@ENTERPRISE

Application Development with @enterprise 11

Groiss Informatics GmbH • 2023

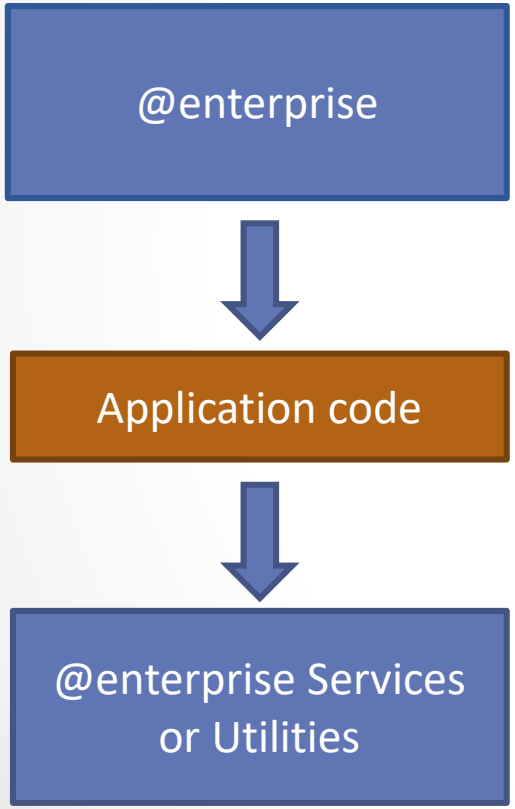
@enterprise – Course material, May 2023

Contents

1. Introduction

@enterprise provides a lot of functions and services. Many of them can be customized or enhanced by application code.

API programs are pieces of code embedded in the @enterprise environment:



An @enterprise component calls the application code, for example: Web-Server, TimerManager, Worklist, Engine.

Application code for some specific business need:

implement a servlet method, timer, worklist adaption, etc. and call

@enterprise Services, like DMS, Workflow engine, store functions, etc.

Structure of the slides

- How to embed the code
  - next slide and section Setting up a project (2)
- How the application is organized
  - Structure of Applications (6)
- What services
  - Servlet methods (3)
  - Persistence (4)
  - Organizational Data (7)
  - Workflow engine (8,9)
  - Document Management (10)
  - User Interface (11)
  - Communication (12)
  - Reporting (14)
- and utilities are available
  - Utilities and Data Structures (5), client side (13), and throughout the slides

Calling application code

Application code is called by @enterprise using Java interfaces or Java reflection. In these slides the interfaces are described using the following structure:

Description: purpose of the API

Definition: Interface definition or, if called by reflection, signature of method

Declaration: where to declare: either in the system configuration, a field in a database object (process, task,...) or in a configuration file.

In Product: Implementation of the API in product, default implementation

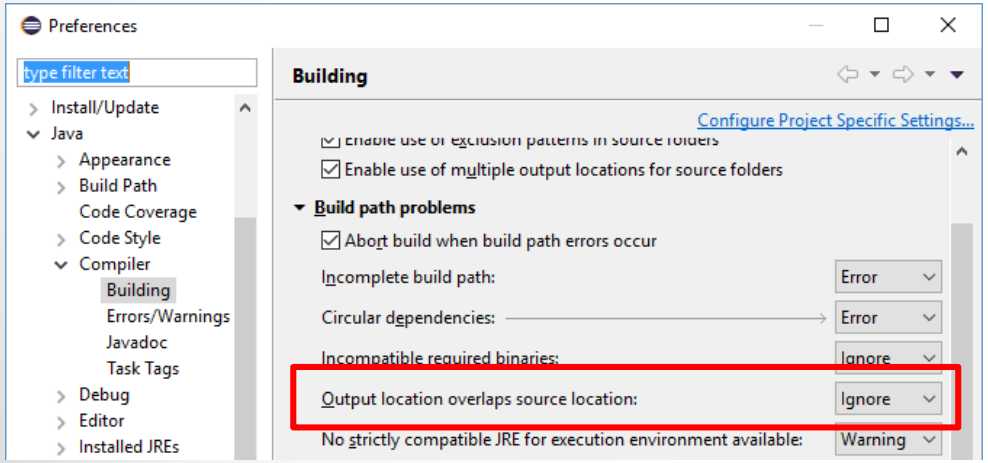
Demos: Implementation in demo package

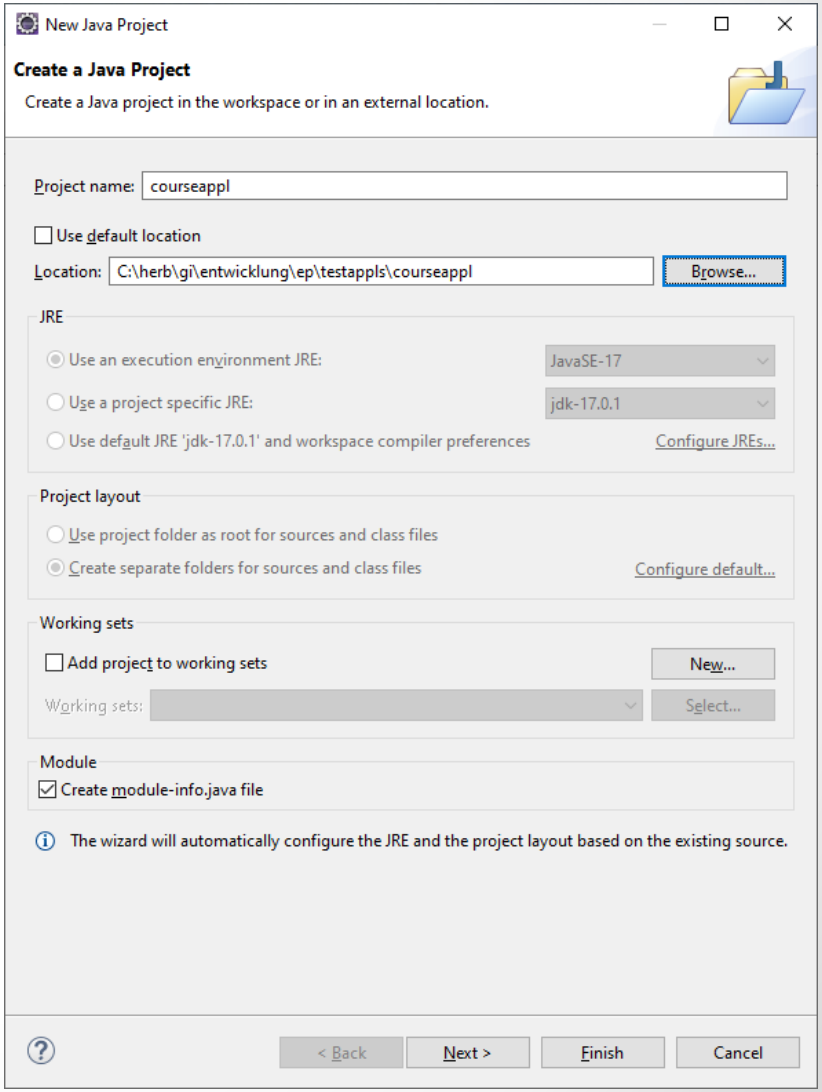
Example: a short example in the slides

2. Setting up a Project

How to run you own code in @enterprise

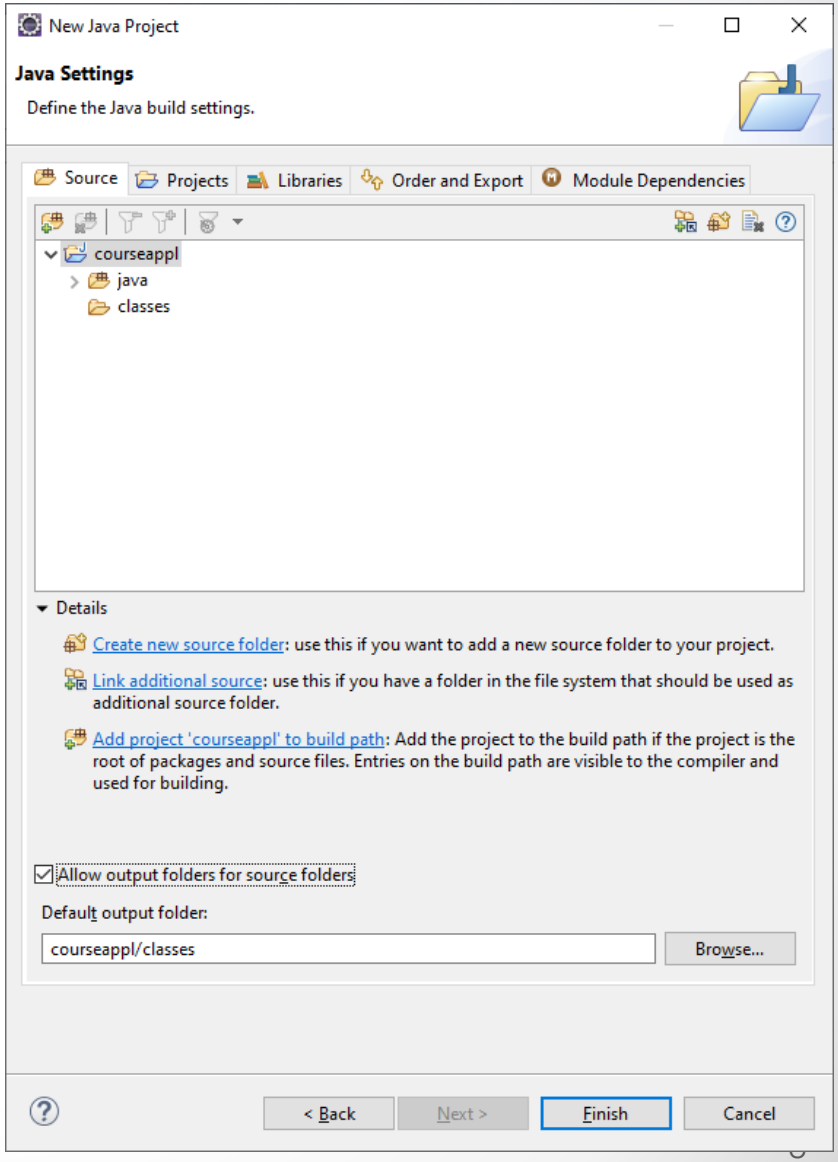
- Example using Eclipse
- Create a Java project
- define a project name
- select "Create project from existing source"





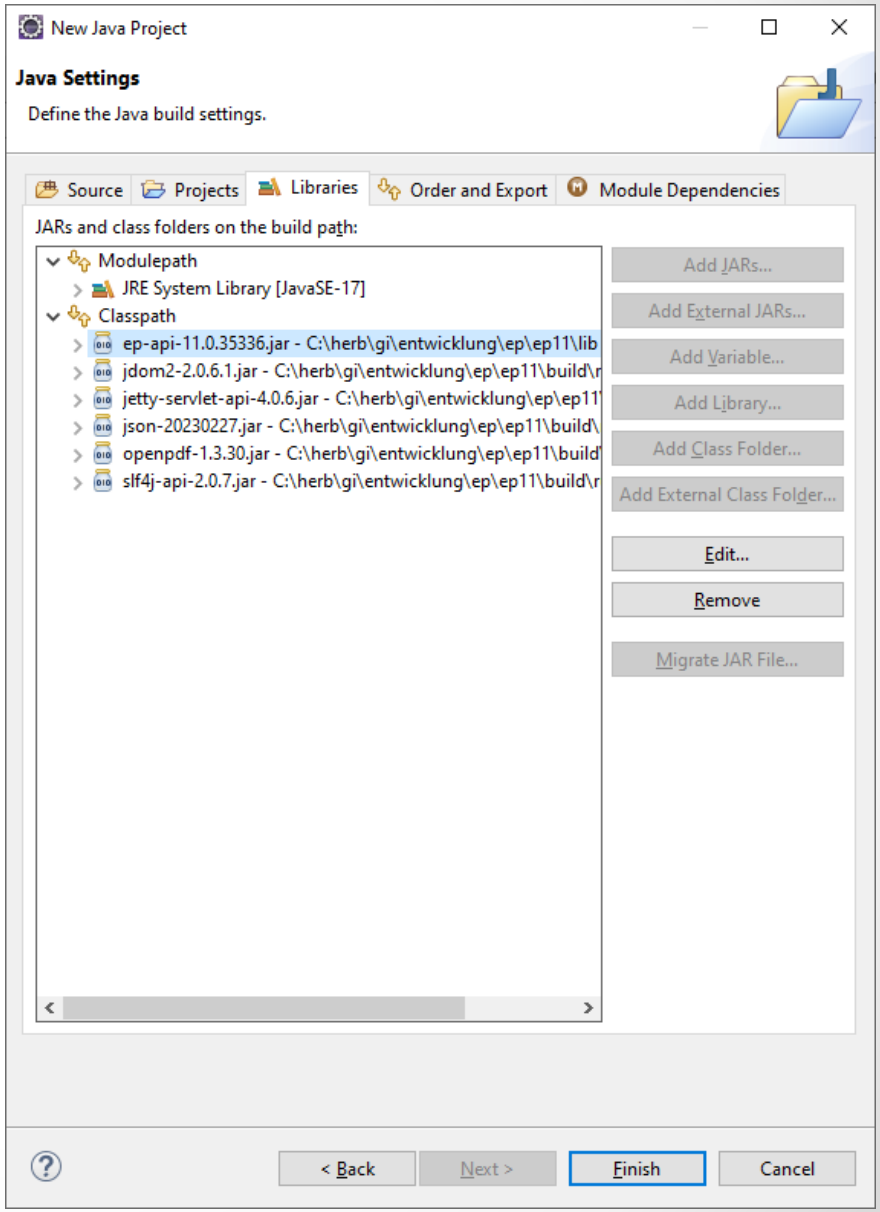
Setting up a Project (2)

- add a `java` folder for the sources
- check "Allow output folders for source folders"
- set default output folder to projectname/classes



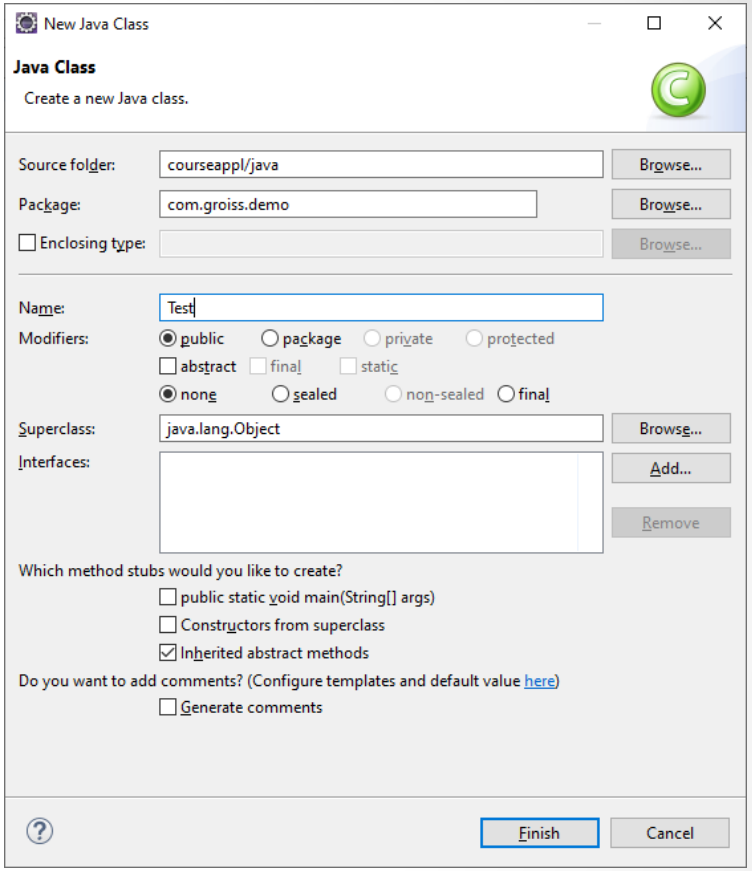
Setting up a Project (3)

- add the libraries from the @enterprise lib directory
  - Either all of them excluding ep_impl-*.jar
  - or – if you know what you need – select the necessary ones.
- click Finish



Setting up a Project (4)

- Create a Java class
- in directory java
- package com.groiss.demo
- class name Test
- click Finish



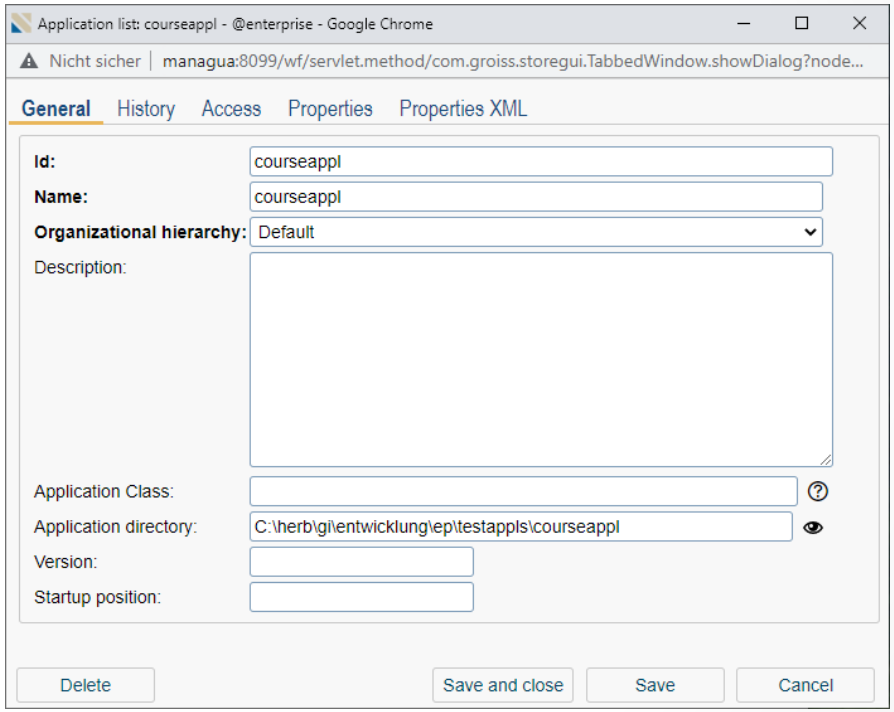
- Add the method

```
@EntryPoint
public void showDate(HttpServletRequest req, HttpServletResponse res)
   throws IOException {
  res.getWriter().println("<html>"+ new Date()+"</html>");
}
```

Setting up a Project (5)

in @enterprise:

- add an application
  - with the eclipse project directory as application directory



- restart
- Test the method

  `http://localhost:8000/wf/servlet.method/com.groiss.demo.Test.showDate`

Debugging your code

Client side debugging:

F12 in most browsers

Server side debugging in Eclipse:

- add to Java call in batch file:
- -Xdebug -Xrunjdwp:server=y,transport=dt_socket,address=<port-number>,suspend=n
- in Eclipse Debug Configurations... set host and port

3. Servlet Methods

- How Java methods are called
- sending pages to the browser
- context and parameters
- static content

File Servlet

FileServlet is the default servlet in the @enterprise context root (normally `/wf`)

requested resources are searched for in the elements of the classpath

Example:

`http://host:port/wf/somedir/somefile`

locale is `de_AT_gi`

resources are searched in the following paths:

1. `lang/de_AT_gi/somedir/somefile`
2. `lang/de_AT/somedir/somefile`
3. `lang/de/somedir/somefile`
4. `lang/default/somedir/somefile`
5. `alllangs/somedir/somefile`

- In case 5 placeholders in the file with the syntax `@@@key@@` are translated to the strings of the current locale.
- Don't place images in alllangs!

Dispatcher

- requests to
  `http://host:port/wf/`**`servlet.method`**`/classname.methodname`
  are handled by the Dispatcher servlet

- the path is interpreted as Java method:
  full qualified class name "." methodname

- method must have one of the following signatures

```
public void method(HttpServletRequest req,
            HttpServletResponse res) throws Exception;

public Page method(HttpServletRequest req) throws Exception;
```

How the Dispatcher works

- parses the URL-path and loads the class
- create a new instance of the class using the default constructor
- checks whether the class has a method with a correct signature

- sets the environment
  - if a session exists (user logged in), the user is set in the ThreadContext
  - the locale is set
  - a transaction is started

- the method is called

- If an error occurred, it is written into the log file and the error page is sent to the browser and the transaction is completed with a rollback
- else the transaction is completed with commit
- and the resulting (HTML-)Page is sent to the browser.

ThreadContext

- the Dispatcher sets context information,
  `com.groiss.util.ThreadContext` can be used to access them:

- current User
  ```
  static Principal getThreadPrincipal()
  ```

- language of this session
  ```
  static Locale getThreadLocale()
  ```

- current request
  ```
  static HttpServletRequest getThreadRequest()
  ```

- set and get further context information
  ```
  static void setAttribute(String key, Object o)
  static Object getAttribute(String key)
  ```

HTMLPage

Write pages to the browser using a template, patterns starting and ending with % can be replaced.

Example:

- Mask:

```
<html>Date in language %language% in %format%:<br>%date%</html>
```

- Java-Code:

```
public Page showNLSDate2(HttpServletRequest req) throws Exception {
     String language = req.getParameter("language");
     String format = req.getParameter("format");
     Locale l = new Locale(language,language);
     HTMLPage p = new HTMLPage("com/groiss/demo/Date.html");
     SimpleDateFormat df = new SimpleDateFormat(
              ("long".equals(format) ? "EEEE, MMMM dd, yyyy" :
        "EEE, MMM dd, yyyy"),l);
     p.substitute("format", format );
     p.substitute("language", language );
     p.substitute("date", df.format(new Date()));
     return p;
}
```

java/com/groiss/demo/HttpDemo.java

XHTMLPage

Page must be in XML structure (XHTML), loaded using XMLParser

Replacements in XML

- using XPath and JDOM

  `p.getDocument()` returns an org.jdom2.Document

- using a Map of components

  `p.get("id")` returns an instance of `com.groiss.gui.Component`

  `p.get("id").setAttribute("value","val")`

JSONPage

- send JSON data to the client

```
JSONPage p = new JSONPage(
  new JSONObject().put("a","b")
);
```

- is never cached

```
Cache-Control: no-cache
Expires: 0
```

=> if you have no result, return `JSONPage.EMPTY_PAGE` to avoid caching

Restricting Access

- Default access is method is accessible for authorized users only
- Define Access to a URL using Annotations (`com.groiss.servlet.Access`)
  - Public: Accessible without authorization
  - User: accessible for authorized users
  - Admin: accessible for admin users, extra login necessary port configurable

- Attach Annotation to package, class or method
  - package: create file package-info.java

```
@Access(Access.mode.Admin)
package com.groiss.htmladmin;
import com.groiss.servlet.Access;
```

  - class or method:

```
@Access(Access.mode.Admin)
public class HTMLImpex { ...
```

Admin Session

- Secure admin actions with Admin annotation
- You can check whether in admin session or not

```
ServletUtils.isAdminSession(req)
```

- `ServletUtils.checkAdminSession` throws an Exception if not in admin session

- Check access based on rights with OrgData methods:

```
public boolean hasRight(User u, Right r, Object o);
public void checkRight(Right r, Object o);
```

SQLInjection and Cross-site scripting

- use prepared statements against SQL Injection
  - WRONG:

```
store.get(User.class, "id='"+req.getParameter("user")+"'");
```

  - RIGHT:

```
store.get(User.class,"id=?",new Object[]{req.getParameter("user")});
```

- Cross-site scripting: embedding script code
  - WRONG:

```
<input name="formtarget" value="%target%">
page.substitute("target",req.getParameter("formTarget"));
```

  - RIGHT:

```
page.substEncoded("target",req.getParameter("formTarget"));
```

This applies to all data from user input: request parameters, form data, etc.

- Don't use file names or other resources as parameters

  can be used to view arbitrary resources

Authorization

Description: customize login

Definition:

```
package com.groiss.org;
public interface HttpAuth {
  public void sendLoginRequest(HttpServletRequest req,
     HttpServletResponse res) throws Exception;
  public Principal checkUser(String user, String passwd,
     String clientAddr) throws Exception;
}
```

`AuthUtil` is used for creating the session

Declaration: Configuration -> Classes -> Authorization Class

In Product: `PasswdAuth, SSLAuth`

Demos: `BasicPasswdAuth, ClientCertDemoAuth, WinPasswdAuth`

Authorization (2)

Example: Implementation of BasicAuth

```
public class BasicPasswdAuth implements HttpAuth {

 public void sendLoginRequest(HttpServletRequest req, HttpServletResponse
    res)
     throws Exception {
    String auth= req.getHeader("Authorization");
    if (auth != null && auth.startsWith("Basic ")) {
      auth = auth.substring(6);
      auth = new String(Base64.decode(auth));
      String userId = auth.substring(0,auth.indexOf(':'));
      String passwd = auth.substring(auth.indexOf(':')+1);
      try  {
      User u = (User)checkUser(userId,passwd,req.getRemoteAddr());
        AuthUtil.authorizeBrowser(req, res, u, req.getRequestURI() +"?"+
          req.getQueryString());
        return;
      } catch (Exception e) {
        com.groiss.util.Settings.logError(e);
      }
    }
    res.setStatus(401);
    res.addHeader("WWW-Authenticate", "Basic realm=\"@enterprise\"" );
    res.getWriter().println();
 }

 public Principal checkUser(String userId, String passwd, String clientAddr)
     throws Exception {
    return AuthUtil.checkUser(userId,passwd, clientAddr);
 }
}
```

java/com/groiss/demo/BasicPasswdAuth.java

4. Persistence

Principles

- Persistence in @enterprise is implemented on top of relational database management systems
- Technical access at a low level is accomplished via JDBC
- An object-relational mapping ("the Store") is provided as an abstraction layer, so virtually no direct JDBC-calls are needed.
- Database connection handling is managed via a pool; assignments of connections to threads is automatic
- Transaction handling is also automatic to a large extend
- Efficient treatment of complex object meshes is supported via lazy loading and caching
- Package: `com.groiss.store`

Basic O-R Mapping Aspects

- Classes of persistent objects must implement the interface `Persistent` (or extend the abstract class `PersistentObject`)

- Each persistent Java class is mapped to one relational table
- Each class member is mapped to one relational tuple (record) in this table
- Each instance variable is mapped to one column of the table

- Object identification:
  - Via instance variable (= table column) `oid`.
  - Is also the primary key of the table.
  - Must be unique (at least) within the table.
  - A new system-wide unique oid is automatically assigned to all new persisted objects
  - It is advisable to use a scheme like classname:oid for unique object identification

Persistent and PersistentObject (1)

- Specify the name of the relational table:
  - `String getTableName()`

- Object Identity / Primary Key (provided by `PersistentObject`)
  - `private long oid;`
  - `public long getOid();`
  - `public void setOid();`

- Declare further instance variables to be persisted:
  - `private T v1;`
  - `public T getV1();`
  - `public void setV1(T v1);`

- Special cases:
  - Static variables are not persisted
  - Instance variables marked as `transient` or `volatile` are not persisted
  - Instance variable named „`filled`“ is not persisted

Persistent and PersistentObject (2)

- Basic Type Mapping between Java and SQL

SQL Column Type	Java Type
char, varchar	String
decimal(x)	int, long
decimal(x,y)	float,double,BigDecimal
BIGINT	long
longvarchar, clob	String, char[]
longvarbinary, blob	byte[]
date, time, timestamp, %DATE%	Date
decimal(20), BIGINT, %OIDTYPE%	Persistent

- Referencing other persistent objects :
  - declare a Java field f with the appropriate type P2 (must extend Persistent)
- Referencing polymorphic persistent objects:
  - If P2 is abstract or an interface or implements the interface HasSubclasses:
  - There must be an additional column in the table definition with the name f_class (type e.g. varchar(100)) to allow proper mapping

DefinedEnums (1)

- Enumerations with a defined value to facilitate bijective mapping between Java enum object and the stored value:
- Interface **`DefinedEnum`**: **`public Object definedValue()`**:
  - currently Strings, Shorts, Integers and Longs are supported.
  - Provide appropriately typed columns in the database table

```
enum IntEnum implements DefinedEnum {
        ZERO(0),ONE(1);
        private final Integer i;
        IntEnum(Integer i) {this.i = i;}
        @Override public Object definedValue() {return i;}
}
```

- For "normal" enums, `Enum.name()` is used instead of `definedValue()`

```
enum ColorEnum {
        WHITE,BLACK;
}
```

DefinedEnums (2)

```
public class EnumExample extends PersistentObject{
        public String getTableName() {
                return "anobjectwithenums";
        }
        private String id;  // assume getters and setters
        private IntEnum iE;
        private ColorEnum cE;
}
create table anobjectwithenums (
 oid %OIDTYPE% not null primary key
 id varchar(20),
 ie integer,
 ce varchar(5));

Store.getInstance().list(
        EnumExample.class,"ie=? and ce=?",
        IntEnum.ONE, ColorEnum.BLACK);
```

Persistent and PersistentObject (3)

- Hooks for additional actions at Insert, Update, Delete and Select

```
public void beforeInsert();        public void afterInsert();
public void beforeUpdate();        public void afterUpdate();
public void beforeDelete();        public void afterDelete();
public void afterRead();
public void isValid(); //called after beforeInsert and beforeUpdate
```

  `PersistentObject` provides empty implementations of the hooks

- Do **not** override other methods of `PersistentObject`
- Another Interface, `PersistentEventHandler<P extends Persistent>` which is not related to the object hierarchy allows one to specify additional actions without „polluting“ the code of the core object class, or to specify common actions for more than one class. Must be registered via:
  - `StoreUtil.addEventHandler(`

```
    PersistentEventHandler<? extends Persistent> peh,
            Class[] ca)
```

Store (1)

- Interface `com.groiss.store.Store` is a facade to the persistence layer
- Get it via `Store.getInstance()`

Important Methods of `Store`:

- `insert(Persistent p)`: inserts `p` into the database
- `update(Persistent p)`: updates `p` in the database
- `delete(Persistent p)`: deletes `p` from the database

- `Persistent get(Class c , long oid)`:
  retrieves an object from the db via its oid
- `Persistent get(Class c, String cond, Object[] params)`: retrieves an object from the database via a parameterized condition `cond` (e.g. "name=? and city=?")

- `List list(Class c)`: a list of all persistent instances of `c`
- `List list(Class c, String cond, String order, Object[] params)`: a list of the instances of `c` satisfying the parameterized condition `cond` ordered by `order`

Store (2)

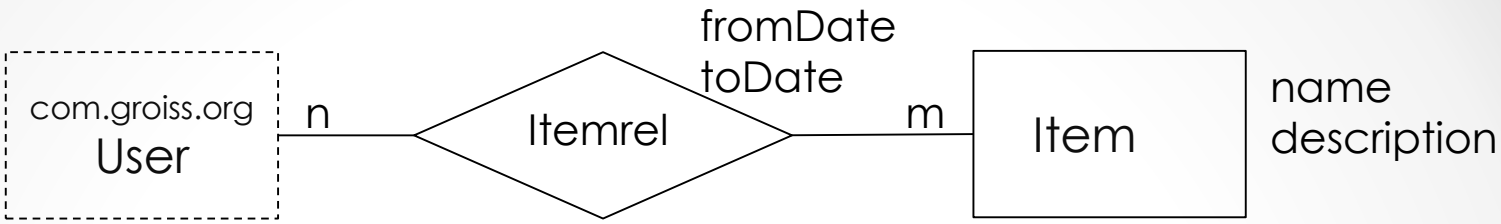
Methods (continued):

- `long getOID()`: return the next unique oid from (usually, there is no need to set this manually, it is taken care of during insert)
- `int count(Class c, String cond, Object[] params)`: count the instances of `c` satisfying the parameterized condition `cond`
- `boolean hasRows (String statement, Object[] params)`: determine if the parameterized SQL-`statement` would return any rows
- `boolean inDatabase(Persistent p)`: determine if `p` is present in the database

Most of the methods which require a Class as parameter are also available as a variant with a class name String. e.g.:

- `int count(String classname, String cond, Object[] params)`: count the instances of class named `classname` satisfying the parameterized condition `cond`

Store Example



- Table definitions:

```
create table res_itemrel (
        oid %OIDTYPE% primary key,
        item %OIDTYPE%,
        userid %OIDTYPE%,
        fromDate %DATE%,
        toDate %DATE%);
```

```
create table res_item(
        oid %OIDTYPE% primary key,
        name varchar(30),
        description varchar(200));
```

- Don‘t forget the indexes :

```
create index res_ir_item on res_itemrel(item);
create index res_ir_userid on res_itemrel(userid);
```

Store Example (2)

Item class:

```
public class Item extends PersistentObject{
        private String name;
        private String description;

        public String getTableName() { return "res_item"; }

        public String toString() {
           Store.getInstance().fill(this);
           return name;
        }

        public void onDelete() {
           Store.getInstance().delete(ItemRelation.class,
              "item=?", this);
        }
}
```

Store Example (2)

ItemRel class:

```
public class ItemRelation extends PersistentObject {
  public String getTableName() { return "res_itemrel";}
  private Item item;
  private User userid;
  private Date fromDate;
  private Date toDate;
  public ItemRelation() {}
  public ItemRelation(Item res, User user, Date from, Date to) {
    item = res; userid = user; fromDate = from; toDate = to;
  }
  public Item getItem() {
    Store.getInstance().fill(this);
        return item;
    } // …other getters
  public String toString() {
    Store.getInstance().fill(this);
    return userid.toString()+":"+item.toString();
  }}
```

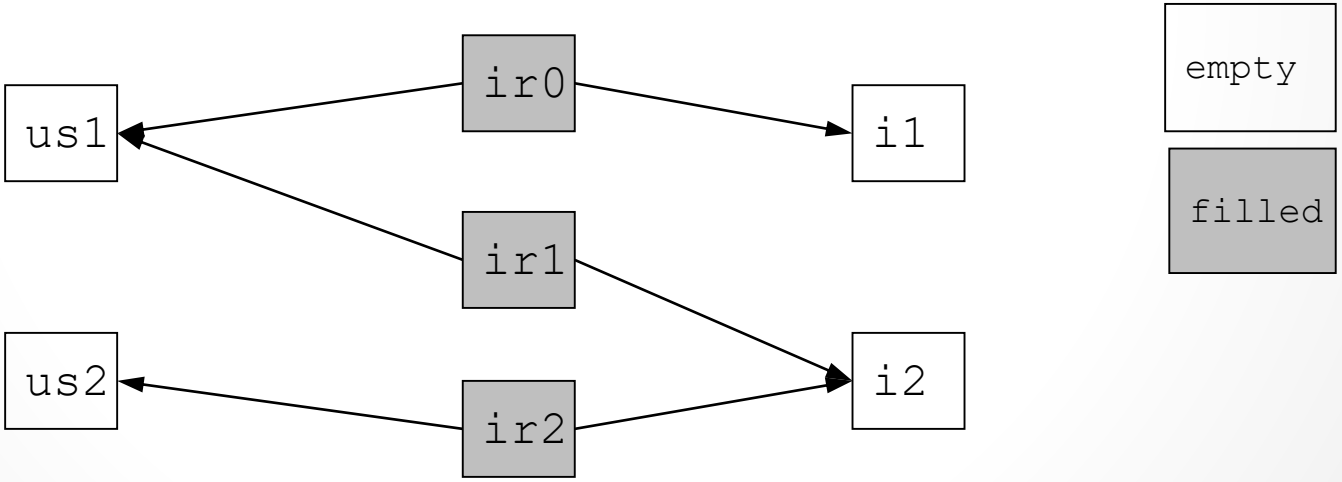
ObjectCache, Lazy Loading and Store.fill()

- Persistent objects are cached per thread and transaction.
- When an object is read from the database:
  - the cache („ThreadCache“ / „ObjectCache“) is checked first.
  - If the object is in the cache, the cached one is returned (maybe after if has been is filled).
  - If the Object is not in the cache, it is put into it, its fields are set from the database.
  - If the object references other persistent objects:
    - If the referenced objects are in the cache: those are used / referred to
    - If the referenced objects are not in the cache: new empty (unfilled) objects are put into the cache and referenced to.
- There (usually) is only one incarnation of a specific persistent object per thread and transaction (after the transaction ends, the cache is cleared).
- There are no inconsistencies because of multiple potentially different incarnations of one object.
- Referenced objects are „lazy loaded“ or can be „loaded on demand“ using Store.fill(Persistent p) In your getters / before they are needed
- Method Persistent.isFilled() can be used to determine if an object is filled or empty.

ObjectCache, Lazy Loading and Store.fill()

- Method Store.getNoCache(Class c, long oid) can be used to retrieve an object (a copy of the object) without any cache interaction
  - no cache check before reading
  - no insertion into cache after reading
  - E.g. in an onUpdate() method to retrieve the old values of the object without destroying the current (uncommitted) ones
- Example: List<ItemRel> irl = store.list(ItemRel.class,

  "item in (select oid from rel_item where name like ?)", null, new Object[]{"C%"});



- irl.get(1).toString(); → fills us1 and i2 (because of Item.toString() and User.toString()

OptimisticLocking

- Allows for coordinated object modification without undue diminishing of concurrency (no long duration locks).
- Additional instance field long transactionid in the class and(!) table
- The transactionid is sent along to the client (e.g. when a form is presented to the user via the browser).
- When the changed object should be updated in the database, the clients transaction id and the transactionid in the database are compared.
- When they are the same (no one else changed the object between retrieval and update), the update is allowed; the transacionid of the object is incremented.
- If the transactionids differ, an exception is thrown.
- Specified in Interface OptimisticLocking:

```
public long getTransactionId();
public void setTransactionId(long tid);
public void increaseTransactionId();
```

- Automatically honored by store.update(Persistent p)

Additional Aspects: Permission Checks

- **Permission Checks**: is an agent allowed to apply an operation to an object?
- The insert, update, delete and list operations of Store do not consider permissions in any way.
- The OrgData facade, obtainable via OrgData.getInstance() provides appropriate methods for this:
- OrgData.insert(Persistent p), OrgData.update(Persistent p), and OrgData.delete(Persistent p) check if the current user is permitted to execute the operation on the object. If the check fails, an Exception is thrown.
- Method OrgData.listWithRightCheck(…) can be used to retrieve a list of objects, the current user is allowed to see according to her permissions.

Additional Aspects: Logging Changes (HasLog)

- **Logging**: store versions of changed objects along with the time and the agent of change
- The insert, update, delete and operations of Store do not consider logging in any way.
- `OrgData` methods do versioning if the object implements the `HasLog` interface.
- Object versions are stored in serialized form in the `avw_log` table
- Can be retrieved via `OrgData.getLogEntries(…)`

Further Aspects: DeferredChanges, HasPermissionList

Some master data instances can have changes that are „deferred“ till a later point in time.

- Marker Interface: `HasDeferredChange` designates that an instance may have outstanding changes.
- Is checked during update and delete via the Store

Some objects can have PermissionLists attached to them.

- Additional instance field `PermissionList acl` in the class and(!) table
- Specified in Interface HasPermissionList:

```
public PermissionList getPermissionList();
public void setPermissionList(PermissionList pl);
public OrgUnit getDefaultOrgUnit();
```

- Automatically honored by Store: insert, update, delete

CheckedPersistent: HasPermissionList and OptimisticLocking

- `CheckedPersistent`: is a convenience class combining `HasPermissionList` and `OptimisticLocking` (those interfaces which require additional fields)

PersistentAspect

Allows one to deviate from the standard automatic handling of different persistence aspects:

- `PersistentAspect`:
  - Enumeration of defined aspects
  - and a bunch of static methods to set / clear them.

- `NO_OPTLOCKING`: do not check for optimistic locking in Store.update
- `NO_DEFERREDCHECK`: do not check for deferred changes in Store.update/delete
- `NO_VALIDATION`: do not call PersistentObject.isValid() during Store.insert/update
- `NO_PERMISSIONCHECK`: do not check Permissions during OrgData.insert/update/delete
- `NO_LOG`: do not write Log entries during OrgData.insert/update/delete
- Can be set on a global or on an object level. Is reset at BeanManager.commit()

PersistentAspect

- `EnumSet<PersistentAspect> get(Object o)`
- `void set(Object o, EnumSet<PersistentAspect> pas)`
- `boolean isSet(Object o, PersistentAspect pa)`
- `SilentCloseable /* for try with resources block */`
  `add(Object o, PersistentAspect... aspects )`
- `void remove(Object o, PersistentAspect... aspects )`
- `void clear(Object o)`
- All calls also available in a global form without the first „Object o“ parameter.
- Example:

```
try {
 PersistentAspect.add(o, PersistentAspect.NO_OPTLOCKING);
 Store.getInstance().update(o);
} finally {
  PersistentAspect.remove(o,PersistentAspect.NO_OPTLOCKING);
}
```

- Alternative using AutoClosable:

```
try (SilentCloseable acpa =
    PersistentAspect.add(o, PersistentAspect.NO_OPTLOCKING)) {
 Store.getInstance().update(o);
} // automatically closed / removed here
```

5. Utilities and Data Structures

- Utility classes
- Timer
- Service
- Holidays
- SessionSynchronization
- ErrorFormatter
- HTML components

Utility classes

- `StringUtil`
  - encoding and decoding values
- `FileUtil`
  - reading from files
- `CalUtil`
  - Methods for parsing and showing date and time values according to the configured formats
- `ThreadContext`
  - get context: request, user
  - set Attributes

- `StoreUtil`
  - `toJSON(persistent)`: returns JSON representation of object
  - `toJsonAsReference(persistent)`: returns

    {objectId: classname:oid, _toString: string-repr}

Timer

Description: execute recurring actions

Definition:

```
package com.groiss.timer;
public interface TimerTask {
   public void run(TimerEntry e, String arg) throws Exception;
   public void abort();
}
```

Declaration: System administration -> Application -> Timer

In Product: several implementations

Demos: `com.groiss.demo.FileGetter`

Timer - Example

Example: Start a process with files from directory

```
public class FileGetter implements TimerTask {
  @Override
  public void run(TimerEntry te, String arg1) throws Exception {
    WfEngine e = WfEngine.getInstance();
    OrgData od = OrgData.getInstance();
    Configuration conf = Configuration.get("demo");
    ThreadContext.setThreadPrincipal(User.SYSADM_USER);
    File in = new File(conf.getProperty("input.dir"));
    File [] files = in.listFiles();
    for (int i = 0; i< files.length; i++) {
      File f = files[i];
      String subject = f.getName();
      int j = subject.indexOf('.');
      String extension = subject.substring(j+1);
      subject = subject.substring(0,j);
      ProcessDefinition pd = e.getProcessDefinition(procid);
      OrgUnit startou = od.getById(OrgUnit.class, conf.getProperty("start.ou"));
      ProcessInstance pi = e.startProcess(pd, null, startou, null, null);
      ....
```

java/com/groiss/demo/FileGetter.java

Service

Description: implement hooks on startup and shutdown

Definition:

```
public interface Lifecycle {
   public default void startup() {}
   public default void shutdown() {}
}
public interface Service extends Lifecycle {
   public default boolean isRunning() { return false; }
   public default void reconfigure() {}
}
```

Declaration:

- Configuration -> Classes -> Services
- an application adapter may implement the service interface

In Product: the configured services

ApplicationAdapter is a subinterface

PropertiesChangeListener

Description: react on change of properties

Definition:

```
public interface PropertiesChangeListener {
    /** Properties have changed.
     * @param events key is property name, pair contains old and new value.
     */
    public void propertiesChange(Map<String,Pair<Object,Object>> events);
}
```

Declaration:

- call Configuration.addPropertiesChangeListener(PropertiesChangeListener l, String... props), for example in startup method of Service (ApplicationAdapter)

Example:

```
Configuration.get().addPropertiesChangeListener(new PropertiesChangeListener() {
  public void propertiesChange(Map<String, Pair<Object, Object>> events) {
    boolean activeFromConfig  = (Boolean)events.get("aclcache.active").second;
    if (!activeFromConfig) {
      ACLCache.deActivate();
    } else if (ACLCache.isActive()) { .. }
  }, "aclcache.active");
```

Holidays

Description: implement the holidays for your country

Definition:

```
package com.groiss.cal;
public interface Holidays {
   public String isHoliday(com.ibm.icu.util.Calendar d);
}
```

Declaration: Configuration -> Calendar -> Holiday Class

In Product: `GermanHolidays, AustrianHolidays`

Demos: `UKHolidays`

Holidays - Example

Example: UKHolidays

```
public String isHoliday(com.ibm.icu.util.Calendar d) {
   if(!(d instanceof GregorianCalendar)) {
        return null;
   }
   GregorianCalendar c = (GregorianCalendar)d;
   int wd = d.get(Calendar.DAY_OF_WEEK);
   if (wd == Calendar.SATURDAY || wd == Calendar.SUNDAY)
        return null;
   int day = c.get(Calendar.DAY_OF_YEAR);
   int year = c.get(Calendar.YEAR);
   int easter = CalUtil.easterDay(year);
   if (day == easter - 2) {
        return "Good Friday";
   } else if (day == easter + 1) {
        return "Easter Monday";
   }
   // move to next monday if on saturday or sunday
   if (day == 1 || wd == Calendar.MONDAY && (day == 2 || day == 3) ) {
        return "New Year's Day";
   }
```

Holidays – Example (2)

```
    day -= c.isLeapYear(year) ? 1 : 0;
    if (day == 359 || wd == Calendar.MONDAY && (day == 360 || day == 361)) {
          return "Christmas Day";
    } else if (day  == 360 || wd == Calendar.TUESDAY && (day==361 || day==362)
                   || wd == Calendar.MONDAY && day == 362) {
          return "Boxing Day";
    }
    if (wd == Calendar.MONDAY) {
          int m = d.get(Calendar.MONTH);
          int md = d.get(Calendar.DAY_OF_MONTH);
          if (md >= 1 && md <=7 && m == Calendar.MAY) {// first monday in may
              return "Early May Bank Holiday";
          }
          // last monday in may
          if (md >= 25 && md <=31 && m == Calendar.MAY) {
              return "Spring Bank Holiday";
          }
          // first monday in august
          if (md >= 25 && md <=31 && m == Calendar.AUGUST) {
              return "Summer Bank Holiday";
          }
    }
    return null;
}
```

java/com/groiss/demo/UKHolidays.java

SessionSynchronization

Description: interface from javax.ejb "allows a stateful session bean instance to be notified by its container of transaction boundaries".

Definition

```
public interface SessionSynchronization {
   public void afterBegin();
   public void beforeCompletion();
   public void afterCompletion(boolean committed);
}
```

Declaration:

```
BeanManager.addBean(beanId, beanclass)
```

Demos:

```
com.groiss.demo.dms.FileStoreBean
```

Formatting Error Messages

Description: change format of error messages

Definition:

```
public interface ErrorFormatter {
  public Page format(Throwable e);
  public JSONObject formatJSON(Throwable e);
}
```

Declaration:

- ApplicationException has a method `setErrorFormatter`
- for the default behavior: Configuration -> Classes -> Error-Formatter

In Product: `com.groiss.gui.DefaultErrorFormatter`

Demos: `DemoErrorFormatter`

Error handling in smartclient

- error handling method in ep/Utils:

  `showErrorMessage(e)` shows a popup

  e is a message or an error object from request.get or post
- Ajax requests: no special handling on server necessary, Dispatcher recognizes ajax call and returns error code and error object.

  Example:

```
{error: "Error 17: In the first step of ...",
 errornumber: 17,
 message: "In the first step of ...",
 title: "Error 17"
 showHTML: false}
```

  `Utils.showErrorMessage` can handle this object.

- Global handler in main page calls `showErrorMessage` for all unhandled Ajax errors. Set `_errorHandlingDone=true` on the error object, if handled by own method.

ErrorFormatter - Example

Example:

```
public class DemoErrorFormatter extends DefaultErrorFormatter {
  public Page format(Throwable e) {
   HTMLPage p = new HTMLPage();
   p.setPage("<html>Oops, en error occured, this is the message:" +
      e.getMessage() + "<html>");
   return p;
  }
  public JSONObject formatJSON(Throwable e) {
    JSONObject result = super.formatJSON(e);
    result.put("title", "Error title..");
    return result;
  }

  public Page test(HttpServletRequest req) {
    ApplicationException e = new ApplicationException("test");
    e.setErrorFormatter(this);
    throw e;
  }
}
```

java/com/groiss/demo/DemoErrorFormatter.java

HTML Components

- package `com.groiss.gui.component` contains Java objects for HTML form elements: Link, Textfield, Textarea, ...

- Usage:
  - in XHTMLPage get(id) returns the java representation of the HTML element with the given id.
  - the components can be inserted into HTML pages:
    - HTMLPage: `p.substitute("placeholder", comp.show());`
    - XHTMLPage: `p.get(id).setContent(comp)`
    - `getRoot()` return the JDOM element

HTMLUtils

- Get the object from a parameter containing `classname:oid`
  - Parameter `object:`

    ```
    Persistent x = HTMLUtils.getObject(req)
    ```
  - other parameters:

    ```
    HTMLUtils.getObject(req, paramname)
    ```

- Create the classname:oid string from a persistent:

  `StoreUtil.toString(Persistent)` => classname:oid

6. Application

groups all information of a workflow application

- Application object in the database
  - Id, name, org.hierarchy, description
  - application class: behavior of application, startup and shutdown action
  - application directory: location of the code and resources in the file system
  - version: defines the version installed => upgrade
  - startup position: string for ordering the application startup
- Application parameters
  - Configuration parameters for the application in appl.prop
- User parameters
  - application specific parameters for each user

- Application dependent objects
  - process definitions, forms, tasks, functions, roles, rights, object classes function groups, GUI configurations, Timers, reports, mail boxes, Web-Services
  - Resource bundle for text strings and error messages

Structure of Applications

appl.prop	configuration file
classes/	Java-classes and resources loaded from classpath
classes/applid/properties.xml	property-file for application- and user-params
classes/applid/reporting.xml	customized reporting
classes/applid/styles.less	The @enterprise style-loader loads the file and appends it to standard styles
classes/applid/masks/	HTML masks
classes/applid/forms/	Forms (templates for form-types)
classes/applid/exports/	Export files
classes/applid/strings.xls	Resource file for internationalization
classes/lang/default/applid/	default for language dependent files and images
classes/lang/<language>/applid/	language dependent files
classes/alllangs/applid/	language independent files (HTML, ..)
classes/alllangs/scripts/applid/	JavaScript files, dojo classes
lib/	jar files
java/	Java source
doc/	documentation

bold path are required, rest is convention

Internationalization

Configuration: System administration -> Application x -> Tab Properties: define a name
Definition: Application x ->Resources: add keys and translations
Use it:

- process names, form names, etc.: use the resource key
- Resources loaded by FileServlet (images, scripts, HTML Pages):
  - loaded from alllangs directory in classpath
  use placeholders in text: @@@key@@
  - loaded from language specific directory

- Resources loaded by @enterprise: Forms, GUI-Configuration
  placeholders in text: @@@key@@ or @@@<applid>:key@@
  applid for @enterprise resources: ep

- Java Code
  - get the application resource
  ```
  ApplicationAdapter applclass = ApplicationAdapter.of("staffprocs");
  Resource res = applclass.getResource();
  ```
  - translate keys
  ```
  res.getString("key");
  ```
  - load HTMLPage: translation is done automatically

Changing Style and Logos

- Style
  - the style used in @enterprise are loaded from the URL `com.groiss.gui.css.StyleConf.loadCSS`
  - files in classpath under <appl-id>/`styles.less` are added to this file in application loading order
- To change any style, use the appropriate CSS selector
- Examples:

.scTaskfunction.<taskfunction-id>	Taskfunction icon
.scEnterpriseLogo, .scEnterpriseLogoLarge	Your enterprise-logo
.scIcon.scXXXX	Selector for @enterprise standard icons
.scProcess.<process_id>	Process icon – visible in process list
.scIcon.scDmsForm.<process_id>.v<version>	Form icon for a process
.scIcon.scProcess.<process_id>.v<version>.scDmsFolder	DMS-Folder icon of a process
.scDept.<dept-id>	Icon of an OrgUnit in organization-tree
.admin_tree\.appl_<appl_id>.dijitIcon	Icon in admin tree

- examples in demos/classes/demo/styles.less
- Use font icons if available

Deployment and Upgrade

package your application:

- most resources are loaded from the classpath and can therefore be packaged in a jar file
- you may need other jar files of third-party libraries
- package the application into a zip file
  - jar files in lib sub-directory
  - classes in classes sub-directory
  - appl.prop in root with the following properties
    - `avw.application.id` application id
    - `avw.application.name` application name
    - `avw.application.class` application class
    - `avw.export.file` comma separated list of XML export files
- use the function "Install/Upgrade application" on the target system

see demos.zip or hr_proc.zip for examples

Deployment and Upgrade (2)

- initial deployment
  - install @enterprise and then the application zip

- application upgrade
  - database-objects: make a new import or import parts of the application
  - Define an upgrade path in application details using either groovy or Java code
    - `Admin.getInstance().importXML(resource)` for new import
    - `StoreUtil.executeScript()` for loading database updates

      use placeholder like `%DATE%` or `%OIDTYPE%`

  - Replace application zip using Install/Upgrade application

Context-Sensitive Help

- appl.prop: pointer to index page

  `avw.application.docu.html=itsm/index.html`

  other html pages in same dir or subdir

  `avw.application.docu=itsm/itsmdoc.pdf`

- Parameter group in properties.xml:

  `<parametergroup name="ITSM" helpctx="itsm/config">`

  itsm ... application id

  config ... an element with this id in a html file in the dir that contains the file defined in `avw.application.docu.html` property (for example: <a id="config">)

- XML configuration

```
<node id=..>
    <helpContext>itsm/problem</helpContext>
    …
```

- HTML form or iframe

  `<html data-ep-helpcontext="sysadm/user"`

- smartclient
  - in widget: attribute `helpContext: "itsm/problem"`

Customized User Manual

Parameters in group "Other Parameters":

- parameter `ep.user.docu` points to docu in pdf format
- parameter `ep.user.docu.html` points to a file in classpath

  for example: `itsm/userdoc/index.html`
- The system searches for an anchor first in customized docu, then in standard docu.
- if `ep.user.docu.shadowall=true`, search only in customized docu

7. Organizational Data

- Use the `OrgData` interface for retrieving information about Users,Roles, Organizational Units and Rights.

```
OrgData.getInstance()
```

- get, list, insert, update, delete

- hasRight, checkRight, hasRightForList

PermissionMapping

Description: change behavior of permission system

Definition:

```
public abstract class PermissionMapping {
   public abstract Class<?>[] forClasses();

   public void init(Agent agent, Right right, List<? extends
    Persistent> wanted){}

   public DNF rewrite(PermissionQuery p) {
        DNF result = new DNF(Arrays.asList(Arrays.asList(p)));
        return result;
   }
}
```

Declaration:

```
OrgData.getInstance().addRule(new
  SupplierPermissionMapping());
```

Demos:

```
SupplierPermissionMapping
```

PermissionMapping - Example

```
public class SupplierPermissionMapping extends PermissionMapping {
  public Class<?>[] forClasses() {
    return new Class[]{ DemoApplication.SUPPLIER_CLASS };
  }

  public DNF rewrite(PermissionQuery p) {
    User u = p.user;
    DMSForm supplier = (DMSForm)p.target;
    boolean isAgent = Store.getInstance().hasRows(
      "select 1 from avw_stepinstance where agent=? and childof in" +
      "(select process from avw_forminstance where form in" +
      "(select oid from form_demo_order_1 where supplier=?))",
      new Object[]{u,supplier});
    if (isAgent) {
      return new DNF(true);
    } else {
      return null;
    }
  }
}
```

java/com/groiss/demo/SupplierPermissionMapping.java

PermissionMapping

- hierarchy of rights

```
//add that VIEW and EDIT imply FIND
PermissionChecker.addToRightMap(Rights.FIND, Rights.FIND);
PermissionChecker.addToRightMap(Rights.FIND, Right.VIEW);
PermissionChecker.addToRightMap(Rights.FIND, Right.EDIT);
```

- DNF = Disjunctive normal form

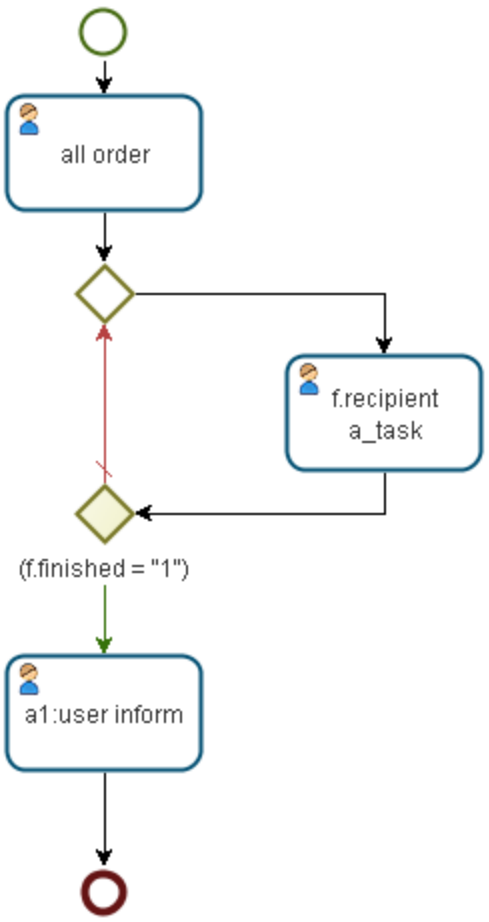
  list of list of PermissionQueries

  {{a,b}, {c,d}} means (a ∧ b) ∨ (c ∧ d)

8. The Workflow Engine

interprets the process graph

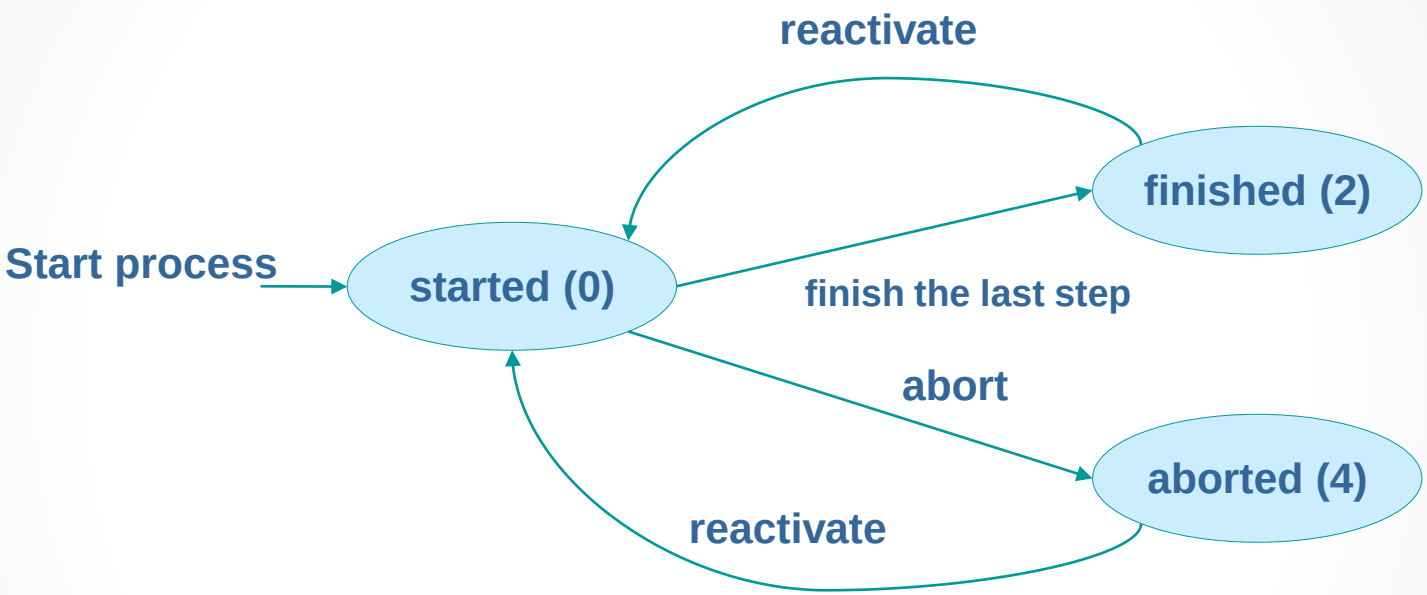
- Nodes (Steps):
  - task-step
  - system-step
  - if: if, while, exit when
  - nop: begin, end, par
  - andjoin
  - orjoin
  - process
- Edges (Flows):
  - normal (black)
  - then (green): condition in source node is true
  - else (red, slash at start): condition is false



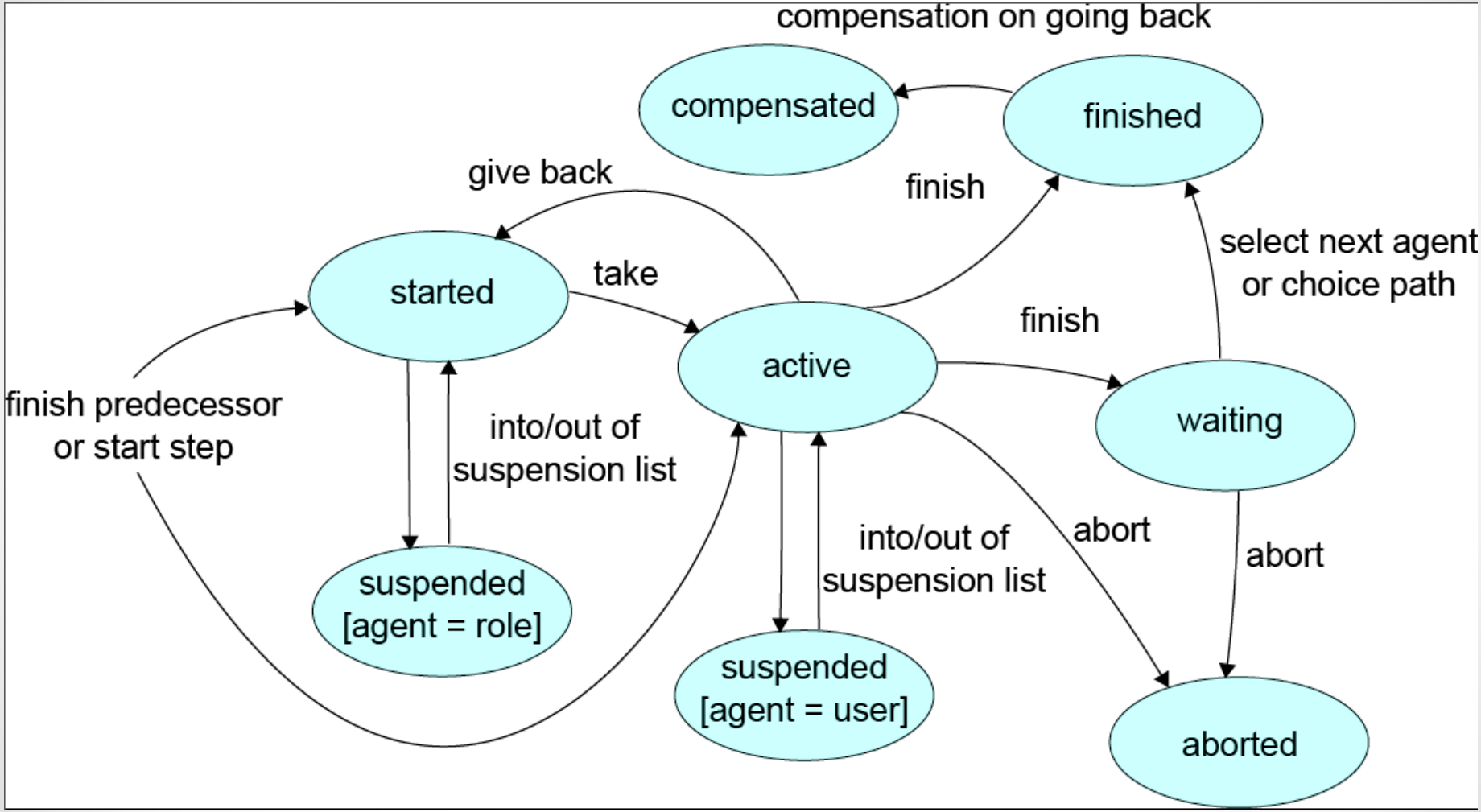
The Workflow Engine (2)

- startProcess
  - creates a ProcessInstance object
  - call step.start with the first step in the process definition
- Step.start
  - case
    - task-step: execute preprocessing, put in worklist of agent
    - system-step: execute method and call finish
    - if: execute condition and call finish
    - nop: call finish
    - process: call startProcess
    - andjoin: if all predecessors are finished call finish
    - orjoin: if first call finish, abort other branches
    - ...
- ActivityInstance.finish
  - check postcondition
  - find successors:
    - if found, call step.start on them
    - else finish parent

Process Instance States



Activity Instance States



Workflow Engine API

- package `com.groiss.wf`
- main interface is **`WfEngine`**
- get instance with `WfEngine.getInstance()`

WfEngine

- find processes
  - `List getStartableProcesses(Application appl)`
  - `List listProcessDefinitions(Application appl)`
  - `ProcessDefinition getProcessDefinition(String id, int version)`
  - `ProcessDefinition getProcessDefinition(String id)`
- start a process
  - `ProcessInstance startProcess(ProcessDefinition p, User u, OrgUnit d, Date duedate, String id)`
  - `ProcessInstance startProcess(ProcessDefinition p, User u, OrgUnit d, Date duedate, String id, DMSForm f)`
- get the worklist
  - `List getWorklist(Application a, boolean withRepr)`
  - `List getRoleWorklist(Application a)`
  - `List getSuspensionList(Application a)`
  - `List getRoleSuspensionList(Application a)`
- get a process instance
  - `ProcessInstance getProcess(String id)`
  - `ProcessInstance getProcess(long oid)`
  - `ProcessInstance getProcess(DMSForm f)`
  - `List getProcesses(String condition)`

WfEngine (2)

- get an activity instance
  - `ActivityInstance getActivityInstance(long oid)`
  - `List getActivityInstances(ProcessInstance process)`
  - `List getActivities(String condition)`
  - `List getActiveTasks(ProcessInstance process)`
  - `List getActiveTasks(ProcessInstance process, User u)`

  inside preprocessing, postcondition, condition, system step:
  - `ActivityInstance getContext()`

WfEngine (3)

- manipulating activity instances
  - `ActivityInstance take(ActivityInstance ai)`
  - `ActivityInstance untake(ActivityInstance ai)`
  - `void finish(ActivityInstance ai)`
  - `void seeLater(ActivityInstance ai, Date d)`
  - `void seeAgain(ActivityInstance ai)`
  - `ActivityInstance goBack(ActivityInstance ai, ActivityInstance ai2, String comment)`
  - `void gotoTask(ProcessInstance process, String taskid, Agent ag, String comment)`
  - `ActivityInstance copyTo(ActivityInstance ai, User u)`
  - `void setAgent(ActivityInstance ai, Agent a)`
  - `void setOrgUnit(ActivityInstance ai, OrgUnit d)`
  - `void setStepAgent(ActivityInstance ai, Agent u)`
  - `void setDescription(ActivityInstance ai, String descr)`
  - `void setDuedate(ActivityInstance ai, Date d)`

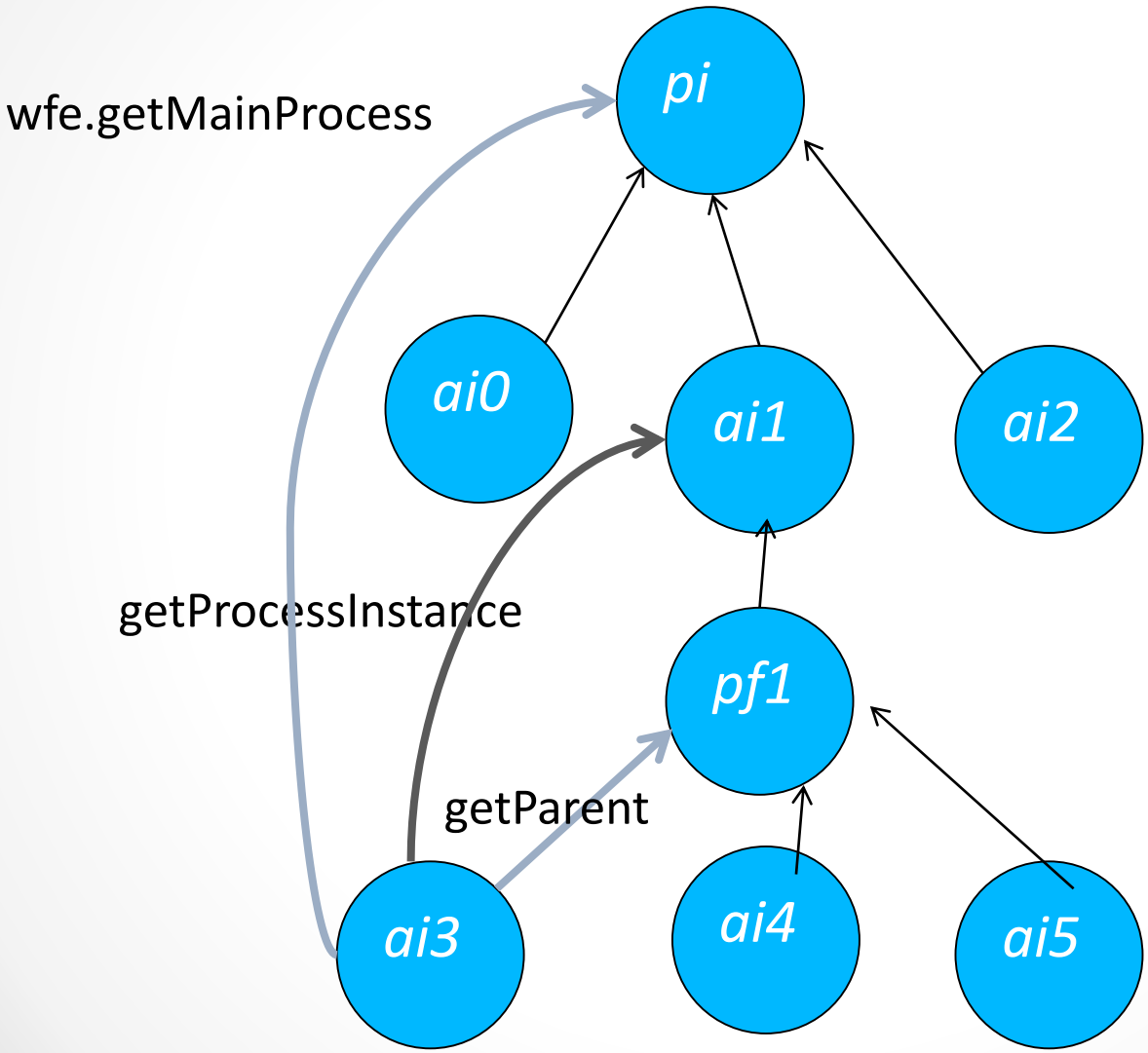
WfEngine (4)

- manipulating process instances
  - `void abort(ProcessInstance process)`
  - `void reactivate(ProcessInstance process)`
  - `void archive(ProcessInstance process)`
  - `void setSubject(ProcessInstance process)`
  - `void setSubjectToString(ProcessInstance process, String str)`

WfEngine (5)

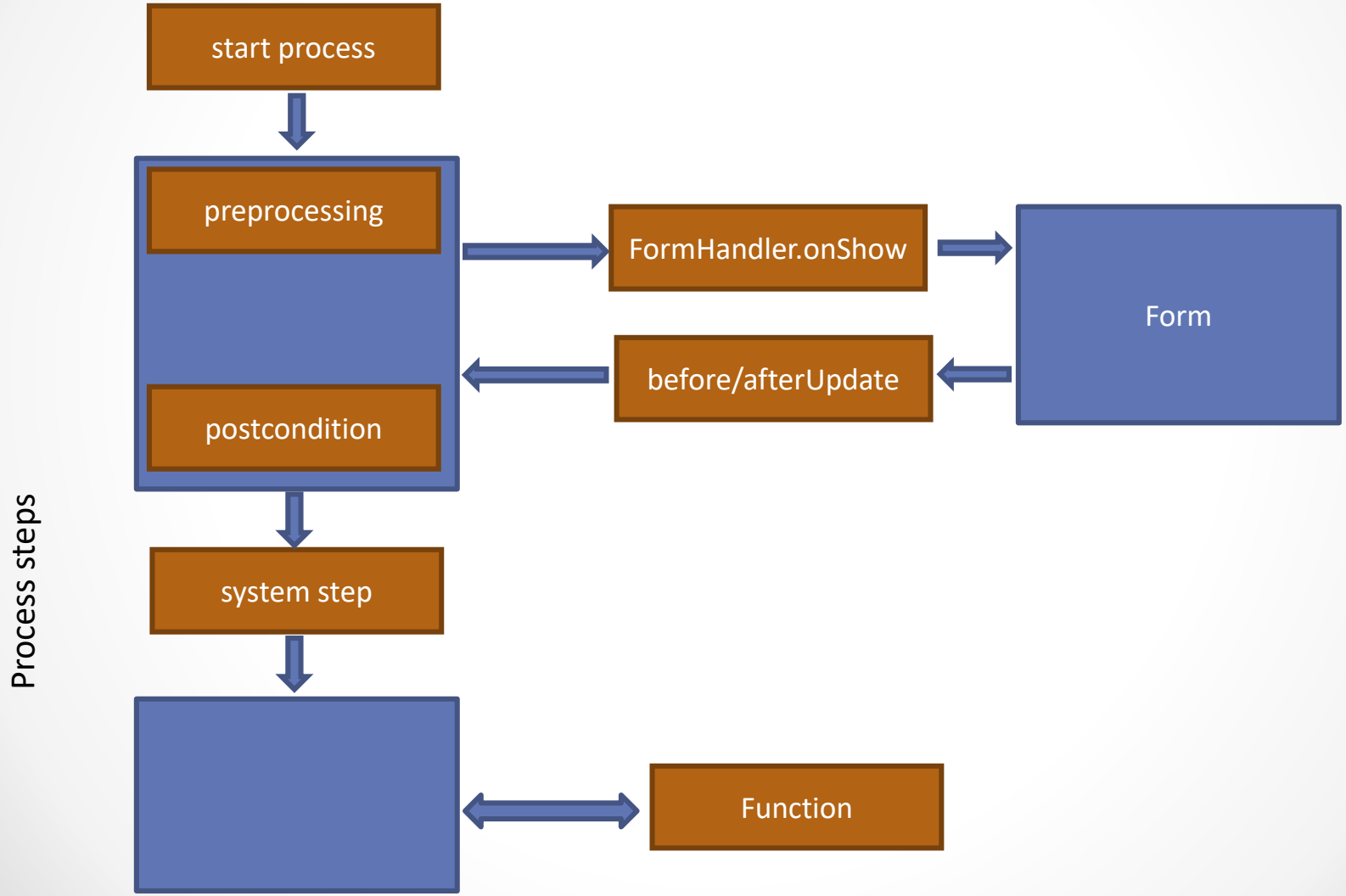
- documents and forms
  - `DMSForm getForm(ProcessInstance pi, String id)`
  - `void updateForm(DMSForm f)`
  - `List getForms(ProcessInstance pi)`
  - `List getNotes(ProcessInstance pi)`
  - `void addDocument(ProcessInstance pi, DocForm f)`
  - `List getDocuments(ProcessInstance pi)`

Tree of activity instances



9. Using the Workflow API

- API Callbacks



API Callbacks in Workflow Execution (2)

- preprocessing, system steps
- escalations
- conditions, postcondition
- toolbar functions
- application behavior
- batch steps
- archiving

Preprocessing, System Steps

Description: perform actions in process context

Definition: Method called via reflection, 0 to n String arguments

```
public void method(String arg1, .., String argn) { ...}
```

Declaration: System steps in process definition, Preprocessing in Task definition

In Product: `com.groiss.wf.SystemAction` contains some useful methods

Demos: `com.groiss.demo.SystemSteps`

Preprocessing, System Steps

Example: Set the activity duedate from a form field

```
public void setDuedate() {
   WfEngine e = WfEngine.getInstance();
   ActivityInstance ai = e.getContext();
   ActivityInstance pi = ai.getParent();
   DMSForm f = e.getForm(pi, "item");
   Date d = f.getField("duedate");
   if (d != null)
        e.setDuedate(ai, d);
}
```

Example: Set a form field to the current agent

```
public void setFieldToAgent()  {
   WfEngine e = WfEngine.getInstance();
   ActivityInstance ai = e.getContext();
   ActivityInstance pi = ai.getParent();
   DMSForm f = e.getForm(pi,"item");
   f.setField("agent", ThreadContext.getThreadPrincipal());
   e.updateForm(f);
}
```

java/com/groiss/demo/SystemSteps.java

Escalation

Example: Send mail to agent

```
public void fire() {
  WfEngine e = WfEngine.getInstance();
  ActivityInstance ai = e.getContext();
  ProcessInstance pi = ai.getProcessInstance();
  if (ai.getAgent() instanceof User) {
    User u = (User)ai.getAgent();
    Admin.getInstance().createMessageTemplate().
      addRecipient(u).
      setSubject(StringUtil.noNull(pi.getSubject())).
      setMimeType("text/plain").
      setBody("Please handle the case "+pi.getId()+
        "\nit is in your worklist since "+ai.getTaken()+
        " and due on "+ai.getDuedate()+".").
      send();
  }
}
```

java/com/groiss/demo/DemoEscalation.java

Conditions in a process, Postconditions

Description: With a postcondition you can perform actions after the user has finished the task. Throwing an exception or returning false prevents the completion of the task.

Conditions in process definitions appear in ifs, loops and choices

Definition: Method called via reflection, 0 to n String arguments, returns boolean

```
public boolean method(String arg1, .., String argn) { ...}
```

Declaration : in process definition, postcondition in Task definition

In Product: `com.groiss.wf.SystemAction` contains some useful methods

Demos: `com.groiss.demo.SystemSteps`

Conditions - Example

Example: postcondition for order: at least one suborder

```
public boolean hasSuborder() {
   WfEngine e = WfEngine.getInstance();
   ActivityInstance ai = e.getContext();
   DMSForm f = e.getForm(ai.getProcessInstance(), "orderform");
   return DMS.getInstance().countSubforms(f, 1) > 0;
}
```

Alternatively throw an exception:

```
   if (DMS.getInstance().countSubforms(f, 1) == 0)
      throw new ApplicationException("At least one suborder!");
```

java/com/groiss/demo/SystemSteps.java

Toolbar Functions

Description: function attached to a form, a task or a process

Definition: Method called via reflection, two possible signatures:

```
public void method(HttpServletRequest req,
                   HttpServletResponse res) { ...}
public Page method(HttpServletRequest req) { ...}
```

the parameter `functionTask` contains the oid of the current activity instance

Declaration:

Define the function: System Administration -> Your application -> Functions

appears in: worklist, history, function list

attach to process or task (or all tasks)

In Product: several functions in the default application

Demos: `com.groiss.demo.DemoFunctions`

Toolbar Functions - Parameter

- **Worklist function (in toolbar or function column)**
  - `nodeid`: id of worklist node (xmlid.nodeid)
  - `functionTask`: oid of activity instance
- **History**
  - `nodeid`: id od worklist node (xmlid.nodeid)
  - `functionTask`: oid of activity instance or process instance (depending whether the function is added to process or task)
- **Form, in toolbar**
  - `functionTask`: oid of activity instance
- **DMS, in toolbar**
  - `folder`: folder (classname:oid)
  - `object`: selected object (classname:oid)
  - `ai`: if folder belongs to an active process (classname:oid)
- **Form tables**
  - `nodeid`: id of worklist node (xmlid.nodeid)
  - `object`: selected object (classname:oid)
- **Subform table**
  - `_src`: mainform (classname:oid)
  - `_id`: subform id
  - `object`: selected object (classname:oid)
- **Reporting**
  - `object`: selected object (classname:oid)
  - `functionTask`: oid of selected object

Toolbar Functions - Example

Example: approve function, set two fields in a form

```
public Page approve(HttpServletRequest req) throws Exception {
   User u = (User)ThreadContext.getThreadPrincipal();
   WfEngine e = WfEngine.getInstance();
   ActivityInstance ai =
        e.getActivityInstance(Long.parseLong(req.getParameter("functionTask")));
   DMSForm f = e.getForms(ai.getProcessInstance()).get(0);
   f.setField("approvedBy", u);
   f.setField("approvedDate", new Date());
   e.updateForm(f);
   e.propagateChange(ai);
   JSONObject result = ClientUtil.getChangesAsJSON(req.getParameter("nodeid"),true);
   return new ActionPage("parent.require(['ep/Utils'],function(Utils) {"+
     "Utils.refreshWorklists(" + result + ",true);});");
}
```

java/com/groiss/demo/DemoFunctions.java

See section 13 of this course for an example of a toolbar-function with client-side code.

Application Adapter

Description: customization of various application properties

Definition:

```
public interface ApplicationAdapter extends HasResource, Service {
 public static ApplicationAdapter of(String applId);
 public static ApplicationAdapter of(Application appl);
 public String getNewProcessId(ProcessInstance pi);
 public void onSeeLater(ActivityInstance ai);
 public void onSeeAgain(ActivityInstance ai);
 public void onChangeAgent(ActivityInstance oldAi, ActivityInstance newAi);
 public void onAbort(ProcessInstance pi);
 public void onReactivate(ProcessInstance pi);
 public void onAddDocument(ProcessInstance pi, DMSFolder f, DMSObject o);
 public void onRemoveDocument(ProcessInstance pi, DMSFolder f, DMSObject o);
 public void notifyUser(User u, ActivityInstance ai);
 public List<Element> getUserProperties(User u);
 public void modifyDetailPanes(KeyedList<String,NavigationTreeNode> nodes,
    StringBuilder title, ProcessInstance pi, ActivityInstance ai);
 public void getToolbarActions(ActivityInstance ai,
    List<Pair<String,List<Pair<String,?>>>> actions);
 public String getVersion();
 public String upgrade(Application appl) throws Exception;
 public void onAnonymize(User u, String newId);
 public void onFinishNotComplete(ActivityInstance ai);
 public void onApplicationInstall(Application appl);
 public void onApplicationUninstall(Application appl);
 public void onCopyProcess(ProcessInstance originPI, ProcessInstance newPI);
}
```

Adapter Class: `DefaultApplicationAdapter`

Declaration: Detail mask of application

Demos: `com.groiss.demo.DemoApplication`

Application Adapter - Example

Example: startup method, add-document hook, and prevent reactivate

```
public void startup(){
  OrgData.getInstance().addRule(SupplierPermissionMapping.class);
}

public void onAddDocument(ProcessInstance pi, DMSFolder f, DMSObject o) {
  if (o instanceof DMSDocForm df &&
    df.getFormType().getId().equals("demo_orderconfirmation") &&
    !DMS.getInstance().listContents(f, ((DMSDocForm)o).getFormType(), true,
    null, null).isEmpty()) {
   throw new ApplicationException("Only one order confirmation in process.",
   Level.DEBUG);
  }

public void onReactivate(ProcessInstance pi) {
  throw new ApplicationException("Not supported in DemoApplication");
}
```

java/com/groiss/demo/DemoApplication.java

Batch Adapter

Description: Execution of a batch step

Definition:

```
public interface BatchAdapter {
    public void startup() throws Exception;
    public void afterCreation(BatchJob job) throws Exception;
    public void doStart(BatchJob job) throws Exception;
    public void beforeCompletion(BatchJob job) throws Exception;
    public void afterCompletion(BatchJob job, boolean commit)
        throws Exception;
    public void doCompensate(BatchJob job)  throws Exception;
}
```

Adapter Class: `NullAdapter`

Declaration: class name in batch step

Demos: `DemoBatchAdpater`

Batch Adapter - Example

Example: write info to a file, finished by url,

```
public void startup()  {
  File mainDir = new File(getMainDirName());
  mainDir.mkdir();
}
public void afterCreation(BatchJob job)  {
  File procDir = new File(getMainDirName(),getProcDirName(job.getContext()));
  procDir.mkdir();
}
public void doStart(BatchJob job)  {
  try {
    String procId = job.getContext().getProcessInstance().getId();
    File procDir=new  File(getMainDirName(),getProcDirName(job.getContext()));
    File outFile = new File(procDir,procId+".out");
    PrintWriter  out = new PrintWriter(new FileWriter(outFile));
    out.println("Output File "+ new java.util.Date());
    DMSForm f =
      WfEngine.getInstance().getForm(
        job.getContext().getProcessInstance(),"f");
    out.println(f.getField("description"));
    out.println(
     "com.groiss.demo.DemoBatchAdapter.notifyFinish?bjOid="+job.getOid());
    out.close();
  } catch (Exception ex) {
    throw new ApplicationException("doStart",ex);
  }
}
```

Batch Adapter – Example (2)

```
public void notifyFinish(HttpServletRequest req, HttpServletResponse res)
     throws Exception {
   long bjOid = Long.parseLong(req.getParameter("bjOid"));
   BatchJob bj = Store.getInstance().get(BatchJob.class,bjOid);
   BatchManager.markJobFinished(bj);
   res.getWriter().println("Done");
}

public void beforeCompletion(BatchJob job)  {
   try {
        String procId = job.getContext().getProcessInstance().getId();
        File procDir = new File(
         getMainDirName(),getProcDirName(job.getContext()));
        File inFile = new File(procDir,procId+".in");
        BufferedReader in = new BufferedReader(new FileReader(inFile));
        String line = in.readLine();
        in.close();
        DMSForm f = WfEngine.getInstance().getForm(
         job.getContext().getProcessInstance(),"f");
        f.setField("description",line);
        Store.getInstance().update(f);
   } catch (Exception ex) {
        throw new ApplicationException("beforeCompletion",ex);
   }
}
```

java/com/groiss/demo/DemoBatchAdapter .java

Archiving

Description: when process instances are removed from the database, move some information to another system.

Definition:

```
package com.groiss.wf;
public interface ProcessArchiver {
  public void archive(java.util.List<ProcessInstance> processes,
    User u);
}
```

Declaration: Configuration -> Classes -> Archiving class

In Product: `com.groiss.wf.NoArchiver`: throws an error, if somebody tries to archive processes

Demos: `XMLArchiver, FileArchiver`

Archiving Example

Example: Use the XML Export for archiving

```
public void archive(List<ProcessInstance> processes, User
   u)throws Exception{
  FileOutputStream os = new FileOutputStream("archive.xml");
  List<Exportable> l = (List)processes;
  try {
    Admin.getInstance().exportXML(processes,
      "procinst_cluster", "export for archive", null,
      os, Settings. getLogWriter());
  } finally {
    os.close();
  }
}
```

java/com/groiss/demo/XMLArchiver.java

Parfor

Description: control the creation of parfor branches without using forms

Definition:

```
public interface ParForIterator  {
        public void init(ActivityInstance ai);
        public boolean hasNext();
        public void next(ActivityInstance ai);
}
```

Declaration: in the parfor node in process definition

Demos: ParforIterator

Parfor Example

```
public class ParIterator implements ParForIterator {
   private int count = 2;

   public void init(ActivityInstance ai) {
      logger.debug("******** ParIterator init");
    }

   public boolean hasNext() {
      return count-- >0;
   }

   public void next(ActivityInstance ai) {
      logger.debug("******** ParIterator next "+count);
   }
}
```

java/com/groiss/demo/ParIterator.java

10. Document Management

- DMS API
- Folder events
- Office templates

DMS API

- `DMSObject`

  Interface for all objects in DMS
- `DMSForm`

  most objects are forms (except DMSLinks)
- `DMSDocForm`

  an object with a form and a document component
- `DMSFolder`

- `com.groiss.dms.DMS` interface

  `DMS.getInstance()`
  - createFolder, createDocForm, createForm, createNote
  - add, remove
  - delete
  - move
  - ...

Handling folder events

Description: provide hooks for add and remove events

Definition:

```
public interface XHTMLFolderFormEventHandler<T extends DMSFolder &
    DMSForm> extends XHTMLFormEventHandler<T> {
  public default void onAdd(T f, DMSObject o) throws Exception {}
  public default void onRemove(T f, DMSObject o) throws Exception {}
}
```

Declaration:

- global in DMS configuration
- for a folder type: formtype definition

Demos: `OrderFolderEventHandler`

Office templates

- Open office is used for creating open office, MS Word and PDF documents.

```
This is the Id: ${$pi/id/text()}
today: ${$today}
```

- Example - Create pdf document

```
DMSDocForm template =
  Store.getInstance().get(OdtTemplate.class,"id='order'");
Map<String,Object> map = new HashMap<String,Object>();
map.put("pi", folder);
map.put("today", CalUtil.showDate(new Date()));
byte[] bytes = DocumentManager.mixin(template, map);
bytes = DocumentManager.convert(new ByteArrayInputStream(bytes),
"odt", "pdf");
```

java/com/groiss/demo/DemoFunctions.java

11. Configure the Client

- Worklist layout
- DMS table layout
- Forms
- Form tables
- Object tables

Configure worklist layout

- Use `<worklist>` in your application XML
- Settings selection model
  - Use `<selection></selection>`
  - Possible arguments are:
    - NONE: no selection
    - ONE: only single selection, extra selection column
    - MULTI: multi selection, extra selection column
    - ROWSINGLE: single selection, clicking direct on row
    - ROWMULTI: multi selection, clicking direct on row
  - Defaults to ROWMULTI
- Defining actions
  - Use the `<actions>` element to add actions
  - Examples
    - `<action id="xmlid.actionId"/>`
    - `<action id="taskfunction:functionId"/>`
    -
```
<action id="finish">
      <action id="finishAndSelect"/>
</action>
```

Configure worklist layout - columns

- Configure columns
  - Use `<columns>` container
  - Use `<column>` to define a single column
  - To use multiple rows per entry use the <row> element
  - A `<column>` may have the following attributes
    - id – the colum-id
    - field – the field of the JSON-object to use
    - name – the display-name
    - visible – default visibility
    - unhideable – the column is not listed in column-picker and is always visible
    - rowSpan – the cell rowSpan (like HTML <table>)
    - colSpan – the cell colSpan (like HTML <table>)
    - jsClass – a client-side-widget implementing `ep/widget/smartclient/grid/Column`

      => see section 13 of this course
  - For information how to style a grid-instance, see: `https://github.com/SitePen/dgrid/blob/master/doc/usage/Styling-dgrid.md`

API for customizing tables

different table handlers

- `Worklist`, `DMSTableHandler`, `FormTableHandler` (subform, configured tables), `ObjectTableHandler`
- common methods

```
public List<T> getList(List<T> list);
public void modifyColumns(List<ColumnDescription> colDescs);
public void modifyTableLine(T f, Map<String,Object> line);
public String lineStyle(T f, String style);
```

- different `init` methods

Worklist Layout Java API

Description: Implement this interface to change the layout of the worklist, if XML configuration is not powerful enough: showing customized information in the columns, modify the list contents, etc.

The methods are called in the given order.

Definition:

```
package com.groiss.wf.html;
public interface Worklist {
   public void init(HttpServletRequest req, WorklistDescription wl, User u);
   public List<ActivityInstance> getList(List<ActivityInstance> l);
   public void getAdditionalData(List<ActivityInstance> instances,
        List<String> splitResult);
   public void modifyColumns(List<ColumnDescription> colDescs);
   public void modifyTableLine(ActivityInstance ai, Map<String,Object>
   line);
   public String lineStyle(ActivityInstance ai, String style);
   public List<Pair<String,String>> listFilters(List<ActivityInstance>
        lines);
}
```

Declaration: in Gui-Configuration: Table-Handler of worklist node

Worklist Layout API (2)

Demos: `DemoWorklist, AdditionalProcDataWL`

Example: Show the subject in a color depending on process priority

```
public class DemoWorklist extends WorklistAdapter {
   WfEngine e = WfEngine.getInstance();

  public void modifyTableLine(ActivityInstance ai,
     Map<String,Object> line) {
    ProcessInstance pi = ai.getProcessInstance();
    int prio = pi.getPriority();
    String color = prio > 50 ? "red" : (prio >30 ? "orange" : null);
    if (color != null) {
      line.put("subject", new CellValue(line.get("subject"),
        Arrays.asList(new Pair<>("style", "color:" + color))));
    }
  }
}
```

java/com/groiss/demo/DemoWorklist.java

DMS table layout

Description: customize the DMS table

Definition:

```
public interface DMSTableHandler {
  public void init(HttpServletRequest req, DMSFolder folder,User u,int mode);
  public List<DMSObject> getList(List<DMSObject> objects);
  public void modifyColumns(List<ColumnDescription> colDescs);
  public void modifyTableLine(DMSObject obj, Map<String, Object> line);
  public void modifyActions(List<Pair<String, Object>> actions);
  public String lineStyle(DMSObject obj, String style);
}
```

Declaration:

- global in DMS configuration
- for a folder type: formtype definition
- for a process: process definition mask

Demos: `OrderFolderTableHandler`

Example

```
public void modifyTableLine(DMSObject obj, Map<String, Object> line) {
  String value = "";
  if (obj instanceof DMSDocForm) {
   if (((DMSDocForm)obj).getFormType().getId().
       equals("demo_deliverynote")) {
     if((Boolean)((DMSForm)obj).getField("checked")){
       value = getResource().getString("yes");
     } else {
       value = getResource().getString("no");
     }
   }
  }
  line.put("form.paid", value);
}
```

java/com/groiss/demo/dms/OrderFolderTableHandler.java

Forms

Two possibilities: XHTMLForms, XForms

- XHTMLForms:
  - HTML documents that are correct XML documents = XHTML
    - one form, containing form fields
  - build-time: form is loaded into @ep, Java class and db table is generated
  - run-time:
    - form is filled with values
    - form field visibilities are used to set fields invisible, read-only, read-write
    - subform tables are inserted
    - buttons are added for insert, update, delete
  - form handler can be used to customize behavior
- XForms
  - type-sensitive rendering (object select, date, number, boolean, etc.)
  - XPath constraints
  - less scripting

Inline subforms

- Example

```
<table><tr><th>itemdate</th><th>timefrom</th><th>timeto</th></tr>
  <tbody subformid="1" formtype="hr_timeitem_1" id="subform_1"
    xf:repeat-
nodeset="/data/form/subform[@id='1']/form[position()!=last()]">
    <tr><td><xf:input ref="itemdate" /></td>
    <td><xf:input ref="timefrom"/></td>
    <td><xf:input ref="timeto"/></td>
</tr>
</tbody>
</table>
<xf:trigger ref="/data/form/subform[@id='1']/buttons">
  <xf:label>new line</xf:label>
  <xf:insert ev:event="DOMActivate" position="after"
    nodeset="/data/form/subform[@id='1']/form" at="index('subform_1')"/>
</xf:trigger>
<xf:trigger ref="/data/form/subform[@id='1']/buttons">
  <xf:label>delete</xf:label>
  <xf:delete ev:event="DOMActivate"
    nodeset="/data/form/subform[@id='1']/form" at="index('subform_1')"/>
</xf:trigger>
<table>
```

Form Handler

Description: change the form, hooks for insert, update, delete

Definition:

```
package com.groiss.dms;
public interface XHTMLFormEventHandler<T extends DMSForm> extends
   ObjectFormHandler<T> {
   public void beforeInsert(T f) throws Exception;
   public void beforeUpdate(T f) throws Exception;
   public void beforeDelete(T f) throws Exception;
   public void afterInsert(T f) throws Exception;
   public void afterUpdate(T f) throws Exception;
   public void afterDelete(T f) throws Exception;
   public void beforeShow(T f, FormContext ctx, HttpServletRequest req)
      throws Exception;
   public void onShow(T f, FormContext ctx, XHTMLPage p,
      HttpServletRequest req) throws Exception;
   public String getName(T f) throws Exception;
   public default String[][] getKeys(T f);
}
```

Declaration: System administration -> Forms, Attribute Event-Handler

Demos: `DemoFormEventHandler`

Form Handler - Example

Example: Set the duedate field of the main form to the max of the subform duedates.

```
public class DemoFormEventHandler extends XHTMLFormEventAdapter<DMSForm> {
  public void beforeUpdate(DMSForm f) {
    DMS dms = DMS.getInstance();
    DMSForm main = dms.getMainForm(f);
    Date subdue = f.getField("duedate");
    Date superdue = main.getField("duedate");
    if (subdue != null && (superdue == null ||
        subdue.after(superdue))) {
      main.setField("duedate", subdue);
      dms.update(main);
    }
  }

  public void beforeInsert(DMSForm f) {
    beforeUpdate(f);
  }
}
```

Configured form table

- Configure form table in XML, can be done with wizard

```
<table id="id_25">
  <name>@@@supplier@@</name>
<classname>com.dec.avw.appl.demo_supplier_1</classname>
</table>
```

- FormTableHandler

```
<tablehandler>com.groiss.demo.SupplierTableHandler</tablehandler>
```

- Column definition is taken from formtype definition, can be overwritten

```
<columns>
  <column id="name" name="@@@ep:name@@" visible="true" />
  <column id="description" name="@@@ep:description@@"
visible="true"/>
```

- Actions: xml keys or taskfunctions

```
<actions>
    <action id="admin.new"/>
    <action id="edit"/>
</actions>
```

Form Table API

Description: customize layout of form tables

Definition:

```
package com.groiss.dms;
public interface FormTableHandler<T extends DMSForm> {
   public void init(HttpServletRequest req, User u);
   public List<DMSForm> getList(List<T> list);
   public void modifyColumns(List<ColumnDescription> colDescs);
   public void modifyTableLine(T f,
        Map<String,Object> line);
   public String lineStyle(T f, String style);
}
```

Declaration:

- subform tables: in form

```
<tablefield class="com.dec.avw.appl.Change_2" id="2"
       tablehandler="com.groiss.itsm.ReleaseChangeTableHandler"/>
```

- form tables configured in XML

Demos: `DemoFormTableHandler`

Using form interfaces

- *applid*_`forms.jar` in the lib directory of the application contains forms interfaces
- Add to build path in your IDE, but not to application zip
- Package `com.groiss.forms.interfaces`
- Interface name: *formid _ version*
- Getter and Setter for all fields:
  - Example: field name „vat“

```
public double getVatField();
public demo_orderitem_1 setVatField(double vat);
```

- Use in form table handlers, form event handler, postcondition, etc.
- Example:

```
public class OrderItemFormHandler implements
  XHTMLFormEventHandler<demo_orderitem_1> {
public void beforeUpdate(demo_orderitem_1 f) {
        …
```

12. Communication

- REST API
- Web Services
- LDAP
- Sending mails
- Handle incoming mails
- File import

REST API

- REST API for workflow operations
- Using OpenAPI (known as Swagger)
  - Specification of APIs: wf/ep-rest/v1/openapi.yaml
  - Interactive Swagger-UI: wf/swagger-ui/index.html

- Examples:
  - Start process

```
POST /wf/ep-rest/v1/process-instances
{
  "processDefinitionId": „order",
  "processDefinitionVersion": 0,
  "dueDate": "2023-02-23T08:30:00Z"
}
```

  - Get worklist content:

```
GET /wf/ep-rest/v1/activity-instances?worklistTypes=user
```

  - Finish an activity:

```
POST /wf/ep-rest/v1/activity-instances/activityinstance:4294967627/finish
```

REST API (2)

process-instances
Operations to start and monitor your processes

- GET `/process-instances` get a list of process instances
- POST `/process-instances` create a new process instances of the specified process definition
- GET `/process-instances/{puid}` get a process instance
- GET `/process-instances/{puid}/activity-instances` get the activity instances of a process instance
- POST `/process-instances/{puid}/start` starts the specified process instance

process-forms

- GET `/process-instances/{puid}/forms` get the list of the forms assigned to this process instance

process-documents

process-notes

activity-instances
Operations to work with the activities of your pocesses and to control their flow

- GET `/activity-instances` get a list of activity instances
- GET `/activity-instances/{auid}` get an activity instance
- POST `/activity-instances/{auid}/changeagent` changes the agent of an activity instance
- POST `/activity-instances/{auid}/finish` finishes the specified activity instance
- POST `/activity-instances/{auid}/take` takes an activity instance
- POST `/activity-instances/{auid}/untake` untakes an activity instance

Web Services

- @enterpise uses the Apache Axis2 framework for calling and providing webservices
- @enterprise is WS-Security enabled (e.g. SecureTokenService for repeated conversations)
- There's an integration for calling/providing webservices
  - Also directly from processdefinition

Call a Web Service

Method 1 - using stubs, created by the Axis2 Code Generator

see the Axis2 website for further information on how to use the Code Generator

Code-Example:

```
XEOXKelagWSStub client = new
  XEOXKelagWSStub(com.groiss.ws.client.ConfigurationContextFactory.
    getClientConfigurationContext(),"http://..."); //create the stub
  SendMessageDocument reqdoc = SendMessageDocument.Factory.newInstance();
  SendMessage req = reqdoc.addNewSendMessage();
  //use setter methods to fill the request doc
  req.setField1("the message");
  req.setField2("...");
  ...
  SendMessageResponseDocument respdoc = client.sendMessage(reqdoc);
  // use getter methods to extract the response data
  String resp = respdoc.getSendMessageResponse().getResultField1();
  ...
```

Call a web service (2)

Method 2 - Don't generate stubs, invoke the web service directly using the Apache Axis2 API

Code-Example:

```
ConfigurationContext ctx =
 com.groiss.ws.client.ConfigurationContextFactory.
   getClientConfigurationContext();
ServiceClient client = new ServiceClient(ctx, "http://...");
Options options = new Options();
options.setAction("urn:echo"); // select operation via WS-Addressing
client.setOptions(options);
//use Apache AXIOM methods to create the payload XML
OMElement payload = ...;

OMElement response = client.sendReceive(payload);
//use Apache AXIOM methods to extract the response data
response.getChildElement("Field1");
```

Call a web service (3)

Method 3 - Don't generate stubs, invoke the web service using @enterprise integration

- Create webservice client in Administration/Applications/your-appl/Webservice clients
  - WSDL is required
- Add the required operations to the new webservice
  - Form supports you by listing only available operations
- Add parameters to the operation
  - Id/XPath-mapping for setting/getting parameter-values

Example

- Call the operation:

```
public void ws(HttpServletRequest req, HttpServletResponse res)throws Exception {
  // get the operation
  WebserviceOperation ws = WebserviceOperation.
    getOperation("tempconvert", "FahrenheitToCelsius");
  // fill the payload
  Map<String,Object> map = new HashMap<>();
  map.put("Fahrenheit", req.getParameter("f"));
  // call the service
  Map<String,Object> inparams = ws.invoke(map);
  // get the result
  res.getWriter().println("result:"+inparams.get("FahrenheitToCelsiusResult"));
}
```

java/com/groiss/demo/ws/DemoWSCall.java

- Preparations for this example:
  - download the wsdl http://www.w3schools.com/xml/tempconvert.asmx?wsdl to application classes directory
  - create new WebService with id tempconvert, type name of wsdl file, select service and operation (TempConvertSoap12) and specify url
  - double click operation, gererate xml, and input and output parameters.

Provide a Web Service

1. Specify or get the WSDL
2. Generate your service skeletons with the Axis2 CLI or Ant-Task [9]
3. Compile the generated sources
4. Package the generated classes
5. Add the new library to your application classpath
6. Subclass the service-skeleton and implement your business logic
7. Modify the services.xml to change the implementation class. This step is required, because it's not recommended to modify the generated source files.
8. Package your services.xml and your WSDL as a Web service archive (.aar)
9. Upload the aar file using the wizard at

   Admin.Tasks -> Communication -> Web Services -> Local Services

Code Example

Subclass the generated Skeleton

```
public class MyService extends DemowsSkeleton {
    @Override
    public MessageRsDocument sendMessage(MessageRqDocument in) {
          MessageRq rq = in.getMessageRq();
          Settings.log("-------------"+rq.getSubject(), 1);
          Settings.log("-------------"+rq.getText(), 1);
          ...

          Principal u = ThreadContext.getThreadPrincipal();
          MessageRsDocument doc = MessageRsDocument.Factory.newInstance();
          MessageRs rs = doc.addNewMessageRs();
          rs.setOut("success");
          return doc;
    }
}
```

Provide a Web Service (2)

- To enable a 'Dispatcher-Like-Behavior' (e.g. auto-commit/rollback, ThreadContext etc.) you have to enable the epcontext-Module which is shipped with @enterprise
- Add the following line to your services.xml (in your .aar file)

  <module ref="epcontext"/>
- If the epcontext-Module is enabled, the request must contain one of the following
  - Username/Password token
  - SAML-Token previously requested from the @enterprise-SecureTokenService

Provide a Web Service (3)

- Use @enterprise integration

- Create webservice server in Administration/Applications/your-appl/Webservice server
  - WSDL is required
- Add the required operations to the new webservice
  - Form supports you by listing only available operations
- Add parameters to the operation
  - Id/XPath-mapping for setting/getting parameter-values

Provide a Web Service (3.1)

- Specify a message-handler for the operation

  com.groiss.ws.server.MessageHandler

```
OMElement inbound(OMElement payload, WebserviceOperation wso,
Map<String,Object> inputParameterValues,Map<String,Object>
outputParameters) throws AxisFault;
```

  - Callback may return the full XML response, or fill the output-parameters-map

LDAP

Description: synchronize users, org.units, etc.

Definition

```
public interface DirectorySyncer {
 public void synchronize(DirectoryServer dirServer,
    DirContext baseContext) throws Exception;
}
```

Declaration: Admin-Tasks -> Communication -> LDAP

Demos: `SimpleDirectorySyncer`

LDAP (2)

Example: Loading users from LDAP

```
public void synchronize(DirectoryServer ds, DirContext ctx) throws Exception{
        NamingEnumeration<Binding> ne = baseContext.listBindings("");
        while (ne.hasMore()) {
                Binding b = ne.next();
                String rdn = b.getName();
                DirContext objectCtx = (DirContext) b.getObject();
                syncObject(rdn, ds, ctx, objectCtx);
        }
    }
  private void syncObject(String rdn, DirectoryServer ds, DirContext
      baseContext, DirContext objectCtx) throws Exception {
        Attributes attribs = objectCtx.getAttributes("");
        Object ldapKey = attribs.get(LDAPKEYATTNAME).get();
        OrgData od = OrgData.getInstance();
        User u = od.getById(User.class,(String)ldapKey);
        if (u != null) { // object exists in @enterprise
                // do nothing
        } else { // create user object
                u = od.createUser();
                setFields(u,attribs);
                u.setActive(true);
                od.insert(u);
        }
    }
```

java/com/groiss/demo/SimpleDirectorySyncer.java

Sending Mails

- Message templates are used for sending mails (notification, escalation, report timer, etc.)
- Send message using a pre-defined template:
- Admin.getInstance().getMessageSender(templateid).send()
- add recipients, set message body with methods of MessageTemplate:

```
Admin.getInstance().getMessageTemplate("templateid").
    setSubject("test").
    addRecipient("xx@acme.com").
    send();
```

- without predefined template:

```
Admin.getInstance().createMessageTemplate().
   addRecipient(
   new Recipient().setAgentString("herbert@groiss.com").
     setRecType(javax.mail.Message.RecipientType.BCC)).
   setSubject("test").
   setLogged(true).
   send();
```

Mail Handler

Description: receive a mail

Definition

```
package com.groiss.mail;
public interface MailHandler2 {
  public boolean receive(javax.mail.Message msg, MailBox mb)
    throws Exception;
}
```

Declaration: Application X -> Mail-boxes

Demos: `MailGetter`

Mail Handler Example

Send the server Info back to the sender.

```
public class MailGetter implements MailHandler2 {
  public boolean receive(Message msg, MailBox mb) throws Exception {
    Admin admin = Admin.getInstance();
    String from = msg.getFrom()[0].toString();
    String body = admin.serverInfo();
    admin.createMessageTemplate().
      addRecipient(from).
      setSubject("Server info").
      setMimeType("text/plain").
      setBody(body).
      send();
    return true;
}
```

java/com/groiss/demo/MailGetter.java

File Import

Description: import tabular data from a file

Definition:

```
public abstract class ImportHandler {
 public Object getKey(Persistent o)  {
   return o.getOid(); }
 public void prepareFile(File f) {}
 public boolean beforeImport(Map<String,Object> m, Map<String,Object> ext) {
   return true; }
 public Object getKey(Map<String,Object> map)  {
   return null; }
 public Persistent getExistingObject(Object key, Map<String,Object> values,
      Map<Object,Persistent> cache) {
   return cache.get(key);
 }
 public boolean beforeInsert(Map<String,Object> m, Persistent o) {
   return true; }
 public boolean beforeUpdate(Map<String,Object> m, Persistent o) {
   return true; }
 public void afterImport(Map<String,Object> m, Persistent o) {}
}
```

Declaration: Define the importHandler in the import.xml file

Demos: import.xml, `UserImporter`

File Import (2)

Description:

The Importer does the following:

- Read the existing members of the class and store in a memory cache.

  The key is either the key field from the configuration or, if none is specified, the `getKey` method is called.
- Call `prepareFile` with the selected or configured File
- for each line in the file:
  - Put the columns in a map
  - call `beforeImport`
  - call `getKey` with the value map
  - call `getExistingObject`
  - if the object is in cache call `beforeUpdate` else `beforeInsert`
  - perform the database operation
  - call `afterImport`
- Close the file

File Import Example

```
OrgData orgdata = OrgData.getInstance();
Role home = orgdata.getRole(Role.HOME);
Map<String, OrgUnit> ous;

public void afterImport(Map<String,Object> m, Persistent o) {
  User u = (User)o;
  OrgUnit ou = orgdata.getHomeOrg(u);
  if (ou == null) {
    ou = getOrg((String)m.get("ou"));
    if (ou != null) {
      UserRole ur = orgdata.createUserRole();
      ur.setUser(u);
      ur.setRole(home);
      ur.setOrgUnit(ou);
      ur.setActive(true);
      orgdata.insert(ur);
    }
  }
}

private OrgUnit getOrg(String id) {
  if (ous == null) {
    ous = new HashMap<String, OrgUnit>();
    for (OrgUnit ou: orgdata.list(OrgUnit.class, null, null, null)) {
      ous.put(ou.getId(), ou);
    }
  }
  return ous.get(id);
}
```

java/com/groiss/demo/UserImporter.java

13. Client side programming

- @enterprise uses the **dojo** library (dojotoolkit.org), currently version 1.17
- Import the main script

```
<script type="text/javascript" src="../scripts/dojo/dojo.js"
    data-dojo-config="parseOnLoad: true">
 </script>
```

- define the style

```
<link href="com.groiss.gui.css.StyleConf.loadCSS" rel="stylesheet"
    type="text/css"></link>
```

contains all required CSS files + application CSS + dojo themes

- import the defined widgets, for example:

```
<script>
require(["ep/widget/DateField",  //necessary for date fields
 "ep/widget/ObjectSelect"]);//necessary for obj. select
```

Dojo API from @enterprise

- All @enterprise Dojo widgets use the AMD (Asynchronous Module Definition) syntax

- Client-callback widgets
  - `ep/widget/smartclient/grid/Column`
  - `ep/widget/smartclient/_Action`
  - `ep/widget/smartclient/ProcessDetails`
  - `ep/widget/smartclient/ProcessDetailsHandler`
- Data widgets
  - `ep/widget/ObjectSelect`
  - `ep/widget/DateField`
- Utils
  - `ep/Utils`
  - `ep/widget/smartclient/StandardDialog`
  - `ep/config!`

ep/Utils

- Collection of common utilities
- Message Utilities
  - `/*promise*/ showErrorMessage(/*string*/error)`
  - `/*promise*/ alert(/*string*/message,/*string?*/title)`
  - `/*promise*/ confirm(/*string*/message,/*object?*/config)`
  - `/*promise*/ yesNoCancel(/*string*/message, /*object?*/ config)`
  - `/*promise*/ prompt(/*string*/message, /*string?*/defaultValue)`

  Usage:

```
Utils.alert(„this is an alert!!“).then(function(){ … });
```

- Permissions
  - `/*promise*/ hasRight(/*string*/ rightId,/*string*/targetObjId)`

  Usage:

```
Utils.hasRight("edit", "com.groiss…:0815").then(
  function(result){
    alert(result.hasRight);
});
```

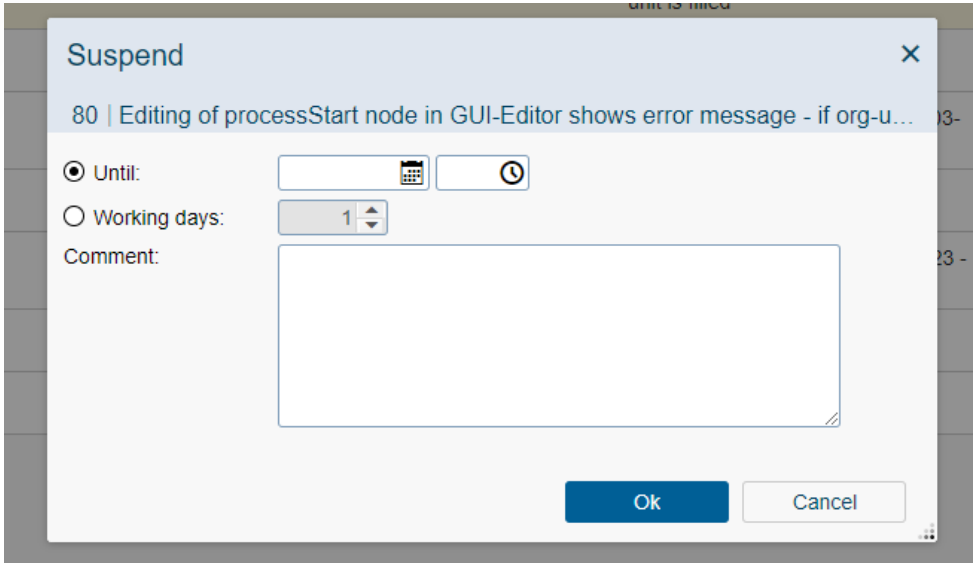
ep/Utils

- Worklist/notification
  - `/*void*/ refreshWorklists(/*object*/ content, /*boolean?*/ showFirst, /*boolean?*/ showDetailsOfFirstAdded, /*boolean?*/ selectAdded)`
    - content – object returned from com.groiss.smartclient.ClientUtil.getChangesAsJSON()
    - showFirst – focus the first worklist with new items
    - showDetailsOfFirstAdded – open the details for the first new item
    - selectAdded – select the new items in the worklists

- Formatting
  - `/*string*/ formatDateTime(/*long,Date*/date)`
  - `/*string*/ formatDate(/*long,Date*/date)`
  - `/*string*/ formatDecimal(/*number*/value)`
  - `/*string*/ formatPersistent(/*object*/object)`
  - `/*string*/ formatMessage(/*string*/message, /*array[string]*/values)`
    - `Utils.formatMessage(„The item {0} contains {1} new elements“,[„elak“,“2“]);`

StandardDialog

- `ep/widget/smartclient/StandardDialog`



- defines a resizable Dialog
- Attributes:
  - showOK: if true the dialog has an OK and a Cancel button otherwise a Close button
  - title: window title
  - subtitle: a second title below the title
  - content: a widget or HTML content
  - labelClose, labelOK, labelCancel: strings to modify the button-labels

StandardDialog (2)

Example:

```
var thePane = new Pane({
  onCommit: function(onComplete) {
    if (!this.validate()) { // content is a form
      return;
    }
    ...some actions
    onComplete(); // closes the dialog
  }
});
var dlg = new Dialog({
  title:"@@@see_later@@", subtitle: subtitleVar,
  content: thePane,
  showOk: true
});
```

Dialog calls the onCommit function of the content when OK button is clicked, it calls the onCancel function if Cancel/Close button is clicked

resize: if content has a resize function, it is called

if content contains HTML, the height of elements with css class "balloon" are resized.

ep/config!

- A custom AMD plugin that can be used to load server-side parameters
- Define property to be used on clientside (properties.xml)
  - `<property name="my.appl.property" allowOnClient="true"/>`
- Load plugin and access property

```
define(["dojo/_base/declare", "ep/config!"],
    function(declare,conf) {
console.debug(conf["applid:my.appl.property"]);
});
```

- Access default @enterprise properties
  - Certain properties are sent to the client – e.g. numberformat, dateformat etc.
  - `conf["DateFormat"]` – without any prefix

Date picker

- HTML Code

```
<input type="text" name="changetime"  id="changetime"
 showTime="false" value="" data-dojo-type="ep/widget/DateField"/>
```

- access the object

```
registry.byId("changetime")

registry.byId("changetime").set("value","1-1-2013");
registry.byId("changetime").get("value");
```

Object Selection

- HTML Code

```
<select name="substitute" id="substitute"
  data-dojo-type="ep/widget/ObjectSelect"
  searchid="myxml.DeptsWithSubdeptsSelect">
  value="['','']">
</select>
```

- Configuration in XML file

```
<nodes>
  ...
  <query id="DeptsWithSubdeptsSelect">
    <classname>com.groiss.org.Dept</classname>
    <attribs>name</attribs>
    <searchAttrs>name,id</searchAttrs>
    <title>@@@ep:dept@@</title>
    <condition>oid in (select superdept from
      avw_flatdepttree)</condition>
  </query>
```

JavaScript toolbar function

- Define in xml

```
<action id="approve2" name="@@@approve@@">
    <onClick>ep/widget/smartclient/demo/Approve</onClick>
    <apply>MULTI</apply>
</action>
```

- Reference node (by id) in worklist actions
- Write the JavaScript file Approve.js
- Write the server-side Java method

Example: Approve

```
define(["dojo/_base/declare",  "ep/widget/smartclient/_Action",
  "dojo/request", "ep/Utils"],
  function(declare, _Action, request, Utils) {
    return declare([_Action], {
      actionPerformed: function(evt) {
        request.post("com.groiss.demo.DemoFunctions.approve2",{
          handleAs: "json",
          data:{ object: this.getSelectedIds(), wlid: this.wlid }
      }).then(function(result) {
        Utils.refreshWorklists(result);
      });
    },
    isEnabled: function() {
      var selection = this.getSelection();
      if(selection.length==0) {return false;}
      for (var i = 0; i< selection.length; i++) {
        if (selection[i].pd.id != "demo_order") {
          return false;
        }
      }
      return true;
    }});
});
```

classes/alllangs/scripts/ep/widget/smartclient/demo/Approve.js

Example (2)

```
public Page approve2(HttpServletRequest req) throws Exception {
   User u = (User)ThreadContext.getThreadPrincipal();
   WfEngine e = WfEngine.getInstance();
   String[] tasks = req.getParameterValues("object");
   for (String aistr: tasks) {
        ActivityInstance ai = StoreUtil.getObject(aistr);
        DMSForm f = e.getForms(ai.getProcessInstance()).get(0);
        f.setField("approvedby", u);
        f.setField("approved", "1");
        e.updateForm(f);
        // may be in another folder
        e.propagateChange(ai);
   }
   // compute changes makes commit
   JSONObject result =
      ClientUtil.computeChanges(req.getParameter("nodeid"), null);
   return new JSONPage(result);
}
```

java/com/groiss/demo/DemoFunctions.java

- computeChanges computes the changes of the worklists triggered by this action
- because the action doesn't change the activity, wfe.propagateChange(ai) is necessary
- in the client code `Utils.refreshWorklists(result)` updates the worklists

Configure worklist layout - Column

- subclass `ep/widget/smartclient/grid/Column`
  - Is used to modify column layout in a grid
  - Possible functions are defined in dgrid docu

  - E.g.
    - renderCell(object,data,td,options)
    - renderHeaderCell(contentNode)
    - format(value)
    - get(object)
    - init()
    - …
  - Fields:
    - grid
    - field
    - label

Column Example

```
define(["dojo/_base/declare",
    "ep/widget/smartclient/grid/Column",
    "dojo/dom-construct"],
  function(declare,
      Column,
      domConstruct) {
    return declare([Column],{
      renderCell:function(object, data, td, options) {
        if (data) {
          td.innerHTML =
            "<div class='dijitIcon scIcon scSuccessful'></div>";
        }
      }
    });
});
```

14. Reporting

Customize reporting

- customized search masks
- functions on report results
- special column values and appearance
- other export formats

Reporting – Customize search mask

```
<script>
 function startSearch() {
       var searchString = document.getElementById("str_value").value;
       var searchOperator = document.getElementById("str_operator").value;
       var reportId =  document.getElementById("reportId").value;
       //call parent require  cause we need the registry of the main window
parent.require(["ep/widget/smartclient/reporting/ReportingResult2","dijit/registry","do
jo/_base/lang"],
          function (ReportingResult,registry,lang){
                    var stack  =registry.byId("_mainstack");
                    var reportingGrid = new ReportingResult({
                              query : {id: reportId},
                              postParams: {comesFromParamMask:"1",
                                        str_value : searchString,
                                        str_operator : searchOperator},
                              showToolbar: true,
                              additionalButtons:
                                        [{title:"@@@ep:back@@",label:"@@@ep:back@@",
                                        iconClass:"scIcon scBackward",
                                        onClick: function () {stack.back();}}]
                              });
                    stack.addChild(reportingGrid);
                    stack.selectChild(reportingGrid);});
}
```

classes/alllangs/demo/reporting/sc_demosearchmask.html

Reporting function

Description: function in toolbar of reporting result

Declaration: define the column containing the parameter value in the export options in the report definition

define the toolbar function in the report options mask (select task-function or enter xml-id)

Definition: signature like toolbar functions

the parameter `object` contains the value(s) in the defined column in the selected line(s)

if the defined column contains a reference to an object (user, process, ...) the object parameter contains the oid. the same applies to the Id column of process instance and activity instance.

Demos: `com.groiss.demo.reporting.DemoToolbarFunctions`

Reporting function – Example Client

```
define(["dojo/_base/declare", "ep/widget/smartclient/_Action",
"dojo/request",  "dijit/registry",  "ep/Utils"],
function(declare, _Action, request, registry, Utils) {
   return declare([_Action], {
     actionPerformed:function(evt) {
         //wizardId refers to reporting Grid
         var reportingGrid = registry.byId(this.wizardId);
         request.post("com.groiss.demo.reporting.DemoToolbarFunctions.abortP
         rocess",{
                 handleAs: "json",
                 data:{object: this.getSelectedIds()}
         }).then(function(data) {
                 if (data.result) {
                         Utils.showErrorMessage(data.result);
                 }
                 reportingGrid.refresh(); //refresh Grid
         });},
     isEnabled:function() {
         var selection = this.getSelection();
          return selection.length>0)
    }
});});
```

classes/alllangs/scripts/ep/widget/smartclient/demo/AbortProcesses.js

Reporting function – Example Server

```
public Page abortProcess(HttpServletRequest req) throws Exception {
  String result;
  String [] ai_oids = req.getParameterValues("object");
  if (ai_oids!=null && ai_oids.length>0) {
    WfEngine wfe = WfEngine.getInstance();
    for (int i = 0; i < ai_oids.length; i++) {
        ActivityInstance ai = StoreUtil.getObject(ai_oids[i]);
        ProcessInstance pi = wfe.getMainProcess(ai);
       wfe.abort(pi,"Aborted by Reporting Demo Function");
    }
    result = ai_oids.length + " Process(es) aborted!";
  }else {
    result = "No processes selected!";
  }

return new JSONPage(new JSONObject().put("result",result));}
```

java/com/groiss/demo/reporting/DemoToolbarFunction.java

Reporting data

Description: customize a column appearance

Definition: interface `ReportingExportable`, default implementation DefaultReportingData

```
public interface ReportingExportable extends
Comparable<ReportingExportable> {
  public int compareTo(ReportingExportable re);
  public String toText(); //export to files
  public Object toHtml(); //export to old client
  public Object getValue();
  public void setValue (Object val) ;
  public Object toJson(); // export to smartclient
  public String getColumnRenderer(); // client column widget
}
```

Declaration: define an attribute using the class in the application specific reporting.xml

In Product: implementations for Date, Persistent, Process, etc.

Demos: `com.groiss.demo.reporting.TrafficSignalDueDate`

Reporting data – Example code

```
public Object toJson() throws JSONException {
    Date duedate = (Date)getValue();
    String imgsrc = "../images/smartclient/status.png"; //normal state
    String altText = CalUtil.showDateTime(duedate);
    if (isOverdue()) {
         //duedate in past
         imgsrc = "../images/smartclient/status-busy.png";
    } else if (willExpire()) {//duedate in next 10 days
         imgsrc = "../images/smartclient/status-away.png";
    }
    return new JSONObject().put("imgsrc", imgsrc).put("text",altText);
    }

//returns client renderer which builds image tag
public String getColumnRenderer() {
         return "ep/widget/smartclient/demo/TrafficLightColumn";
}
```

java/com/groiss/demo/reporting/TrafficSignalDueDate.java

Reporting data – reporting.xml

```
<Schema xmlid="demo" name="DemoApplication">
  <!-- add new column to entity processInstance...-->
  <entity xmlid="processInstance" table="avw_stepinstance"
        tablealias="pi">
        <attribute xmlid="pi_coloredduedate"
        name="@@@ep:duedate@@ (with Trafficlights)"
        class="com.groiss.demo.reporting.TrafficSignalDueDate">
               <select>duedate</select>
        </attribute>
</entity>
</Schema>
```

Reporting exporter

Description: customize export formats

Definition: interface `com.groiss.reporting.export.ReportingExporter`

- Subinterface `ClientSideExporter` for smartclient support
- Subinterface `FileReportingExporter` for downloads and ReportingTimer support.

Declaration: define an exporter using the class in exporter mapping in the application specific reporting.xml

In Product: implementations for Excel, CSV, Charts, SVG-Charts, XML etc.

Demos: `com.groiss.demo.reporting.FileSystemExporter`

Reporting exporter – reporting.xml

```
<Schema xmlid="demo" name="DemoApplication">
<!-- add file system exporter to exporter mapping of default
reporting.xml -->
    <mapping xmlid="exporter">
        <mapentry key="fileystem"
        value="com.groiss.demo.reporting.FileSystemExporter" />
   </mapping>
…
</Schema>
```

Contact

Groiss Informatics GmbH
Strutzmannstraße 10
A- 9020 Klagenfurt
Austria
Tel: +43 463 504694

E-Mail: support@groiss.com
Web: www.groiss.com