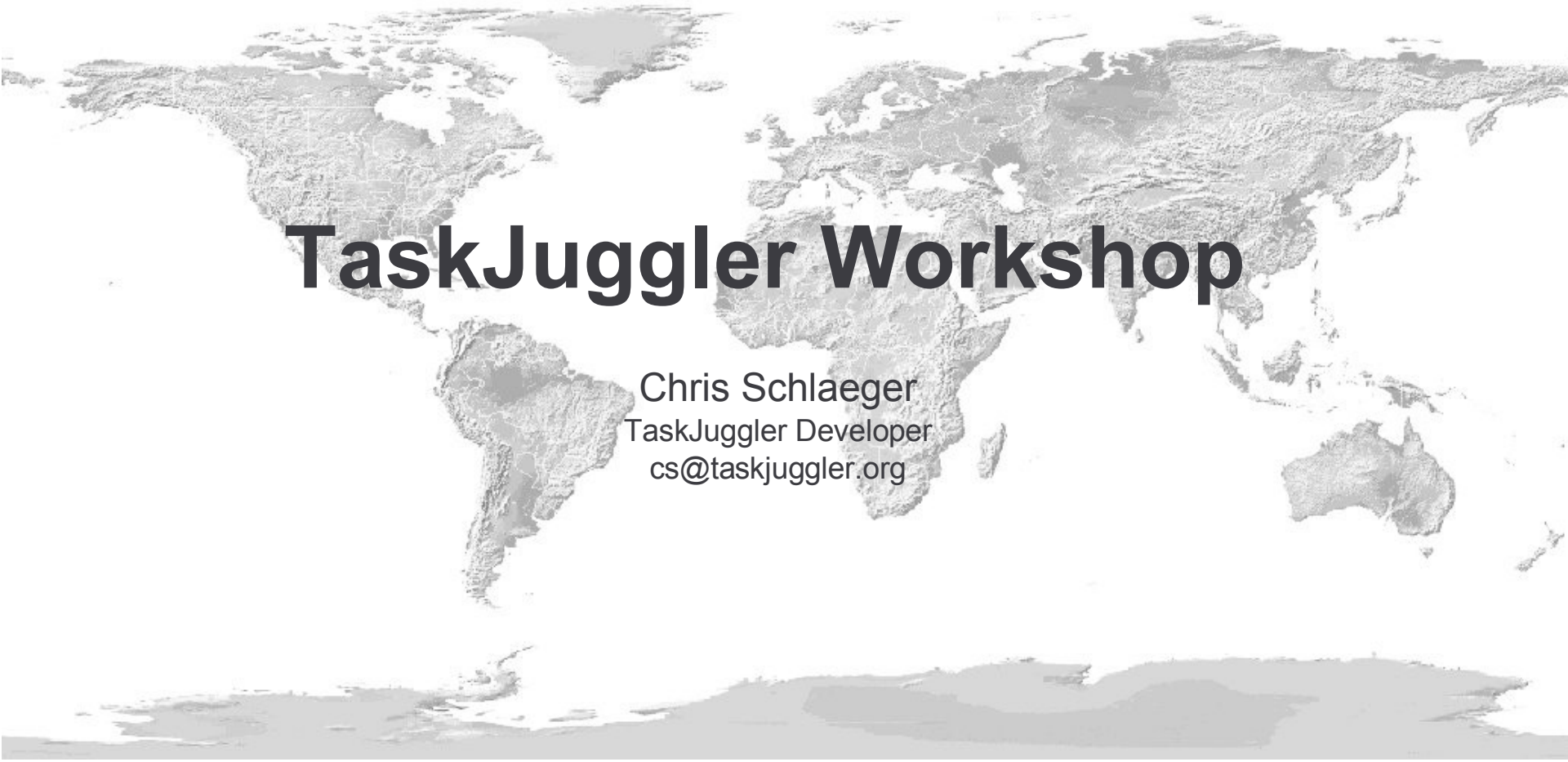


TaskJuggler Workshop

Chris Schlaeger
TaskJuggler Developer
cs@taskjuggler.org

Version: 1.1
Date: 2006-06-13

Workshop Agenda (Part I)

- What is TaskJuggler?
- Working with the User Interface
- A first Project Plan
- Generating a Report
- Defining Resources
- Defining Accounts
- Capturing the Work Breakdown Structure
- Using Shifts and Limits
- Using Task Priorities to control the Scheduling
- Working with Macros
- Generating Reports

Workshop Agenda (Part II)

- Working with Include Files
- Creating Custom Templates
- Advanced Reports
- Collaborating with other Projects and Project Managers
- Tracking the Project Status
- Documenting the Project Evolvement
- User defined Attributes
- Playing with multiple Scenarios

TaskJuggler Workshop (Part I)

What is TaskJuggler?

- Project was founded in 2001 by a small team of developers working for SUSE
- OpenSource project with internal and external contributors
- Website: http://www.taskjuggler.org
- Until 2004 it was command line only
- New graphical user interface was released in March 2005
- Active user community from academia and industry
- Discussion forums are available on the website
- TaskJuggler is not a Gantt chart editor.
- It's a project planning and tracking tool.

Working with the User Interface

Interactive Demonstration of the following features:

- Launching TaskJuggler from the menu
- Creating a new Project from a template file
- Access to the manual and the F2 keyword help
- Editor Settings
- Auto-completion and indentation
- Date insertion and modification with CTRL-D
- Scheduling a project
- Handling syntax and scheduling errors
- Explanation of the list browsers and the editor and report tabs
- Loading of an existing Project
- Browsing of various reports
- Demonstration of keyboard navigation to all GUI elements

Basic Components of a Project

- Project Header

```
project myProject “My Project” “1.0”
        2005-11-01 - 2006-03-31
```

- Task Definitions

```
task prjstart “Project Start” {
  start 2005-11-05
}
task step1 “Step 1” {
  depends prjstart
  duration 2w
}
```

Specifying a Task Duration

- Three methods of specifying a duration directly
  - Calendar time: `duration`
  - Working time: `length`
  - Resource time: `effort`
- Must be used with a unit

  `min, h, d, w, m, y`
- Examples:
  - `duration 1.5y` (1.5 calendar years)
  - `length 2w` (10 working days)
  - `effort 2w` (10 resource days)
- Conversion factors: `yearlyworkingdays`, `dailyworkinghours`

Exercise No. 1

Task:
Create a project plan consisting of 5 different tasks.

Time: 10 minutes

Steps

- Create a project with the “Blank” Template
- Enter 5 tasks with different task durations
- 4 should happen in sequence
- 1 should happen simultaneously with another task
- Schedule the project and make sure there are no errors
- The summary report should show a task count of 5

Generating a Report

- Available Format Types

  Interactive, HTML, CSV, XML, Export, iCalendar
- Available Content Types

  Task Reports, Resource Reports, Account Reports, Calendar Reports, Status Reports
- Definition of a Report

```
resourcereport “Resource List”

taskreport “Project Overview” {
  columns no, name, start, end, chart
}
```

Using Flags to mark Objects

- Use flags to mark certain groups of tasks.
- Flags must be declared before they can be used.
- Flag names are TaskJuggler IDs. They must consist only of letters, numbers and underscores.

```
flags important

task foo “Foo Task” {
  flags important
}
```

Filtering of Report Content

- Content of reports can be limited to the exact amount needed.
- Show only columns you are interested in
- Show only tasks you are interested in

```
hidetask important
```

- Show only resources you are interested in

```
hideresource ~team
```

- Sort content by up to 3 criteria

```
sorttasks tree, startup
sortresources nameup
```

Getting Help

- The TaskJuggler Manual
  - Available as PDF document
  - On the web: http://www.taskjuggler.org/docs.php
  - In the menu: Select Help->TaskJuggler Handbook or Press F1
  - Context sensitive keyword help:

    Move the cursor in the editor over any TaskJuggler keyword and press F2. You will get a detailed explanation of the keyword including it's meaning, context and other related information.

Structure of a TaskJuggler Project

- TaskJuggler processes files from top to bottom. So the order of things inside the file matter. You cannot reference properties, that haven't been defined yet. The following order is a good guideline:
  - Project Header
  - Macro definitions
  - Flag declarations
  - Shift definitions
  - Account definitions
  - Resource definitions
  - Task definitions
  - Bookings
  - Reports

Exercise No. 2

Task:
Generate a report for your first project.

Time: 5 minutes

Steps

- Look up the possible columns in the TaskJuggler manual
- Be sure to include the start and end date of tasks in the report
- Use flags to filter out 2 tasks
- Sort the remaining tasks by name from last to first

Defining Resources

- Resource definitions have similar format as task definitions
- Many attributes get inherited from enclosing scope.

```
resource team “Developer Team” {
  rate 300  # daily cost
  resource john “John Doe”
  resource wilma “Wilma Flintstone”
  resource paul “Paul McCartney” {
    rate 250
  }
}
```

Assigning Resources to Tasks

- Simple case: 1 task, 1 resource

```
task foo “Foo Task” {
  effort 5d
  allocate joe
}
```

- Allocating multiple resources

```
task foo “Foo Task” {
  effort 10w
  allocate wilma, paul, joe
}
```

Assigning Resources to Tasks (Cntd.)

- Allocating a team

```
task foo “Foo Task” {
  effort 5d
  allocate team
}
```

- Mandatory allocations

```
task foo “Foo Task” {
  effort 10w
  allocate wilma
  allocate projector { mandatory }
}
```

Assigning Resources to Tasks (Cntd.)

- Specifying alternative resources

```
task foo “Foo Task” {
  effort 5d
  allocate joe { alternative paul }
}
```

- Using the selection function

```
task foo “Foo Task” {
  effort 5d
  allocate wilma {
    alternative joe, paul
    select maxloaded
  }
}
```

Assigning Resources to Tasks (Cntd.)

- Making sure that a resource does not change once it has been selected

```
task foo “Foo Task” {
  effort 5d
  allocate wilma {
    alternative joe, paul
    persistent
  }
}
```

- Whichever resource is available first, does all of the work.

Resource Efficiency

- You can specify differences in the ability to do work by using the `efficiency` attribute. Use with care as this can lead to micromanagement!
- You can model an opaque team of 5 people

```
resource team “5 Guys” {
  efficiency 5.0
}
```

- You can model resources that don't do any work

```
resource projector “Projector” {
  efficiency 0.0
}
```

Timing Resolution

- By default TaskJuggler uses 1 hour time slots.
- This can be changed using the timingresolution keyword.
- All time values are always rounded to a timing resolution boundary.
- Changing this value has a massive influence on performance and memory consumption.

```
project myproject “My Project” “1.0”
        2005-11-04 - 2006-03-06 {
  timingresolution 10min
}
```

Commenting your Project

- Comments allow you to put any additional information about your project right into the project file.
- There are 2 types of comments:
  - Single line comment: `# Some wise words`
  - Multi-line comments:

```
/* A whole lot of
 * more wise
 * words */
```

- Comments can be used to temporarily disable certain parts of the project file.

Exercise No. 3

Task:
Plan 2 meetings for 2 different teams each so TaskJuggler prevents conference room conflicts.

Time: 10 minutes

Steps

- Define 2 teams with at least 3 members each
- Define 2 conference rooms
- Define the meeting tasks each lasting 2.5 hours
- Allocate the teams and meeting rooms appropriately.

Short Break

Please be ready to continue in 5 Minutes!

Name Spaces

- TaskJuggler properties like Tasks, Accounts and Resources have separate name spaces.
- It's ok to have a task and a resource with ID foo.
- The Task name space is hierarchical. All other name spaces are flat.
- There can be a task foo.foo, but not a resource.

```
task foo “Foo” {
  task foo “Foo”    // This is ok!
}
resource foo “Foo” {
  resource foo “Foo” // This is an error!
}
```

Defining Accounts

- Format for Account definitions

```
account acc1 “Cost Accounts” cost {
  account salaries “Salary cost”
  account material “Material cost”
}
account acc2 “Revenue Accounts” revenue {
  account payments “Customer Payments”
}
```

Crediting Costs to Accounts

- Cost Type 1: Running costs

```
task job “A Job” {
  effort 2w
  allocate joe
  account salaries
}
```

- Cost Type 2: Start or End payments

```
task payment “Customer Payment” {
  start 2005-12-24
  milestone
  startcredit 50000
  account payments
}
```

Accounting Reports

- Currency unit must be defined in the project header

```
project myProjectId "My Project" "1.0"
      2005-11-02 - 2005-12-31 {
  currency "$"
  currencyformat "(" ")" "," "." 2
}
```

- Accounting Reports are currently available in HTML or CSV

```
htmlaccountreport “Project-Cash-Flow.html” {
  columns no, name, total, monthly
  accumulate
}
```

Exercise No. 4

Task:
Calculate the P&L for a 3 year project taking development costs and expected customer payments into account.

Time: 15 minutes

Steps

- Define the project outline with 4 consecutive tasks.
- Define an opaque team of 5 people and assign them to all tasks
- Define 3 customer payments as milestones.
- Define cost and revenue accounts.
- Create an accounting report that shows your cash flow.

Milestones

- Tasks that don't have any kind of duration specification are called milestone.
- You can mark a task to be a milestone, but it still must not have any duration specification.

```
task foo “Foo Task” {
  start 2005-11-04
}
task foo “Foo Task” {
  start 2005-11-04
  milestone
}
```

Capturing the Work Breakdown Structure

- Start with a project outline

```
task project “My Project” {
  start 2005-11-01
  task plan “Planning Phase” {
    task prd “Write PRD” {
      duration 2w
    }
  }
  task dev “Development Phase”
  task qa “Testing Phase”
  task rel “Release Phase”
  task maint “Maintenance Phase”
}
```

- Then add details as you learn them.

Specifying Task Dependencies

- Task B depends on Task A

```
task a “Task A” { start 2005-11-01 }
task b “Task B” { depends a }
```

- Task A precedes Task B

```
task a “Task A” { precedes b }
task b “Task B” { start 2005-11-01 }
```

- Relative and absolute Dependency Specifications

```
task p “Project” {
  task a “Task A” { start 2005-11-01 }
  task b “Task B” { depends !a }
  task c “Task C” { depends p.b }
}
```

Scheduling Direction

- Attributes that cause ASAP scheduling:
  - `start, depends`
- Attributes that cause ALAP scheduling:
  - `end, precedes`
- Explicit specification of the scheduling mode:
  - `scheduling asap`
  - `scheduling alap`
- The last implicit or explicit specification rules

Avoid mixing ASAP and ALAP tasks in the same project! Either plan everything from start to end or vice versa.

Logical Expressions for Filters

- Logical Filter expressions consist of flags, functions and operators
- Supported operators (subset):

  & (and), | (or), ~ (not), > (larger), < (smaller)
- Expressions can be grouped with parentheses
- Many query functions supported

  e.g. `isChildOf(ID), isMilestone(), treeLevel()`
- Example:

```
taskreport “Task Overview (Important ones)” {
  rolluptask (treelevel() > 1) & ~important
}
```

Exercise No. 5

Task:
Create a release plan for a software project that outlines the project phases. Generate a report with only the important milestones.

Time: 15 minutes

Steps

- All tasks should be relative to a start milestone
- Start with an outline of the phases
- Add more detailed tasks
- Mark important milestones
- Generate the report
- Then change the start date and watch the impact on the report.

Short Break

Please be ready to continue in 5 Minutes!

Defining Working Hours

- Global definition in the Project Header

```
project myprj “My Project” “1.0” 2005-11-01 -
  2006-04-01 {
  workinghours {
    mon 8:00 – 12:00, 13:00 – 17:00
    tue 8:00 – 12:00, 13:00 – 17:00
    wed 8:00 – 12:00, 13:00 – 17:00
    thu 8:00 – 12:00, 13:00 – 17:00
    fri 8:00 – 12:00, 13:00 – 17:00
    sat off
    sun off
  }
}
```

Defining Working Hours (Cntd.)

- Different working hours for some resources

```
resource team “Team” {
  workinghours {
    mon off
    fri 8:00 – 12:00
  }
  resource john “John Doe” {
    workinghours {
      mon 13:00 – 17:00
    }
  }
  resource wilma “Wilma Flintstone”
}
```

Defining Shifts

- Shifts are defined sets of working hours per week

```
shift fullTime “Full Time Shift” {
  workinghours mon 8:00 – 12:00, 13:00 – 17:00
  workinghours tue 8:00 – 12:00, 13:00 – 17:00
  workinghours wed 8:00 – 12:00, 13:00 – 17:00
  workinghours thu 8:00 – 12:00, 13:00 – 17:00
  workinghours fri 8:00 – 12:00, 13:00 – 17:00
  workinghours sat off
  workinghours sun off
  shift partTime “Part Time Shift” {
    workinghours mon off
    workinghours wed off
  }
}
```

Using Shifts

- Shifts are used to modify standard working hours during specified intervals

```
resource wilma “Wilma Flintstone” {
  shift partTime 2005-12-01 – 2006-01-01
}
```

- Shifts can be used to limit resource allocations to a task during certain intervals

```
task foo “Foo Task” {
  effort 2w
  allocate john
  shift partTime 2005-12-01 – 2006-01-01
}
```

Limiting Resource Usage per Interval

- Limits can be used to limit the usage of a resource or an allocation of resources to a task to a certain maximum per interval. Supported intervals are day, week or month.

```
resource john "John Doe" {
  limits { dailymax 2h weeklymax 6h }
}
task foo "Foo Task" {
  duration 60d
  allocate r2 {
    limits { weeklymax 3d monthlymax 2w }
  }
}
```

Using Task Priorities to control the Scheduling

- The priority attribute controls the probability that a tasks gets the allocated resources
- The default priority is 500

```
task secUpds “Security Updates” {
  duration 2m
  allocate paul
  limits { dailymax 1h }
  priority 700
}
task calls “Handle customer calls” {
  duration 2m
  allocate paul
  priority 300
}
```

Exercise No. 6

Task:
Create a shift plan for the next 4 weeks for a team of system administrators.

Time: 15 minutes

Steps

- Define the various tasks of the system administration group
- Define your team
- Prioritize the tasks and allocate the resources
- Generate a shift plan for one of the team members that only list his or her tasks
- Generate an overview plan that shows all assignments

End of TaskJuggler Workshop (Part I)

TaskJuggler Workshop (Part II)

Workshop Agenda (Part II)

- Working with Include Files
- Creating Custom Templates
- Advanced Reports
- Collaborating with other Projects and Project Managers
- Tracking the Project Status
- Documenting the Project Evolvement
- User defined Attributes
- Playing with multiple Scenarios

Working with Macros

- Macros are somewhat flexible text fragments that can be inserted multiple times once they have been defined.
- Macro names must have at least one uppercase letter
- Definition of a Macro

```
macro allocateGrp [ allocate john, wilma ]
```

- Using a defined Macro

```
task foo “Foo Task” {
  effort 20d
  ${allocateGrp}
}
```

Working with Macros (Cntd.)

- Parts of macros can be replaced during insertion time by using parameters.
- Definition of a Macro with parameters

```
macro defTask [ task ${1} “${1} Task” ]
```

- Calling a Macro with parameters

```
${defTask “foo”}
${defTask “bar”}
```

- Result of the expanded Macros

```
task foo “foo Task”
task bar “bar Task”
```

Creating Custom Templates

- TaskJuggler comes with several custom templates but the you can add your own templates as well
- Custom templates need to be put into

  ${HOME}/.kde/share/apps/taskjuggler/templates/en_US
- The project start and end date can be automatically set to the current date (and current date + 180 days) when using `@@projectstart@@` and `@@projectend@@` instead of the dates in the templates.

Working with include Files

- To include another file into your project file, put an include statement into your project:

```
include “sometasks.tji”
```

- Project files must have a .tjp extension, include files must have a .tji extension.
- Tasks in the include file can be included as sub-tasks of some other task.

```
include “sometasks.tji” { taskprefix foo }
```

- Include statements may only be used in the project header or outside of all property definitions.
- `supplement` keyword can be used to add attributes to already defined properties

Exercise No. 7

Task:
Take the release plan project and break it into several files.

Time: 15 minutes

Steps

- Create an include file for the resource definitions divided into 2 teams
- Add some sub tasks of one tasks into 2 additional include files
- Allocate some resources so that one team is allocated in each of the include files
- Learn how to navigate the project with the browsers

Advanced Reports

- Good reports show exactly the amount of information you want to show. Nothing more and nothing less.
- TaskJuggler supports many filter mechanism to limit the reports to the right amount of data
  - Show only the `columns` that matter
  - `hidetask, hideresource, hideaccount`
  - `rolluptask, rollupresource, rollupaccount`
  - limit the report interval with `start` and end dates
  - values are reported in the right format and unit:
    - `loadunit, timeformat, shorttimeformat, barlabels, showprojectids`
- In tree-mode sorting parents are always included, no matter what the filters say. Use a different sorting mode to avoid this if undesired.

Advanced Reports (Cntd.)

- The default scenario ID needed for some query functions is `plan`
- In tree-mode sorting parents are always included, no matter what the filters say. Use a different sorting mode to avoid this if undesired.

Excluding details from reports

- Limiting the report period

```
taskreport “Task List” {
  period 2001-12-01 +2w
}
```

- Excluding tasks or resources

```
hidetask <LOGICAL EXPRESSION>
rolluptask <LOGICAL EXPRESSION>
hideresoruce <LOGICAL EXPRESSION>
rollupresource <LOGICAL EXPRESSION>
```

Customizing Column Headers and Cells

- The default column title can be replaced

```
taskreport “Task List” {
  columns no, name, effort { title “Work” }
}
```

- In HTML reports links can be added to headers and table cells

```
htmltaskreport "LinkURL.html" {
  columns hierarchindex, name,
          monthly { subtitleurl "Monthly-Detail-
   $${month}.html" }
}
htmltaskreport "LeafEfforts.html" {
  columns hierarchindex, name,
          effort { hidecelltext ~isLeaf() }
}
```

Adding Information to Reports

- Adding a Headline

```
taskreport “Task List” {
  headline “The tasks of my project”
}
```

- Adding a Caption

```
htmlresourcereport “Resources.html” {
  caption “List of all the hard working men and
  women on the project.”
}
```

- Adding a copyright (must be done in the header)

```
project myPrj “My Project” “1.0” 2005-11-01 -
 2006-04-01 {
  copyright “2005 Big Business, Inc.”
}
```

Advanced HTML Reports

HTML Reports can be customized by adding an inline stylesheet, and HTML fragments at the top and bottom of the report.

```
rawhead
  '<table align="center" border="2" cellpadding="10"
    style="background-color:#f3ebae; font-size:105%">
  <tr>
    <td><a href="Tasks-Overview.html">Tasks Overview</a></td>
    <td><a href="Staff-Overview.html">Staff Overview</a></td>
    <td><a href="Accounting.html">Accounting</a></td>
    <td><a href="Calendar.html">Calendar</a></td>
  </tr>
    </table>
  <br>'
```

CSV (Colon Separated Values) Report

- CSV is a simple text-form exchange format to export data to office suites like OpenOffice.org
- CSV reports are available in 3 types
  - `csvtaskreport`
  - `csvresourcereport`
  - `csvaccountreport`
- Right click on a CSV report in the report browser and select “Generate Report”
- Then the resulting report file can be loaded with OpenOffice.org. Use “comma” as a separator.
- Set the mime-type definition of `text/x-csv` to OpenOffice.org to automatically lauch OOo from the report browser

Exercise No. 8

Task:
Generate several different reports for your project.

Time: 10 minutes

Steps

- Generate an HTML task report that only contains tasks allocated to team 1
- Generate a CSV report with all tasks and efforts and import it into OpenOffice.org

Short Break

Please be ready to continue in 5 Minutes!

Tracking the Project Status

- TaskJuggler is helping you a lot when tracking your project status. If no other information is provided, it assumes that all tasks have progressed as planned.
- Simple way to provide status information

```
task foo “Foo Task” {
  effort 2w allocate john
  complete 75
}
```

- Detailed way to provide status information

```
supplement resource john {
  booking 2003-06-08 2003-06-09 t1 { sloppy 2 }
  booking 2003-06-11 2003-06-12 t1 { sloppy 2 }
}
```

Generating Export Files

- TaskJuggler can export a scheduled project in the same text format that the unscheduled project was provided in.
- The amount of information that is exported can be controlled by the `properties` attribute.
- E.g. export only the resource bookings for a certain week:

```
export "Week1Bookings.tji" {
  properties bookings
  start 2000-01-01
  end 2000-01-08
}
```

Scheduling in Projection Mode

- The bookings up to the current date may vary from the original plan.
- TaskJuggler can then schedule a new plan based on the amount of work that has happened already.
- The “current” date is user configurable.

```
project prj "Project" "1.0" 2003-06-05 -
        2003-07-05 {
  now 2003-06-15
  scenario plan "Plan" {
    projection
  }
}
```

Generating Status Reports

- Status reports are only available in HTML format
- Status reports include the following items
  - Tasks that should have been finished already
  - Work in progress
  - Tasks that have been completed
  - Upcoming new tasks

```
htmlstatusreport “StatusReport-Week45.html”
```

Exercise No. 9

Task:
Generate a status report report for the project 4 weeks after the start.

Time: 15 minutes

Steps

- Export bookings for the first 4 allocated weeks
- Remove or modify some bookings and include the export file in your project file
- Generate a status report for the week 4 weeks after the first work started
- Generate a new project plan based on the currently completed work

Working with multiple Project Plans

Export reports can be standalone project files or includeable sub-projects depending on the filename extension used in the report definition.

```
export "SubProject.tji" {
  taskattributes all
  hideresource 0
}
export "FullProject.tjp" {
  taskattributes all
  hideresource 1
}
```

Different Export Files

- *.tjp Export files have a project header
- *.tji Export files have no project header
- The other content can be controlled by `properties` and the usual filter mechanisms
- The following properties are supported:

  `all, bookings, shifts, tasks, resources`

Dealing with multiple Project IDs

- All tasks of sub-projects keep their original project ID
- To include a sub-project, the project ID needs to be declared first

```
projectids myproject1, myproject2
```

- Export reports (*.tji) already contain this declaration

Exporting a sub-Project

- Parts of a project can be turned into a standalone Project file.
- Use `taskroot` to identify the sub-tree you want to export. All sub tasks of the root task will be exported.

```
export “SubProject.tjp” {
  taskroot myproject.jimsWork
  taskattributes all
  hideresource 0
}
```

- Use `hidetask` to export only certain sub-tasks of the root task

```
hidetask ~isChildOf(myproject.jimsWork.foo)
```

Exercise No. 10

Task:
Combine 2 independent project into a summary project.

Time: 15 minutes

Steps

- Turn the 2 task files into independent projects
- Export the scheduled projects
- Include them into a summary project
- Generate a summary report

Short Break

Please be ready to continue in 5 Minutes!

Documenting the Project Evolvement

- Adding notes to tasks

```
task foo “Foo Task” {
  note “This might be difficult.”
}
```

- Adding status notes to tasks

```
task foo “Foo Task” {
  statusnote “We have unexpected problems.”
}
```

- Keeping a Journal of events

```
task foo “Foo Task” {
  journalentry 2005-10-20 “We ran into problems.”
  journalentry 2005-10-21 “Informed customer.”
  journalentry 2005-10-25 “Problem solved.”
}
```

User Defined Attributes

- The attribute set of tasks, resources and accounts can be extended by the user.
- There are two types of attributes available

  Strings and URLs
- User defined attributes do not impact the scheduling. They are for documentation and reporting purposes only.
- They can be used in all reports like the build-in attributes.
- User Defined attribute IDs must start with a capital letter

User Define Attributes (Cntd.)

```
project ca "Custom Attributes" "1.0" 2003-05-28 -
        2003-06-28 {
  extend task {
    reference MyLink "My Link"
    text MyText "My Text"
  }
}

task t "Task" {
  start 2003-05-28
  MyLink "http://www.taskjuggler.org" { label "TJ Web"
  }
  MyText "TaskJuggler is great!"
}
```

Playing with multiple Scenarios

```
project prj "Example" "1.0" 2005-05-29 - 2005-07-01 {
  scenario plan "Planned Scenario" {
    scenario actual "Actual Scenario"
    scenario test "Test Scenario" {
      disabled
    }
  }
}

task t "Task" {
  start 2005-05-29
  actual:start 2005-06-03
  test:start 2005-06-07
}
```

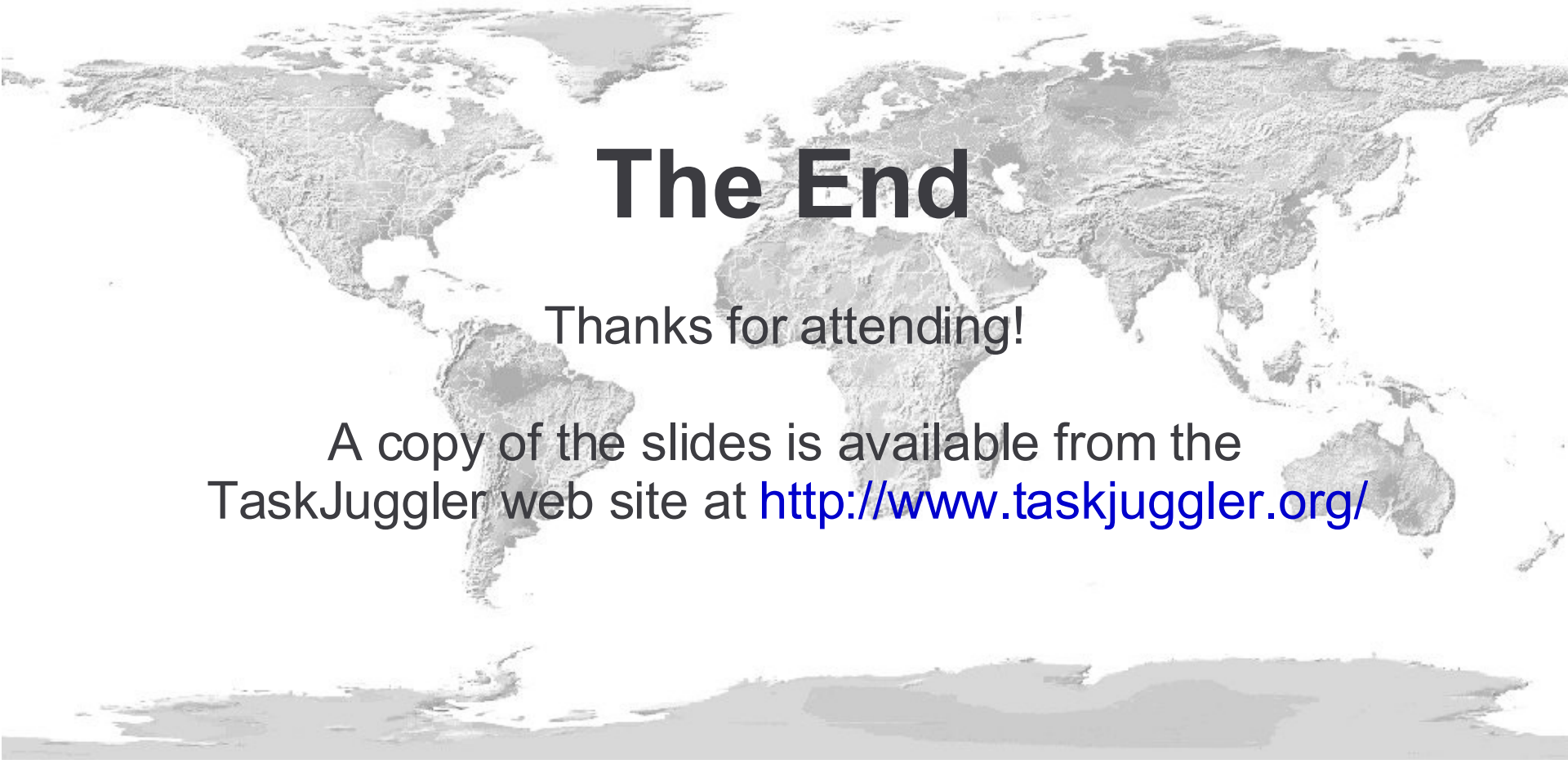
Exercise No. 11

Task:
Add another slightly different scenario to the project plan from Exercise 7.

Time: 15 minutes

Steps

- Create an additional scenario definition
- Look in the manual for scenario specific values
- Add a few changes for the 2nd scenario
- Add an email attribute to your resources
- Generate an HTML task report that compares both scenarios
- Generate a list with all resources including their email address



The End

Thanks for attending!

A copy of the slides is available from the TaskJuggler web site at http://www.taskjuggler.org/