Space Details

Key:	VAAP
Name:	901022 Video Annotation Analysis and Presentation
Description:	
Creator (Creation Date):	brian (Feb 05, 2010)
Last Modifier (Mod. Date):	brian (Feb 05, 2010)

Available Pages

- Home
  - Computers
    - Alaskanwind
    - Dione
    - Eione
    - Seadog
    - Ulikai
  - Data Analysis Framework
  - Data Products
    - Artifact Relationships
  - Migrating to Equinox
  - Web Application Technologies
    - JEE6 CDI
    - JEE6 EJB3
    - JEE6 JPA2.0
    - JEE6 Servlet3.0

This page last changed on Feb 14, 2012 by brian.

901022 - Video Annotation Analysis and Presentation (VAAP)

Infrastructure

- Source Code
  - Wiki
- Computers

Notes

- Web Application Technologies
- Migrating to Equinox
- Data Products
- Data Analysis Framework

Computers

This page last changed on Mar 06, 2012 by brian.

Computers that are involved with running DSG at MBARI

Computer	Role
Alaskanwind	Western Flyer database server
Dione	External database server
Eione	External web server
Equinox	*Master database server
Seadog	Western Flyer Web and Web application server
Seaspray	Internal Web and Web application server
Ulikai	Weekly processing of data products

Alaskanwind

This page last changed on Jan 10, 2012 by brian.

About alaskanwind.wf.mbari.org

Alaskanwind hosts the MS SQL Server 2008 with the following databases:

Database	Role
DSG	DSG data
DSG_KB	The VARS_KB database that's been copied and has URL's modified to point to local references

DSG_KB info

The DSG_KB database is a copy of the local VARS_KB with the following SQL executed after copy:

```
UPDATE
Media
SET
url = REPLACE(url,  'http://dsg.mbari.org/images/dsg', 'http://seadog.wf.mbari.org:8082/www/vars/knowledgebase/images')
WHERE
url IS NOT NULL
GO
```

DSG info

The DSG database is a copy of DSG from Equinox with the following SQL applied:

```
UPDATE
URLArtifact
SET
url = REPLACE(url,  'http://seaspray.shore.mbari.org/dsg', 'http://seadog.wf.mbari.org:8082/www/dsg')
WHERE
url IS NOT NULL
GO
```

More Info

For notes on the setup of the DSG on Alaskanwind see https://oceana.mbari.org/jira/browse/VAAP-33

Dione

This page last changed on Mar 06, 2012 by brian.

About Dione

Dione (dione.mbari.org) is a Windows VM that hosts the eternal (DMZ) SQL Server 2008 instance. This server hosts all the external VARS databases as well as SSDS.

Eione

This page last changed on Sep 10, 2012 by brian.

About Eione

> **Important**
>
> Eione is aliased with the CNAME *dsg.mbari.org*

Eione is a linux VM whose sole purpose is to host an apache web server and a tomcat application server in the DMZ. It's hostname is actually *eione.mbari.org*. Content for Eione should be placed in `/var/www/website`, which is pretty much the only directory non-root users can write to. The following directories are of special interest:

/var/www/website/frameGrabs

This directory contains water-marked annotation images. All non-embargoed images are watermarked and copied here.

/var/www/website/images/dsg

This is a symlink to `/mbari/dsg/kbimages` which, in-turn, is the share \\Atlas\VideoLab\VARS_Images mounted on Eione.

Tomcat and the Deep Sea Guide

The Deep Sea Guide is hosted on tomcat. Notes on the tomcat installation can be found at https://oceana.mbari.org/jira/browse/VAAP-42

Seadog

This page last changed on Feb 09, 2012 by brian.

About seadog.wf.mbari.org

Seadog acts as the web application server for the Deep-sea Guide on the Western Flyer. All content is servered via an instance of Tomcat. All DSG content and applications exist in `/opt/dsg`.

Tomcat

Tomcat resides in `/opt/dsg/tomcat/apache-tomcat-7.0.23` and is visible via port 8082. It has been configured to run on non-default ports so that it does not interact with another Tomcat on Seadog that is running JIRA on the default ports. This instance of Tomcat has been modified to serve static DSG content stored in `/opt/dsg/www`.

"Configure Tomcat to Serve Static Files

Suppose you have a /var/project/images directory which stores a number of images for your project. If you want this directory to be exposed through http protocol, all you have to do is to add a <Context> parameter to the <Host> section of Tomcat's server.xml:

```
<Server port="8025" shutdown="SHUTDOWN">
...
<Service name="Catalina">
...
<Engine defaultHost="localhost" name="Catalina">
...
<Host appBase="webapps"
autoDeploy="false" name="localhost" unpackWARs="true"
xmlNamespaceAware="false" xmlValidation="false">
...
<Context
docBase="/var/project/images"
path="/project/images" />
</Host>
</Engine>
</Service>
</Server>
```

Note that docBase parameter is set to the path on hard drive and path parameter is set to the context path for your files. Now, if you have a /var/project/images/sample.png file, it can be seen at http://localhost:8084/project/images/sample.png.

Source

The startup script is located at `/etc/init.d/tomcat-dsg`. It is run under the user *tomcat*. Files added under `/opt/dsg` should all belong to the group *tomcat*!!. Brian Schlining has sudo permissions on `/sbin/service` and `/sbin/chkservice`. To restart tomcat run:

```
sudo /sbin/service tomcat-dsg restart
```

Configure Tomcat *Manager* application to handle large files

When I attempted to upload the `dsg.war` I got the following error:

```
org.apache.tomcat.util.http.fileupload.FileUploadBase$SizeLimitExceededException:
the request was rejected because its size (53887121) exceeds the configured maximum
(52428800)
```

By default the Manager app only allows WARS upto 50MB to be uploaded; the dsg.war was just slightly larger. To resolve this, I edited `\webapps\manager\WEB-INF\web.xml` and changed the max-file-size and max-request-size to something larger. The block to be edited looks like:

```
    <multipart-config>
    <!-- 52MB max -->
    <max-file-size>52428800</max-file-size>
    <max-request-size>52428800</max-request-size>
    <file-size-threshold>0</file-size-threshold>
    </multipart-config>
```

Static Content

Static content was copied as follows:

1. **Knowledgebase Images**

   ```
   ssh seadog.wf.mbari.org
   cd /opt/dsg/www/vars/knowledgebase
   scp -r brian@eione.mbari.org:/var/www/website/images/dsg images
   ```
2. **Deep-sea Guide Artifacts**

   ```
   ssh seadog.wf.mbari.org
   cd /opt/dsg/www/dsg
   scp -r brian@seaspray.shore.mbari.org:/var/www/html/dsg/artifacts artifacts
   ```

Database Replication

Replication from equinox.shore.mbari.org to alaskanwind.wf.mbari.org is run nightly. The following SQL is executed after replication:

```
-- For DSG_KB run the following:

UPDATE
Media
SET
url = REPLACE(url, 'http://seaspray.shore.mbari.org/dsg',
'http://seadog.wf.mbari.org:8082/www/vars/knowledgebase/images')
WHERE
url IS NOT NULL


-- For DSG run the following:

UPDATE
URLArtifact
SET
url = REPLACE(url, 'http://seaspray.shore.mbari.org/dsg',
'http://seadog.wf.mbari.org:8082/www/dsg')
WHERE
url IS NOT NULL
```

Ulikai

This page last changed on Mar 06, 2012 by brian.

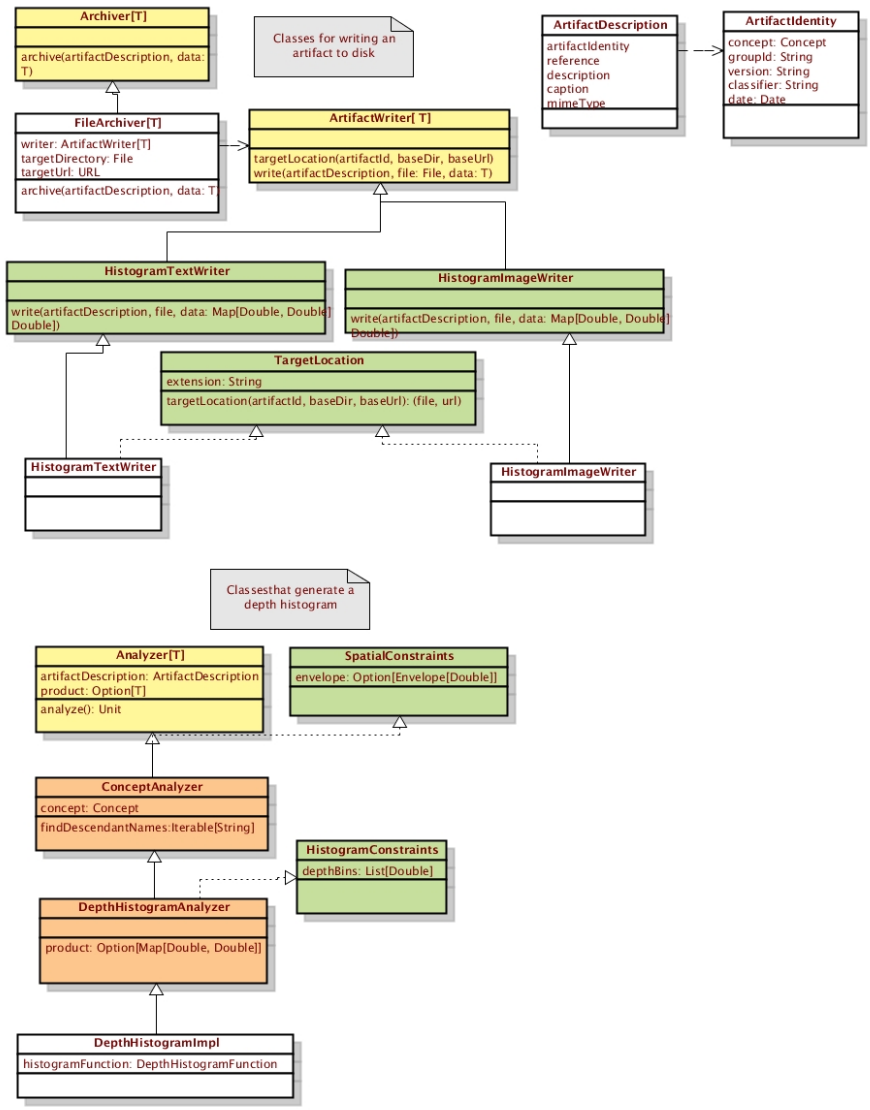
About Ulikai

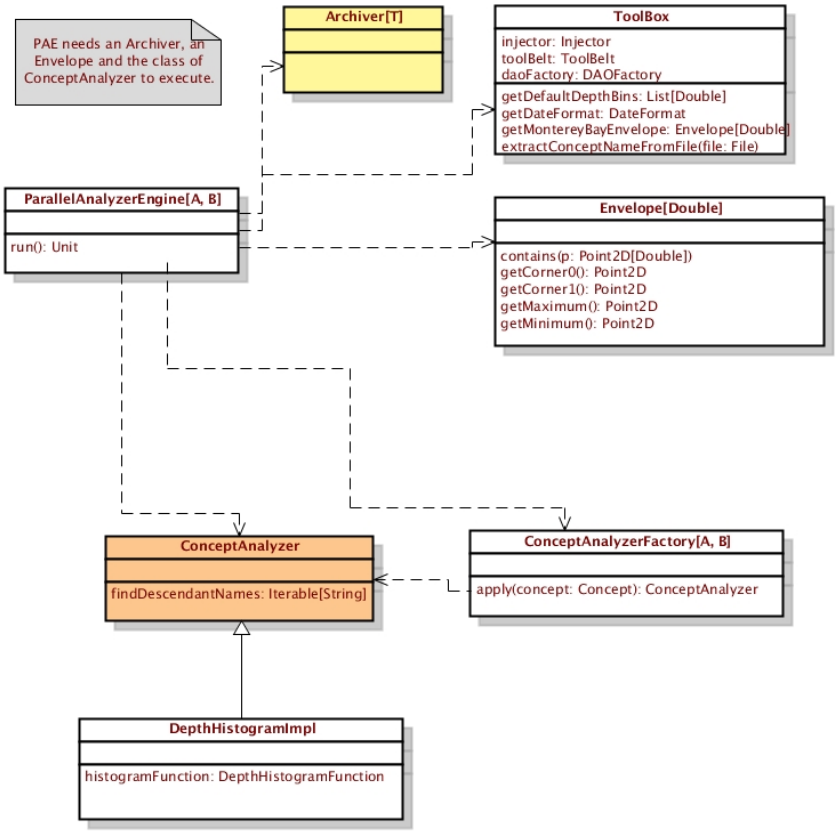
This VM is running RHEL 6.2. It has 12GB of RAM available and 4 CPU's. Essentially, it's a stock linux VM with JDK7 installed. The weekly processing is run as the user 'brian'. Data products should be written to \\Atlas\DeepSeaGuide which can be mounted as:

OS	mount path
Windows	\\atlas\DeepSeaGuide
Macs	cifs://atlas/DeepSeaGuide
NFS	atlasnfs:/ifs/mbarisnaponly/DeepSeaGuide
NIS	/mbari/DeepSeaGuide

Data Analysis Framework

This page last changed on Sep 28, 2010 by brian.




PAE needs an Archiver, an Envelope and the class of ConceptAnalyzer to execute.
Archiver[T]
ToolBox
injector: Injector
toolBelt: ToolBelt
daoFactory: DAOFactory
getDefaultDepthBins: List[Double]
getDateFormat: DateFormat
getMontereyBayEnvelope: Envelope[Double]
extractConceptNameFromFile(file: File)
ParallelAnalyzerEngine[A, B]
run(): Unit
Envelope[Double]
contains(p: Point2D[Double])
getCorner0(): Point2D
getCorner1(): Point2D
getMaximum(): Point2D
getMinimum(): Point2D
ConceptAnalyzer
findDescendantNames: Iterable[String]
ConceptAnalyzerFactory[A, B]
apply(concept: Concept): ConceptAnalyzer
DepthHistogramImpl
histogramFunction: DepthHistogramFunction

Data Products

This page last changed on Jul 13, 2010 by brian.

Overview

Brian Schlining sent out this message on January 4th 2010:

> Happy new year everyone,
> With the start of the new year, I'm kicking off the work for the Video Annotation and Presentation project (VAAP) (see https://mww.mbari.org/resources/2010_Proposal_Process/phase_I_pdfs/901022_Video_Annotation_Analysis_and_Presentation.pdf)
> As per management request, my first task is to figure out what VARS-derived data products are useful for your 'Biodiversity Initiative' work. Since the original idea of the VAAP project was to use techniques that we've already developed, here's the grocery list of work-that-has-been-done-before:
> - Separating benthic annotations from midwater annotations
> - Generating grids (e.g. maps) from annotations that are useful for spatial analysis
> - Creating vertical distributions (i.e. both altitude and depth, either aggregated or as a timeseries)
> - Normalizing annotation data base on ROV effort (very roughly normalized)
>
> We can slice-and-dice the annotations in any way you see fit. Some examples are:
> - by taxa
> - by functional group (if you provide me with a list of organisms in the group you are interested in)
> - by spatial bounds
>
> As a first cut, it might be best to generate products that give 'the big picture'. In my mind the big picture is:
> - what is the horizontal and vertical spatial distribution of the critters (and for Charlie, the rocks 
>   -Do these distributions change over time.
>
> My naive suggestions for a first cut of data products would be:
> **1) Spatial grids of benthic annotations by taxa or functional groups.**
> a) An aggregate grid for each taxa/group spanning all annotations within Monterey Bay. This would provide an overview of where a taxa/group has been found
> b) Grids for each taxa/group representing discrete time periods (for example, one grid for each year or grids for each oceanographic season). This would give some indication of change in the community structure.
> **2) Vertical distributions of taxa/groups**
> a) Aggregate vertical distributions (i.e. annotations from any time)
> b) vertical distributions for discrete time periods (again, either yearly or by oceanographic season)
> c) Same as a and b above but normalized by ROV effort
> **3) A time series product, perhaps vertical distribution vs time?**
> Please let me know if these products are useful for you. If you have other ideas please let me know. Again, these just represent a starting point; other data products can and will be added later as you determine what questions you're interested in examining for the Biodiversity Initiative.

No scientist responded to the email.

Product relations

Artifact Relationships

Tentative Products

Aggregated over entire data series from surface to 4000m with Monterey Bay

What spatial bounds do scientists want?

EXPD

1. 1m Histogram of EXPD data as text file
2. 1m Histogram of EXPD data as plot
3. 1m Histogram of EXPD normalized by total counts as text file
4. 1m Histogram of EXPD normalized by total counts as plot

VARS

The following will use a *Concept* and all it's children for an analysis. So for example, an analysis of *Squid* would be an aggregate analysis of all squid.

1. 1m Histogram of VARS data for each concept as text file
2. 1m Histogram of VARS data for each concept as plot

Normalized products

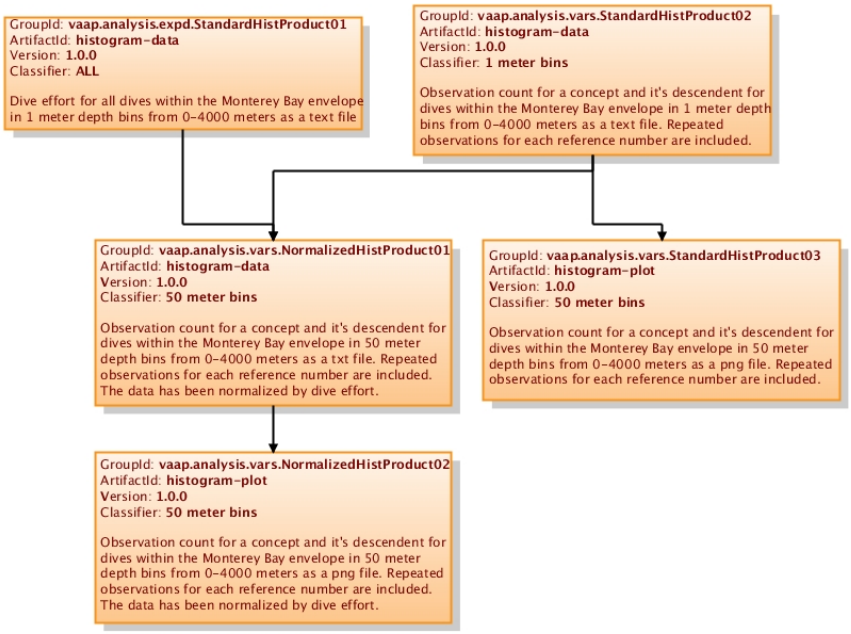
1. Normalized vertical distribution
    a. 1st normalize the VARS distribution by total counts
    b. Then normalize by dive effort (using normalized EXPD vertical distribution)

Artifact Relationships

This page last changed on Jul 13, 2010 by brian.

Overview

I'll be storing references to the various artifacts in the VARS Knowledgebase. Here's how the standard products reference each other...



Migrating to Equinox

This page last changed on Apr 05, 2010 by brian.

THe migration of VARS from SQL Server 2000 on Solstice to SQL Server 2008 on Equinox is documented at https://oceana.mbari.org/jira/browse/VAAP-1

This page last changed on Apr 05, 2010 by brian.

Web Application Technologies

VAAP will be making extensive use of the vars API hosted at http://code.google.com/p/vars-redux/. Since this is a pure Java API, the Web Application stack will also be written in Java to leverage as much as existing code as possible. In this wiki page, I'll be taking notes to help define the web technologies to use when developing VAAP.

JEE 6

JEE6 EJB3
JEE6 Servlet3.0
JEE6 JPA2.0
JEE6 CDI

Grails

Wicket

I've been working through the Wicket in Action book. Since there is a very nice separation between the presentation layer and the Java code, this would allow me to pull in external designers to work on this. I'm going to use wicket as the presentation layer.

This page last changed on Feb 05, 2010 by brian.

Context and Dependency Injection API

- Basics `@Inject @LoggedIn User user` user = what, LoggedIn = which one (not string based!)
- Can coexist with `@Resource` and `@PostConstruct`
- Can inject fields, methods or constructors
  - Constructor (My prefered type) Example:

```
public class CheckoutHandler {
@Inject
CheckoutHandler(@LoggedIn User user,
PaymentProcessor processor,
@Default Card cart) {
// ...
}
}

// Multiple qualifer example
public class CheckoutHandler {
@Inject
CheckoutHandler(@LoggedIn User user,
@Reliable
@PayBy(CREDIT_CARS)
PaymentProcessor processor,
@Default Card cart) {
// ...
}
}
```

- Easy to write your own qualifiers. Just create an annotation and annotate it with `@Qualifier`:

```
@Qualifer
@Retention(RUNTIME)
@Target({FIELD, TYPE})
public @interface Red{}
```

- ManagedBean
  - Anything injected is a *bean*. Examples include:
    - EJB Session beans
    - Plain classes with `@ManagedBean`
    - Any class CDI can discover in a module
    - Example:

```
@Reliable
@PayBy(CREDIT_CARD)
@ManagedBean
public class ReliableCreditCardPaymentProcessor implements PaymentProcessor {

void pay(Amount amount) throws PaymentException {
...
}
}
```

    - Same example as an EJB:

```
@Reliable
@PayBy(CREDIT_CARD)
//@ManagedBean
@Stateless
public class ReliableCreditCardPaymentProcessor implements PaymentProcessor {

void pay(Amount amount) throws PaymentException {
...
}
```

```
}
```

- Same example as a Request-scoped Bean:

```
@Reliable
@PayBy(CREDIT_CARD)
@RequestScoped
@ManagedBean
public class ReliableCreditCardPaymentProcessor implements PaymentProcessor {

void pay(Amount amount) throws PaymentException {
...
}
}
```

    - Scopes allowed everywhere: `@ApplicationScoped, @RequestScoped`
    - Also allowed in web apps: `@SessionScoped`

- Named beans. (Built in support for Unified EL)
    - You can name a bean with `@Named("cart")`. The refer to it in a JSP page using the EL:

```
<h.commandButton value="Checkout" action="#{cart.checkout}">
```

- Events with annotation-based event model
    - A bean `@Observes` an event:

```
void onLogin(@Observes LoginEvent event) { ...}
```

      Another bean fires an event using the `Event.fire(T event)` method

JEE6 EJB3

This page last changed on Feb 05, 2010 by brian.

EJB 3

Notes from https://www.sun.com/offers/details/java_ee6_javabeans.xml

- EJB allows 'interface-less' session beans.
- EJB's can now be packaged in WAR files. EAR's are not required
- Profiles
  - Web profile: Local Session Beans, CMT/MBT, Declarative Security, Interceptors
  - Full profile: Lite + Message-Driven Beans, Timer Servie, Async method calls, CMP/BMP Entity Beans
- Portable JNDI names:
  - Globally unique name:

    ```
    java:global[/<app-name>]/<module-name>/<ejb-name>
    ```
  - Unique name within same application:

    ```
    java:app/<module-name>/<ejb-name>
    ```
  - Unique name within defining module:

    ```
    java:module/<ejb-name>
    ```
  - Example:

    ```
    @Stateless
    public class HelloBean implements Hello {
    public String sayHello(String msg) {
    return "Hello, " + msg;
    }
    }
    ```

    If deployed as hello.jar, JNDI entries are:

    ```
    java:global/hello/HelloBean
    java:app/hello/HelloBean
    java:module/HelloBean
    ```
- Embeddable EJBContainer for testing or JSE use:

```
EJBContainer c = EJBContainer.createEJBContainer();
Bank bank = (Bank) c.getContext().lookup("java:global/bank/BankBean");
// ...
c.close();
```

- New Features
  - Singletongs
    - Allows shared state
    - Designed for concurrent access
    - Lots in common with stateless/stateful beans
    - Example:

      ```
      // EJB
      @Singleton
      // @Startup // optionally create this bean eagerly
      public class SharedBean {
      private SharedData shared;

      // Note we can do things during construction and teardown (optional)
      @PostConstruct
      private void init() {
      shared = ...;
      }

      @PreDestroy
      private void destroy() {
      ...
      ```

```
}

public int getXYZ();
return shared.xyz;
}


}

// Client
@Stateless
public class FooBean {
// Inject reference to Singleton bean
@EJB
private SharedBean shared

public void foo() {
int xyz = shared.getXYZ();
...
}
}
```

    - Concurrecy options: Single threaded (default), Container Managed (via method-level locking metadata), Bean Managed (all concurrent invocation can access bean instance)
- Improvements to EJB timer service.
    - richer calendar-base way to express timer expressions (like cron, but better)
    - Automatic Timer Creation:

```
@Stateless
public class BankBean {

@PersistenceContext EntityManager accountDB;
@Resource javax.mail.Session mailSession;

// Callback the last day of each month at 8 a.m.
@Schedule(hour="8", dayOfMonth="Last")
void sendMonthlyBankStatements() {
...
}
}
```

- Asynchrounouse Session Bean Invocations
    - Fire and Forget or async results via `Future<V>`
    - Available for Stateful, Stateless and Singleton beans
    - Example of Fire and Forget:

```
@Stateless public class DocBean {

@PersistenceContext EntityManager resultsDB;
@EJB DocBean myself;

public void processDocument(Document doc) {
myself.doAnalysisA(document);
myself.doAnalysisB(document);
}

@Asynchronous public void doAnalysisA(Document d) { //... }
@Asynchronous public void doAnalysisB(Document d) { //... }
}
```

    - Example of Future:

```
@EJB Processor proc;
Task task = new Task(...);
Future<int> computeTask = processor.compute(task);
...
int result = computeTask.get();

// EJB
```

```
@Stateless
public class ProcessorBean implements Processor {
@PersistenceContext EntityManager db;

@Asynchronous
public Future<int> compute(Task t) {
// perform computation
int result = ...;
return new javax.ejb.AsynchResult<int>(result);
}
}
```

This page last changed on Feb 05, 2010 by brian.

JEE JPA 2.0

Notes from

- Automatic Orphan Deletion

```
@Entity public class Order {
@Id int orderId;
...
@OneToMany(cascade=PERSIST, orphanRemoval=true)
Set<Item> lineItems;
}
```

- Validation
  - Automatic validation upon lifecycle events: PrePersist, PreUpdate, PreRemove
  - Validation-mode element in `persistence.xml`: **AUTO, CALLBACK, NONE**
  - Example:

```
@Entity public class Employee {
@Id Integer id
String name;
@Max(15) Integer vacationDays;
@Valid Address worksite;
// ...
}

@Embeddable public class Address {
@Size(max=30) String street;
@Size(max=20) String city;
@Zipcode String zipcode;
// ...
}
```

This page last changed on Feb 05, 2010 by brian.

JEE6 Servlet 3.0

Notes from https://www.sun.com/offers/details/java_ee6_servlet.xml

- Added annotations for declarative programming and generics:

```
/*
You can use web.xml to override values
*/
@WebServlet // defines a servlet
@WebFilter // defines a filter
@WebListener // defines a listener
@WebInitParam // defines an init param
@ServletSecurity // security constraints
@MultipartConfig // file upload
```

- `@WebServlet`
  - MUST specifiy url mapping
  - Default name of servlet is full qualified class name
  - Must still extend `HttpServlet`
  - Example:

```
package com.foo;
@WebServlet(name="MyServlet", urlPattern="/myApp/*")
public class MyServlet extends HttpServlet {
public void doGet(HttpServletRequest req, HttpServletResponse res) { ... }
}
```

- Enable use of frameworks without alot of boiler plate configuration. Modularize web.xml
  - Dynamic registration of Servlets and Filters
    - Performed during `ServletContext` intialization;
      - `ServletContext.add<Servlet|Filter>`
      - `ServletContext.create<Servlet|Filter>`
      - `ServletContext.find<Servlet|Filter>`
  - Example:

```
// Registration
ServletRegistration.Dynamic dynamic =
servletContext.addServlet("DynamicServlet", "com.foo.MyServlet");
dynamic.addMapping("/dynamicServlet");
dynamic.setAsyncSupported(true);

// Lookup
ServletRegistration declared = servletContext.getServletRegistration("DeclaredServlet");
declared.addMapping("/declaredServlet");
declared.setInitParameter("param", "value");
```

- Modularized web.xml
  - `web-fragment.xml`
  - bundled in framework jar fil in `META-INF`
  - Almost identical to `web.xml`
  - Only jars in `WEB-INF/lib` will be scanned for `web-fragment.xml` files
  - Example:

```
<web-fragment>
<servlet>
<servlet-name>welcome</servlet-name>
...
</web-fragment>
```

  - Ordering is from JSF.
    - `web.xml` may declare ordering of fragments via `<absolute-ordering>`
    - fragments can specify ordering with `<before>` and `<after>`
- `ServletContainerInitializer`
  - Expresses interest in classes via @HandleTypes

    - Container scans webapp for classes that match `@HandleTypes` and passes them to `ServletContainerIntializer`
- Resource sharing
    - Static (e.g. images) and JSP resources no longer confined to document root of web application
    - May be place in `WEB-INF/lib/<*.jar>/META-INF/resources`
    - Resources in document root take precedence over those in bundled jars
- Asynch support
    - Must declare `@WebServlet(asyncSupported=true)`
    - Then call `AsyncContext ctx = ServletRequest.startAsync(req, res)`
    - `AsyncContext` can then either:

      ```
      dispatch(String path)
      start(Runnable action)
      ```
    - Then paired with `complete()`
    - `ServletRequest.isAsyncSupported()`
        - True if **all** <filter|servlet> support async in the filter chain **and** the request dispatch chain
        - Can be configured in `web.xml`, Annotation, or API
- Security
    - Added support for:

      ```
      // Declared using https contraints with @ServletSecurity
      HttpServletRequest.authenticate
      HttpServletRequest.login
      HttpServletRequest.logout
      ```