

Assignment 1

VU Entwurfsmethoden für Verteilte Systeme

SS2010

Group evs024

Team members:

- Matthias Steinböck, 0527943, 535

Table of contents

HOW TO DEPLOY THE SOLUTION	1
HOW TO TEST THE SOLUTION	2
HOW THE SOLUTION WORKS - PATTERNS	2
HOW THE SOLUTION WORKS – CLASS DIAGRAMM	3
HOW THE SOLUTION WORKS – SEQUENCE DIAGRAMM	4

How to deploy the solution

Requirements: To deploy the solution, a running tomcat-server is required.

The solution comes with an ant build-script. This ant-script supports the following commands:

- clean – deletes everything created by compiling the solution
- compile – compiles all java-files and creates a deployable war-file
- run – deploys the war file to a running tomcat
- stop – removes the deployed service

The ant-script „run“ deploys the created ex1.war to a running tomcat-server. It is deployed to / (root).

How to test the solution

The solution comes with a doc-directory. In this directory you find this document with the source-files for the diagrams and a shell-script called „test.sh“. It contains runnable curl-commands to test the „User“-Resource. The same is possible with the required resources „Item“, „Rack“ and „Placement“.

Furthermore you can use the more comfortable RESTClient which provides a GUI and is available for several OS'es: <http://code.google.com/p/rest-client>.

How the solution works

Packages

The framework uses three main packages and two subpackages. The three main-packages are:

- framework: all necessary classes to run the default behaviour are located here
- models: here the user can put his pojos. They must not use primitive types [required update-behaviour]
- services: the user can put his services here (see annotations for further explanation)

Patterns

I chose the following remoting patterns: Server Request Handler, Invoker and Marshaller (realized as Strategy). These are the three Basic-Patterns described in the lecture for the server-side. They separate the code so that each component gets interchangeable.

Helper

Furthermore I created two helper-classes called „Context“ and „ServiceAndModelMapper“.

The Context-class can be used by the service to access the database or to get the requested ID or the sent object as the requested communication-method (JSON or XML).

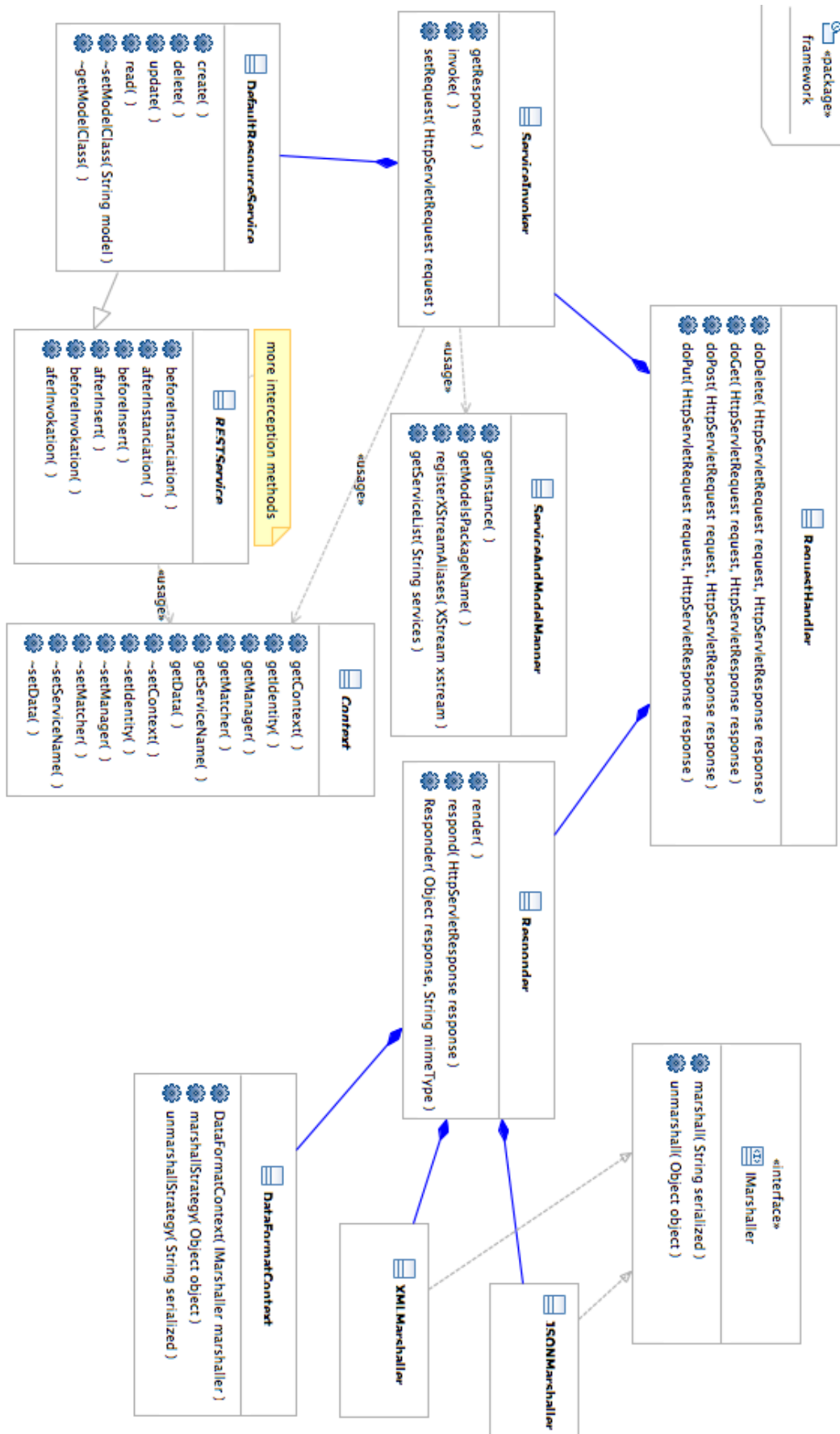
The ServiceAndModelMapper is used by the framework itself to look the models and services up. It registers the DefaultResourceService for every found model.

Annotations

Annotations are used extensively by the framework. If you create a service and place it in the services-package it can be found by the framework. The following annotations can be used:

- Service(url=„Item“) – this is a class-annotation and defines the basis-url the service can be found under. All models are registered under their simple name (case sensitive). If you overwrite the default-urls for the models, you have to extend the DefaultResourceService and then you can overload the interceptors provided by RESTService or add additional functions.
- Get(regexp=„pattern“) – this method-annotation defines that the following function is possibly used by GET-requests. If the URL the client used matches with the pattern, this function is used (first method matching is used, if you extend DefaultResourceService, your Service is checked first)
- Post, Put and Delete – these method-annotations work the same way as the Get-annotation but only for other HTTP-Methods

EVS2010



How the solution works – sequence diagramm

