TDP024

Maven, Skriv-uppgiften och Program to an Interface

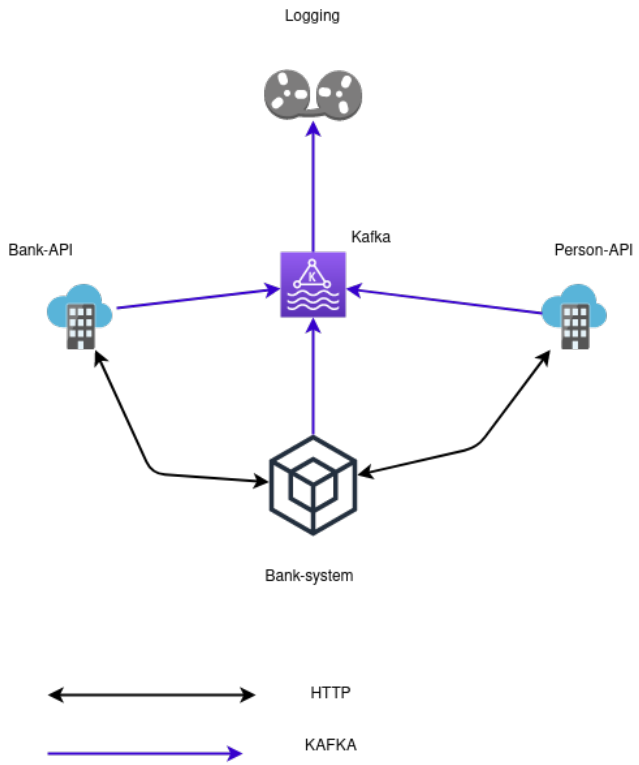
Anders Fröberg anders.froberg@liu.se

Oooops sorry

- Fel datum i webreg
- Missat att lägga till er till repot
- Dessutom har jag glömt att lägga upp slides

Have you made your chooice

Write down on the board

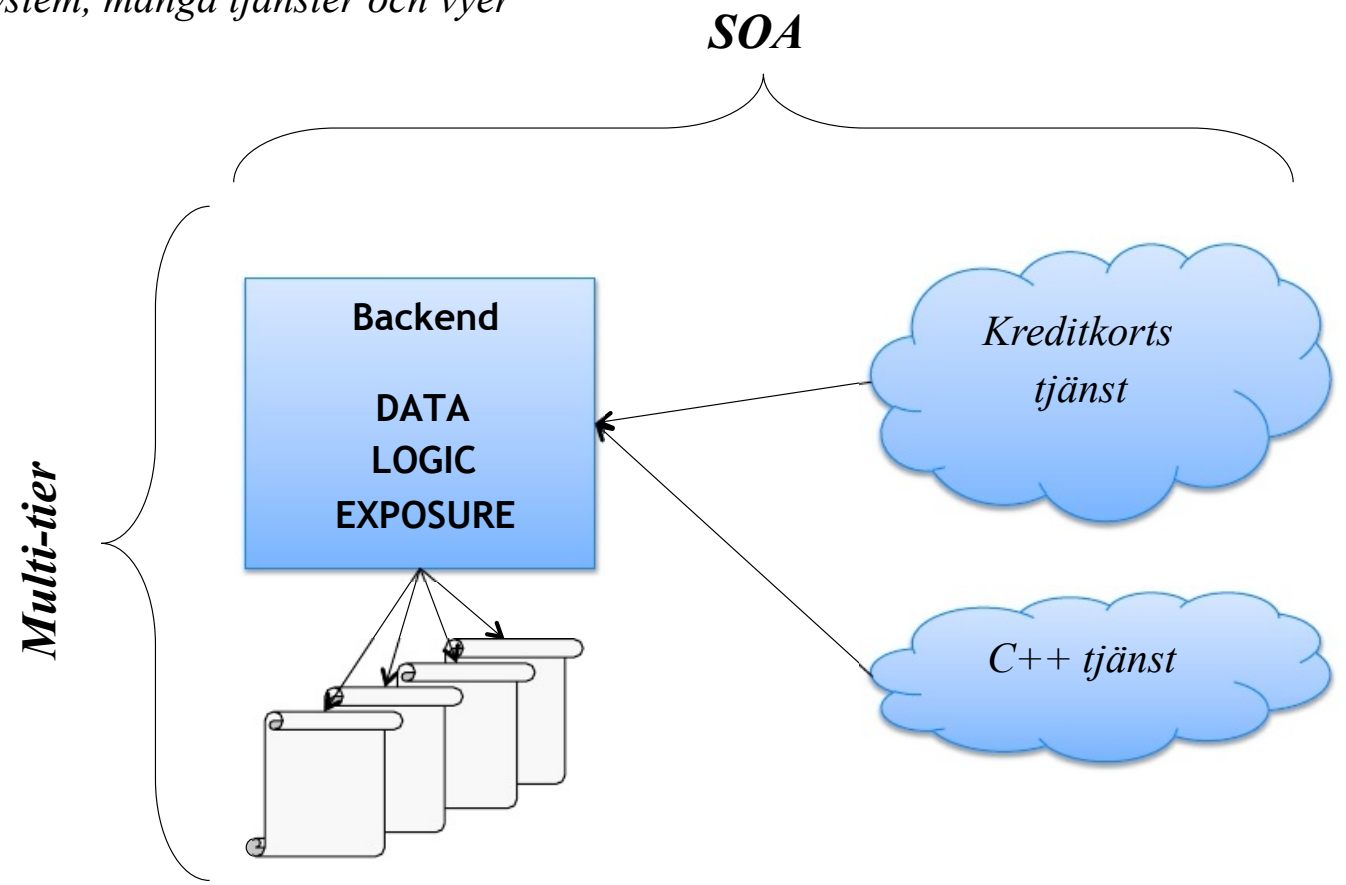


SOA Principer - Summering

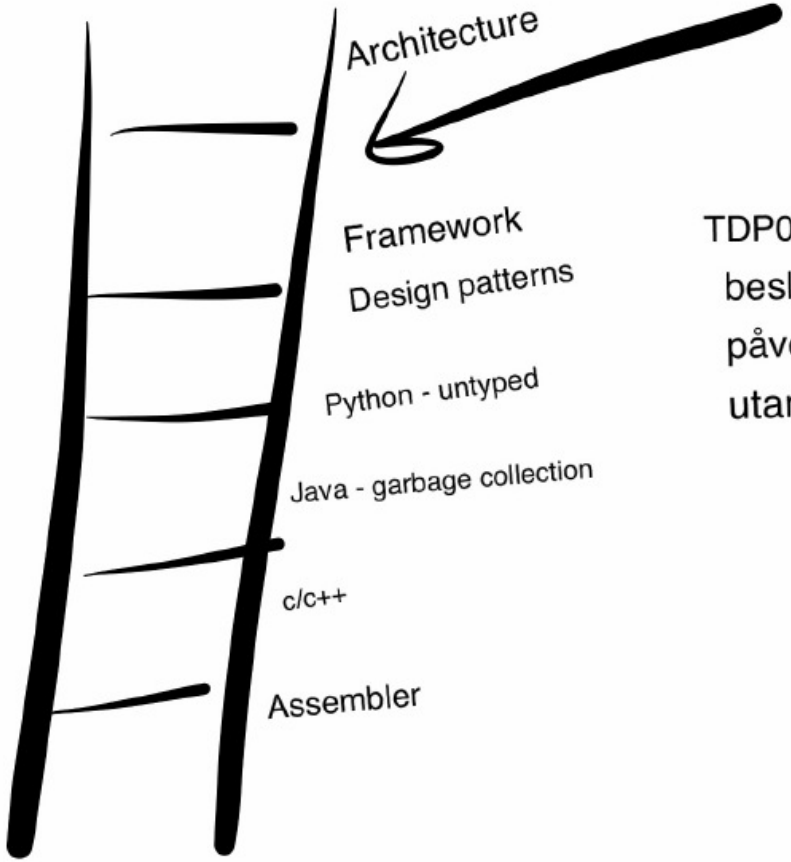
1. Contract
2. Abstraction
3. Autonomy
4. Stateless

Enterprise Systems

Ett system, många tjänster och vyer



Abstraktion



TDP024 hamnar högt i abstraktionen,
besluten vi tar i våra scenarion
påverkar inte bara oss själva
utan hela organisationen.

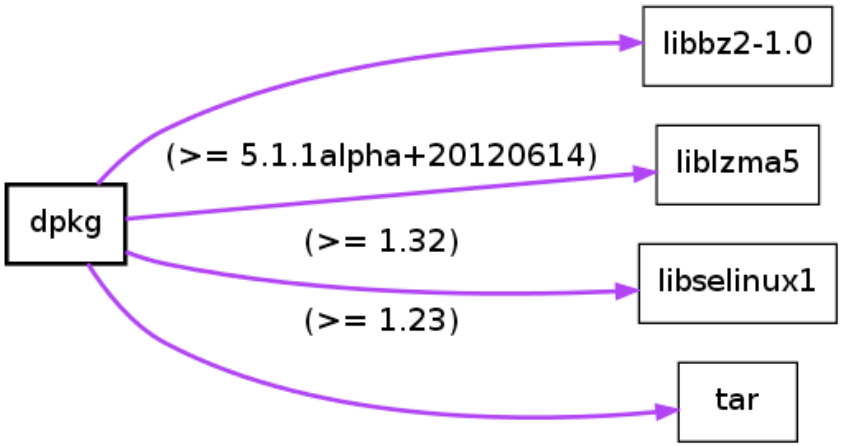
Maven

- Build and Packaging

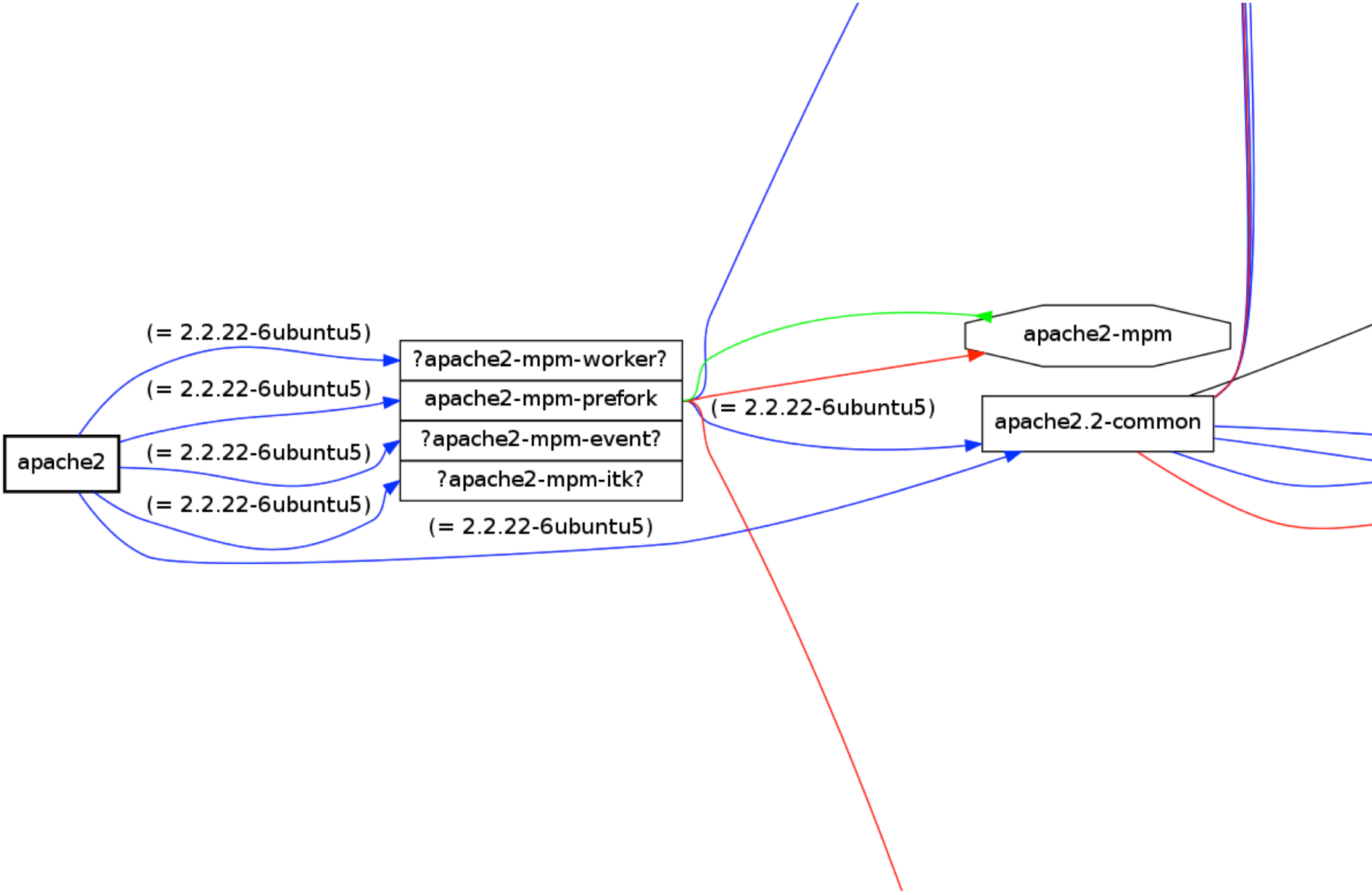
Scenarios - actions

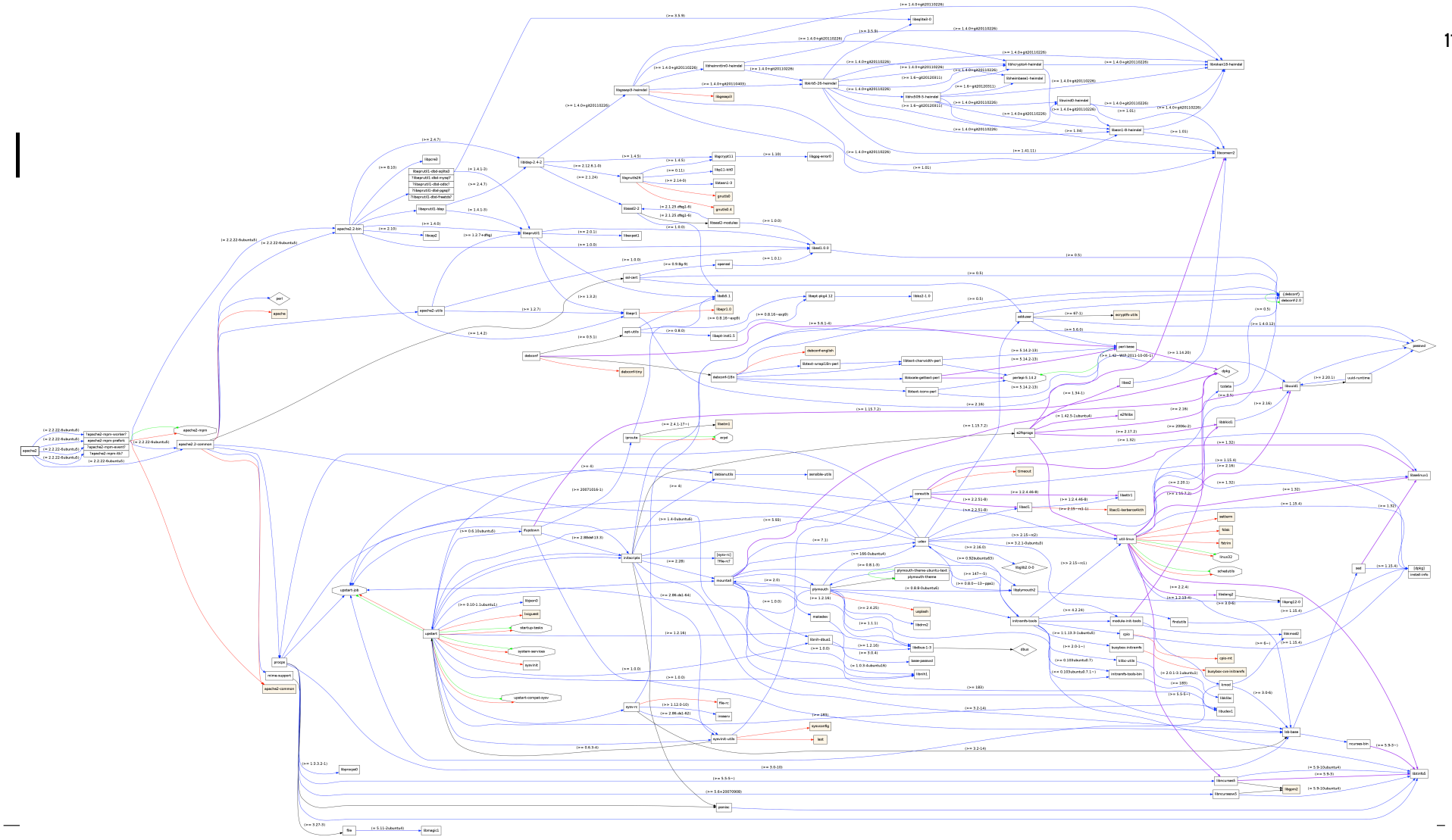
1. Many groups of developers, multiple changes — Package management system
2. Building artefacts based on multiple files with dependencies — build scripts
3. Conducting multiple actions with inter-dependencies on multiple files … — Flexible build system

Package management systems



Dep


apache2
(= 2.2.22-6ubuntu5)
(= 2.2.22-6ubuntu5)
(= 2.2.22-6ubuntu5)
(= 2.2.22-6ubuntu5)
(= 2.2.22-6ubuntu5)
?apache2-mpm-worker?
apache2-mpm-prefork
?apache2-mpm-event?
?apache2-mpm-itk?
(= 2.2.22-6ubuntu5)
apache2-mpm
apache2.2-common



Dependency management issues

- Is a request to modify the current software component graph satisfiable?
  - Are additions compatible with other components?
  - Are deletions safe with respect to other dependencies?
- Given a component, determine versions of other components we can safely rely on

Dependency management issues

- Y depends on X >= 1.8. X makes binary incompatible changes from v. 1.9 to v. 2.0…
- Can components be installed from local sources as well as from remote?
- Should OS-specific dependency management or language-specific be used?

Software package management systems

Name	Environment	Format
NuGet	.Net CLR	XML
Gradle	JVM	XML
dpkg/APT	Linux	Ar archive
Rubygems	Ruby	Ruby
MSI	Windows	In-file DB
BSD Ports	OS X/Linux/BSD	Makefile
...		

Maven

Maven is a project management tool which encompasses a project object model, a set of standards, a project lifecycle, a dependency management system, and logic for executing plugin goals at defined phases in a lifecycle. When you use Maven, you describe your project using a well-defined project object model, Maven can then apply cross-cutting logic from a set of shared (or custom) plugins

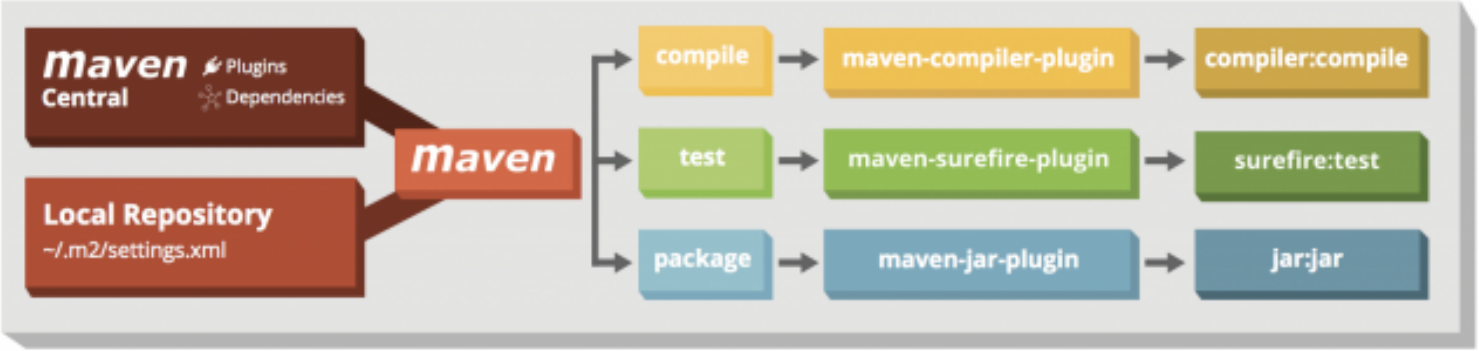
Maven - Convention Over Configuration

```
${basedir}/src/main/java

${basedir}/src/main/resources

${basedir}/src/test
```

Lifecycle , Phases and plugins
mvn clean compiler:compile package

- Three built-in Lifecycles
  - default, clean and site
- Phases in a lifecycle
  - validate, compile, test, package, verify, install, deploy
  - pre-*, post-*, or process-*
    - are not called from the cli (often used in testing)
- Phase are made of Plugin goals
  - compile complier:compile

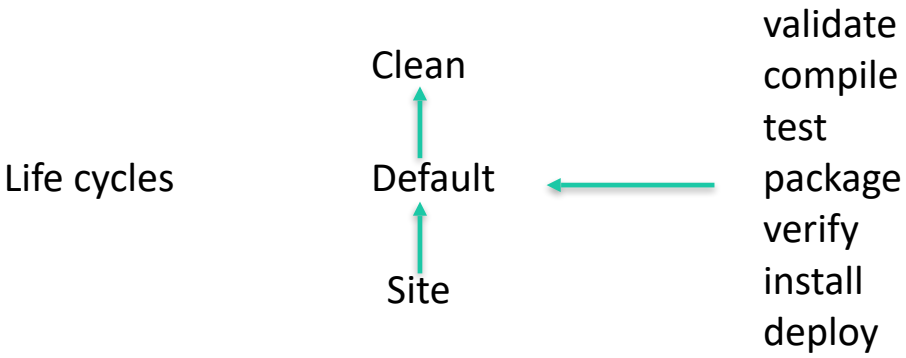
Maven- Plugins



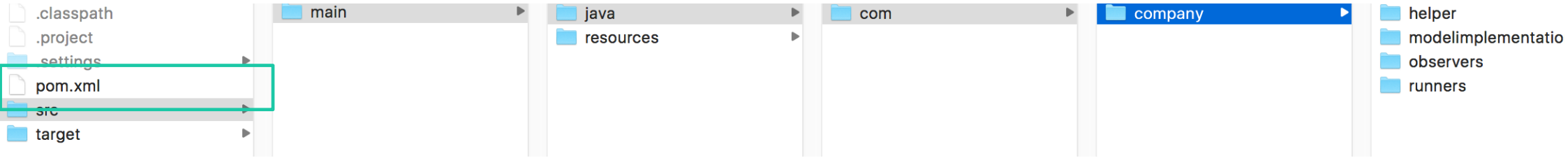
Maven


Maven™

```
mvn -h
```


Life cycles
Clean
Default
Site
validate
compile
test
package
verify
install
deploy

Maven – structure




```
mvn archetype:generate -DgroupId=com.mycompany.app -DartifactId=my-app
-DarchetypeArtifactId=maven-archetype-quickstart -DinteractiveMode=false
```

Demo

TDP024 – IN A NUTSHELL

- I examensordningen står det att för alla kandidatexamina skall bland andra följande mål uppnås: (Även andra examina än kandidatexamina har nästan identiska mål)

1. visa förmåga att söka, samla, värdera och kritiskt tolka relevant information i en problemställning samt att kritiskt diskutera företeelser, frågeställningar och situationer

2. visa sådan färdighet som fordras för att självständigt arbeta inom det område som utbildningen avser

3. visa förmåga att självständigt identifiera, formulera och lösa problem samt att genomföra uppgifter inom givna tidsramar

4. visa förmåga att muntligt och skriftligt redogöra för och diskutera information, problem och lösningar i dialog med olika grupper

TDP024 – Informationssökning

Visa förmåga att söka, samla, värdera och kritiskt tolka relevant information i en problemställning samt att kritiskt diskutera företeelser, frågeställningar och situationer

Individuell rapport och seminarier 1Hp

- Sök och välj ut tre bra forskningsartiklarna relaterade till kursens innehåll .
- Skriv en sammanfattning och analys av alla tre artiklarna på en A4-sida vardera.
- Värdera varje artikel - Varför är den bra

Uppgift

Individuell rapport och seminarie 1Hp

Denna uppgift går ut på att genomföra en fördjupning i aktuell forskning inom kursens områden samt träna på att utvärdera vetenskaplig litteratur. Samt diskutera innehållet av de ni läst på ett seminarie med de andra studenterna.

Sök och välj ut tre bra forskningsartiklarna relaterade till kursens innehåll . Skriv en sammanfattning och analys av alla tre artiklarna på en A4-sida vardera. Som språk kan antingen svenska eller engelska använda. Sammanfattningen/analysen ska utförligt besvara följande frågor:

- Vad är artikelns syfte?
- Hur har det presenterade arbetet lagts upp (tillvägagångssätt/metod)?
- Vilka resultat kommer artikeln fram till?
- Vad finns det för begränsningar i arbetet?

Avsluta varje analys med att redogöra varför artikel är en bra artikel.

Hitta

- http://scholar.google.se/
- http://dl.acm.org/
- http://ieeexplore.ieee.org/Xplore/home.jsp

Vad gör en artikel bra?

- Vem
  - Hur många artiklar
  - Erkänd inom området
- Var
  - Tidskrift eller Konferensbidrag (ev Workshop)
  - Ranking och Impact Factor
    - (http://www.robertfeldt.net/advice/se_venues/)
- Vilka andra
  - Hur många andra har citerat artikeln

Program to an interface

- **”Program to an interface”** är en metod att organisera och bygga sitt system på som är grundläggande för alla system som skall vara modulära och hållbara.
- För en programmerare är det bland de viktigaste kunskaper man kan ta med sig in i ett projekt.
- Projekt som väljer enklare metoder, eller hittar genvägar runt, har en drastiskt förkortad livslängd.
- Det är på abstraktionsnivån *programmering* som denna metod befinner sig, inte på samma abstraktionsnivå som SCRUM eller arkitektur, etc.

Program to an interface

- **I den här kursen** är det omöjligt att klara laborationerna utan att ha förståelse för ”program to an interface”.
- Eventuella avvikelser från metoden i laborationerna kommer ge komplettering.
- I arbetslivet är det en självklarhet att man har denna kunskap med sig, och deltar man i ett projekt som inte använder sig av ”program to an interface” ska man kämpa för att de börjar med det.

Program to an interface

- **I korthet:**
- Systemet man bygger skall knytas samman och bero på kontrakt mellan olika delar.
- Det skall inte bero på en viss implementation.
- T ex. ett system skall vara beroende på *”sorterar en lista”*, inte på *”bubble sort”.*
- Att ”sortera en lista” är ett kontrakt, jag säger till ett annat system *”sortera denna lista”* och får tillbaka en sorterad lista. Det spelar dock inte mig någon roll hur detta gick till, jag har skrivit kod som jobbar mot *kontraktet* inte mot en viss implementation av kontraktet.

Program to an interface

```
public interface Sort {
    List sort(List list);
}
```

När jag skriver kod så jobbar jag mot interfacet, jag vet att alla klasser som uppfyller kontraktet kommer ha en funktion som heter sort och tar en lista och returnerar en lista.

```
public class BubbleSort implements Sort {
   public List sort(List list) {
      //Do bubble sort
   }
}
```

```
public class QuickSort implements Sort {
    public List sort(List list) {
       //Do quick sort
    }
}
```

Program to an interface

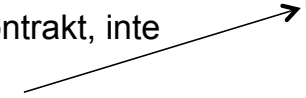
```
public interface Sort {
    List sort(List list);
}
```

```
public class BubbleSort implements Sort {
   public List sort(List list) {
     //Do bubble sort
    }
}
```

```
public class QuickSort implements Sort {
    public List sort(List list) {
       //Do quick sort
    }
}
```

```
public class QuantSort implements Sort {
    public List sort(List list) {
       //Do quick sort
    }
}
```

När någon sedan kommer fram till en snabbare sorteringsalgoritm så behöver jag inte ändra i min kod, för den är skriven mot ett kontrakt, inte mot en implementation.



Program to an interface

```
public class ListUtils {

   private Sort sort;

   public List findSmallest(List list) {
      list = sort.sort(list);
      return list.get(0);

   }

}
```

*Interface, det står **inte**:*
private BubbleSort sort;

Använder en metod som jag vet finns enligt kontraktet, oavsett vilken implementation jag använder.

”That’s a null pointer exception!”

Dependency Injection

```
public class ListUtils {

  private Sort sort;

  public ListUtils(Sort sort) {
    this.sort = sort;
  }

  public List findSmallest(List list) {
    list = sort.sort(list);
    return list.get(0);

  }

}
```

ListUtils väljer inte vilken implementation den skall använda. I sin konstruktor så accepterar den en instans av en implementation. (Notera att vi fortfarande endast använder interfaces).

Detta kallar vi **dependency injection** det består alltså av en klass som har en instansvariabel som blir satt i konstruktorn.

Dependency Injection

```
public static void main(String [] args) {
   ListUtil listUtil = new ListUtil(new QuickSort());
   int min = listUtil.findSmallest([21,13,55,34])
}
```

Någon gång måste man välja implementation, i detta fall är det koden som skapar ett nytt objekt av ListUtil som måste välja vilken implementation som skall användas.

För den intresserade så går det även att få bort detta beroende. Med hjälp av *”Inversion of Control”* kan man få ”program to an interface on steroids”.

Du anger aldrig vilken implementation som skall användas i koden, utan detta löses ”runtime” från t ex en XML fil som säger vilken implementation av Sort som skall användas just nu.

Vi har inte tid med ”Inversion of Control” i denna kurs, men det rekommenderas till den intresserade.

Program to an interface

- Enhetstestning drar också stora fördelar från att fokusera på kontrakt snarare än implementation.
- ”Test the contract, not the implementation”

- Om vi skriver våra test mot kontrakt så kan vi testa oändligt många implementationer av samma kontrakt.
- Om någon kommer och påstår att de har en ny implementation som är bättre, så kan man testa implementationen mot redan existerande tester.

Program to an interface

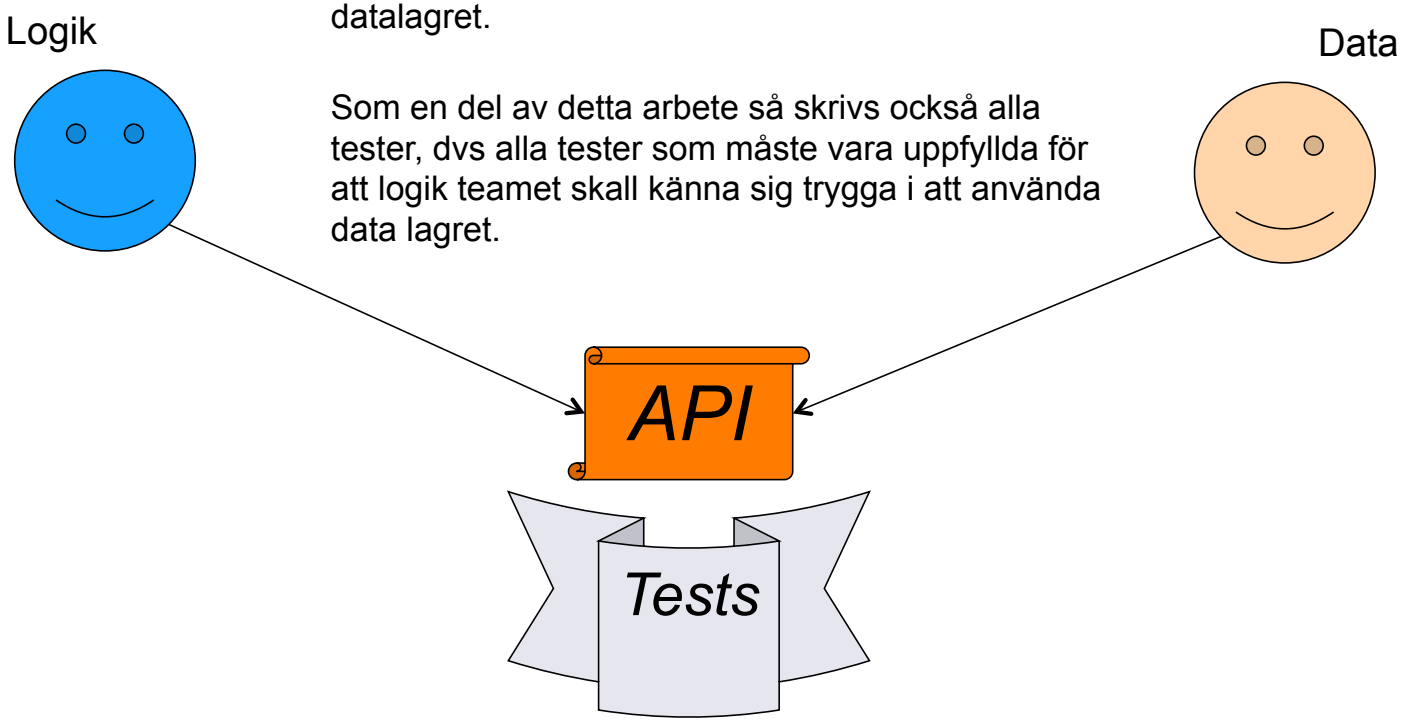
```
public class Test {

   //--- Unit under test ---//
   private Sort sort = new BubbleSort();
   //----------------------//

   public void test() {
      List a = [4,3,5,2,6];
      List b = sort.sort(a);
      Assert.assertTrue(b[0] == 2);
   }

}
```

- Koden i testet är oberoende av implementationen.
- Innan vi kör testet kan vi byta implementation.
- Om någon kommer och påstår att de har en bättre implementation, så tar vi deras kod och instansierar deras lösning istället.
- Testen är exakt desamma.

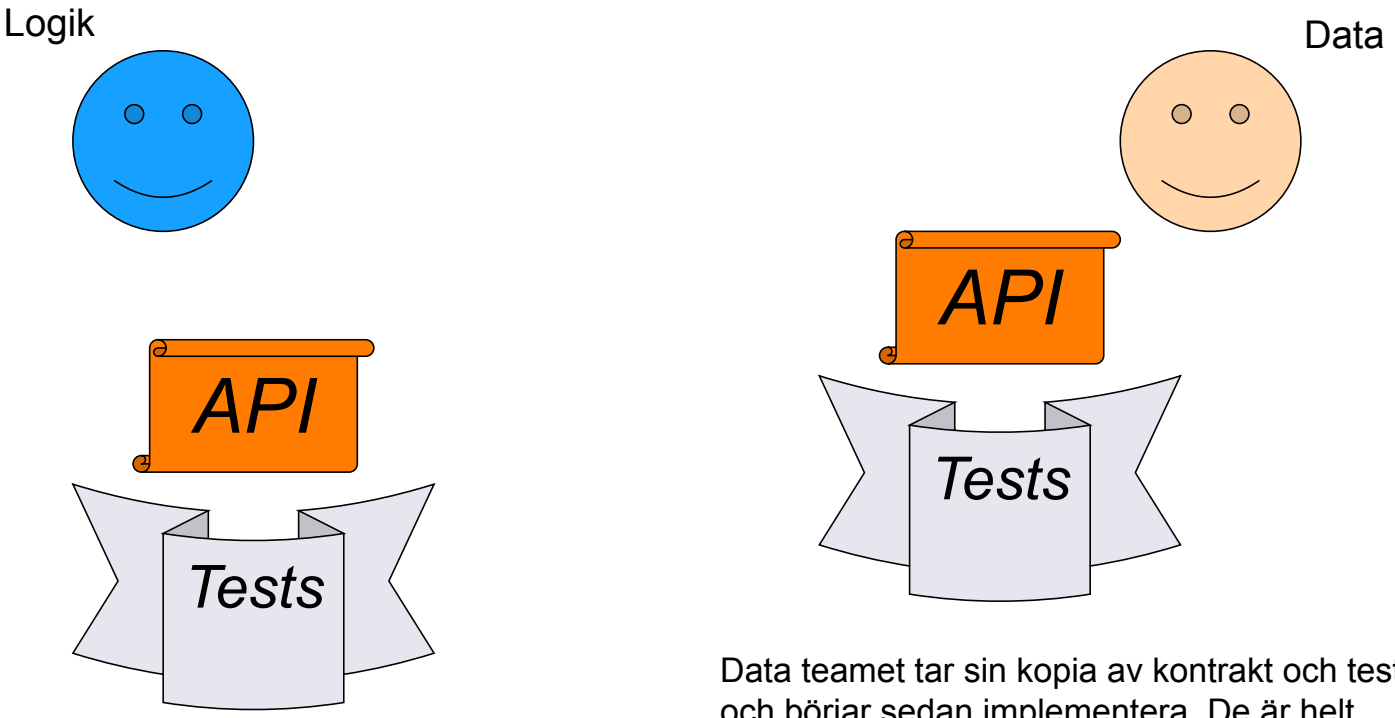
Data – Logic - Web

Data och Logik teamen sitter tillsammans och kommer överens om ett API (dvs ett kontrakt) för vilka funktioner som kommer behövas från datalagret.

Som en del av detta arbete så skrivs också alla tester, dvs alla tester som måste vara uppfyllda för att logik teamet skall känna sig trygga i att använda data lagret.

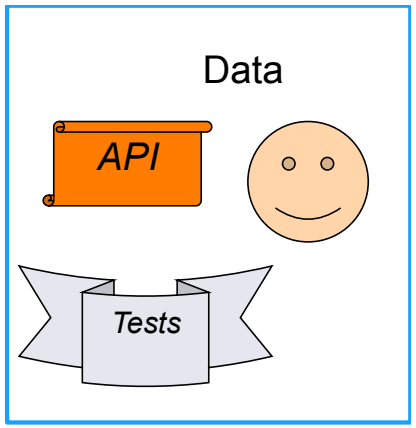


Data – Logic - Web

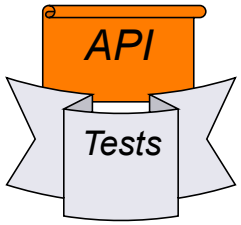


Data teamet tar sin kopia av kontrakt och tester och börjar sedan implementera. De är helt oberoende av alla andra delar av projektet. Och resten av projektet är helt oberoende av deras implementationsdetaljer.

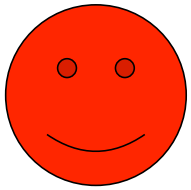
Data – Logic - Web



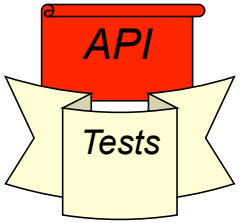
Logik



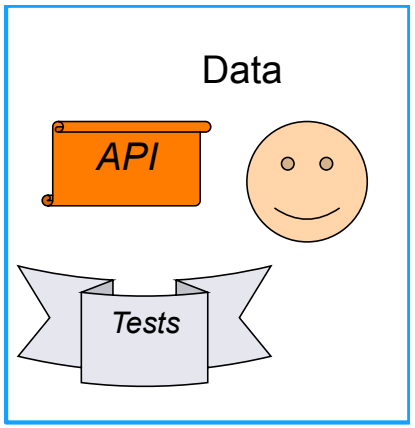
Web

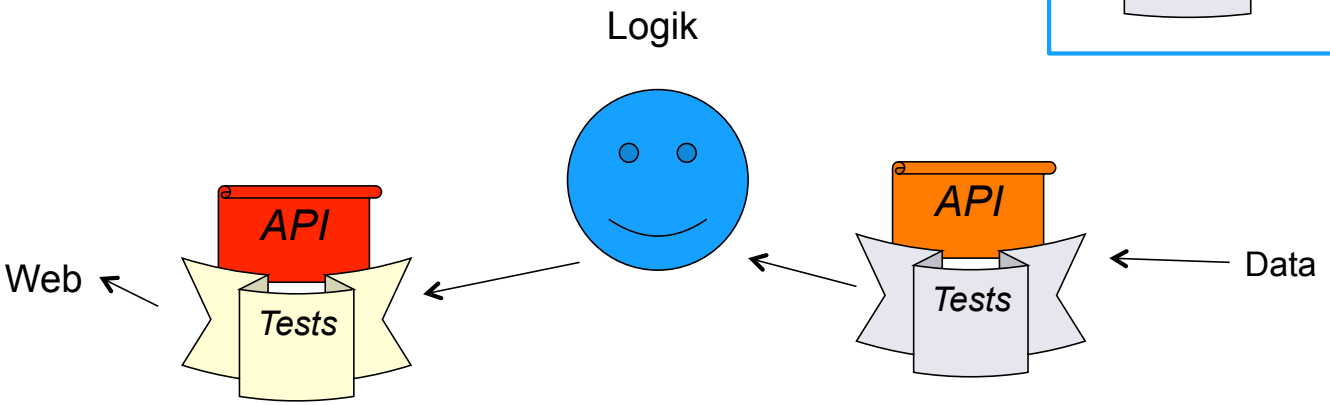


Web och logik teamen gör precis samma sak. De kommer överens om ett kontrakt för kommunikation mellan web och logiklagret. I denna process skrivs även tester.



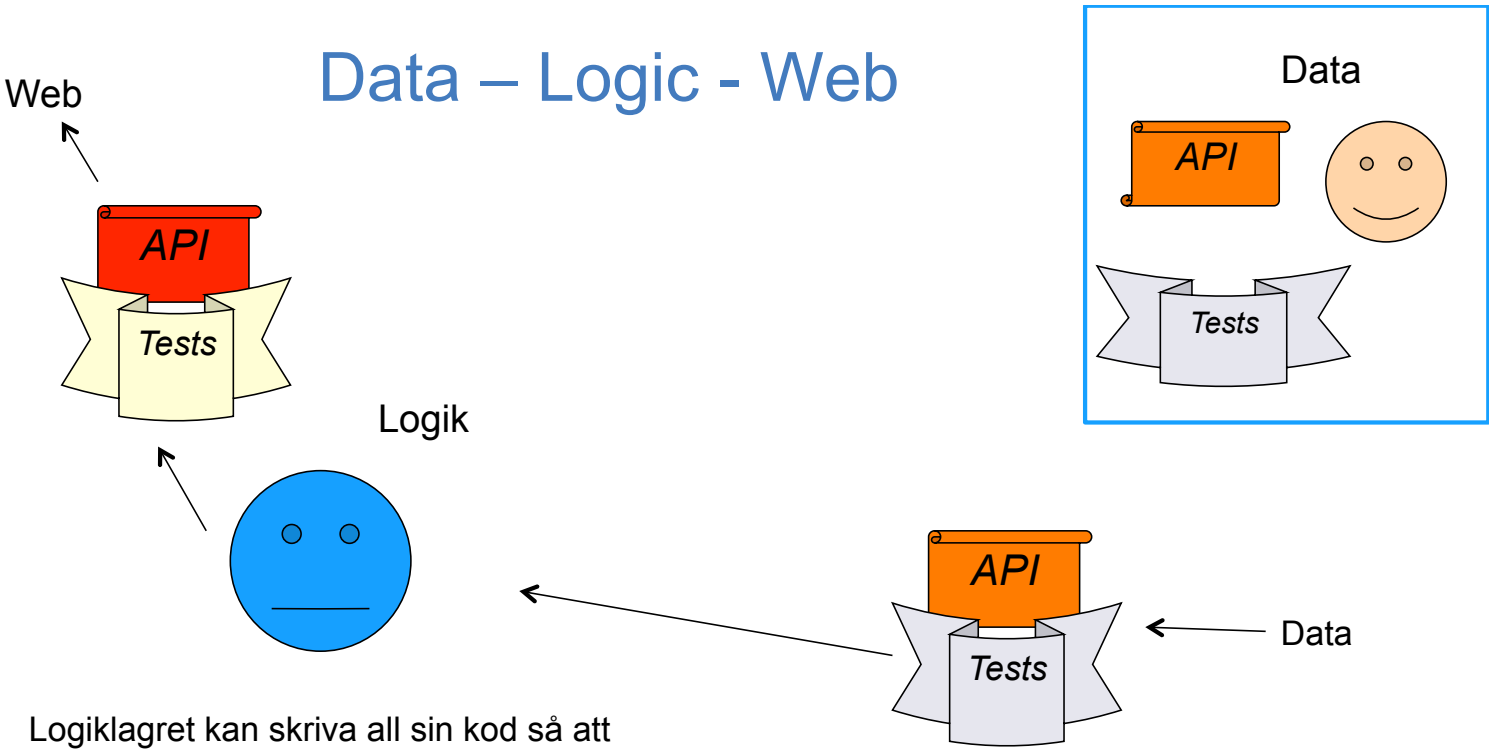
Data – Logic - Web





Nu har logik lagret ett kontrakt som de vet kommer bli uppfyllt (samt tester). Och de har ett kontrakt som de skall uppfylla (samt tester).

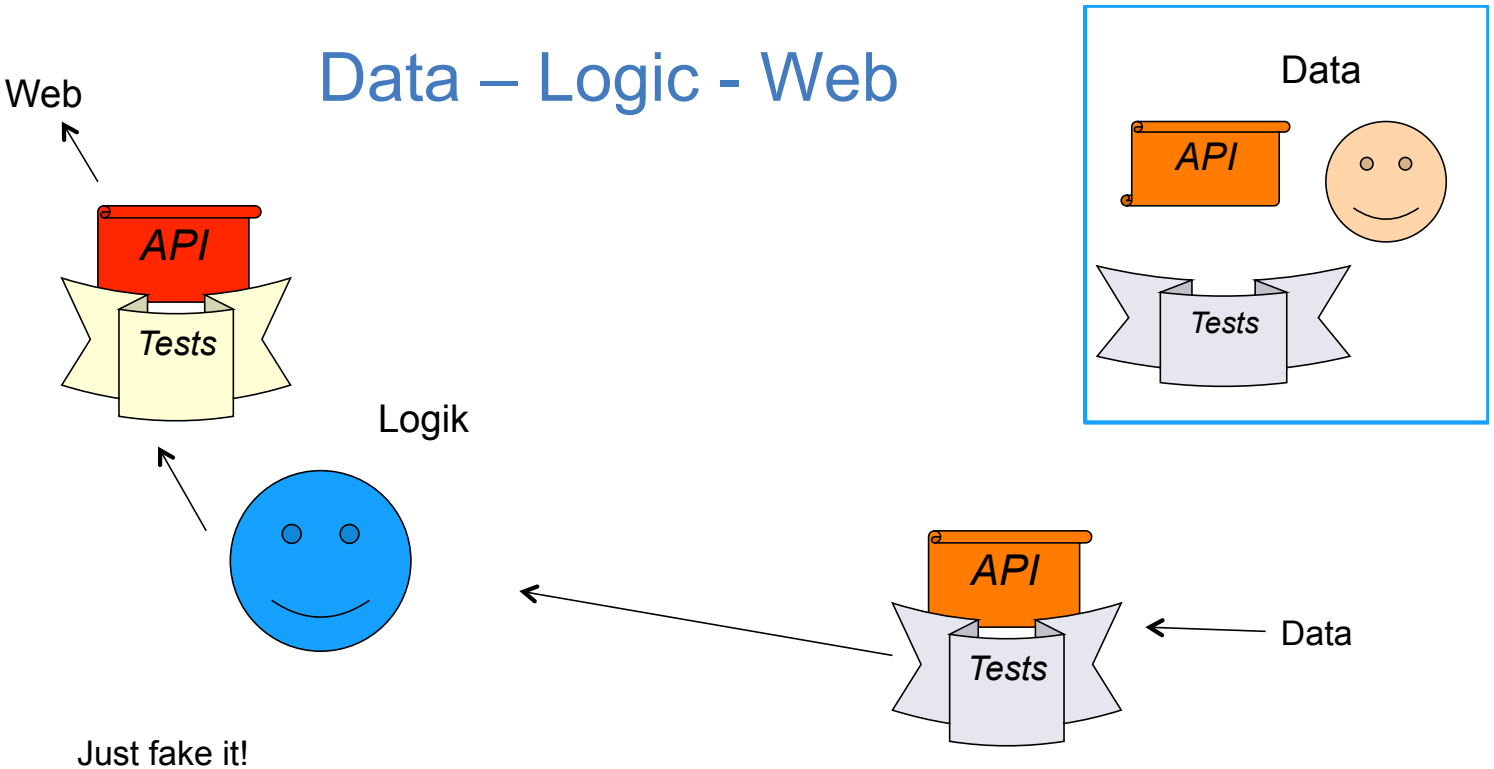
Data – Logic - Web



Logiklagret kan skriva all sin kod så att den uppfyller kontraktet mot web. Eftersom logiklagret **ALLTID** jobbar mot kontraktet så behöver man inte vänta på en implementation från datalagret.

Men hur kan man testa sin logik utan att kunna skapa instanser av de kontrakt som data lagret ska leverera? Måste man vänta på att data lagret blir klara?

Data – Logic - Web



Just fake it!

Logik lagret har ju kontrakten från data lagret, de kan skapa sitt egna lilla fejk datalager som de kan använda för att testa med.

Detta kallas ofta *”mocking”* eller *”creating mock objects”*

Mock objects - Mocking

```
public class Test {

   //--- Unit under test ---//
   private Sort sort = new Sort() {
      public List sort(List sort) {
        return [2];
      }
   }
   //----------------------//

   public void test() {
      List a = [4,3,5,2,6];
      List b = sort.sort(a);
      Assert.assertTrue(b[0] == 2);
   }

}
```

- Så vi skapar en egen implementation av Sort (till dess att data lagret är klara med sin) som vi kan testa med.
- Den behöver inte vara bra, så länge den gör vad den ska.
- När data lagret sedan är klara så stoppar vi in deras implementation istället.