactory"
      class="org.springframework.orm.jpa.LocalContainerEntityManagerFactoryBean">
    <property name="dataSource" ref="dataSource"/>
    <property name="packagesToScan" value="spring.kickstart"/>
    <property name="jpaVendorAdapter">
        <bean class="org.springframework.orm.jpa.vendor.HibernateJpaVendorAdapter">
            <property name="showSql" value="true"/>
            <property name="generateDdl" value="true"/>
            <property name="database" value="HSQL"/>
        </bean>
    </property>
</bean>

<jdbc:embedded-database id="dataSource">
    <jdbc:script location="classpath:schema.sql"/>
    <jdbc:script location="classpath:test-data.sql"/>
</jdbc:embedded-database>
```

© 2012 Raible Designs

Spring Data

Spring Data makes it easier to build Spring-powered applications that use new data access technologies such as non-relational databases, map-reduce frameworks, and cloud based data services as well as provide improved support for relational database technologies.

http://www.springsource.org/spring-data

© 2012 Raible Designs

Plain JPA :: Before

```
/**
 * Service interface for {@link Customer}s.
 *
 * @author Oliver Gierke
 */
public interface CustomerService {

    Customer findById(Long id);

    Customer save(Customer customer);

    List<Customer> findAll();

    List<Customer> findAll(int page, int pageSize);

    List<Customer> findByLastname(String lastname, int page, int pageSize);
}
```

© 2012 Raible Designs

Plain JPA :: Before

```
/**
 * Plain JPA implementation of {@link CustomerService}.
 *
 * @author Oliver Gierke
 */
@Repository
@Transactional(readOnly = true)
public class CustomerServiceImpl implements CustomerService {

    @PersistenceContext
    private EntityManager em;

    @Override
    public Customer findById(Long id) {
        return em.find(Customer.class, id);
    }

    @Override
    public List<Customer> findAll() {

        return em.createQuery("select c from Customer c", Customer.class).getResultList();
    }

    @Override
    @Transactional
    public Customer save(Customer customer) {
        if (customer.getId() == null) {
            em.persist(customer);
            return customer;
        } else {
            return em.merge(customer);
        }
    }
```

© 2012 Raible Designs

JPA with Spring Data :: After

```
/**
 * Repository to manage {@link Customer} instances.
 *
 * @author Oliver Gierke
 */
public interface CustomerRepository extends
        CrudRepository<Customer, Long>, JpaSpecificationExecutor<Customer> {

    Page<Customer> findByLastname(String lastname, Pageable pageable);
}
```

https://github.com/SpringSource/spring-data-jpa-examples

© 2012 Raible Designs

What's New in Servlet 3.0

- Programmatic definition of filters, servlets, listeners and URL patterns
  - @WebServlet, @WebFilter, @WebListener
  - @WebInitParam
  - @MultipartConfig
- Web Fragments
- Asynchronous Servlet and Comet Support

© 2012 Raible Designs

Hardest thing about Servlet 3?

Is getting the Maven dependency right ...

```
<!-- Servlet 3.0 -->
<!-- http://stackoverflow.com/questions/1979957/maven-dependency-for-servlet-3-0-api -->
<dependency>
    <groupId>org.glassfish</groupId>
    <artifactId>javax.servlet</artifactId>
    <version>3.0</version>
    <scope>provided</scope>
</dependency>
```

© 2012 Raible Designs

Spring + Servlet 3.0

- WebApplicationInitializer for programmatic configuration
- Instead of:

```
<servlet>
    <servlet-name>kickstart</servlet-name>
    <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
    <load-on-startup>1</load-on-startup>
</servlet>

<servlet-mapping>
    <servlet-name>kickstart</servlet-name>
    <url-pattern>*.htm</url-pattern>
</servlet-mapping>
```

© 2012 Raible Designs

Servlet 3.0

- WebApplicationInitializer for programmatic configuration
- You use:

```
import org.springframework.web.WebApplicationInitializer;
import org.springframework.web.servlet.DispatcherServlet;

import javax.servlet.ServletContext;
import javax.servlet.ServletRegistration;

public class KickstartWebAppInitializer implements WebApplicationInitializer {

    @Override
    public void onStartup(ServletContext container) {

        ServletRegistration.Dynamic dispatcher =
                container.addServlet("dispatcher", new DispatcherServlet());
        dispatcher.setLoadOnStartup(1);
        dispatcher.addMapping("*.htm");
    }

}
```

© 2012 Raible Designs

Servlet 3.0

- WebApplicationInitializer with XmlApplicationContext

```
import org.springframework.web.WebApplicationInitializer;
import org.springframework.web.context.support.XmlWebApplicationContext;
import org.springframework.web.servlet.DispatcherServlet;

import javax.servlet.ServletContext;
import javax.servlet.ServletRegistration;

public class KickstartWebAppInitializer implements WebApplicationInitializer {

    @Override
    public void onStartup(ServletContext container) {
        XmlWebApplicationContext appContext = new XmlWebApplicationContext();
        appContext.setConfigLocation("/WEB-INF/applicationContext.xml");

        ServletRegistration.Dynamic dispatcher =
                container.addServlet("dispatcher", new DispatcherServlet());
        dispatcher.setLoadOnStartup(1);
        dispatcher.addMapping("*.htm");
    }
}
```

© 2012 Raible Designs

Servlet 3.0

- WebApplicationInitializer with JavaConfig

```
public class KickstartWebAppInitializer implements WebApplicationInitializer {

    @Override
    public void onStartup(ServletContext container) {
        // Create the 'root' Spring application context
        AnnotationConfigWebApplicationContext rootContext =
                new AnnotationConfigWebApplicationContext();
        rootContext.register(AppConfig.class);

        // Manage the lifecycle of the root application context
        container.addListener(new ContextLoaderListener(rootContext));

        // Create the dispatcher servlet's Spring application context
        AnnotationConfigWebApplicationContext dispatcherContext =
                new AnnotationConfigWebApplicationContext();
        dispatcherContext.register(DispatcherConfig.class);

        // Register and map the dispatcher servlet
        ServletRegistration.Dynamic dispatcher =
                container.addServlet("dispatcher",
                    new DispatcherServlet(dispatcherContext));
        dispatcher.setLoadOnStartup(1);
        dispatcher.addMapping("/");
    }
}
```

© 2012 Raible Designs

@MVC Resources

```
<!-- View Resolver for JSPs -->
<beans:bean id="viewResolver"
            class="org.springframework.web.servlet.view.InternalResourceViewResolver">
    <beans:property name="viewClass" value="org.springframework.web.servlet.view.JstlView"/>
    <beans:property name="prefix" value="/WEB-INF/pages/"/>
    <beans:property name="suffix" value=".jsp"/>
</beans:bean>

<beans:bean id="messageSource"
            class="org.springframework.context.support.ResourceBundleMessageSource">
    <beans:property name="basename" value="messages"/>
    <beans:property name="useCodeAsDefaultMessage" value="true"/>
</beans:bean>

<context:component-scan base-package="spring.kickstart"/>

<!-- Enables the Spring MVC @Controller programming model -->
<annotation-driven/>

<resources mapping="/resources/**" location="/resources/"/>

<!-- Maps '/' requests to the 'index' view -->
<view-controller path="/" view-name="index"/>
```

© 2012 Raible Designs

@MVC Resources

```
<link rel="stylesheet" type="text/css"
      href="${ctx}/resources/styles/deliciouslyblue/theme.css" title="default" />
<link rel="alternate stylesheet" type="text/css"
      href="${ctx}/resources/styles/deliciouslygreen/theme.css" title="green" />

<script type="text/javascript" src="http://code.jquery.com/jquery-1.7.1.min.js"></script>
<script type="text/javascript" src="${ctx}/resources/scripts/application.js"></script>
```

© 2012 Raible Designs

Web Performance Best Practices

- Optimizing caching
- Minimizing round-trip times
- Minimizing request overhead
- Minimizing payload size
- Optimizing browser rendering
- Optimizing for mobile

http://code.google.com/speed/page-speed/docs/rules_intro.html

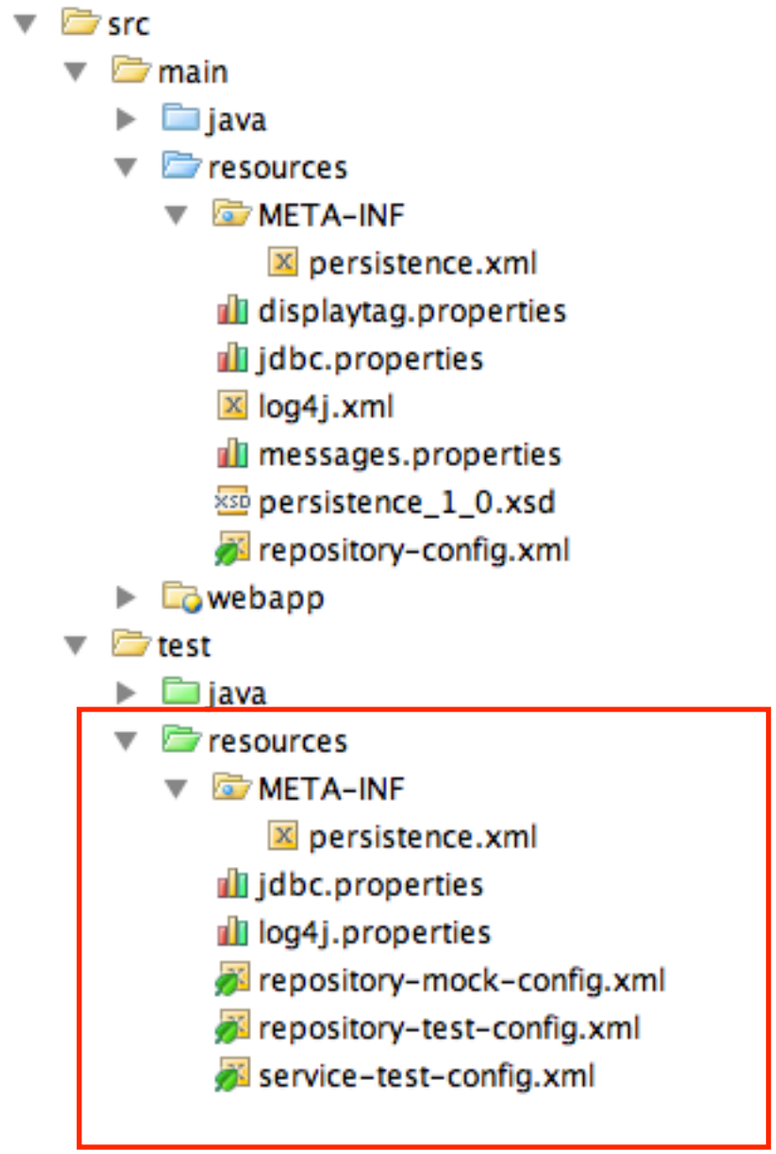
© 2012 Raible Designs

Wro4j

- Open Source Java project for optimization of web resources
- Provides concatenation and minimization of JS and CSS
- Gzip, YUI Compressor, JsHint, JsHint, CssLint, LESS, SASS, CoffeeScript, Dojo Shrinksafe

© 2012 Raible Designs

Profiles



© 2012 Raible Designs

XML Profiles

- Profiles allow for environment-specific bean definitions
- Activated with:
  - -Dspring.profiles.active=test
  - ctx.getEnvironment().setActiveProfiles("dev")
  - <init-param> spring.profiles.active in web.xml
  - @Profile("prod") with @Configuration

© 2012 Raible Designs

Wro4j and Profiles

```
<group name="scripts">
    <js>http://code.jquery.com/jquery-1.7.1.min.js</js>
    <js>/resources/scripts/application.js</js>
</group>
```

```
<filter-mapping>
    <filter-name>wro4j</filter-name>
    <url-pattern>/assets/*</url-pattern>
</filter-mapping>
```

```
<% if ("dev".equals(System.getProperty("spring.profiles.active"))) { %>
<script type="text/javascript" src="http://code.jquery.com/jquery-1.7.1.min.js"></script>
<script type="text/javascript" src="${ctx}/resources/scripts/application.js"></script>
<% } else { %>
<script type="text/javascript" src="${ctx}/assets/scripts.js"></script>
<% } %>
```

© 2012 Raible Designs

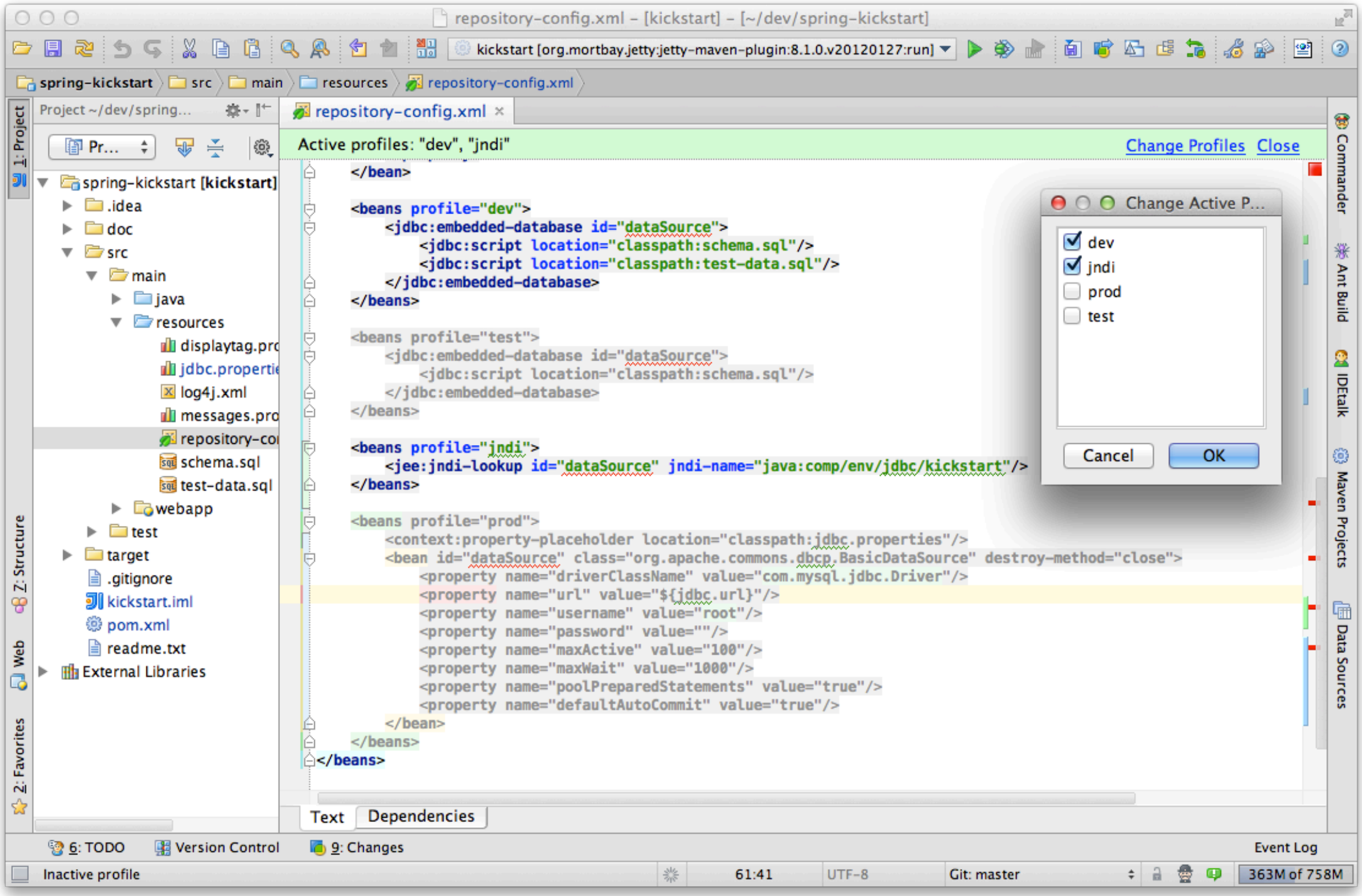
XML Profiles

```
<beans profile="dev">
    <jdbc:embedded-database id="dataSource">
        <jdbc:script location="classpath:schema.sql"/>
        <jdbc:script location="classpath:test-data.sql"/>
    </jdbc:embedded-database>
</beans>

<beans profile="test">
    <jdbc:embedded-database id="dataSource">
        <jdbc:script location="classpath:schema.sql"/>
    </jdbc:embedded-database>
</beans>
```

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

<beans profile="prod">
    <context:property-placeholder location="classpath:jdbc.properties"/>
    <bean id="dataSource" class="org.apache.commons.dbcp.BasicDataSource" destroy-method="close">
        <property name="driverClassName" value="${jdbc.driverClassName}"/>
        <property name="url" value="${jdbc.url}"/>
        <property name="username" value="${jdbc.username}"/>
        <property name="password" value="${jdbc.password}"/>
        <property name="maxActive" value="100"/>
        <property name="maxWait" value="1000"/>
        <property name="poolPreparedStatements" value="true"/>
        <property name="defaultAutoCommit" value="true"/>
    </bean>
</beans>
```

© 2012 Raible Designs

IntelliJ Profile Activation



© 2012 Raible Designs

@Profile

```
import org.springframework.context.annotation.Profile;

import java.lang.annotation.ElementType;
import java.lang.annotation.Retention;
import java.lang.annotation.RetentionPolicy;
import java.lang.annotation.Target;

@Target(ElementType.TYPE)
@Retention(RetentionPolicy.RUNTIME)
@Profile("dev")
public @interface Dev {
}
```

```
@Repository
@Dev
@Transactional(readOnly = true)
public class CustomerRepositoryImpl implements CustomerRepository {

    @PersistenceContext
    private EntityManager em;
```

© 2012 Raible Designs

Cache Abstraction

- Not an implementation, but an abstraction
- Frees you from writing logic, but does not provide stores
  - JDK ConcurrentMap
  - EhCache
- @Cacheable and @CacheEvict
- Can use Spring EL for “key” and “condition” attributes
- Annotations triggered by <cache:annotation-driven/>

© 2012 Raible Designs

Test Support for JavaConfig and Profiles

- Spring 3.0 added @Configuration (JavaConfig)
- TestContext Framework provides annotation-driven testing support
- Spring 3.1 adds “loader” attribute to @ContextConfiguration for @Configuration

© 2012 Raible Designs

Spring 2.0 Testing

```
public class CustomerRepositoryTest extends AbstractJpaTests {

    private CustomerRepository customerRepository;
    private EntityManagerFactory emf;

    public void setCustomerRepository(CustomerRepository customerRepository) {
        this.customerRepository = customerRepository;
    }

    public void setEntityManagerFactory(EntityManagerFactory emf) {
        this.emf = emf;
    }

    protected String[] getConfigLocations() {
        return new String[] {"repository-test-config.xml"};
    }

    protected void onSetUpInTransaction() throws Exception {
        EntityManager em = EntityManagerFactoryUtils.getTransactionalEntityManager(emf);
        Customer c = new Customer();
        c.setName("Test");
        c.setCustomerSince(new Date());
        em.persist(c);

        super.onSetUpInTransaction();
    }

    public void testAddCustomer() {
        Customer c = new Customer();
        ...
```

© 2012 Raible Designs

TestContext with XML

```
@RunWith(SpringJUnit4ClassRunner.class)
@ContextConfiguration(locations = {"/repository-config.xml"})
@Transactional
@TransactionConfiguration(transactionManager = "transactionManager", defaultRollback = false)
public class CustomerRepositoryTest {

    @PersistenceContext
    private EntityManager em;
    @Autowired
    private CustomerRepository customerRepository;

    @Before
    @Transactional
    public void onSetUpInTransaction() {
        Customer c = new Customer();
        c.setName("Test");
        c.setCustomerSince(new Date());

        em.persist(c);
    }

    @Test
    public void testAddCustomer() {
        Customer c = new Customer();
        c.setName("New");
        c.setCustomerSince(new Date());
        customerRepository.save(c);
        List<Customer> customers = customerRepository.findAll();
        assertTrue(customers.contains(c));
    }
```

© 2012 Raible Designs

TestContext with @Configuration

```
@RunWith(SpringJUnit4ClassRunner.class)
@ContextConfiguration(loader = AnnotationConfigContextLoader.class)
@Transactional
@TransactionConfiguration(transactionManager = "transactionManager", defaultRollback = false)
public class CustomerRepositoryTest {

    @Configuration
    static class ContextConfiguration {
        @Bean
        public CustomerRepository customerRepository() {
            return new CustomerRepositoryImpl();
        }
    }

    @PersistenceContext
    private EntityManager em;
    @Autowired
    private CustomerRepository customerRepository;

    @Before
    @Transactional
    public void onSetUpInTransaction() {
        Customer c = new Customer();
        c.setName("Test");
        c.setCustomerSince(new Date());

        em.persist(c);
    }

    @Test
    public void testAddCustomer() {
        Customer c = new Customer();
```

© 2012 Raible Designs

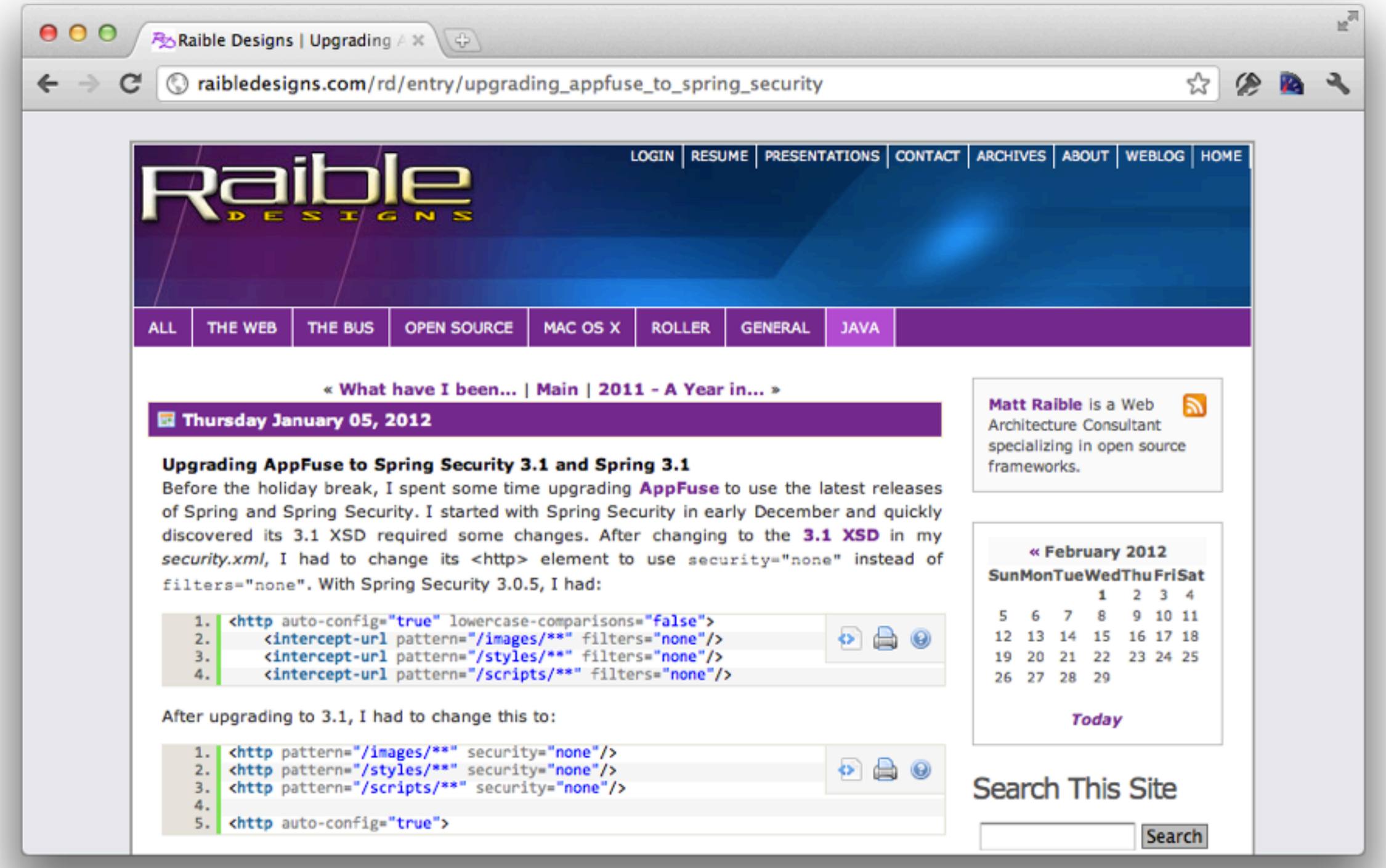
Summary

- Spring 3.1 has a lot of new stuff
  - @Cacheable
  - Profiles
  - Servlet 3.0 Support
  - Better Testing Support

- Compilation errors when upgrading?
  - mvn dependency:tree | grep spring

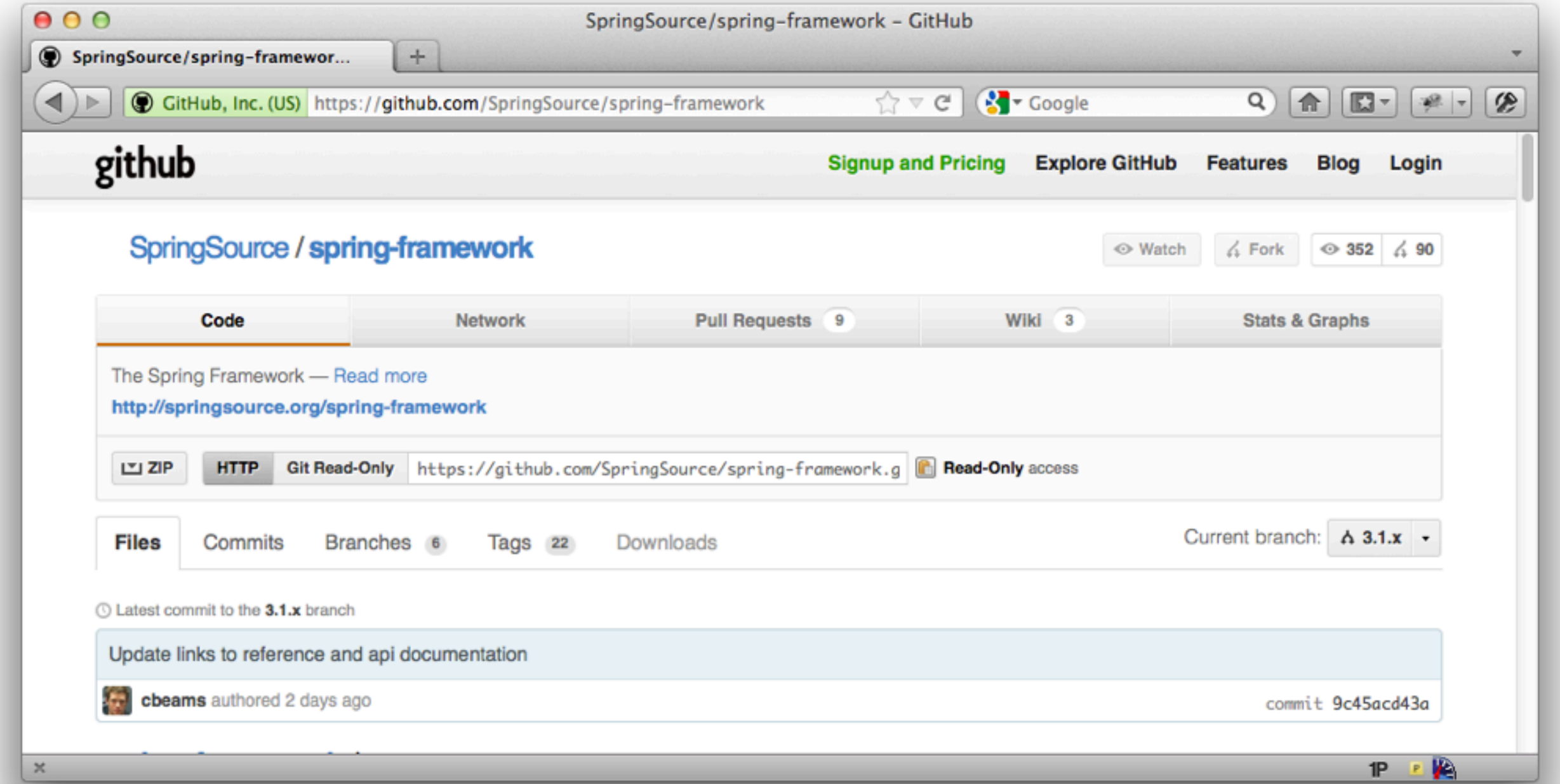
© 2012 Raible Designs

My Upgrade Story



© 2012 Raible Designs

Get the Source



https://github.com/SpringSource/spring-framework

© 2012 Raible Designs

Learn More

- This Week in Spring on springsource.org
- blog.springsource.com
- SpringSource Developer Channel on YouTube
- Stackoverflow.com
- Google it!

© 2012 Raible Designs

heroku

Heroku for Java

by Adam - Aug 25, 2011

We're pleased to announce the public beta of Heroku for Java. Java is the fourth official language available on the **Cedar** stack.

Java is, by **many measures**, the world's most popular programming language. In addition to its large and diverse developer base, it offers a huge ecosystem of libraries and tools, an extremely well-tuned VM for fast and reliable runtime performance, and an accessible C-like syntax.

But there are also many criticisms commonly leveled against the language. We'll take a closer look at Java's strengths and weaknesses in a moment, but first:

Heroku for Java in 2 minutes

BETA
CLOUD FOUNDRY™
DEPLOY & SCALE YOUR APPLICATIONS IN SECONDS

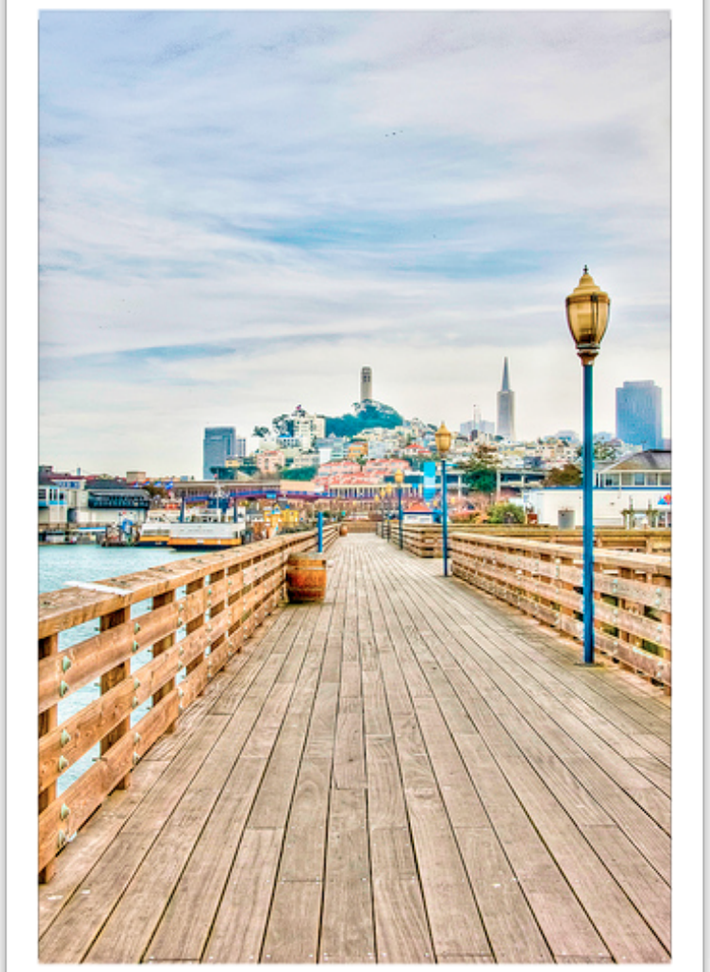
CloudBees

© 2012 Raible Designs

Questions?

matt@raibledesigns.com

- http://raibledesigns.com
- http://twitter.com/mraible



My Presentations

- http://slideshare.net/mraible
- http://raibledesigns.com/rd/page/publications

© 2012 Raible Designs