



How to port an application to GAP



WeiLong Ueng
Academia Sinica Grid Computing

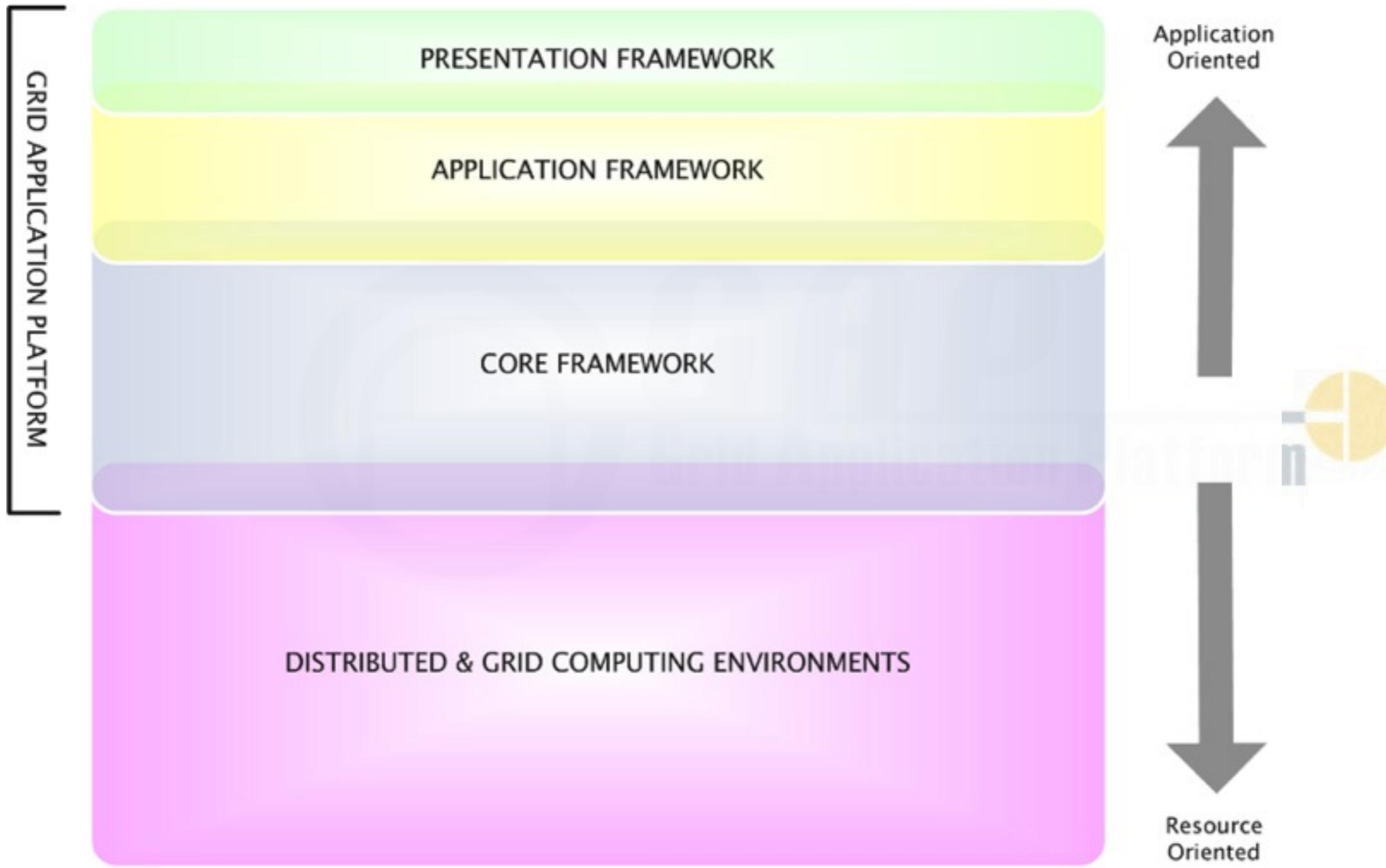


Outline

- GAP Service Architecture
- Adding new utility application
- An Example : SHELL utility application
- Use SHELL utility application in CORE Framework
- Implement Corresponding Logic in Application Framework
- How to use deployed application in your program
- Summary

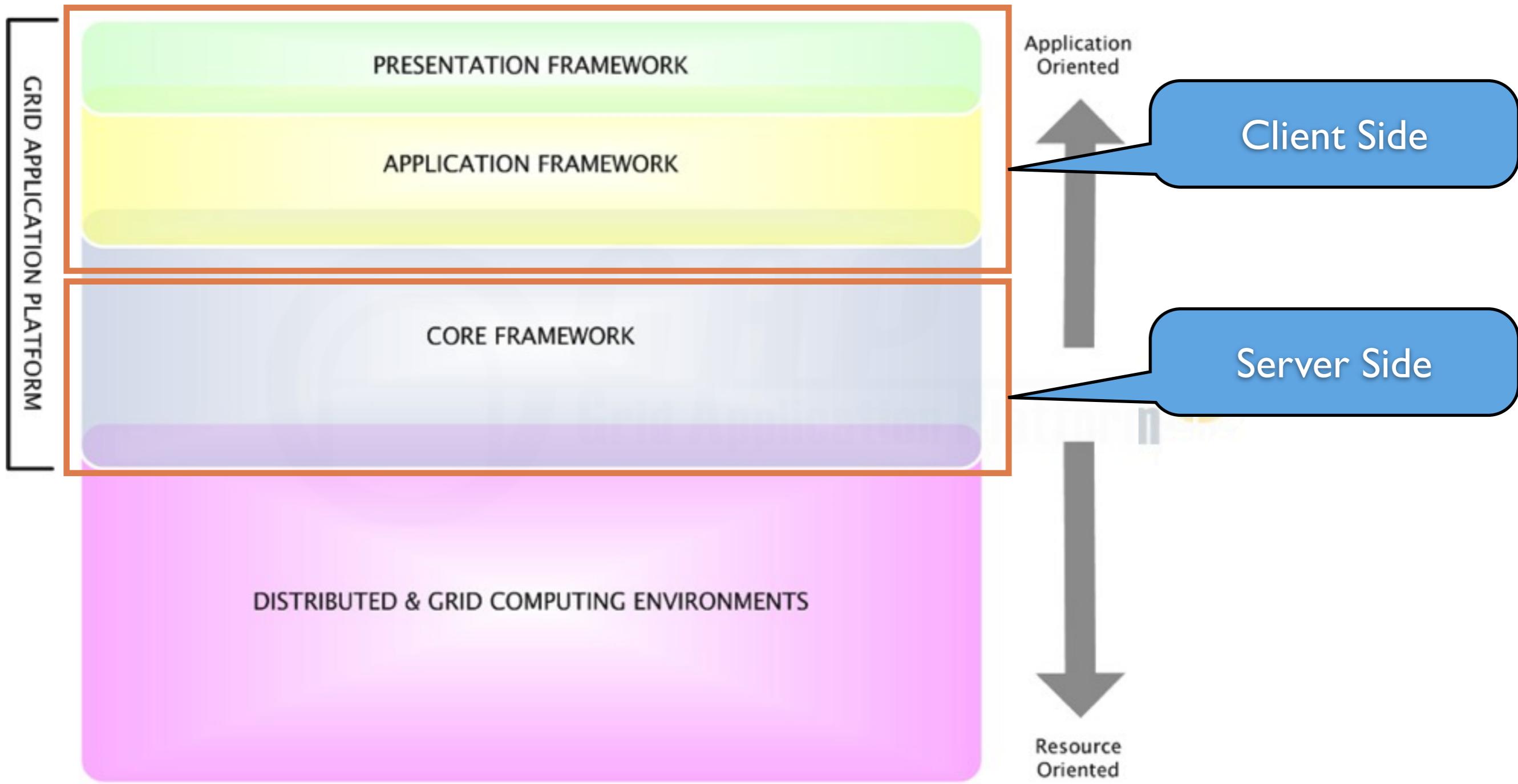


GAP Service Architecture



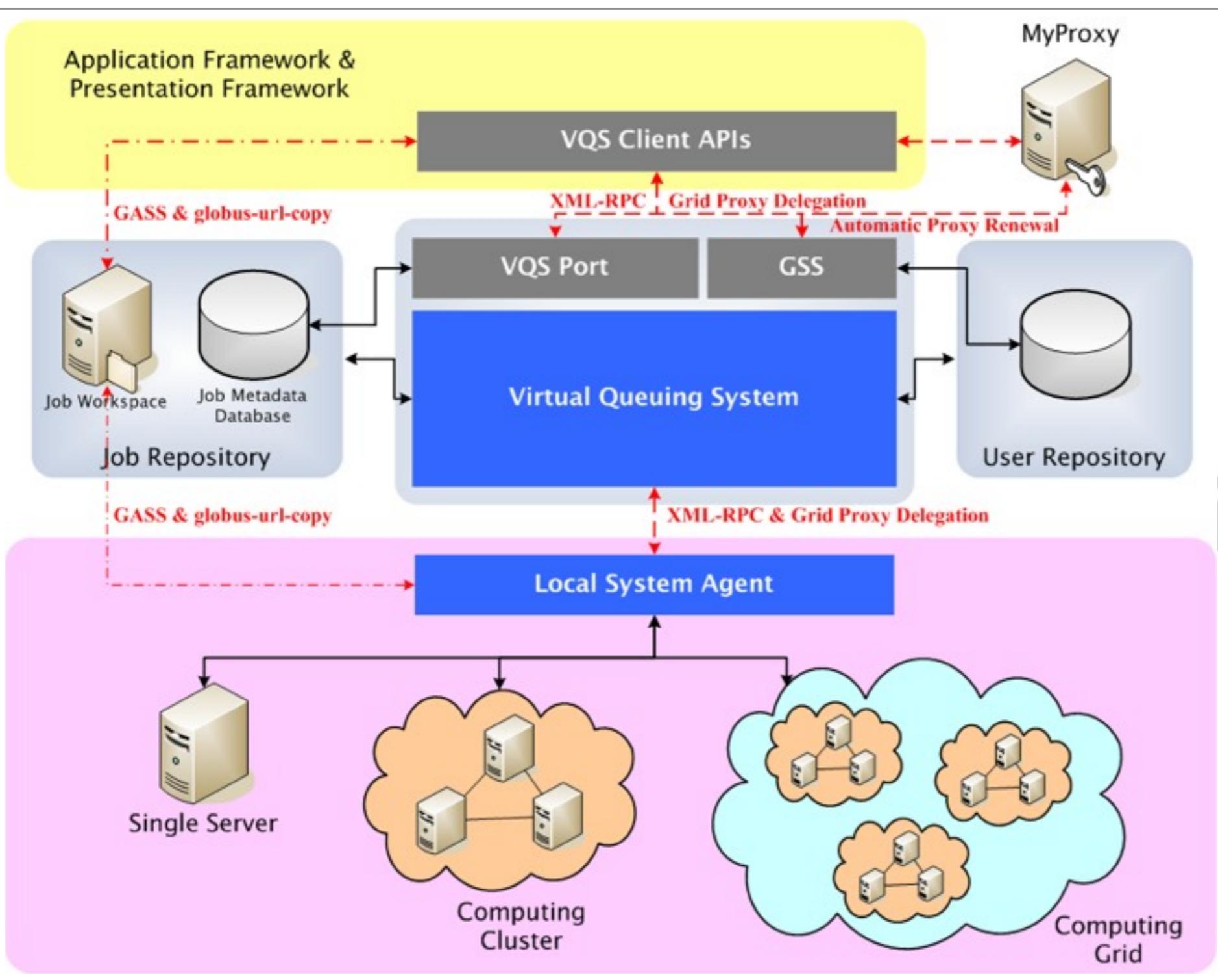


GAP Service Architecture





GAP Service Architecture



• Core Framework

- ▶ **LSA** (Local System Agent) is responsible for dealing with the underlying computing environment.

- ▶ **VQS** (Virtual Queueing System) directly faces the client and communicate with LSA.

- ▶ **VQSCClient** - The client APIs for communicating with VQS

• Application Framework

- ▶ A set of common APIs for handling advanced application logic

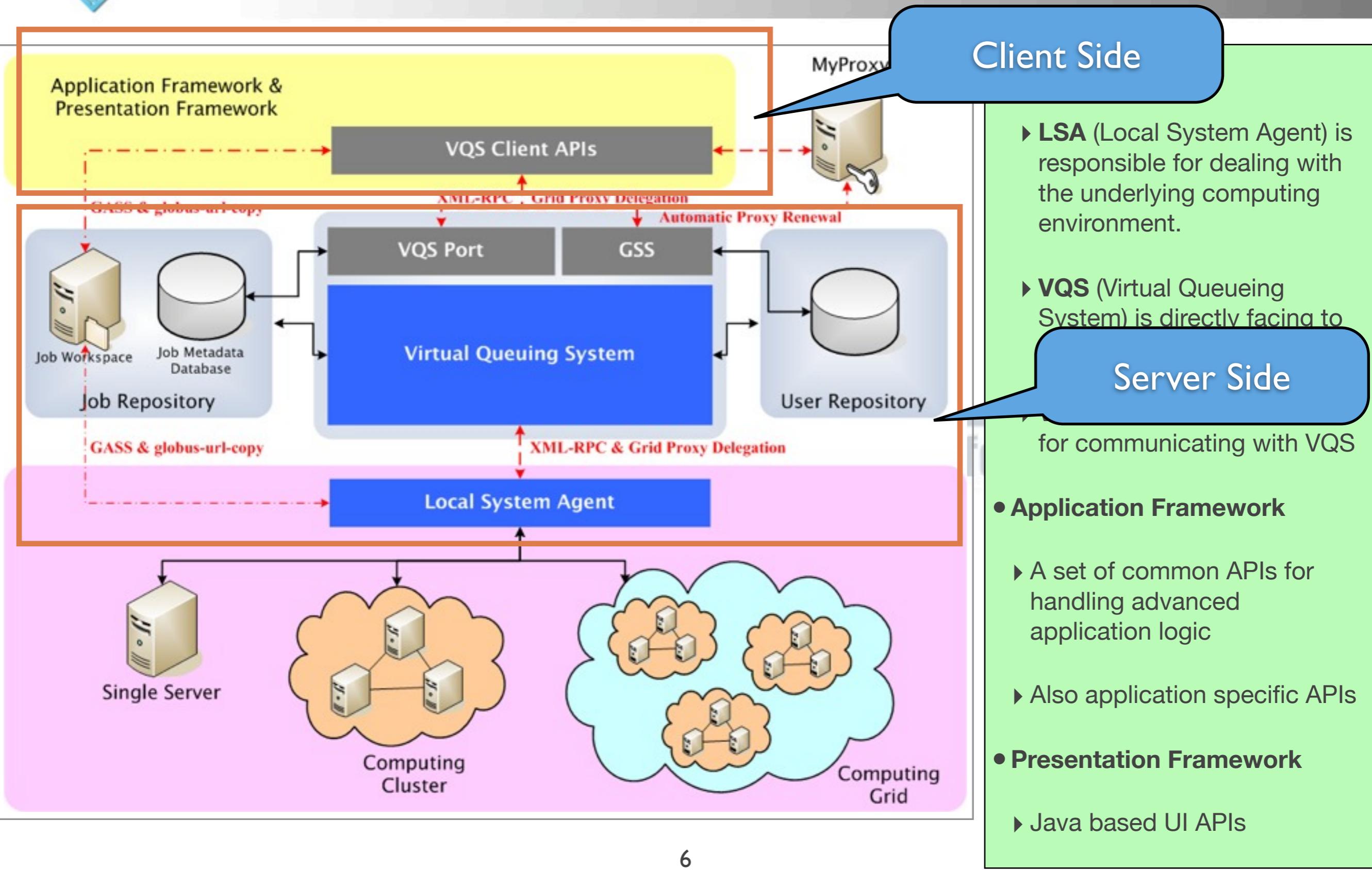
- ▶ Also application specific APIs

• Presentation Framework

- ▶ Java based UI APIs

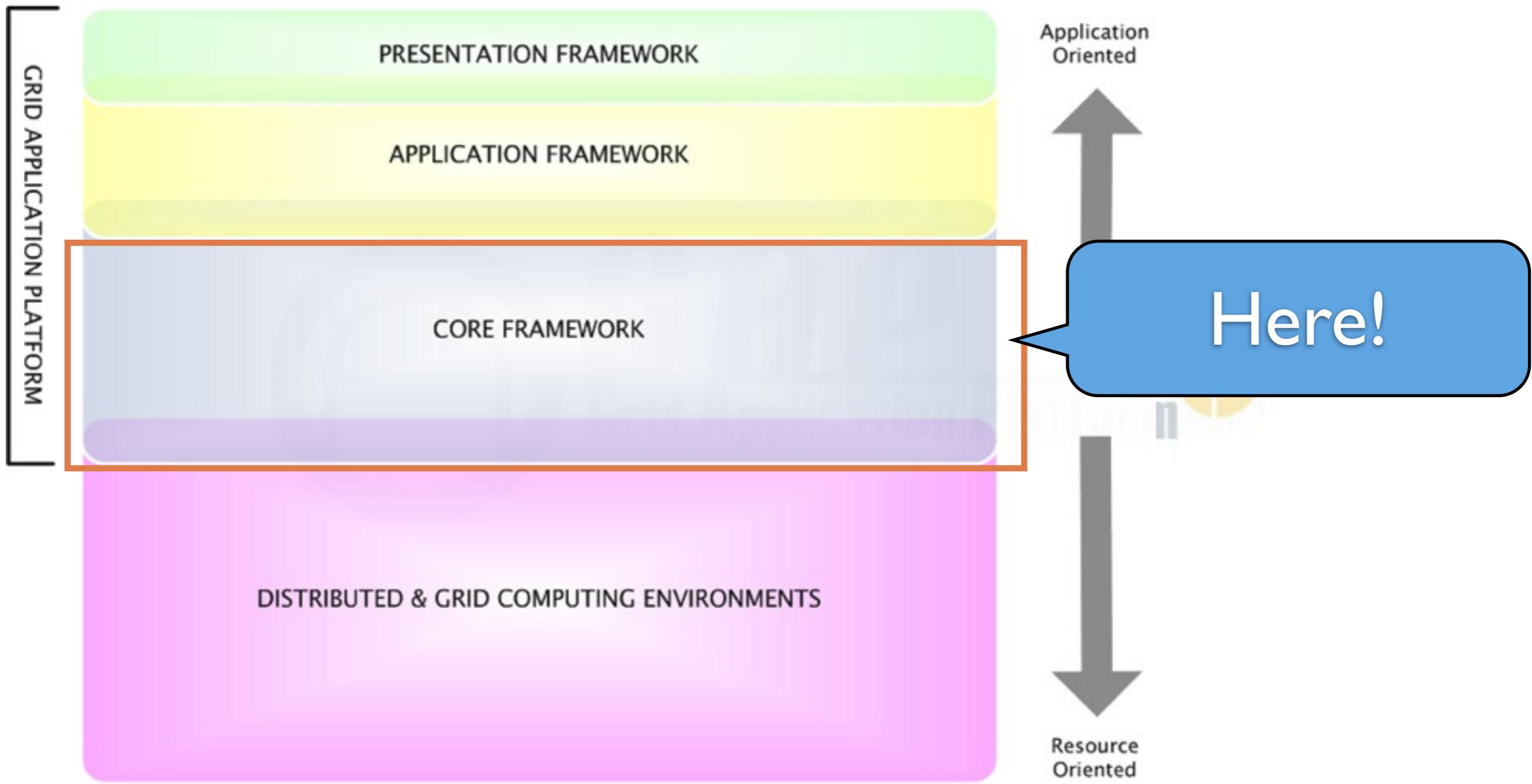


GAP Service Architecture





Adding new utility application





Adding new utility application

1. Design and implement the *Utility Application* on **LSA**
 - in reality is a simplified executable (e.g. a shell script) taking into account the execution logic runs on a specific environment
 - describe it with an **app.xml** and index it in the **app.list**
2. Register the utility application on **VQS**
 - add entries in the table **appdb_applist**
3. Follows the development guide-line of the Application Framework to build an *Advanced Application*
 - or extends the **CommandLineApplication** class to build a command-line based interface for running this new application



An Example : SHELL utility application (1)

- The **SHELL** Utility Application allows to run an arbitrary bash script on LCG
- it's equivalent to
 - % `source <the_bash_script>`



An Example : SHELL utility application (2)

- Design and implementation of the application on **LSA**
 - name: **SHELL**
 - version: **1.0.0.1cg**
 - executable: **source** (available everywhere on LCG)
 - argument: the name of the arbitrary script (which is transferred from client to **VQS** to **LSA**)
 - requirement on LCG: **other.GlueCEInfoTotalCPUs>=1**
- In this case:
 - no real implementation is needed
 - the only thing is to make an **app.xml** to describe it



An Example : SHELL utility application (3)

- The directory hierarchy of SHELL utility application
 - GAP_HOME of LSA is usually \$HOME/opt/gap

\$GAP_HOME/etc/app.list

/apps/shell/

/apps/shell/app.xml

/apps/shell/xxx/





An Example : SHELL utility application (4)

```
<?xml version="1.0"?>
<classads>

... other application index ...

<c>
  <a n="lsa_appid"><s>SHELL</s></a>
  <a n="lsa_appbase"><s>${gap.home}/apps/shell</s></a>
</c>

... other application index ...

</classads>
```

\$GAP_HOME/etc/app.list

```
<?xml version="1.0"?>
<classads>
<c>
  <a n="lsa_appid"><s>SHELL</s></a>
  <a n="lsa_appinfo"><s>Running a given BASH script on LCG</s></a>
  <a n="lsa_appversion"><s>1.0.0.lcg</s></a>
  <a n="lsa_appexec"><s>source</s></a>
  <a n="lsa_appmode"><s>_lsa_batch</s></a>
  <a n="lsa_appreqr"><e>other.GlueCEInfoTotalCPUs>=1</e></a>
  <a n="lsa_appstagein"><b v="f"/></a>
</c>
</classads>
```

app.xml in “lsa_appbase”



An Example : SHELL utility application (5)

- ***Utility application attributes in app.xml***

- **Isa_appid** - Application ID (The Global Unique Application ID). if not equivalent to the registration in the app.list, the value here will be overrided
- **Isa_appinfo** - An abstractive description of this application
- **Isa_appexec** - The executable of the application.
- **Isa_appenv** - The script that sets the runtime environment for the executable. It will always be “source”-ed before running the executable.



An Example : SHELL utility application (6)

- ***Utility attributes in app.xml***

- **lisa_appmode** - The mode of application execution
 - '_lisa_batch' : Single CPU batch mode
 - '_lisa_parallel' : Parallel Job mode
- **lisa_appreqr** - The requirements for placing the jobs on the backend queueing systems. The format should be LQS dependent.
- **lisa_appstagein** - Specify if the executable needs to be staged in the computing node.
- **lisa_appinputs** - The inputsandbox as a list.
- **lisa_appoutputs** - The outputsandbox as a list



An Example : SHELL utility application (7)

- registration on **VQS**

- id: **SHELL**
- name: **SHELL**
- class: **utility**
- VO: **dteam, biomed, twgrid, apesci, euasia, gilda**

```
[t-ap27] /home/ga > util_register.sh SHELL SHELL gilda 'Run arbitrary Bash script on GILDA'
```

id	name	class	vo	abstract
SHELL	SHELL	utility	apesci	Run arbitrary Bash script
SHELL	SHELL	utility	biomed	Run arbitrary Bash script
SHELL	SHELL	utility	dteam	Run arbitrary Bash script
SHELL	SHELL	utility	euasia	Run arbitrary Bash script
SHELL	SHELL	utility	gilda	Run arbitrary Bash script on GILDA
SHELL	SHELL	utility	twgrid	Run arbitrary Bash script



Essential link for SHELL utility application

```
vQS [57]: j = new Job(global.user);  
vQS [58]: j.setName("create manually");  
vQS [59]: j.setAppName("SHELL");
```

VQSCClient - appName

j.submit() accepted by VQS

[t-ap27] /home/ga > show_util.sh SHELL	id	name	class	vo	VQS - database abstract
SHELL	SHELL	SHELL	utility	apesci	Run arbitrary Bash script
SHELL	SHELL	SHELL	utility	biomed	Run arbitrary Bash script
SHELL	SHELL	SHELL	utility	dteam	Run arbitrary Bash script
SHELL	SHELL	SHELL	utility	euasia	Run arbitrary Bash script
SHELL	SHELL	SHELL	utility	gilda	Run arbitrary Bash script on GILDA
SHELL	SHELL	SHELL	utility	twgrid	Run arbitrary Bash script

Utility application id matched on LSA

```
<c>  
  <a n="lsa_appid"><s>SHELL</s></a>  
  <a n="lsa_appbase"><s>${gap.home}/apps/shell</s></a>  
</c>
```

LSA - app.list



Use SHELL utility application directly in CORE Framework

● Sample code for using the utility in CORE framework

```
User user = getUser();                                // Get the instance of User somewhere

String userScript = "/home/mason/sayhello.sh";

List<Job> jobs = new ArrayList<Job>();

Job j = new Job(user);

j.setAppName("SHELL");                                Specify the SHELL app id

j.setTag(Long.toString(UniqueID.get()));    // Set the tag for grouping jobs.
j.setNprocs(1);
j.setName("Run a bash script test");           // Job description
j.setTimeout(timeout);                         // Set the timeout for this job

j.addInputData(userScript);                      // Set your bash script.

j.saveOptions(new String[] {
    (new File(userScript)).getName()}); // Let your bash script become the
                                         // argument of 'source'

List outputs = new ArrayList();
outputs.add("SHELL.stdout");
outputs.add("SHELL.stderr");
j.setOutputList(outputs);                         // Set the outputs for this job
jobs.add(j);                                     // Add to the job list.

JobManager.submitJobs(user, jobs);                // Submit the jobs.
```



Using SHELL utility in command line

- command-line application for **VQSCClient**
 - inherit `net.asgc.gap.core.ui.app.CommandLineApplication`
 - implement the methods:
 - `config()` for setting up the application arguments
 - `preProcess()` for preparing/creating underlying Job objects
- example implementations:
 - `net.asqc.gap.core.ui.app.shell.SHELL`

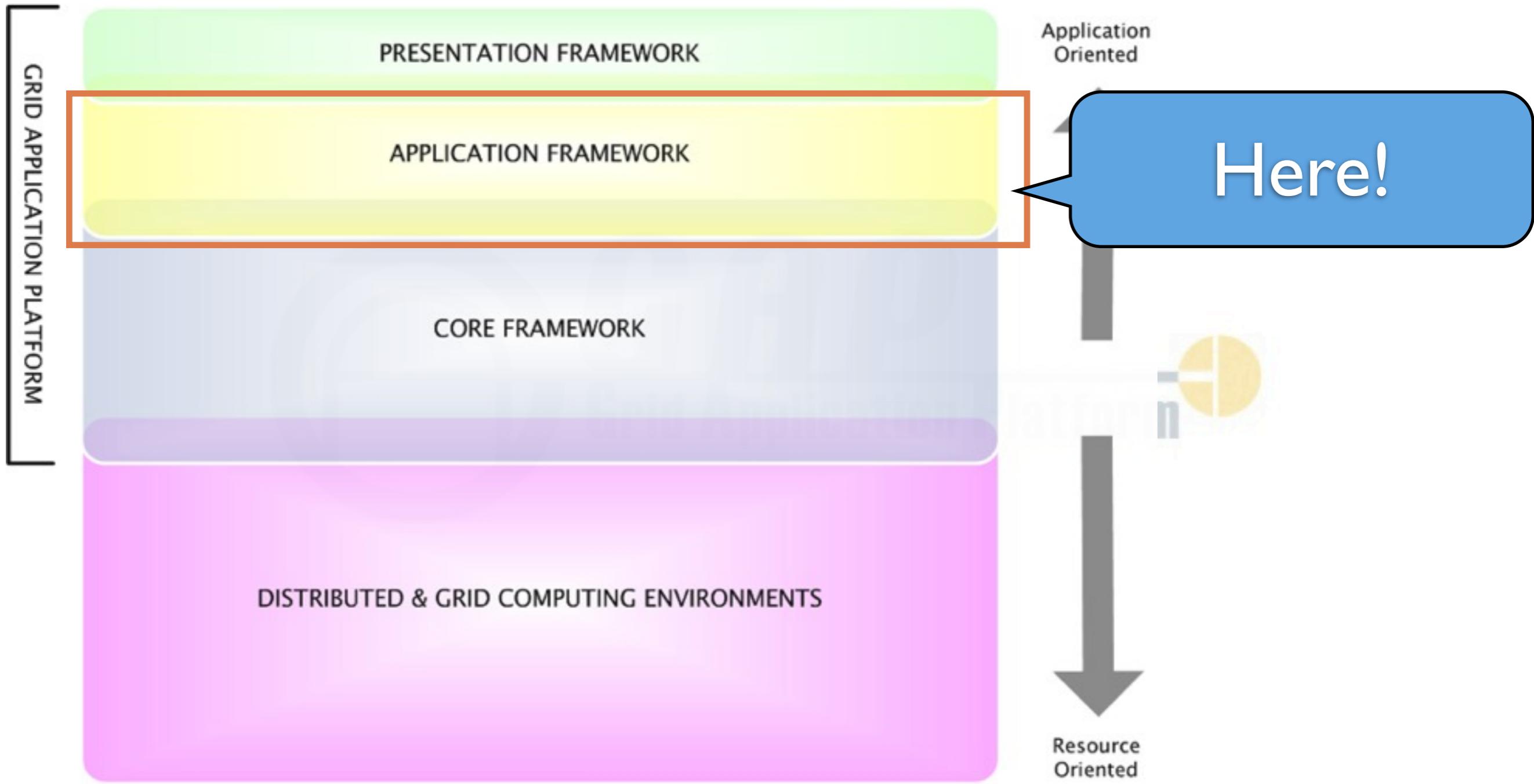


Implement Corresponding
Logic in Application Framework





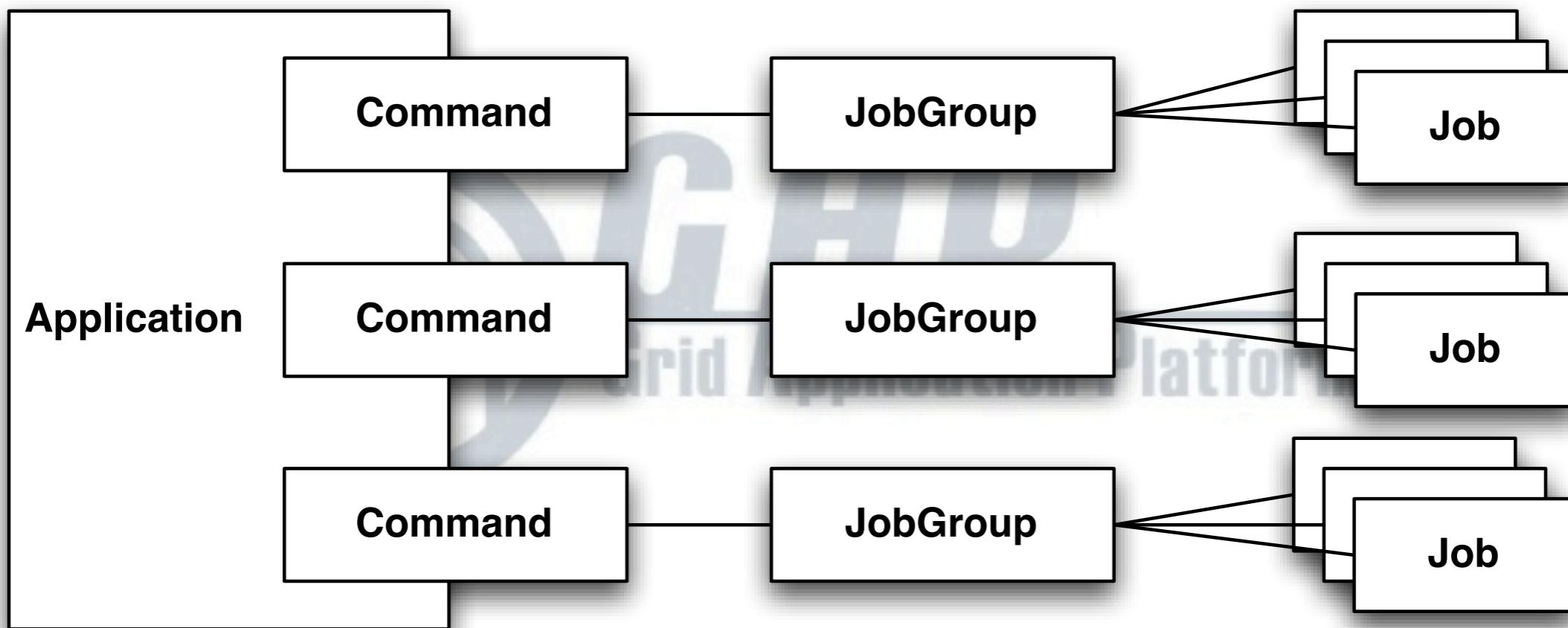
Implement Corresponding Logic in Application Framework





Classes in Application Framework

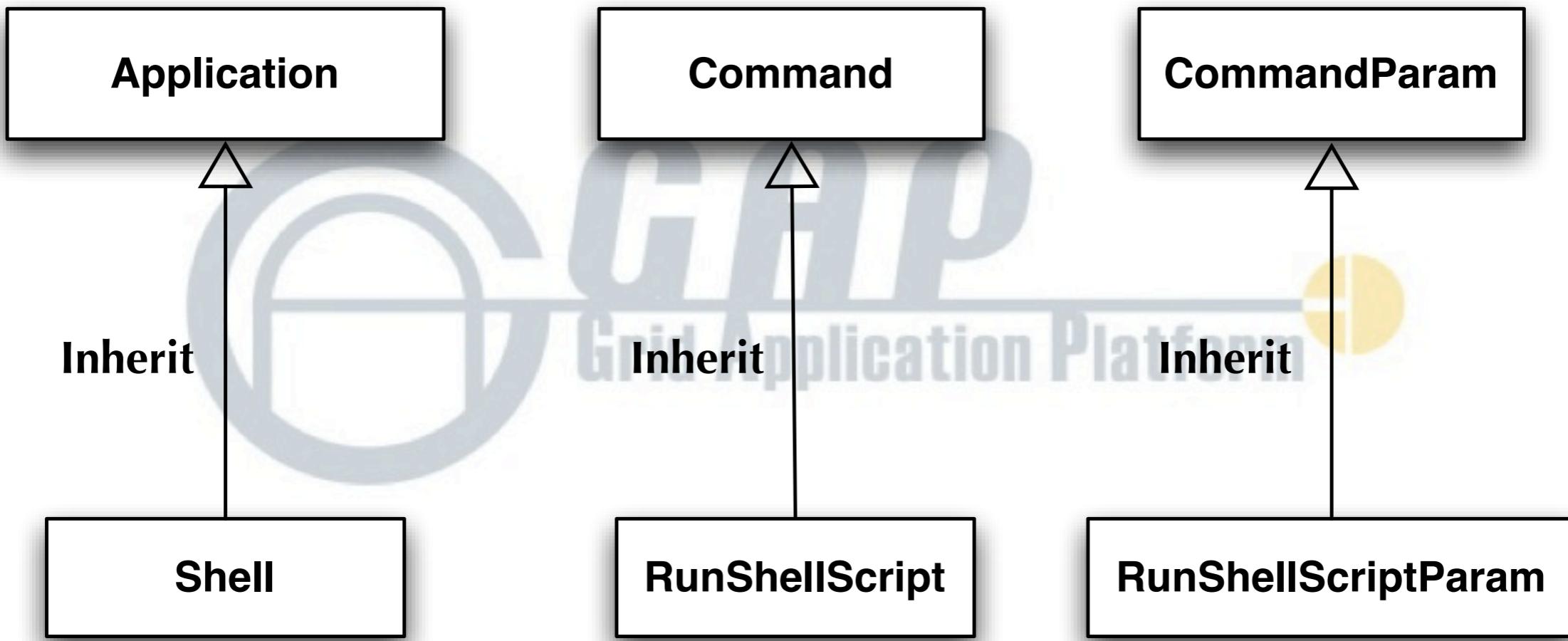
- *The Relationship between the classes in Application Framework*





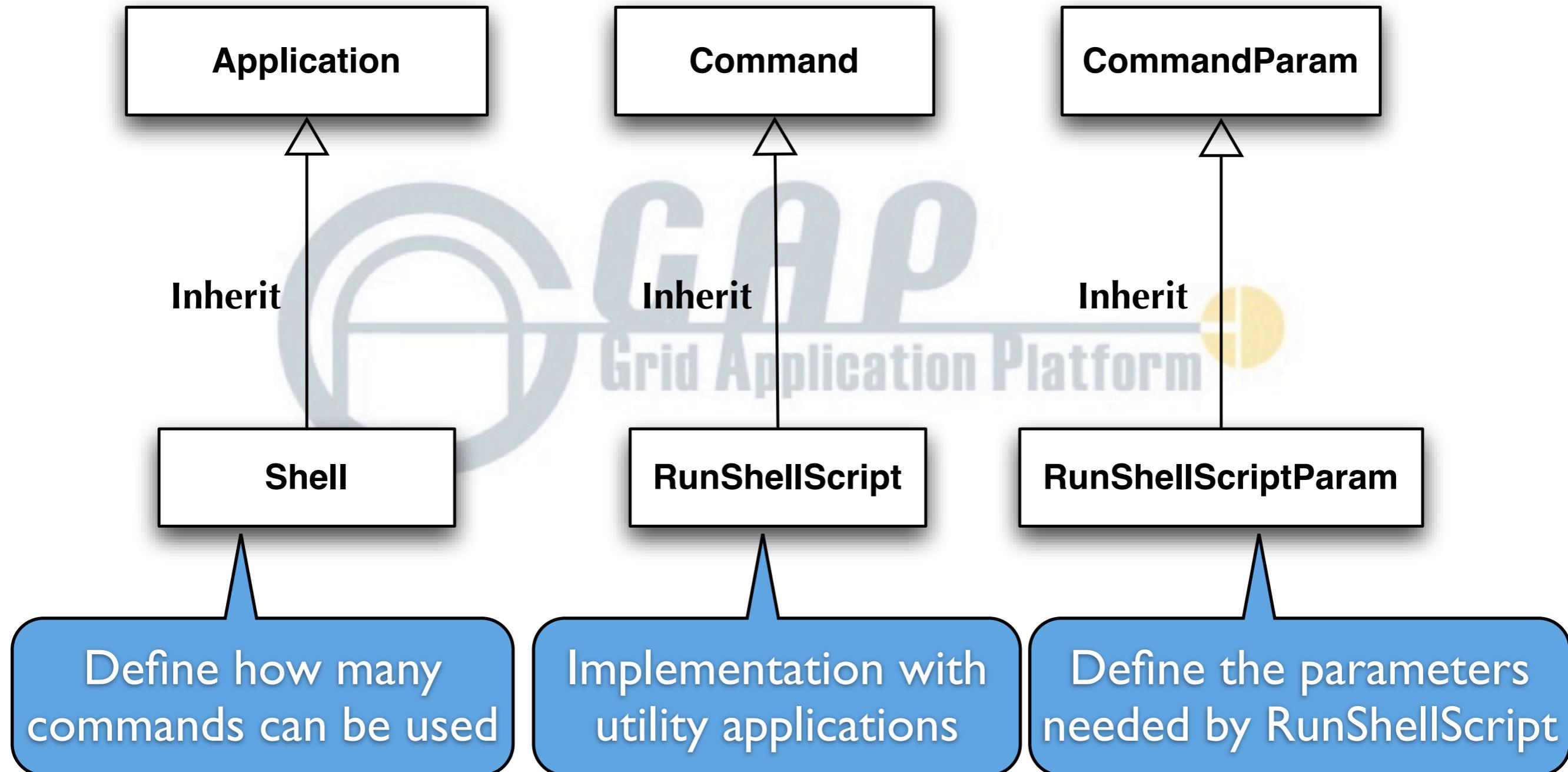
Classes in Application Framework(cont.)

- *The application implementation for using SHELL utility application*



Classes in Application Framework(cont.)

- *The application implementation for using SHELL utility application*





Implementation for SHELL utility in Application Framework

- **RunShellScriptParam Inherits CommandParam to define the inputs for RunShellScript**

```
public class RunShellScriptParam extends CommandParam {  
    /** Creates a new instance of CommandTestDriveParam */  
    public RunShellScriptParam() {  
        this.put("shellInputs", new ArrayList<File>());  
        this.put("shellOptions", new String[]{});  
    }  
  
    public void setShellScript(String script){  
        this.put("shellScript",new File(script));  
    }  
  
    public void setOptions(String[] options){  
        this.put("shellOptions",options);  
    }  
  
    public String[] getOptions(){  
        String[] options = (String[])this.get("shellOptions");  
        if(options == null) options = new String[]{};  
        return options;  
    }  
  
    public void setInputs(List<File> inputs){  
        this.put("shellInputs", inputs);  
    }  
  
    public List<File> getInputs(){  
        return (List<File>) this.get("shellInputs");  
    }  
  
    public void setNumberOfRuns(int runs) {  
        this.put("numOfRuns", new Integer(runs));  
    }  
  
    public File getShellScript(){  
        return (File)this.get("shellScript");  
    }  
  
    public int getNumberOfRuns() {  
        return this.get("numOfRuns")==null ? 1 : ((Integer) this.get("numOfRuns")).intValue();  
    }  
}
```

Specify the bash script

Assign the
script options

Specify other input files
needed by the script

Set the number
of run



Implementation for SHELL utility in Application Framework(cont.)

- **RunShellScript Inherits Command to implement the command logic**

- ▶ **validateParameter**
- ▶ **configureCommand**

```
public class RunShellScript extends Command {  
  
    private static Logger logger = Logger.getLogger(Command.class.getName());  
  
    /**  
     * Creates a new instance of RunShellScript  
     */  
    public RunShellScript() {}  
    public RunShellScript(Application srcApp) {  
        super(srcApp);  
        this.setDescription("Run a shell script");  
    }  
  
    @Override  
    protected boolean validateParameter(CommandParam param) {  
        ...  
    }  
  
    @Override  
    protected boolean configureCommand(CommandParam param, JobGroup bojs) {  
        ...  
    }  
}
```

Validate
Parameter

Configure
Command



Implementation for SHELL utility in Application Framework(cont.)

- **Validate the parameters.**

```
@Override
protected boolean validateParameter(CommandParam param) {
    RunShellScriptParam p = (RunShellScriptParam) param ;
    File f = p.getShellScript();
    if(!f.exists()) {
        return false;
    } else {
        return true;
    }
}
```

Here we simply verify the existence of the script.



Implementation for SHELL utility in Application Framework(cont.)

● Configure the Command according to the parameters.

```
@Override
protected boolean configureCommand(CommandParam param, JobGroup bojs) {
    RunShellScriptParam p = (RunShellScriptParam) param ;
    File f = p.getShellScript();
    boolean ick = false;

    for ( int i=0; i<p.getNumberOfRuns(); i++) {
        try {
            Job j = bojs.allocateJob(p.getTaskName().equals("") ? this.getCommandClassName() : p.getTaskName(), "SHELL", 1);
            String[] paramOptions = p.getOptions();
            String[] options = new String[paramOptions.length + 1];
            options[0] = f.getName();
            for(int x = 1; x < options.length; x++){
                options[x] = paramOptions[x-1];
            }
            j.saveOptions(options);

            // Set inputlists of the job
            List inputs = new ArrayList();
            inputs.add(f.getAbsolutePath());

            Iterator<File> inputIter = p.getInputs().iterator();
            File tmp = null;

            while(inputIter.hasNext()){
                tmp = inputIter.next();
                inputs.add(tmp.getAbsolutePath());
            }

            j.setInputList(inputs);

            // Set outputlists of the job
            List outputs = new ArrayList();
            outputs.add("SHELL.stdout");
            outputs.add("SHELL.stderr");
            j.setOutputList(outputs);

        } catch (JobAllocationFailedException ex){logger.warning(ex.getMessage());break;
        } catch (Exception ex) {logger.warning(ex.getMessage());break;}

        ick = true;
    }
    return ick;
}
```

Create the job object with the corresponding utility application

Utility Name

Assign executable options

Specify input files

Specify output files



Implementation for SHELL utility in Application Framework(cont.)

● The logic of ‘performCommand’ in generic ‘Command’ super class.

```
public abstract class Command {  
    ... other methods ...  
  
    public CommandParam performCommand(CommandParam param)  
        throws PerformCommandFailedException,  
               InvalidCommandParamException,  
               JobAllocationFailedException {  
  
        if(!this.validateParameter(param)) {  
            throw new InvalidCommandParamException();  
        }  
  
        JobGroup bojs = new JobGroup(this.getSrcApp().getUser());  
        bojs.setCommandName(this.getCommandName());  
        bojs.setCommandClassName(this.getCommandClassName());  
        bojs.setDescription(param.getDescription());  
        bojs.setSubmittedTime(new java.sql.Timestamp(new Date().getTime()));  
  
        if(!this.configureCommand(param,bojs)) {  
            throw new PerformCommandFailedException();  
        }  
  
        if(! bojs.submitCachedAllocatedJobs()) {  
            throw new PerformCommandFailedException("Some jobs failed to submit.");  
        }  
  
        // archive job group attributes  
        try {  
            JobGroupDAO dao = JobGroupDAO.getInstance(this.getSrcApp().getUser());  
            dao.store(bojs, this.getSrcApp());  
        } catch (JobGroupDAOException ex) {  
  
            // rollback by deleting the submitted jobs  
            JobManager.deleteJobs( this.getSrcApp().getUser(), bojs.getJobList());  
  
            throw new PerformCommandFailedException(ex.getMessage());  
        }  
  
        CommandOutput out = new CommandOutput();  
        out.setJobGroup(bojs);  
        return out;  
    }  
    ... other methods ...  
}
```

Call your own defined
parameters validation method

Call your own defined
command configuring method

Submit all the jobs in the
JobGroup



Implementation for SHELL utility in Application Framework(cont.)

- ***Shell inherits Application to define how many Commands will be used***

```
public class Shell extends Application {  
  
    /** Creates a new instance of Shell */  
    public Shell(User user) {  
        super(user);  
        this.setAppName("SHELL");  
        this.setCommand(new RunShellScript(this));  
    }  
}
```

Add RunShellScript
command to this
Shell Application

- ***The instance of 'Application' acts as a controller for executing commands***

```
Shell sh = new Shell(user);  
sh.performCommand("RunShellScript", params);  
sh.performCommand("Other Command", other_params);  
... perform more commands if they are available ...
```

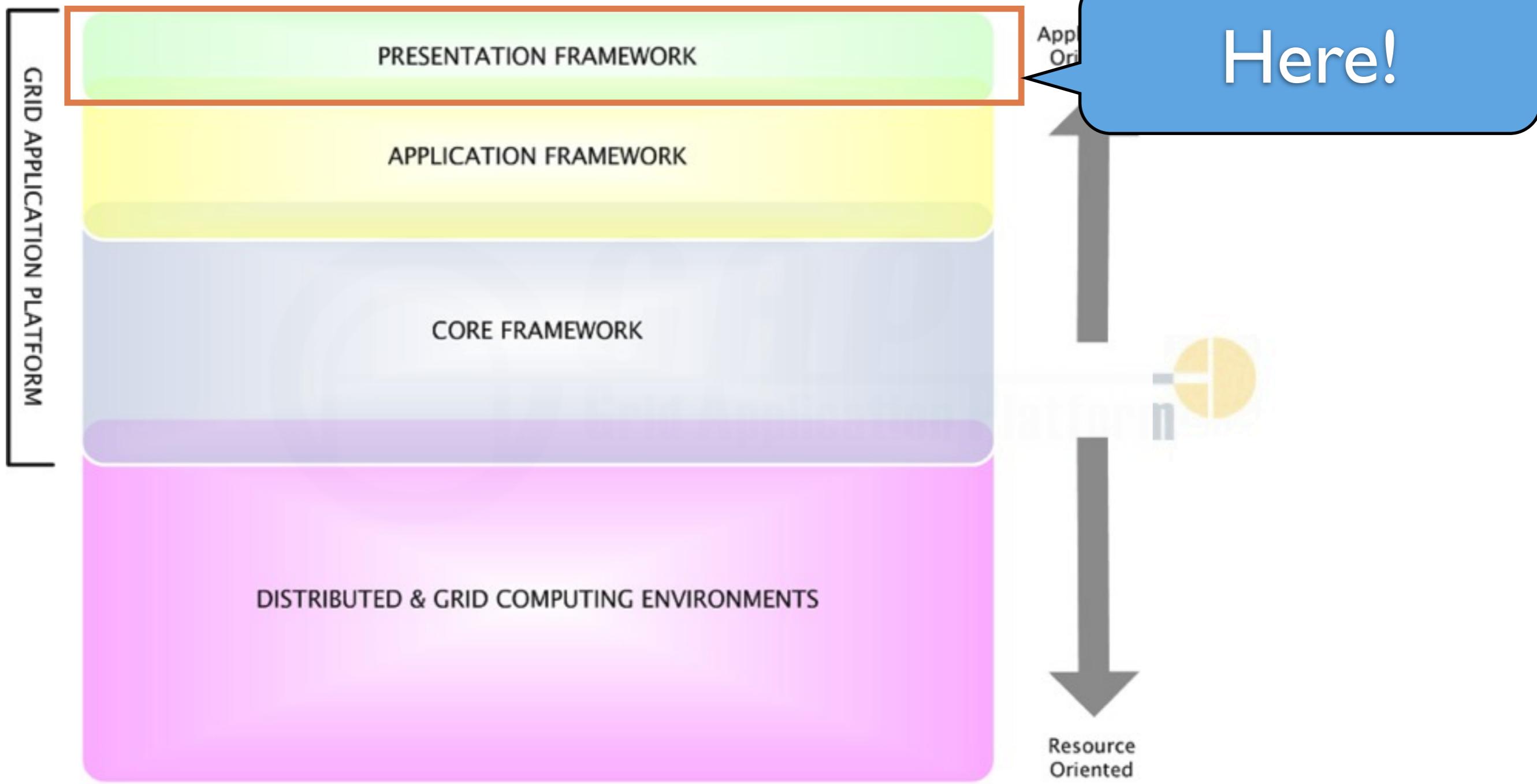


How to use deployed application in your program





How to use deployed application in your program





Use Shell Application in your program

- Put this code slice in your program

```
String vqs_id = "mason";
String password = "vqs password";
String passphrase = "ca passphrase";
int lifeTime = 43200;

String userScript = "/home/mason/say.sh";
int numRuns = 2;
String taskName = "Say something";
String[] options = new String[] {"HelloTest"};

VQSCientProperties.load();

User u = UserManager.getManager().getUser( vqs_id,
                                             System.getProperty("vqs.uservo"),
                                             System.getProperty("vqs.jobpool"));

u.login(System.getProperty("vqs.porturl"), password, passphrase, lifeTime);

Shell sh = new Shell(u);

RunShellScriptParam param = new RunShellScriptParam();
param.setShellScript(userScript);
param.setNumberOfRuns(numRuns);
param.setTaskName(taskName);
param.setOptions(options);

CommandOutput output = (CommandOutput) sh.performCommand("RunShellScript", param);
```



Use Shell Application in your program

- Put this code slice in your program

```
String vqs_id = "mason";
String password = "vqs password";
String passphrase = "ca passphrase";
int lifeTime = 43200;
```

Specify vqs account,
password, ca's passphrase
and the proxy life time

```
String userScript = "/home/mason/say.sh";
int numRuns = 2;
String taskName = "Say something";
String[] options = new String[] {"HelloTest"};
VQSCClientProperties.load();
```

Initiate the user, and login VQS

```
User u = UserManager.getManager().getUser( vqs_id,
                                             System.getProperty("vqs.uservo"),
                                             System.getProperty("vqs.jobpool"));

u.login(System.getProperty("vqs.porturl"), password, passphrase, lifeTime);
```

```
Shell sh = new Shell(u);

RunShellScriptParam param = new RunShellScriptParam();
param.setShellScript(userScript);
param.setNumberOfRuns(numRuns);
param.setTaskName(taskName);
param.setOptions(options);
```

Simply submit the SHELL job

```
CommandOutput output = (CommandOutput) sh.performCommand("RunShellScript", param);
```



Job submission through Application Framework in command line interface

- *Prepare the BASH script in your home directory*
 - **Say “say.sh”**
- *The content of BASH script - “say.sh”*

```
#!/bin/sh

echo "'`hostname`' says" $1
```



Job submission through Application Framework in command line interface

```
vQS [1]: login();  
vqs username: mason  
vqs password: ****  
grid passphrase: *****  
[INFO] connection to tcp://t-ap27.grid.sinica.edu.tw:10010?  
wireFormat.maxInactivityDuration=0&jms.useAsyncSend=true  
[INFO] listen on queue: uqueue.mason  
[INFO] Grid proxy is initialized with lifetime: 43200 secs.  
[INFO] Proxy has been delegated to t-ap27.grid.sinica.edu.tw:10006  
Elapsed time: 11 sec.
```

Login VQS

```
vQS [2]: s = app2("SHELLApp");
```

Create the application using Shell application

```
vQS [3]: s.config();
```

Configure job interactively

```
Task name: Say hello
```

```
The BASH script(Absolute Path): /Users/masonhsiusng/say.sh
```

Specify 'say.sh' BASH script

```
How many times: 3
```

Execute 3 times

```
Specify the options needed by the BASH script: heyhey
```

```
vQS [4]: s.run();
```

Execute this application

```
[INFO] Job "434d133b:1209a07ddf8.1" submitted  
[INFO] Job "434d133b:1209a07ddf6.1" submitted  
[INFO] Job "434d133b:1209a07ddf7.1" submitted
```

3 jobs are actually submitted

```
vQS [5]: print(s.getStandardOutput());
```

Print the standard output

```
[INFO] Job outputs fetched in "/Users/masonhsiusng/.gap/mason/jobs/434d133b/1209a07ddf7/1"  
[INFO] Job outputs fetched in "/Users/masonhsiusng/.gap/mason/jobs/434d133b/1209a07ddf6/1"  
[INFO] Job outputs fetched in "/Users/masonhsiusng/.gap/mason/jobs/434d133b/1209a07ddf8/1"  
'node3' says heyhey  
'wn002' says heyhey  
'wn08.marie.hellasgrid.gr' says heyhey
```

Summary



GRID APPLICATION PLATFORM



Application Oriented



Resource Oriented



Presentation Framework
focuses on the UI
design



Application Framework
controls the work flow
of the application



CORE Framework
focuses on defining the
generic model for the
developer



Thanks for your attention !

Any Question?

