JSF 2.3 Pre-Public Review EC Update

Ed Burns
JSF Spec Co-Lead
Java EE Spec Group
January, 2017
Safe Harbor Statement

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle’s products remains at the sole discretion of Oracle.
My Plan for Your Time Investment

1. JCP EC Template
2. JSF 2.3 is the most community driven Oracle-lead JSR
3. Schedule
3. Feature Review
JCP EC Template
Business/Marketing/Ecosystem Justification

• Why do this JSR?
  – JSF is an important part of the Java EE platform and must be updated to keep current and fix long standing issues.

• What’s the need?
  – Java EE and Java SE 8 have some compelling new features, we’d like to take advantage of them in JSF.
  – We need to continue to improve the integration of the Java EE platform specs.

• How does it fit in to the Java ecosystem?
  – At a technical level, it highly leverages Servlet, CDI, Bean Validation and EL
  – At a community level, it validates the JCP as a source for Java collaboration.
Business/Marketing/Ecosystem Justification

• Is the idea ready for standardization?
  – Yes. It continues the tradition of taking existing ideas, and standardizing them.
Expert Group Members (alphabetical)

• Burns, Edward Oracle
• Caputo, Frank individual
• Civici, Cagatay individual
• Fyten, Ken ICEsoft Technologies Inc.
• Griffin, Neil Liferay, Inc
• Juneau, Josh individual
• Leathem, Brian Red Hat
• Mann, Kito Individual

• Mueller, Michael Individual
• Nicolucci, Paul IBM
• Riem, Manfred Oracle
• Scholtz, Bauke ZEEF
• Tijms, Arjan ZEEF
• Uribe, Leonardo Irian Solutions GmbH
Expert Group Operations

• We had a face-to-face at JavaOne 2016
• We’ve always made use of the java.net collaboration infrastructure
  – Anyone can create issues in the JIRA
  – Any EG member can top-post to the jsr372-experts mailing list
  – All mails to that list are mirrored to the “users” list:
    • users@javaserverfaces-spec-public.java.net
Deliverables

• Spec Document
  – JavaDocs
  – RenderKitDocs
  – VDLDocs (for “tags” to be used in your JSF Pages)
  – PDF document

• Reference Implementation, integrated into Java EE 8 Reference Implementation

• TCK
Publicity

- JavaOne presentations
- Community Blogs
- @jsf_spec twitter handle
Collaboration with other community groups

• Lots of cross-membership with Portlet
• Expert Group members do represent their own communities
  – PrimeFaces
  – Liferay
  – MyFaces implementation and community
Implementations

- Mojarra, Oracle Reference Implementation
- MyFaces, Apache implementation, used in WebSphere
Licenses

• Standard Oracle Spec Licenses
• RI licensed same as GlassFish
JSF: Keeping the C in JCP since 2004
History of JSF Community Involvement

Some Highlights

• JSF was the first part of Java to be open sourced
  – https://community.oracle.com/blogs/edburns/2004/06

• JSF has a community created logo
  – https://java.net/jira/browse/JAVASERVERFACES_SPEC_PUBLIC-980

• JSF 1.0 had hugely significant contributions from many vendors
  – IBM: Portlet integration JSR-168
  – Oracle: UIComponent Model
  – BEA: Managed Bean model
  – Sun: Spec leadership and many fundamental aspects
History of JSF Community Involvement

Some Highlights

• JSF 2.0
  – Facelets, mainly contributed by Jacob Hookom
  – Ajax, collaboratively developed, but heavily influenced by IceFaces and Ajax4JSF

• JSF 2.2
  – Resource Library Contracts from Frank Caputo
  – HTML5 Friendly Markup from Imre Osswald
JSF Historical Perspective

2004

- Spec Lead (Sun)
- Community

95%
JSF Historical Perspective

2008

- Spec Lead (Sun) 20%
- Community 80%
JSF Historical Perspective

2009

- Spec Lead (Sun): 40%
- Community: 60%
JSF Historical Perspective

2010

- Spec Lead (Oracle): 40%
- Community: 60%
JSF Historical Perspective

2013

- Spec Lead (Oracle): 25%
- Community: 75%
JSF Historical Perspective

Poll: JSF and Spring MVC tie

2014

- Spec Lead (Oracle): 20%
- Community: 80%
JSF Historical Perspective

2016

- Spec Lead (Oracle)
- Community

10%
90%
JSF 2.3 and the Community
JSF 2.3 JSR-372

Section 2: Request

2.1 Please describe the proposed Specification:

This JSR aims to improve the clarity of the existing JavaServer Faces (JSF) specification and introduce a small, targeted set of new features as directed by community contribution. Requirements are grouped into four major categories: Small-scale new features, community driven improvements, and platform integration.

Small-scale new features
While the scope of new features is intentionally limited, it is important to continue to evolve JSF to support the growing needs of existing users. To that end, an enhancement to the JSF Ajax API is proposed: namely Ajax method invocation. This feature allows invoking CDI managed bean methods directly from Ajax, allowing the response to be sent using Java EE 7 standard JSON.

Community driven improvements
JSF has always been indisputably community focused. This tradition continues in JSF 2.3. While the set of issues addressed is very much up to what the expert group is willing to contribute, some obvious suggestions include multi-field validation, @Inject FacesContext, EL performance optimizations, and cross-form Ajax clarifications.

Platform integration
Previous versions of JSF intentionally lagged one version behind the Java EE version in which it was bundled. This release abandons this approach in favor of being able to leverage platform features from Java EE 8 and Java SE 8.

Support for MVC 1.0
Several existing features of JSF may make sense to expose for use by the MVC 1.0 specification, also in Java EE 8. These features may include but are not limited to: The Facelets View Declaration Language, JSF native CDI scopes, and faces flows.
JSF 2.3 JSR-372

Small, targeted set of new features directed by community contribution.
JSF 2.3 JSR-372

Section 2: Request
2.1 Please describe the proposed Specification:

This JSF aims to improve the clarity of the existing JavaServer Faces (JSF) specification and introduce a small, targeted set of new features as directed by community contribution.

- Small-scale new features, community driven improvements, and platform integration.

Small-scale new features
While the scope of new features is intentionally limited, it is important to continue to evolve JSF to support the growing needs of existing users. To that end, an enhancement to the JSF Ajax API is proposed: namely Ajax method Invocation. This feature allows invoking CDI managed bean methods directly from Ajax, allowing the response to be sent using Java EE 7 standard JSON.

Better CDI Integration

Community driven improvements
JSF has always been indisputably community focused. This tradition continues in JSF 2.3. While the set of features is limited, it is up to what the expert group is willing to contribute, some obvious suggestions include multi-field validation, performance optimizations, and cross-form Ajax clarifications.

Platform Integration
Previous versions of JSF intentionally lagged one version behind the Java EE version in which it was bundled. This release abandons this approach in favor of being able to leverage platform features from Java EE 8 and Java SE 8.

Support for MVC 1.0
Several existing features of JSF may make sense to expose for use by the MVC 1.0 specification, also in Java EE 8. These features may include but are not limited to: The Facelets View Declaration Language, JSF native CDI scopes, and faces flows.
JSF 2.3 JSR-372

Small, targeted set of new features directed by community contribution

Section 2: Request
2.1 Please describe the proposed Specification:

This JSR aims to improve the clarity of the existing JavaServer Faces (JSF) specification and introduce a small, targeted set of new features as directed by community contribution.

Small-scale new features
While the scope of new features is intentionally limited, it is important to continue to evolve JSF to support the growing needs of existing users. To that end, an enhancement to the JSF Ajax API is proposed: namely, Ajax method invocations. This feature allows invoking CDI-managed beans directly from Ajax, allowing the response to be sent using Java EE 7 standard JSON.

Better CDI Integration

Ajax Improvements

JSF has always been indisputably community focused. This tradition continues in JSF 2.3. While the set of features adds up to what the expert group is willing to contribute, some drivers include multi-field validation, @InheritFacesContext, EL performance optimizations, and cross-form Ajax clarifications.

Platform Integration
Previous versions of JSF intentionally lagged one version behind the Java EE version in which it was bundled. This release abandons this approach in favor of being able to leverage platform features from Java EE 8 and Java SE 8.

Support for MVC 1.0
Several existing features of JSF may make sense to expose for use by the MVC 1.0 specification, also in Java EE 8. These features may include but are not limited to: The Facelets View Declaration Language, JSF native CDI scopes, and faces flows.
JSF 2.3 JSR-372

Small, targeted set of new features directed by community contribution

Ajax Improvements

Better CDI Integration

Direct Dependence on EE 8
JSF 2.3 JSR-372

Section 2: Request

2.1 Please describe the proposed Specification:

This JSR aims to improve the clarity of the existing JavaServer Faces (JSF) specification and introduce a small, targeted set of new features as directed by community contribution.

Requirements are grouped into three major categories: Small-scale new features, community driven improvements, and platform integration.

Small-scale new features

While the scope of new features is intentionally limited, it is important to continue to evolve JSF to support the growing needs of existing users. To that end, an enhancement to the JSF Ajax API is proposed: namely Ajax method invocationl This feature allows invoking CDI managed bean methods directly from Ajax, allowing the response to be sent using Java EE 7/standard JSON.

Community driven improvements

JSF has always been indisputably community-focused. This tradition continues in JSF 2.3. While the set of features have been prioritized up to what the expert group is willing to contribute; some obvious suggestions include multi-field validation, @Inject FacesContext, EL performance optimizations, and cross-form Ajax clarifications.

Platform integration

Previous versions of JSF intentionally lagged one version behind the Java EE version in which it was bundled. This release abandons this approach in favor of being able to leverage platform features from the Java EE 8 version which it is bundled. This release abandons this approach in favor of being able to leverage platform features from the Java EE 8 and Java SE 8.

Support for MVC 1.0

Several existing features of JSF may make sense to expose for use by the MVC 1.0 specification, also in Java EE 8. These features may include but are not limited to: The Faceslets View Declaration Language, JSF native CDI scopes, and faces flows.

Direct Dependence on EE 8

Ajax Improvements

Better CDI Integration

Small, targeted set of new features directed by community contribution
JSF 2.3 and the Community

Activism
JSF 2.3 and the Community

• JSF EG Member Josh Juneau’s famous blog post
  – Used publically available information, thanks to our adherence to JCP transparency rules, to highlight the activity level of JSF over time
  – Made the case that the level was insufficient
JSF 2.3 and the Community

EG Mail traffic per Month

EG Mail traffic per Month
JSF 2.3 and the Community

GitHub Commits

- Though currently Mojarra GitHub is a mirror of Mojarra on java.net git


Contributions to master, excluding merge commits
JSF 2.3 Completion Schedule
JSF 2.3 Completion Schedule

• Shared with JSF EG on 2016-12-09
  – 2017-01-12 All JSF 2.3 related EG spec content delivered to list
  – 2017-01-20 30 Day Public Review commences
  – 2017-01-27 Any remaining EG-driven implementation work done
  – 2017-03-14 Final Approval Ballot commences
JSF 2.3 Completion Schedule

• Additional Details for EC
  – 2017-01-12 All JSF 2.3 related EG spec content delivered to list
  – 2017-01-20 30 Day Public Review commences
  – 2017-01-27 Any remaining EG-driven implementation work done
    • 2017-02-21 14 Day Public Draft Approval Ballot commences
    • 2017-03-06 14 Day Public Draft Approval Ballot completes
    • 2017-03-07 7 Day Proposed Final Draft of spec to JCP PMO
      – Also submit final RI and TCK for Final Approval Ballot
  – 2017-03-14 Final Approval Ballot commences
    • 2017-03-27 Final Approval Ballot completes
    • 2017-03-28 Disband JSF EG
Feature Review
JSF 2.3 Big Ticket New Features, done by the JSF EG

http://arjan-tijms.omnifaces.org/p/jsf-23.html

• Better CDI Integration
  – Way more things are injectable
  – Finally marking legacy managed beans as deprecated
• Websocket Integration
• Ajax Method Invocation
• Class Level Bean Validation
• Java Time support
• UIData and UIRepeat improvements
## JSF 2.3 Big Ticket New Features

http://arjan-tijms.omnifaces.org/p/jsf-23.html

<table>
<thead>
<tr>
<th>Artefact</th>
<th>EL name</th>
<th>Qualifier</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application</td>
<td>#{application}</td>
<td>-</td>
<td>java.lang.Object</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>javax.servlet.ServletContext</td>
</tr>
<tr>
<td>ApplicationMap</td>
<td>#{applicationScope}</td>
<td>@ApplicationMap</td>
<td>java.util.Map&lt;String, Object&gt;</td>
</tr>
<tr>
<td>CompositeComponent</td>
<td>#{cc}</td>
<td>(Not injectable)</td>
<td>javax.faces.component.UIComponent</td>
</tr>
<tr>
<td>Component</td>
<td>#{component}</td>
<td>(Not injectable)</td>
<td>javax.faces.component/UIComponent</td>
</tr>
<tr>
<td>RequestCookieMap</td>
<td>#{cookie}</td>
<td>@RequestCookieMap</td>
<td>java.util.Map&lt;String, Object&gt;</td>
</tr>
<tr>
<td>FacesContext</td>
<td>#{facesContext}</td>
<td>-</td>
<td>javax.faces.context.FacesContext</td>
</tr>
<tr>
<td>Flash</td>
<td>#{flash}</td>
<td>-</td>
<td>javax.faces.context.Flash</td>
</tr>
<tr>
<td>FlowMap</td>
<td>#{flowScope}</td>
<td>@FlowMap</td>
<td>java.util.Map&lt;Object, Object&gt;</td>
</tr>
<tr>
<td>HeaderMap</td>
<td>#{header}</td>
<td>@HeaderMap</td>
<td>java.util.Map&lt;String, String&gt;</td>
</tr>
<tr>
<td>HeaderValuesMap</td>
<td>#{headerValues}</td>
<td>@HeaderValuesMap</td>
<td>java.util.Map&lt;String, String[]&gt;</td>
</tr>
<tr>
<td>InitParameterMap</td>
<td>#{initParam}</td>
<td>@InitParameterMap</td>
<td>java.util.Map&lt;String, String&gt;</td>
</tr>
<tr>
<td>RequestParameterMap</td>
<td>#{param}</td>
<td>@RequestParameterMap</td>
<td>java.util.Map&lt;String, String&gt;</td>
</tr>
</tbody>
</table>
## JSF 2.3 Big Ticket New Features

http://arjan-tijms.omnifaces.org/p/jsf-23.html

<table>
<thead>
<tr>
<th>Artefact</th>
<th>EL name</th>
<th>Qualifier</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Request</td>
<td>#{request}</td>
<td>(Not injectable)</td>
<td><code>java.lang.Object</code></td>
</tr>
<tr>
<td>RequestMap</td>
<td>#{requestScope}</td>
<td>@RequestMap</td>
<td><code>java.util.Map&lt;String, Object&gt;</code></td>
</tr>
<tr>
<td>ResourceHandler</td>
<td>#{&quot;resource&quot;}</td>
<td>-</td>
<td><code>javax.faces.application.ResourceHandler</code></td>
</tr>
<tr>
<td>Session</td>
<td>#{session}</td>
<td>(Not injectable)</td>
<td><code>java.lang.Object</code></td>
</tr>
<tr>
<td>SessionMap</td>
<td>#{sessionScope}</td>
<td>@SessionMap</td>
<td><code>java.util.Map&lt;String, Object&gt;</code></td>
</tr>
<tr>
<td>View</td>
<td>#{view}</td>
<td>-</td>
<td><code>javax.faces.component.UIViewRoot</code></td>
</tr>
<tr>
<td>ViewMap</td>
<td>#{viewScope}</td>
<td>@ViewMap</td>
<td><code>java.util.Map&lt;String, Object&gt;</code></td>
</tr>
<tr>
<td>ExternalContext</td>
<td>#{externalContext} (new)</td>
<td>-</td>
<td><code>javax.faces.context.ExternalContext</code></td>
</tr>
</tbody>
</table>
JSF 2.3 Big Ticket New Features
WebSocket Integration

```html
<!DOCTYPE html>
<html lang="en" xmlns="http://www.w3.org/1999/xhtml" xmlns:f="http://xmlns.jcp.org/jsf/core"
         xmlns:h="http://xmlns.jcp.org/jsf/html">
<h:body>
    <h:form id="form">
        <h:commandButton id="button" value="push" action="#{pushBean.send}"
                         f:ajax />
    </h:form>
    <f:websocket channel="push" onmessage="function(message
    {document.getElementById('form:button').value=message;}" />
    <div id="user" />
    <f:websocket channel="user" user="user" onmessage="function(message)
    {document.getElementById('user').innerHTML=message;}" />
    <div id="view" />
    <f:websocket channel="view" scope="view" onmessage="function(message)
    {document.getElementById('view').innerHTML=message;}" />
</h:body></html>
```
JSF 2.3 Big Ticket New Features

WebSocket Integration

```java
@Model
public class PushBean {
    @Inject @Push
    private PushContext push;
    @Inject @Push
    private PushContext user;
    @Inject @Push
    private PushContext view;

    public void send() {
        push.send("pushed!");
        user.send("pushed!", "user");
        view.send("pushed!");
    }
}
```

```java
import javax.websocket.Endpoint;
import javax.websocket.EndpointConfig;
import javax.websocket.Session;
public class FakeEndpoint extends Endpoint {
    @Override
    public void onOpen(Session session, EndpointConfig config) {
        // https://java.net/jira/browse/WEBSOCKET_SPEC-240
    }
}
```
JSF 2.3 Big Ticket New Features

Ajax Method Invocation

<!DOCTYPE html>
<html lang="en" xmlns="http://www.w3.org/1999/xhtml" xmlns:f="http://xmlns.jcp.org/jsf/core"
xmlns:h="http://xmlns.jcp.org/jsf/html">
  <h:body>
    <h:form>
      <h:commandScript name="foo" autorun="true" action="#{spec613.setResult('foo')}"
render=":result" />
      <h:commandScript name="bar" action="#{spec613.setResult('bar')}" render=":result" />
    </h:form>
    <h:outputText id="result" value="#{spec613.result}" />
  </h:body>
</html>
JSF 2.3 Big Ticket New Features

GetEvalScripts

@Named
@RequestScoped

public class SomeBean {
  @Inject
  private FacesContext context;

  public void eval() {
    context.getPartialViewContext()
      .getEvalScripts()
      .add("alert('After response')");
  }
}
JSF 2.3 Big Ticket New Features

Class Level Bean Validation

```html
<h:form prependId="false" id="form">
    <h:panelGrid columns="2">
        <h:outputText value="Password" />
        <h:inputSecret id="password1" value="#{backingBean.password1}"/>
        <h:outputText value="Password again" />
        <h:inputSecret id="password2" value="#{backingBean.password2}"/>
        <h:commandButton id="submit" value="Submit"/>
    </h:panelGrid>
    <h:messages id="err"/>
    <f:validateWholeBean value="#{backingBean}" copierType="SerializationCopier"
        validationGroups="com.sun.faces.test.javaee8.validateWholeBean.PasswordValidationGroup"/>
</h:form>
```
JSF 2.3 Big Ticket New Features

Class Level Bean Validation

```java
@Named
@RequestScoped
@Password(groups = PasswordValidationGroup.class)
public class BackingBean implements Serializable {
    private String password1;
    private String password2;
    // ...
}

@Constraint(validatedBy = {PasswordValidator.class})
@Documented
@Target(TYPE)
@Retention(RUNTIME)
public @interface Password {
    String message() default "Password fields must match";
    Class<?>[] groups() default {};
    Class<? extends Payload>[] payload() default {};
}
```
JSF 2.3 Big Ticket New Features

Class Level Bean Validation

```java
public class PasswordValidator implements ConstraintValidator<Password, BackingBean> {

    @Override
    public void initialize(Password constraintAnnotation) {} 
    @Override
    public boolean isValid(BackingBean value, ConstraintValidatorContext context) {
        boolean result;
        result = value.getPassword1().equals(value.getPassword2());
        return result;
    }
}
```
JSF 2.3 Big Ticket New Features

Java DateTime Support

```xml
    <h:inputText id="localDateTimeValue2" value="#{backingBean.localDateTime2}" />
    <f:convertDateTime type="localDateTime" />
</h:inputText>

    <h:outputText id="localDateTimeValue2" value="#{backingBean.localDateTime2}" >
    <f:convertDateTime type="localDateTime" timeStyle="short" />
</h:outputText>

    <h:outputText value="localTime1" />

    <h:inputText id="localTime1" value="#{backingBean.localTime1}">
    <f:convertDateTime type="localTime" />
</h:inputText>

    <h:outputText id="localTimeValue1" value="#{backingBean.localTime1}" >
    <f:convertDateTime type="localTime" />
</h:outputText>

    <h:outputText value="localTime2" />
```
JSF 2.3 Big Ticket New Features

UIData and UIRepeat Improvements

• Supports Iterable