Konferenz für Java™, Enterprise Architekturen, SOA

a division of Red Hat

Seam 3

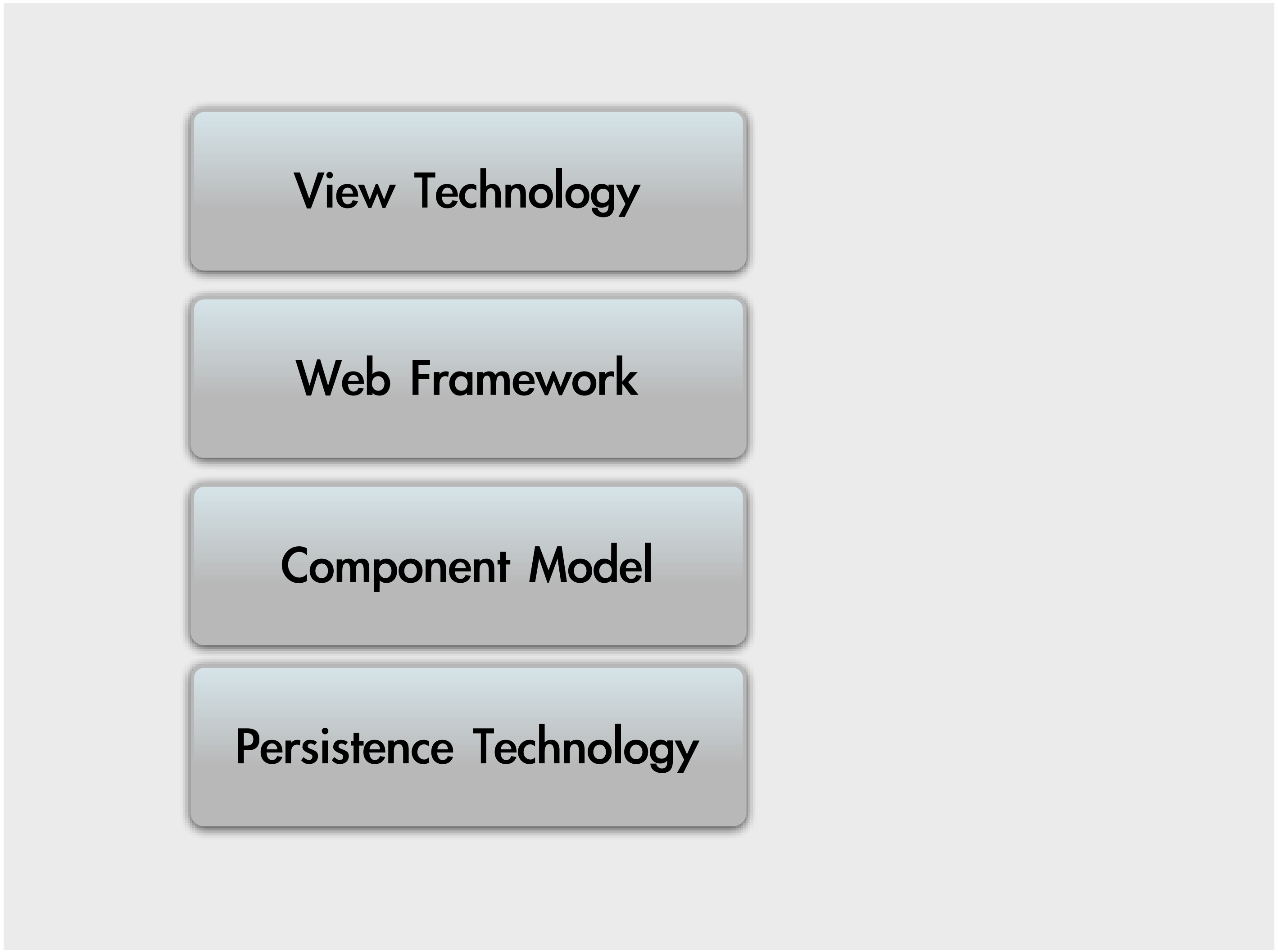
Pete Muir
JBoss, a Division of Red Hat

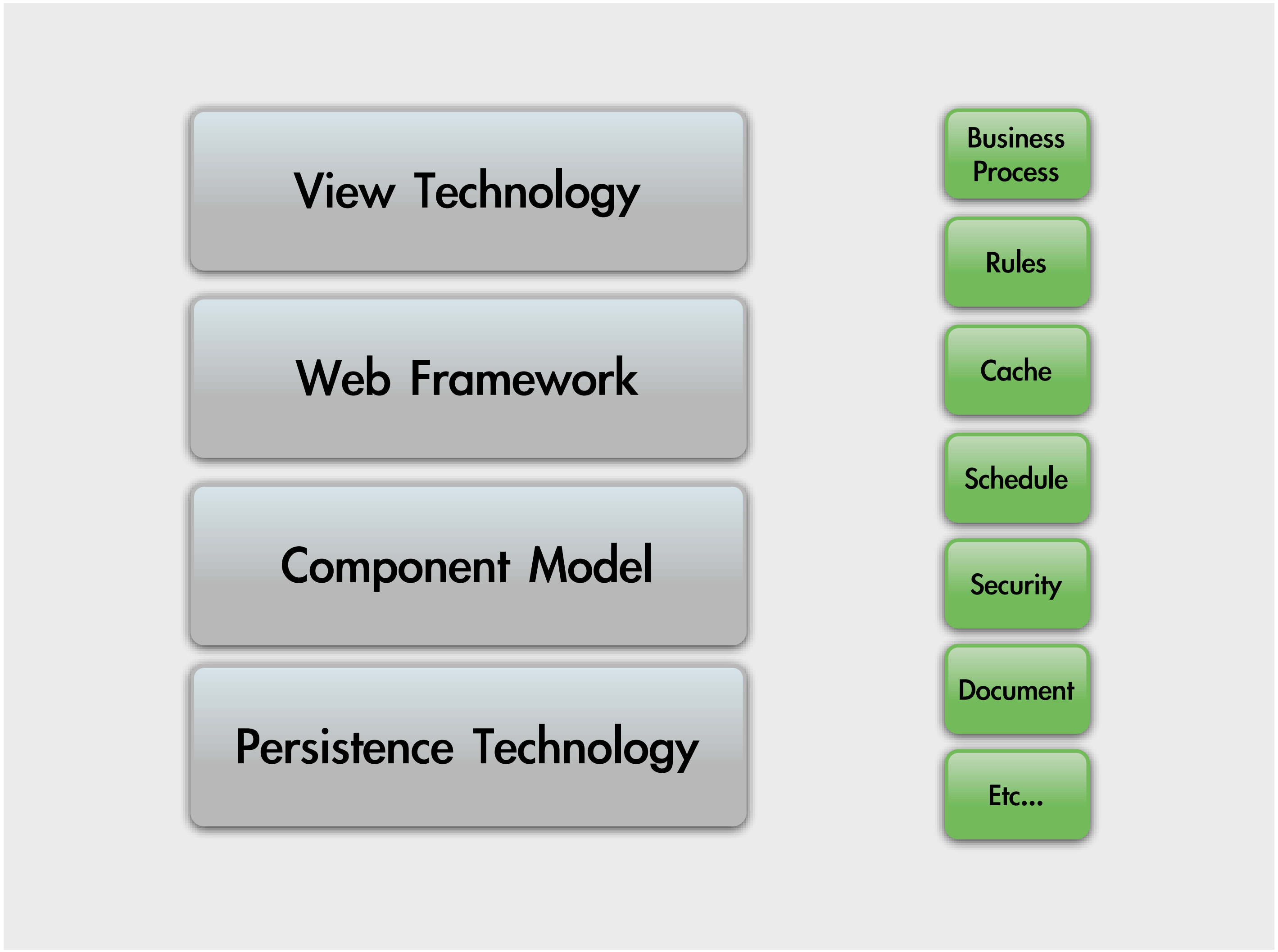
Road Map

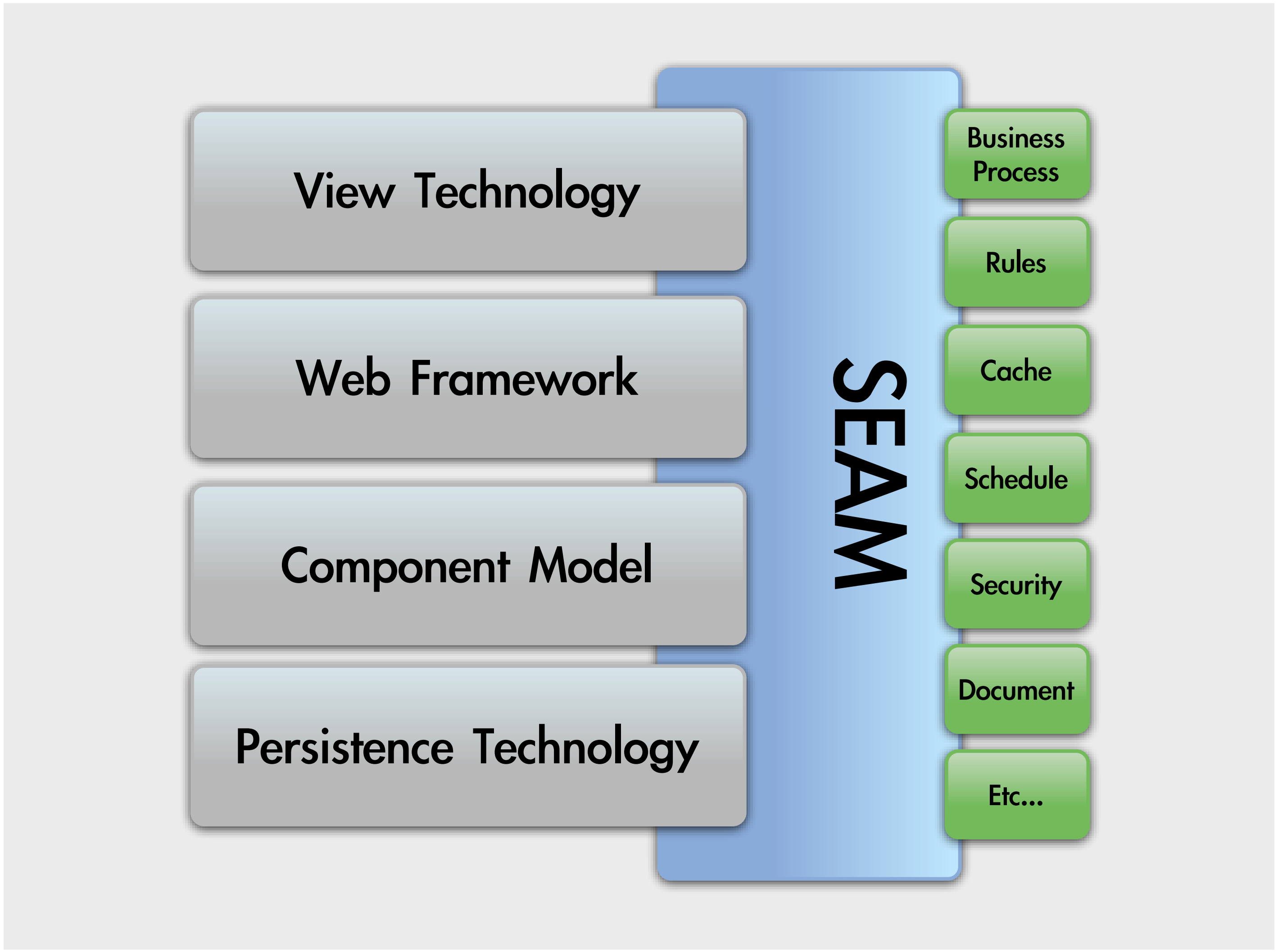
- Introduction
- Java EE 6
- Java Contexts and Dependency Injection
- Seam 3

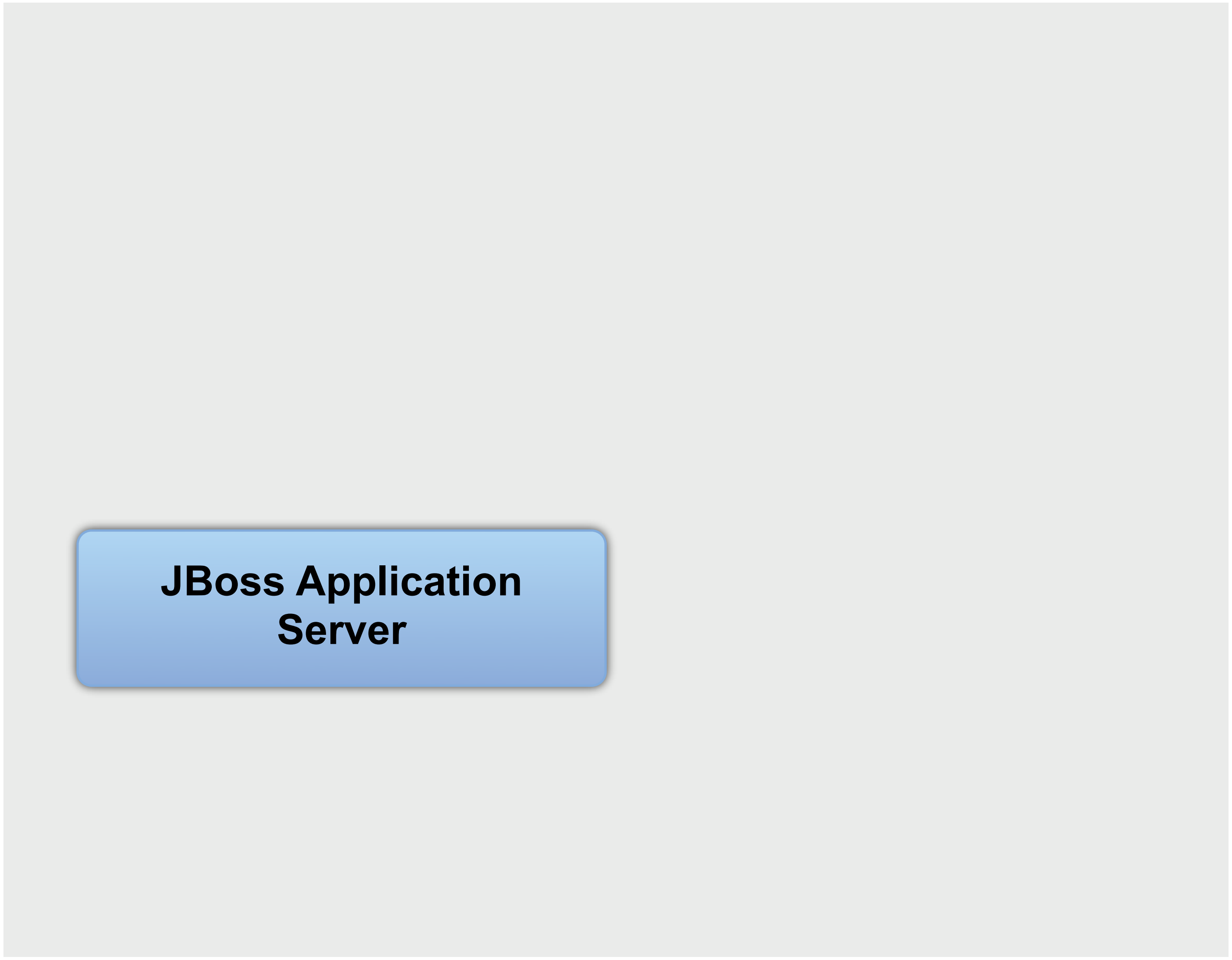
Mission Statement

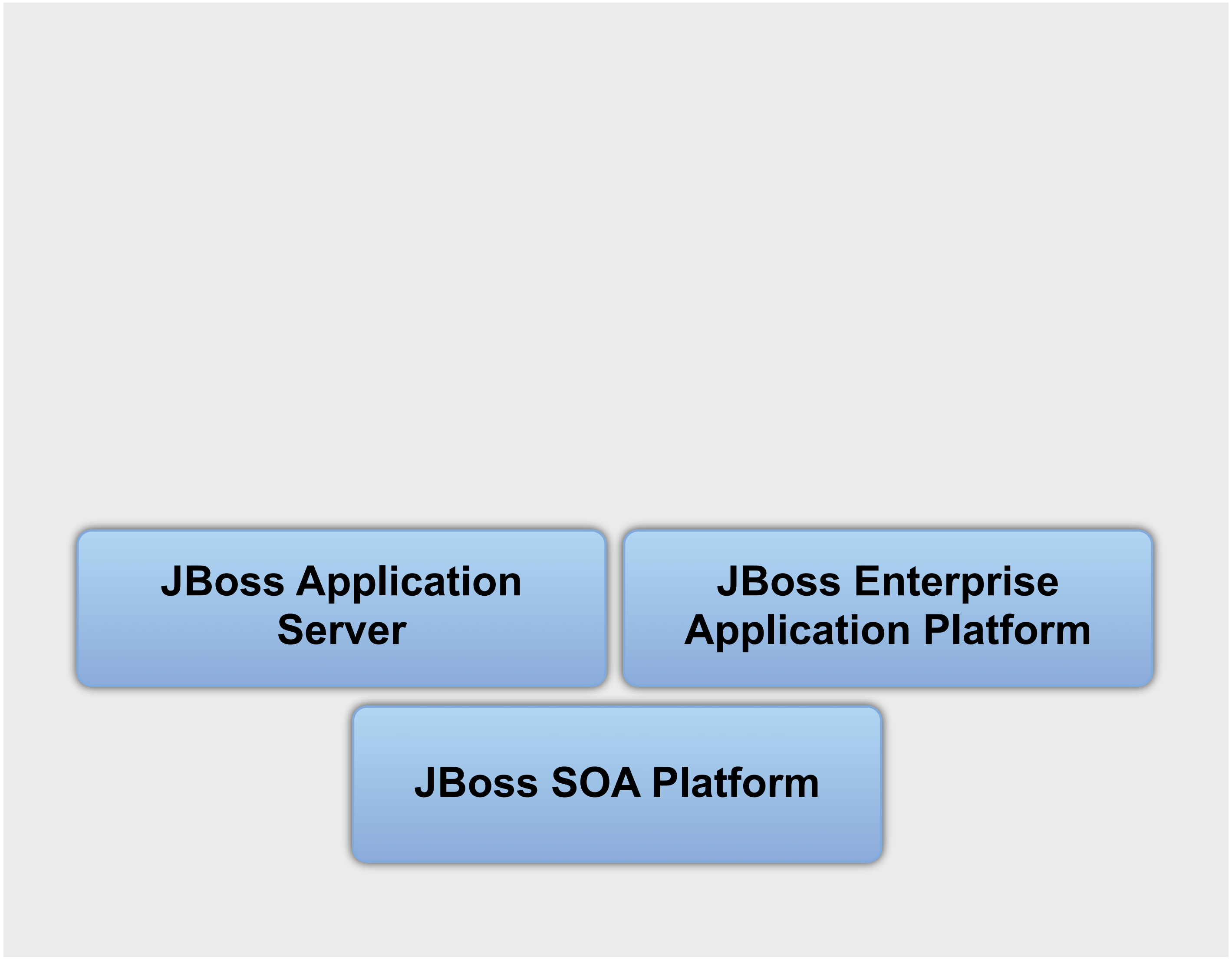
To provide a fully integrated development platform for building rich Internet applications based upon the Java EE environment.

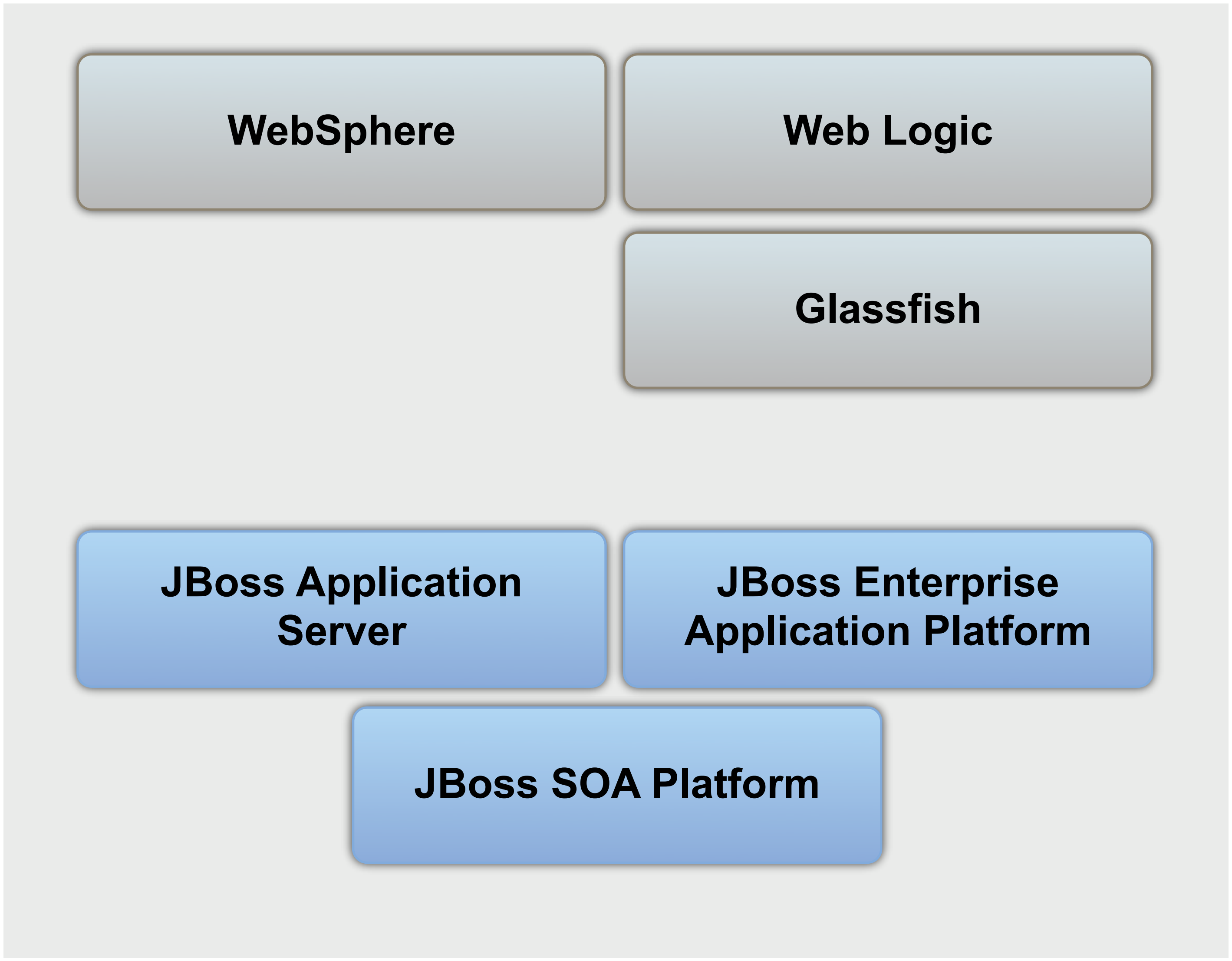

View Technology
Web Framework
Component Model
Persistence Technology

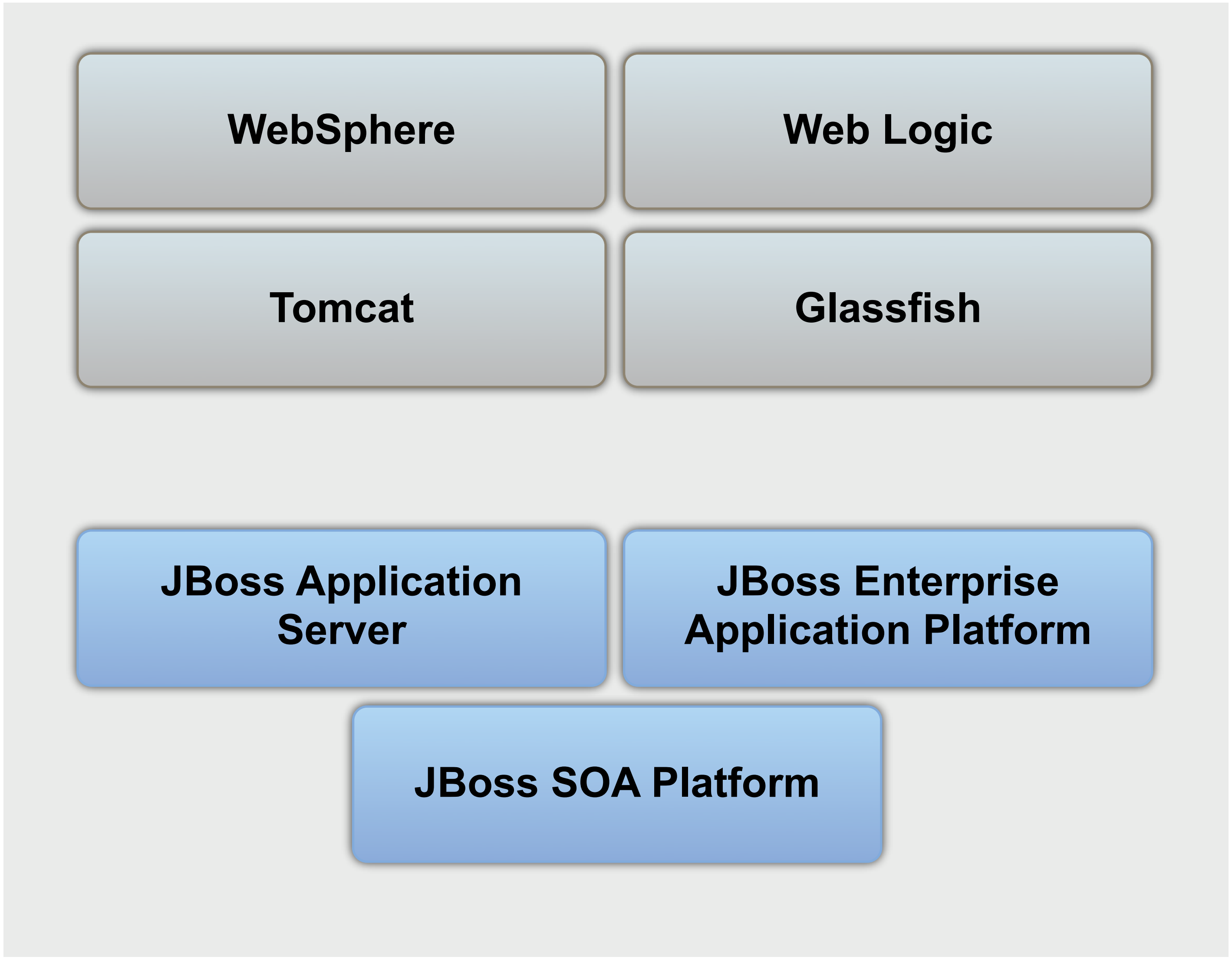

View Technology
Web Framework
Component Model
Persistence Technology
Business Process
Rules
Cache
Schedule
Security
Document
Etc...


View Technology
Web Framework
Component Model
Persistence Technology
SEAM
Business Process
Rules
Cache
Schedule
Security
Document
Etc...


JBoss Application Server


JBoss Application Server
JBoss Enterprise Application Platform
JBoss SOA Platform


WebSphere
Web Logic
Glassfish
JBoss Application Server
JBoss Enterprise Application Platform
JBoss SOA Platform


WebSphere
Web Logic
Tomcat
Glassfish
JBoss Application Server
JBoss Enterprise Application Platform
JBoss SOA Platform

The Seam Stack

- A complete solution for developing applications based upon Java EE standards
- The stack consists of...
  - Java EE integration
  - Declarative security
  - Page flows (jPDL) and business processes (jBPM)
  - JavaScript remoting or AJAX through view layer
  - Email, graphics, PDF, and XLS
  - Spring integration and more...
  - Other view layers - Wicket, Flex

Getting started...is easy!

- Focus on delivering useful functionality to users early on, then repeat
  - Project generator, component templates, reverse engineering tool (from Java classes or database)
- Eclipse tooling – JBoss Developer Studio (JBDS)
  - Key benefit of working in Java is refactoring
  - JBoss tooling takes every advantage of it
  - Forms for all configuration files
  - Visual page and navigation editors
- Incremental hot deployment

Road Map

- Introduction
- Java EE 6
- Java Contexts and Dependency Injection
- Seam 3

Web Profile

- The EE 6 web profile removes most of the “cruft” that has developed over the years
  - Full stack for building web applications
  - mainly the totally useless stuff like web services, EJB 2 entity beans, etc
  - some useful stuff like JMS is also missing, but vendors can include it if they like

JSF2

- Convention over configuration
- Facelets added to spec
  - no change to view definition
- Easy component creation
  - inspired by facelets
- View parameters
  - state held in bookmarkable URL
  - easily generate links and buttons for navigation
- Built in resource handling

JSF2

- Integrated AJAX support
  - inspired by Ajax4jsf, IceFaces
- Exception handling
  - good ootb behavior, pluggable
- New event systems
  - including declarative events
- Bean validation integration
- Project stages
  - add extra debug info automatically

EJB 3.1

- No interface views
  - great for prototyping!
- Embeddable EJB
  - for unit testing
- Singletons
- Async support
- natural timer syntax
- Portable global JNDI

And more...

- Bean Validation 1.0
  - annotation-based validation API
  - integrates cleanly with JSF2, JPA2, Spring etc. for model based validation
- JPA 2.0
  - typesafe criteria query API
  - many more O/R mapping options - no need to use Hibernate APIs :-)

JSR-299

- Spec formally known as “Web Beans”, now Java Contexts and Dependency Injection
- defines a unifying dependency injection and contextual lifecycle model
  - a new, rich, dependency management model
  - designed for use with stateful objects
  - integrates the “web” and “transactional” tiers
  - makes it much easier to build applications using JSF and EJB together
  - includes a complete SPI allowing third-party frameworks to integrate cleanly in the EE 6 environment

Road Map

- Introduction
- Java EE 6
- Java Contexts and Dependency Injection
- Seam 3

Contextual objects

- What can be injected?
  - (Almost) any Java class
  - EJB session beans
  - Objects returned by producer methods
  - Java EE resources (Datasources, JMS topics/queues, etc)
  - Persistence contexts (JPA EntityManager)
  - Web service references
  - Remote EJBs references

Loose coupling

- Events, interceptors and decorators enhance the loose-coupling that is inherent in this model:
  - event notifications decouple event producers from event consumers
  - interceptors decouple technical concerns from business logic
  - decorators allow business concerns to be compartmentalized

Essential ingrediants

- API types
- Binding annotations
- Scope
- Deployment type
- A name (optional)
- Interceptor bindings
- The implementation

Simple Example

```
public class Hello {
   public String hello(String name) {
      return "hello" + name;
   }
}
```

Any Java Bean can use these services

```
@Stateless
public class Hello {
   public String hello(String name) {
      return "hello" + name;
   }
}
```

So can EJBs

Simple Example

```
public class Printer {

   @Current Hello hello;

   public void hello() {
      System.out.println( hello.hello("world") );
   }
}
```

@Current is the default (built in) binding type

Web Bean Names

```
@Named("hello")
public class Hello {
   public String hello(String name) {
      return "hello" + name;
   }
}
```

By default not available through EL.

```
@Named
public class Hello {
   public String hello(String name) {
      return "hello" + name;
   }
}
```

If no name is specified, then a default name is used. Both these beans have the same name

JSF Page

```
<h:commandButton value=”Say Hello”
                 action=”#{hello.hello}”/>
```

Calling an action on a bean through EL

Binding Types

- A binding type is an annotation that lets a client choose between multiple implementations of an API at runtime
  - Binding types replace lookup via string-based names
  - `@Current` is the default binding type

Define a binding type

```
public
@BindingType
@Retention(RUNTIME)
@Target({TYPE, METHOD, FIELD, PARAMETER})
@interface Casual {}
```

Creating a binding type is really easy!

Using a binding type

```
@Casual
public class Hi extends Hello {
   public String hello(String name) {
      return "hi" + name;
   }
}
```

We also specify the `@Casual` binding type. If no binding type is specified on a bean, `@Current` is assumed

Using a binding type

```
public class Printer {
   @Casual Hello hello;
   public void hello() {
      System.out.println( hello.hello("JBoss") );
   }
}
```

Here we inject the `Hello` bean, and require an implementation which is bound to `@Casual`

Scopes and Contexts

- Dependent scope, @Dependent
- Built-in scopes:
  - Any servlet
    - `@ApplicationScoped`
    - `@RequestScoped`
    - `@SessionScoped`
  - JSF requests - `@ConversationScoped`
- Custom scopes

Scopes

```
@SessionScoped
public class Login {
   private User user;
   public void login() {
      user = ...;
   }
   public User getUser() { return user; }
}
```

Session scoped

Scopes

```
public class Printer {

   @Current Hello hello;
   @Current Login login;

   public void hello() {
      System.out.println(
         hello.hello( login.getUser().getName() ) );
   }
}
```

No coupling between scope and use of implementation

Producer methods

- Producer methods allow control over the production of a Web Bean where:
  - the objects to be injected are not required to be instances of Web Beans
  - the concrete type of the objects to be injected may vary at runtime
  - the objects require some custom initialization that is not performed by the Web Bean constructor

Producer methods

```java
@SessionScoped
public class Login {
   private User user;
   public void login() {
      user = ...;
   }

   @Produces
   User getUser() { return user; }
}
```

Producer methods

```
public class Printer {
   @Current Hello hello;
   @Current User user;
   public void hello() {
      System.out.println(
         hello.hello( user.getName() ) );
   }
}
```

Much better, no dependency on Login!

Java EE Resources

- To inject Java EE resources, persistence contexts, web service references, remote EJB references, etc, we use a special kind of producer field declaration:

```
public class PricesTopic {
   @Produces @Prices
   @Resource(name="java:global/env/jms/Prices")
   Topic pricesTopic;
}
```

```
public class UserDatabasePersistenceContext {
   @Produces @UserDatabase
   @PersistenceContext
   EntityManager userDatabase;
}
```

Road Map

- Introduction
- Java EE 6
- Java Contexts and Dependency Injection
- Seam 3

Key themes

- Loose coupling
  - pick and choose the modules you want
- Portability
  - run the modules in **any** JCDI environment
- Full stack
  - all core modules tested together, with examples showing how to use
- Sandbox for new module ideas

Environments

- JSR-299 is targeted at
  - Java EE
  - Java EE web profile
  - Use with EJB 3.1 Embeddable
- Web Beans add supports for
  - Servlet containers like Tomcat or Jetty
  - Java SE
-

Integrations

- Other DI frameworks
  - Seam 2 bridge
  - Spring bridge
  - Guice bridge
- A Seam 2 native layer for supporting Seam style dependency injection
- View layers
  - JCDI requires JSF support
  - Wicket
  - Flex and other planned

Portable modules

- can be used in any JSR-299 environment
  - jBPM
  - log injection (choose between log4j and jlr, parameter interpolation
  - Seam Security
  - Reporting
    - Excel
    - PDF
  - Mail
  - Javascript remoting

Web Beans

- JSR-299 forms the core of Seam
- Web Beans is the Reference implementation
  - Currently feature complete preview
- Download it, try it out, give feedback!
- Supported in this release:
  - JBoss 5.1.CR1
  - GlassFish V3 build 46
  - Tomcat 6.0.x
  - Jetty 6.1.x

Seam 3

- Just getting started!

Q & A

http://in.relation.to/Bloggers/Pete

http://www.seamframework.org/