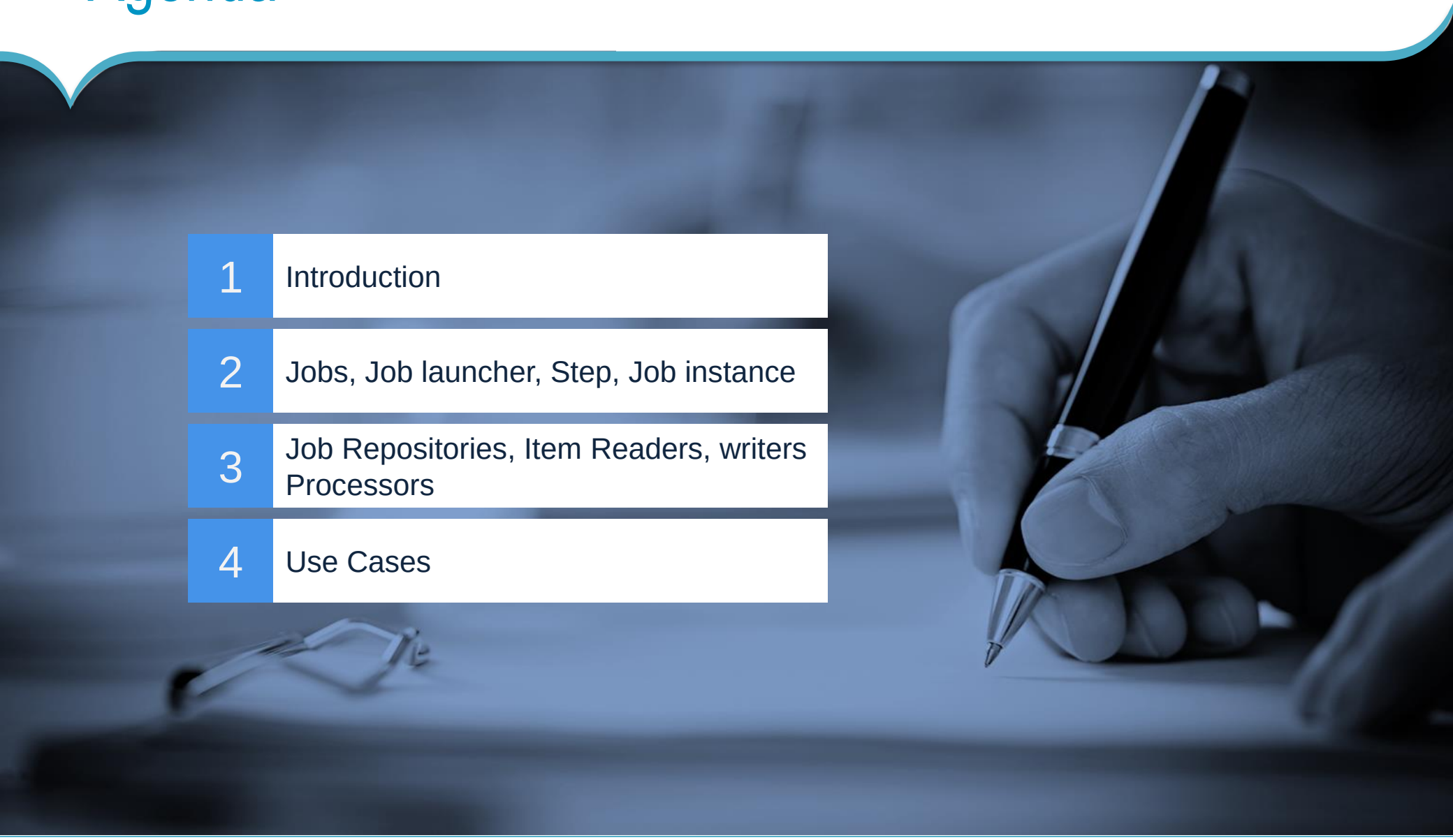
Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

Spring Batch

2017

People matter, results count.

Agenda



1	Introduction
2	Jobs, Job launcher, Step, Job instance
3	Job Repositories, Item Readers, writers Processors
4	Use Cases

Introduction

Introduction - Spring Batch

- Open source framework
- Depends on spring framework
- Definition of batch processing:

"Batch processing is the execution of a series of programs ("jobs") on a computer without manual intervention"

- Batch application executes a series of jobs, where input data is read, processed and written without any interaction.

Introduction - Spring Batch

- Process large amount of data
- Transaction management, job processing, resource management, logging, tracing, conversion of data, interfaces, etc.
- Divided in three main parts:
  - Reading the data (from a database, file system, etc.)
  - Processing the data (filtering, grouping, calculating, validating…)
  - Writing the data (to a database, reporting, distributing…)
- Contains features and abstractions
- Allowing the application programmers to configure them, repeat them, retry them, stop them, executing them as a single element or grouped (transaction management), etc.
- Contains classes and interfaces for the main data formats, industry standards and providers like XML, CSV, SQL, Mongo DB, etc.

Jobs

- Jobs are abstractions to represent batch processes, that is, sequences of actions or commands that have to be executed within the batch application.

```
<job id="eatJob"
xmlns="http://www.springframework.org/schema/batch">
            <step id="stepCook" next="stepEntries">
              <tasklet>
                        <chunk reader="cookReader" writer="cookProcessor"
                 processor="cookWriter" commit-interval="1" />
              </tasklet>
            </step>
……
</job>
```

Job Launcher

- Implementations of its run() method take care of starting job executions for the given jobs and job parameters.

Job Instance

- Representing a single run for a given Job
- unique and identifiable
- Job instances can be restarted in case they were not completed successfully and if the Job is restart able.
- Otherwise an error will be raised.

Steps

- Compose a Job (and a Job instance).
- A Step is a part of a Job and contains all the necessary information to execute the batch processing actions that are expected to be done at that phase of the job.
- Steps in Spring Batch are composed of
  - ItemReader,
  - ItemProcessor
  - ItemWriter
- Simple or extremely complicated depending on the complexity of their members.
- Steps also contain configuration options for their processing strategy, commit interval, transaction mechanism or job repositories that may be used.
- Spring Batch uses normally chunk processing, that is reading all data at one time and processing and writing “chunks” of this data on a preconfigured interval, called commit interval.

Step

- A **Step** is a domain object that encapsulates an independent, sequential phase of a batch job and contains all of the information necessary to define and control the actual batch processing.

  Steps can be processed in either of the following two ways.
- Chunk
- Tasklet

- The Tasklet which is a simple interface with just one method execute. Using this we can perform single tasks like executing queries, deleting files.

Parameter	Tasklet	Chunk
When to use	Suppose the job to be run a single granular task then Tasklet processing is used.	Suppose the job to be run is complex and involves executing of tasks involving reads, processing and writes the we use chunk oriented processing
How it works	No aggregation, just the task gets executed.	It involves reading an input, processing it based on the business logic and then aggregating it till the commit-interval is reached and finally writing out the chunk of data output to a file or database table.
Usage	Its not used commonly.	Its the most common way of executing a Step.
Use Case	Usually Used in scenarios invloving a single task like deleting a resource or executing a query .	Usually used in scenarios where multiple aggregated steps need to be run like copying, processing and transferring of data .
Example	<job id="taskletJob"> <step id="callingStoredProc"> <tasklet ref="callProc"/> </step> </job>	<job id="sampleJob" job-repository="jobRepository"> <step id="step1"> <tasklet transaction-manager="transactionManager"> <chunk reader="itemReader" writer="itemWriter" commit-interval="10"/> </tasklet> </step> </job>

Steps

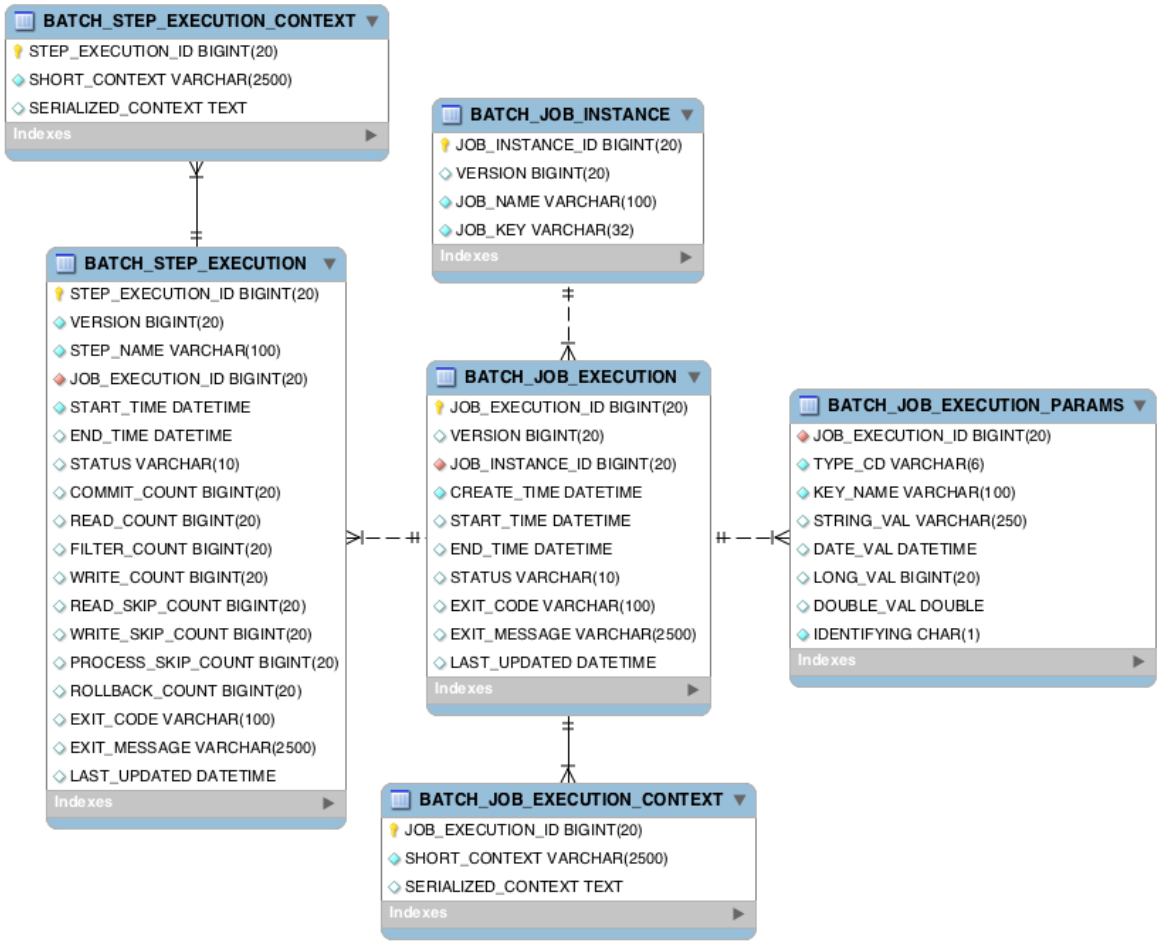
```
<step id="step" next="nextStep">
 <tasklet>
  <chunk reader="customItemReader" writer="customItemWriter" processor="customItemProcessor"
commit-interval="10" />
 </tasklet>
</step>
```

```
@Bean
public Step step1(StepBuilderFactory stepBuilderFactory,
                  ItemReader reader, ItemWriter writer,
                  ItemProcessor processor) {
        /* it handles bunches of 10 units */
        return stepBuilderFactory.get("step1")
                                 . chunk(10).reader(reader)
                                 .processor(processor).writer(writer).build();
}
```

Job Repositories

- Responsible of the storing and updating of metadata information related to Job instance executions and Job contexts.
- Spring stores as metadata information about their executions, the results obtained, their instances, the parameters used for the Jobs executed and the context where the processing runs.
- The table names are very intuitive and similar to their domain classes counterparts, in this link there is an image with a very good summary of these tables:

Job Repositories



Iteam Readers

- Responsible of the data retrieval.
- They provide batch processing applications with the needed input data.
- Here is a list of some readers provided by Spring Batch:
  - AmqpItemReader
  - AggregateItemReader
  - FlatFileItemReader
  - HibernateCursorItemReader
  - HibernatePagingItemReader
  - IbatisPagingItemReader
  - ItemReaderAdapter
  - JdbcCursorItemReader
  - JdbcPagingItemReader
  - JmsItemReader
  - JpaPagingItemReader
  - ListItemReader
  - MongoItemReader
  - Neo4jItemReader
  - RepositoryItemReader
  - StoredProcedureItemReader
  - StaxEventItemReader

Item Writers

- Writing the data to the desired output database or system
- Here is a list of some of these provided writers

  - AbstractItemStreamItemWriter
  - AmqpItemWriter
  - CompositeItemWriter
  - FlatFileItemWriter
  - GemfireItemWriter
  - HibernateItemWriter
  - IbatisBatchItemWriter
  - ItemWriterAdapter
  - JdbcBatchItemWriter
  - JmsItemWriter
  - JpaItemWriter
  - MimeMessageItemWriter
  - MongoItemWriter
  - Neo4jItemWriter
  - StaxEventItemWriter
  - RepositoryItemWriter

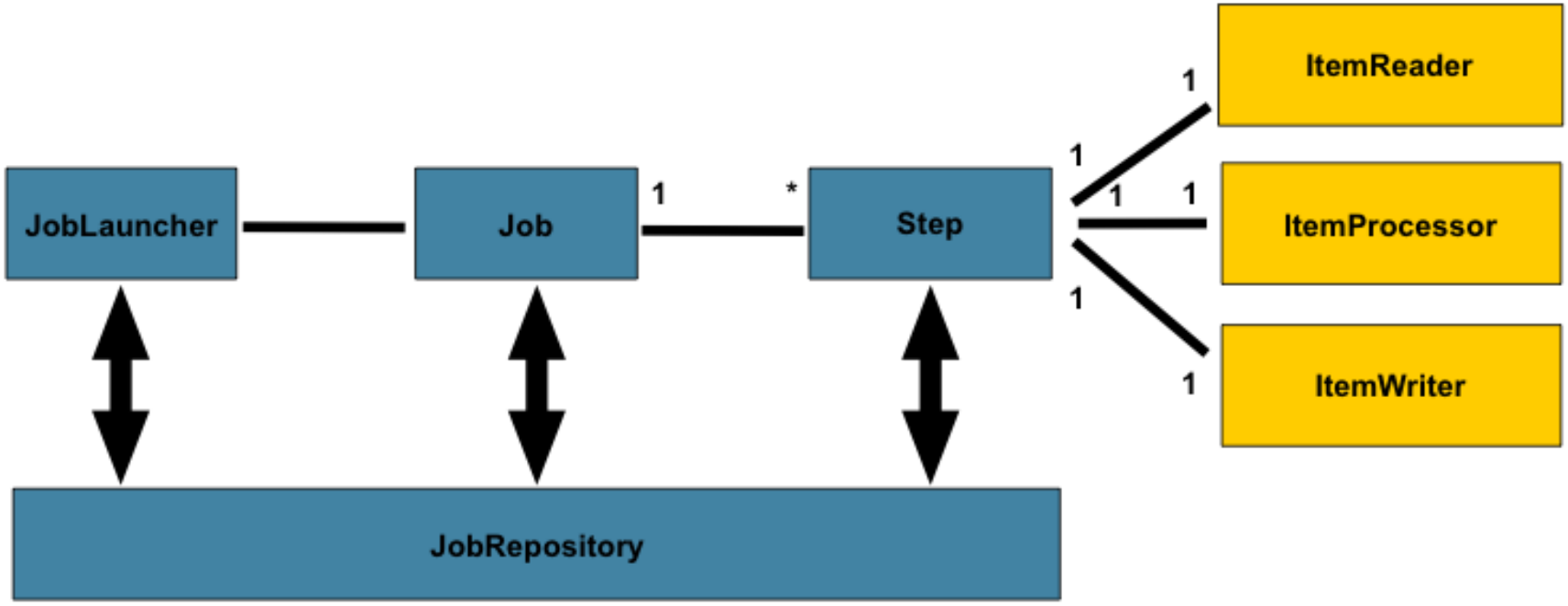
Item Processors

- In charge of modifying the data records converting it from the input format to the output desired one

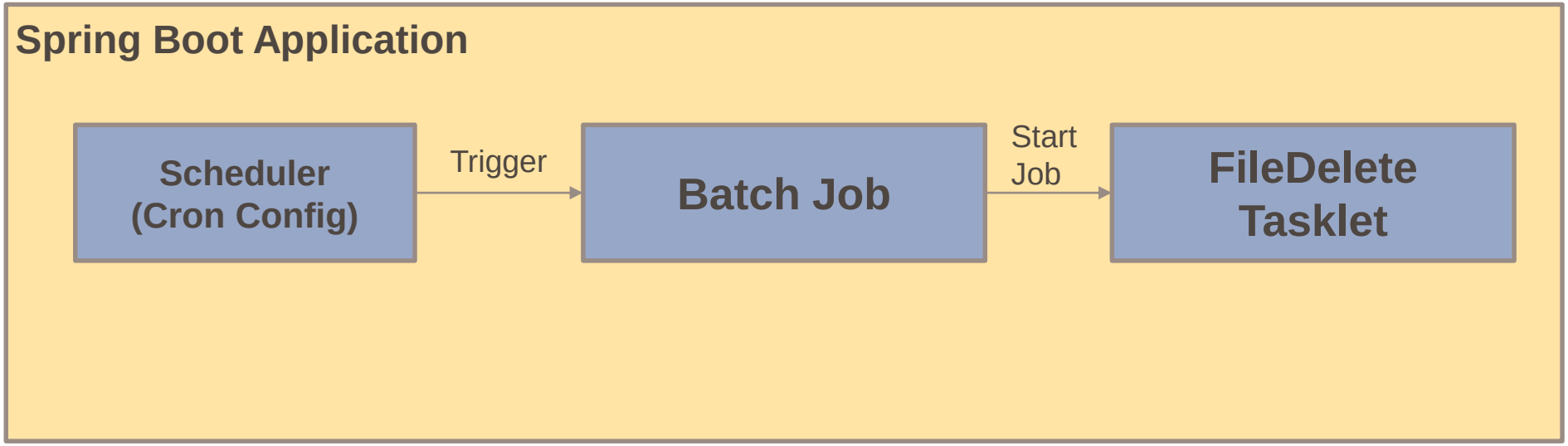
Use cases

- **Conversion Applications**
- **Filtering or validation applications**
- **Database extractors**
- **Reporting**

Spring Batch Reference Model



Job Scheduler



Summary

- **Job Repositories**
- **Job Launcher**
- **Step – Tasklets, Chunks**
- **Item Reader, Item Processor, Item Writer**

THANK YOU

Capgemini

CONSULTING.TECHNOLOGY.OUTSOURCING

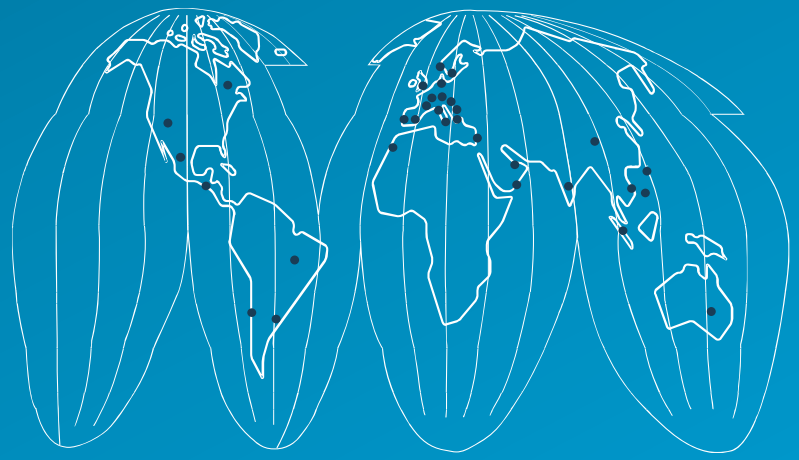
People matter, results count.

About Capgemini

With more than 145,000 people in 40 countries, Capgemini is one of the world's foremost providers of consulting, technology and outsourcing services. The Group reported 2014 global revenues of EUR 10.5 billion.

Together with its clients, Capgemini creates and delivers business and technology solutions that fit their needs and drive the results they want. A deeply multicultural organization, Capgemini has developed its own way of working, the Collaborative Business Experience™, and draws on Rightshore®, its worldwide delivery model.

Rightshore® is a trademark belonging to Capgemini



www.capgemini.com