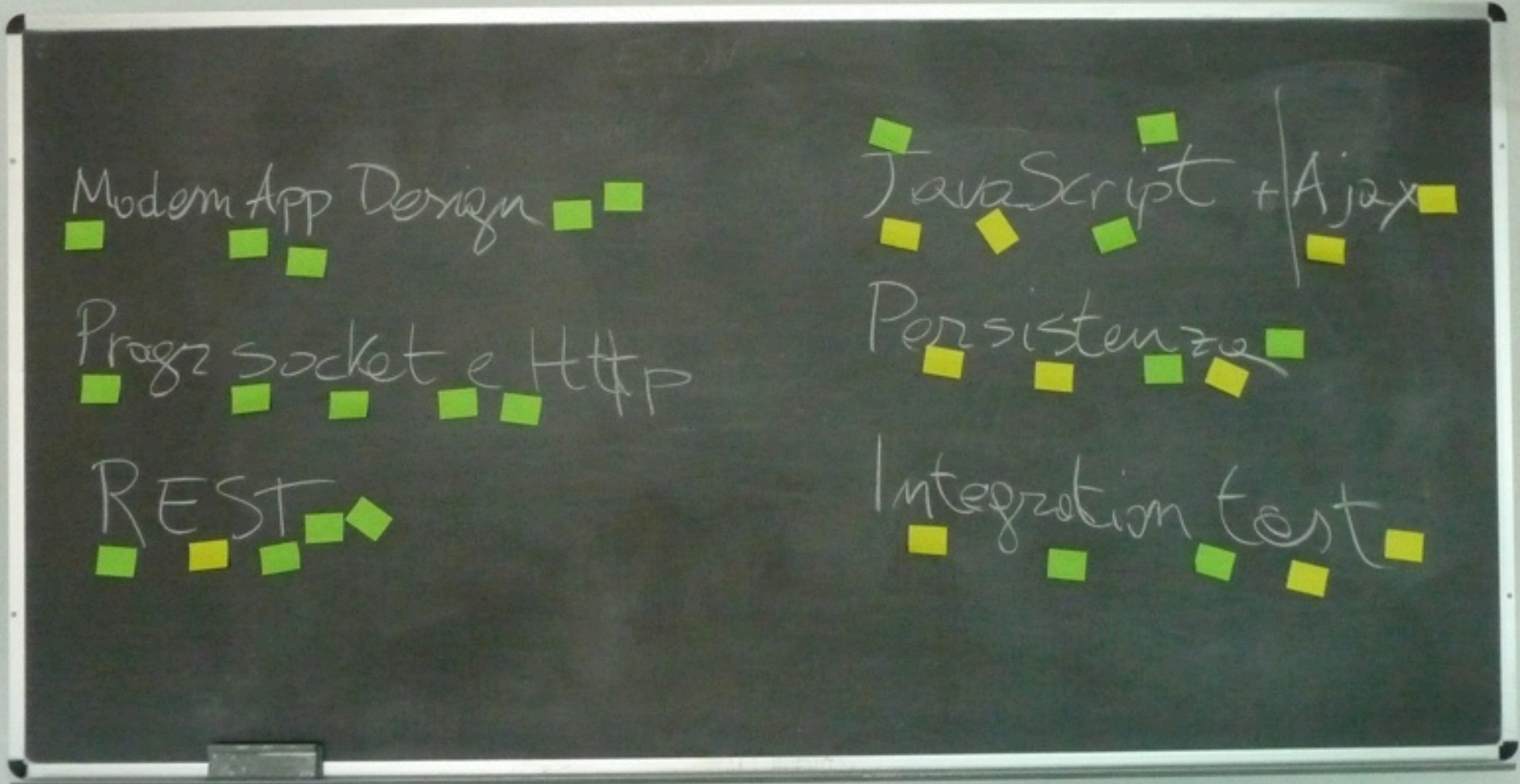
Programmazione web libera dai framework

Matteo Vaccari
matteo.vaccari@xpeppers.com

Chi son io?

- Ho sviluppato applicazioni web in PHP, Java, Ruby (on Rails)
- In particolare Rails, Wicket, Spring, Hibernate
- Insegno Applicazioni Web I e II all'Insubria
- Lavoro in *XPeppers* come consulente e mentor


Modern App Design
Progr socket e Http
REST
JavaScript + Ajax
Persistenza
Integration test

Perché usiamo i framework?

Davvero risparmiamo tempo?

- Dibattere su quale framework sia il migliore
- Imparare a usare il framework
- Capire come si fa a fare X
- Seguire le mailing list
- Aggiornare l’app all’ultima versione del FW

E i rischi?

- Performance insufficiente?
- Contiene errori che non sappiamo correggere? Errori intermittenti?
- Si scopre che non supporta la feature *X*
- E se poi il nostro framework passa di moda?

Gli oggetti scomparsi

```
# controllers/employees_controller.rb
class EmployeesController < ApplicationController
  def index
    @employees = Employee.find(:all)
  end
end

# views/employees/index.html.erb
<table>
  <% for employee in @employees %>
  <tr>
    <td><%= employee.name %></td>
  </tr>
  <% end %>
</table>

# Missing objects....(pseudocodice)
controller = router.find_controller(request.uri) # un EmployeeController
action = router.find_action(request.uri)         # "index"
controller.send(action)

# e ancora...
stream = OutputStream.new
view = View.new("employees/index.html.erb")
view.render_on(stream)
```

Davvero, perché usiamo i framework?

I framework incrementano il costo, la complessità e il rischio

Usa la forza *degli* **oggetti**, Luke!

Programmare *a oggetti*

```
// Qui ficcanasiamo troppo
cane.getCorpo().getCoda().scodinzola();


// Tell, don’t ask!
cane.esprimiContentezza();
```

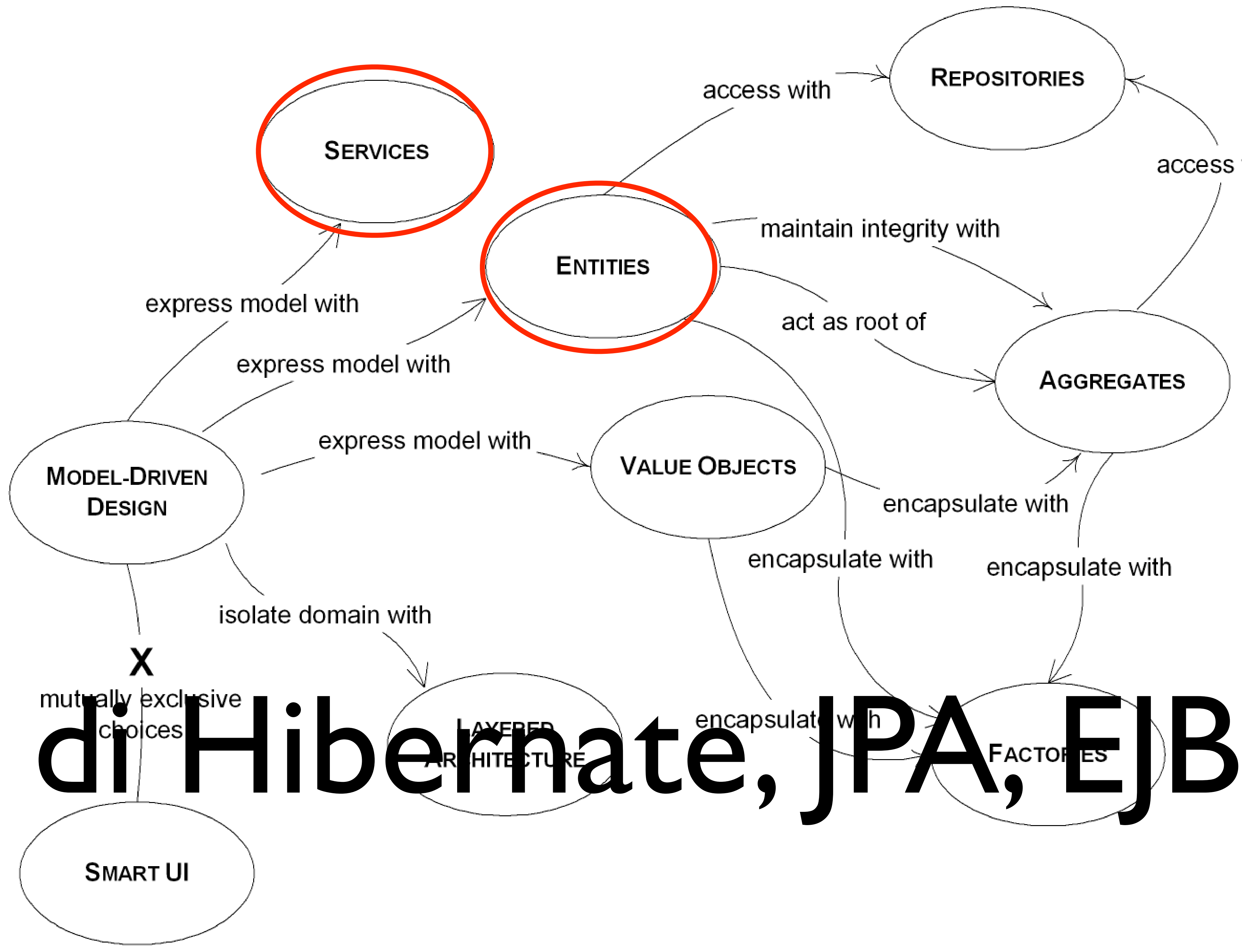
Steve Freeman et al.: Mock Roles, not Objects

```java
@Entity
@Name("user")
@Table(name="users")
public class User implements Serializable {

   private String username;
   private String password;
   private String name;

   public User(String name, String password, String username) {
      this.name = name;
      this.password = password;
      this.username = username;
   }

   public User() {}

   public String getPassword()  {
      return password;
   }

   public void setPassword(String password) {
      this.password = password;
   }

   public String getName() {
      return name;
   }

   // ...
```

Questo non è un oggetto...

è una *struttura dati!*

Il lato oscuro del DDD



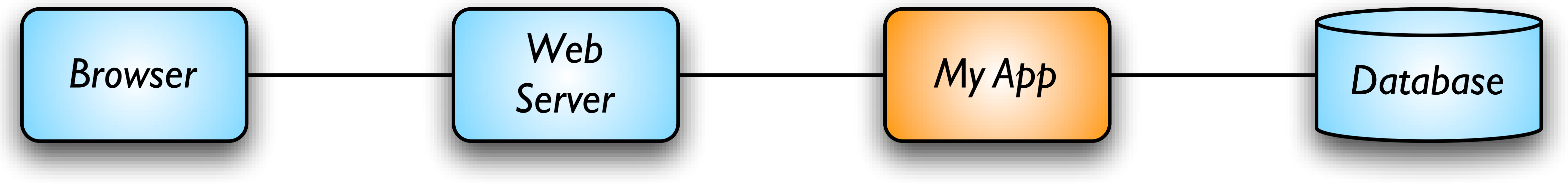
E di Hibernate, JPA, EJB3...

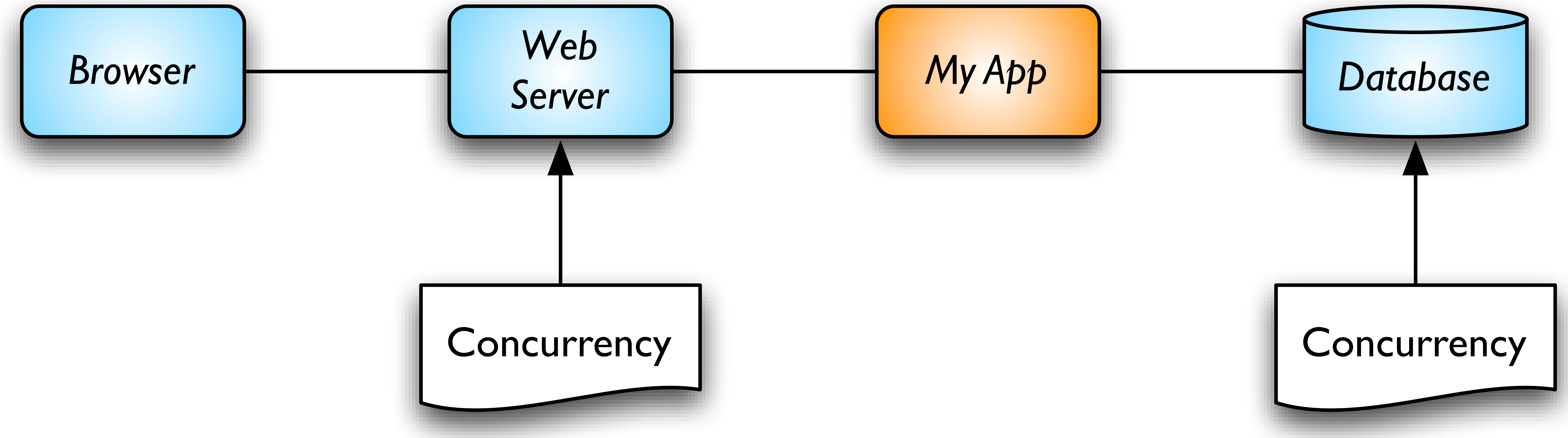
Il *domain model* si riduce a uno *schema dei dati*

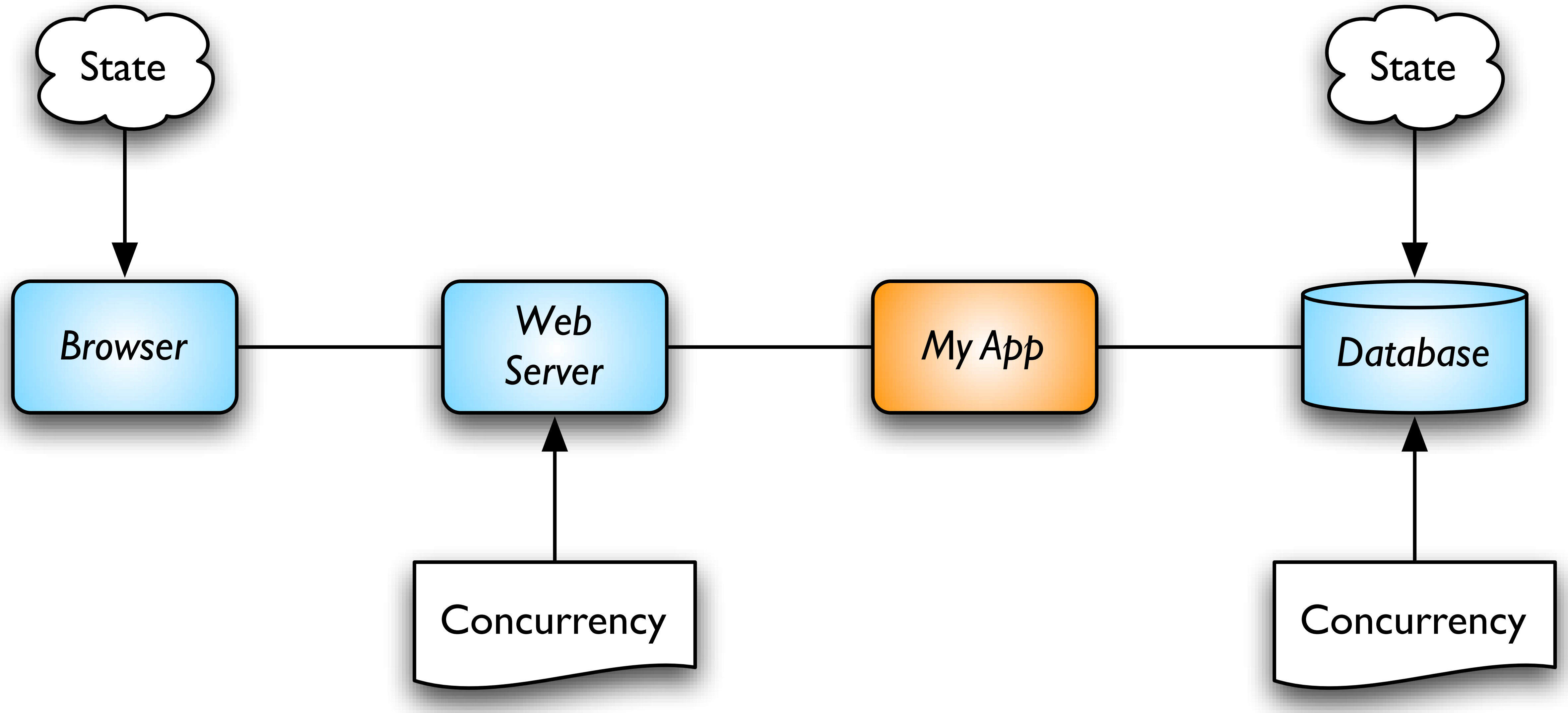
Procedure (non oggetti)

Contro la paura impariamo:

- A programmare *bene*. A oggetti.
- Gli standard di base: *HTTP, URI, HTML, CSS*
- I nostri strumenti: linguaggio, web server, database


Browser
Web Server
My App
Database


Browser
Web Server
My App
Database
Concurrency
Concurrency


State
State
Browser
Web Server
My App
Database
Concurrency
Concurrency

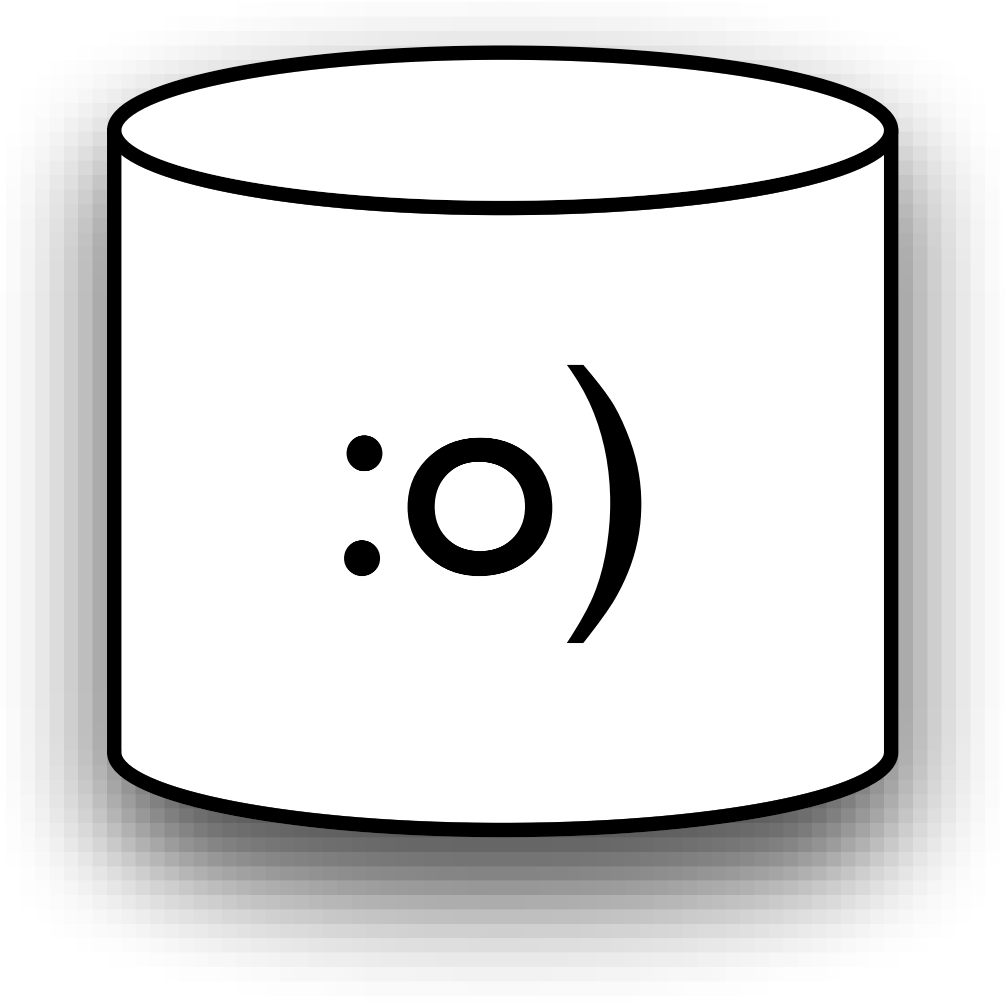
Una servlet come punto di partenza

```
public class MyOnlyServlet extends HttpServlet {

   @Override
   protected void service(HttpServletRequest request,
                          HttpServletResponse response) throws ... {
      MainPage page = new MainPage();
      page.service(request.getParameter("foo"));
   }

}
```

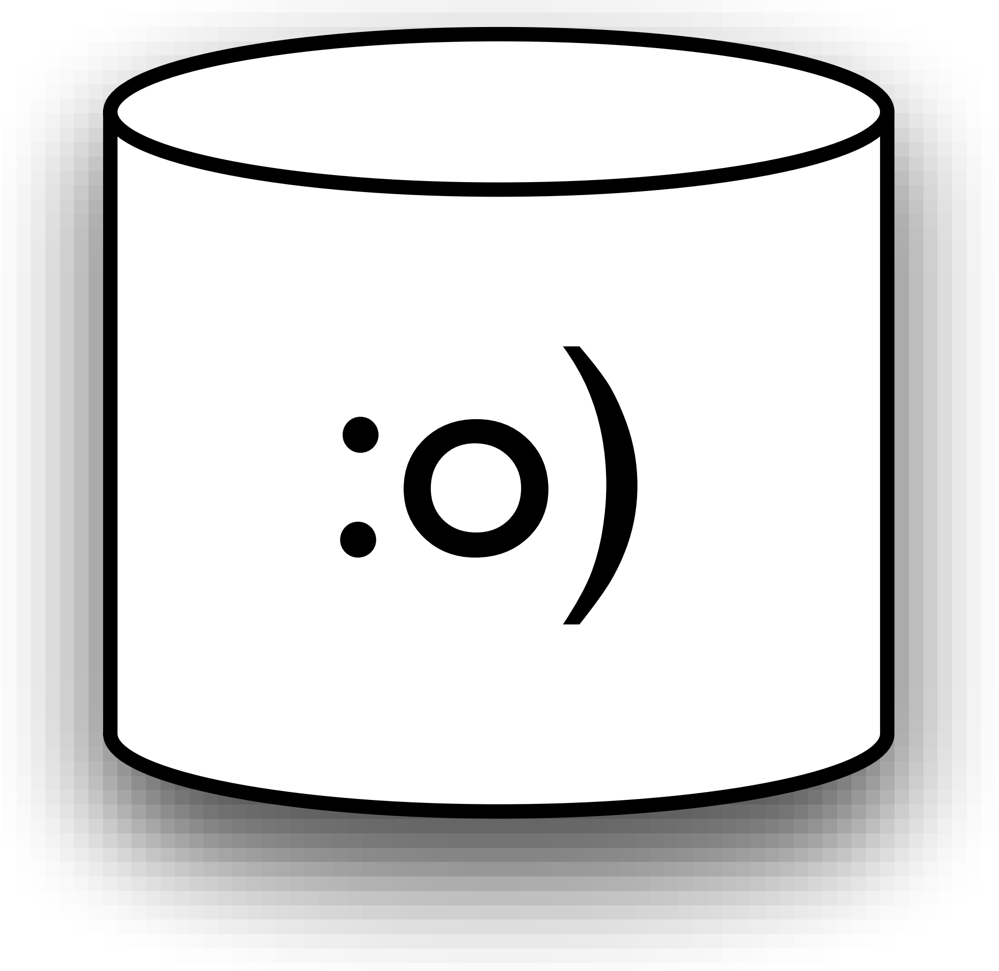
Usare la dependency injection

```java
@Override
protected void service(HttpServletRequest request,
                       HttpServletResponse response) throws ServletException, IOE

   Connection connection = new JndiDataSource("java:comp/env/jdbc/CourseDB")
      .getConnection();
   CourseCatalogue courses = new JdbcCourseCatalogue(connection);
   CoursesApplication app = new CoursesApplication(courses);

   app.service(request, response);
}
```

The database is your friend



The database is your friend



- Transazioni
- Concorrenza
- Ricerche

We don't need no EJBs!

Una transazione HTTP
==
Una transazione DB

```
@Override
protected void service(HttpServletRequest request, HttpServletResponse response) th
    Connection connection = null;
    try {
        DataSource source = new JndiDataSource("java:comp/env/jdbc/CourseDB");
        connection = source.getConnection();
        CourseCatalogue courses = new JdbcCourseCatalogue(connection);
        CoursesApplication app = new CoursesApplication(courses);
        app.service(request, response);
        connection.commit();
    } catch (Exception e) {
        rollback(connection);
        throw new ServletException(e);
    } finally {
        close(connection);
    }
}
```

Una semplice interfaccia al DB

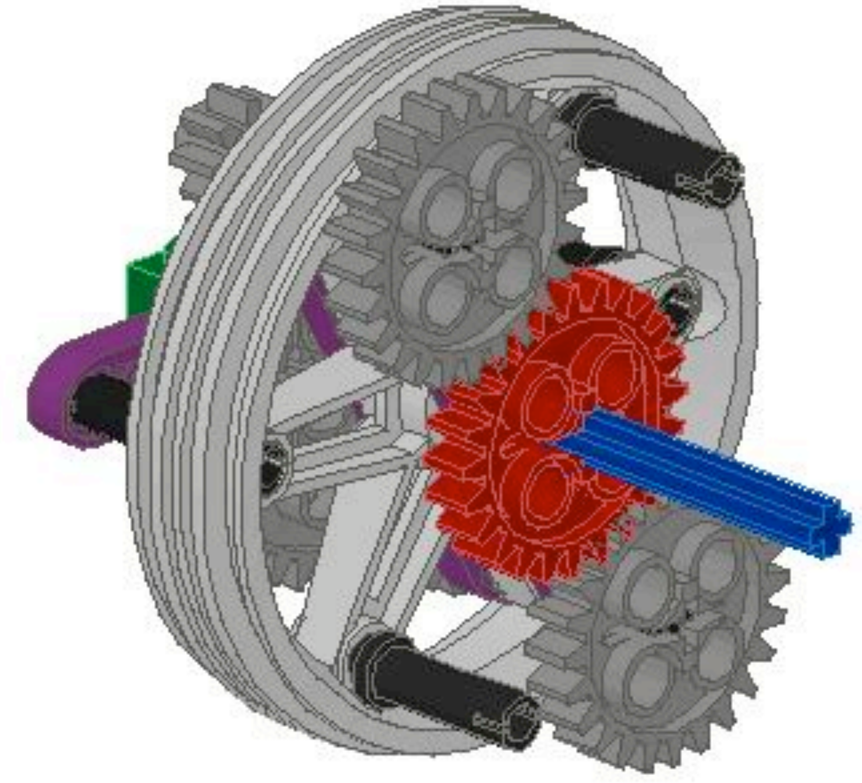
```
public interface Database {

    Map<String, Object> selectOneRow(String sql, Object ... params);

    void execute(String sql, Object ... params);

    List<Map<String, Object>> selectMultipleRows(String sql, Object ... params);

}
```

```java
public class DatabaseCarrierRepository implements CarrierRepository {

    final Database database;

    @Override
    public List<Channel> findAllChannels(CarrierView view) {
        String query = "SELECT ID, NAME FROM CHANNEL ORDER BY NAME";
        List<Map<String, Object>> rawChannels = database.selectMultipleRows(query);

        List<Channel> channels = new ArrayList<Channel>();
        for (Map<String, Object> map : rawChannels) {
            Channel channel = new Channel(map.get("NAME").toString(),
                        ((BigDecimal) map.get("ID")).intValue());
            channels.add(channel);
        }
        return channels;
    }
```

Il “routing” delle pagine

```
// GeometryServlet.service()
protected void service(HttpServletRequest request, HttpServletResponse response)
   List<PageComponent> pages = new ArrayList<PageComponent>();

   pages.add(new WelcomePage());
   pages.add(new SquareAreaPage());
   pages.add(new TriangleAreaPage());
   pages.add(new SphereVolumePage());

   GeometryApplication app = new GeometryApplication(pages);
   app.service(request, response);
}

// GeometryApplication.service()
public void service(HttpServletRequest request, HttpServletResponse response) thr
   for (PageComponent component : components) {
      if (component.wantsToHandle(request)) {
         response.getWriter().print(component.toHtml());
         return;
      }
   }
   response.sendError(404);
}
```

Project automation



http://technicbricks.blogspot.com/

Quick feedback

```
#!/bin/bash
# script/server.sh
# start app on port 8080

ant war || exit 1
java -jar lib/winstone-0.9.10.jar --warfile target/*.war  $*
```

Server up in < 2s

```
$ script/server.sh
Buildfile: /Users/matteo/work/conferences/webtech/projectPortfolio/build.xml

prepare:

compile:

war:

BUILD SUCCESSFUL
Total time: 0 seconds
[Winstone 2010/11/10 00:18:04] - Beginning extraction from war file
[Winstone 2010/11/10 00:18:04] - No webapp lib folder found - /private/var/folders/Cb/CbG3BVbs
[Winstone 2010/11/10 00:18:04] - HTTP Listener started: port=8080
[Winstone 2010/11/10 00:18:04] - AJP13 Listener started: port=8009
[Winstone 2010/11/10 00:18:04] - Winstone Servlet Engine v0.9.10 running: controlPort=disablec
```

Dominare il database

```
#!/bin/bash
# script/create_databases.sh
# create and populate databases for development and test environments

echo 'Drop databases...'
mysqladmin -uroot --force drop db
mysqladmin -uroot --force drop db_test

echo 'Create databases...'
mysqladmin -uroot create db
mysqladmin -uroot create db_test
echo "grant all on db.* to db@localhost identified by 'db';" |  mysql -uroot
echo "grant all on db_test.* to db@localhost identified by 'db';" |  mysql -uroot

echo 'Build schema...'
cat db/*.sql | mysql -udb db -pdb
cat db/*.sql | mysql -udb db_test -pdb

echo 'Populate development...'
mysql -udb -pdb db < db/populate_db.sql

echo 'Done!'
```

Incremental SQL scripts

```
$ ls
001_create_contents.sql
002_add_columns_to_users.sql
003_add_filtri_per_operatore.sql
004_add_custom_fields.sql
005_add_publisher_issues.sql
006_add_columns_content.sql
007_add_media_parade_things.sql
008_alter_publication_issue.sql
009_create_audit_log.sql
010_create_phones.sql
011_add_media_parade_codes.sql
012_add_media_partner_codes.sql
013_add_alias_services.sql
014_delete_custom_field_name_fk_from_user.sql
015_add_indexes.sql
016_add_more_indexes.sql
...
```

```sql
alter table contents
        add pull_downloads int,
        add ivr_downloads  int;
update schema_info set version = 6;
```

Generare HTML

```java
public class Course {

  // ...

  public void renderOn(CourseView view) {
    view.setCourseTitle(title);
    view.setCourseDescription(description);
  }

  // ...
}
```

Testable

```java
@Test
public void rendersCourseView() throws Exception {
   Course course = new Course("A Title", "A Description");
   FakeCourseView view = new FakeCourseView();

   course.renderOn(view);

   assertEquals("A Title - A Description", view.toHtml());
}
```

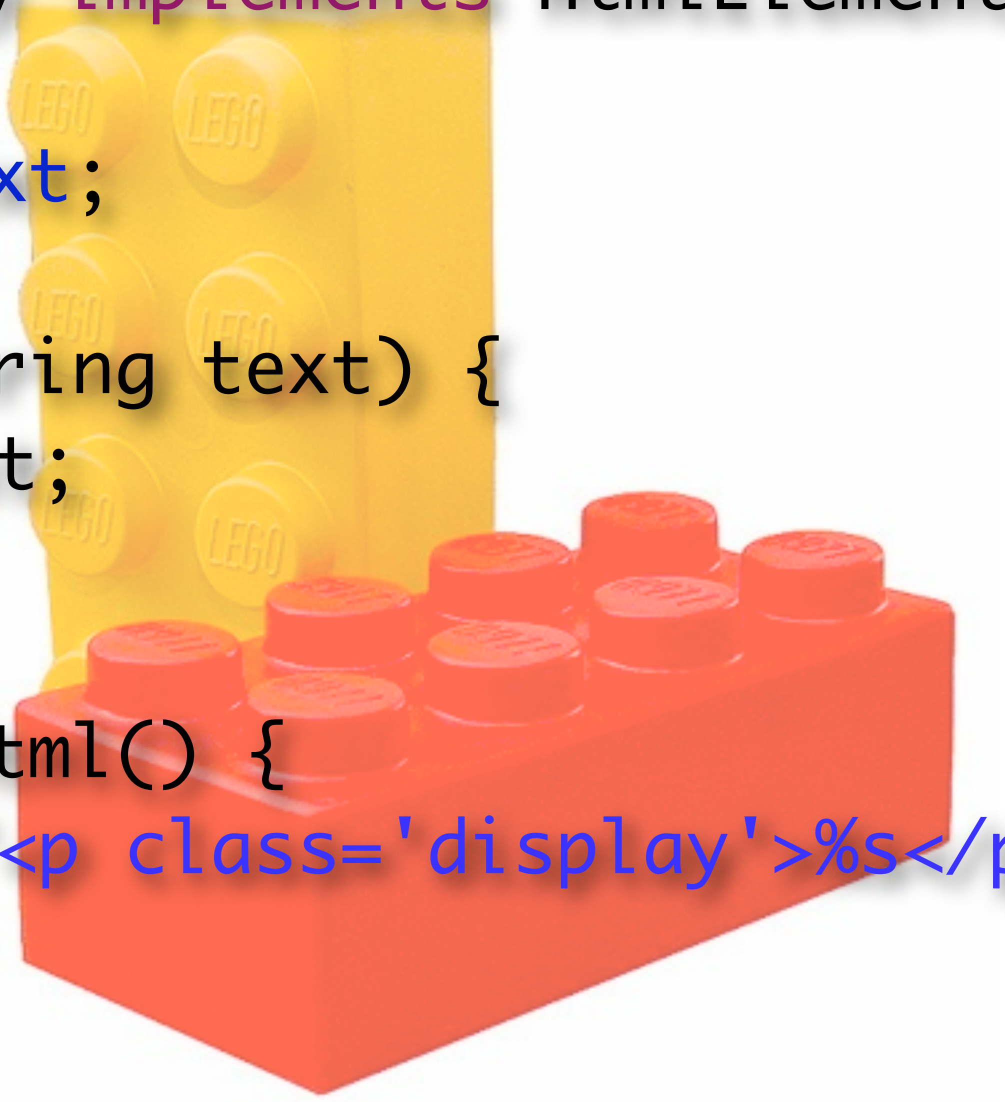
Generare HTML: template

```java
class FreemarkerCourseView implements CourseView {
   Map context = new HashMap();

   public void setCourseTitle(String title) {
      context.put("title", title);
   }

   public void renderOn(Writer writer) throws IOException, TemplateException {
      Configuration configuration = new Configuration();
      Template template = configuration.getTemplate("coursePage.ftl");
      template.process(context, writer);
   }

   public String toHtml() {
      StringWriter writer = new StringWriter();
      renderOn(writer);
      return writer.toString();
   }
}
```

Generare HTML: oggetti

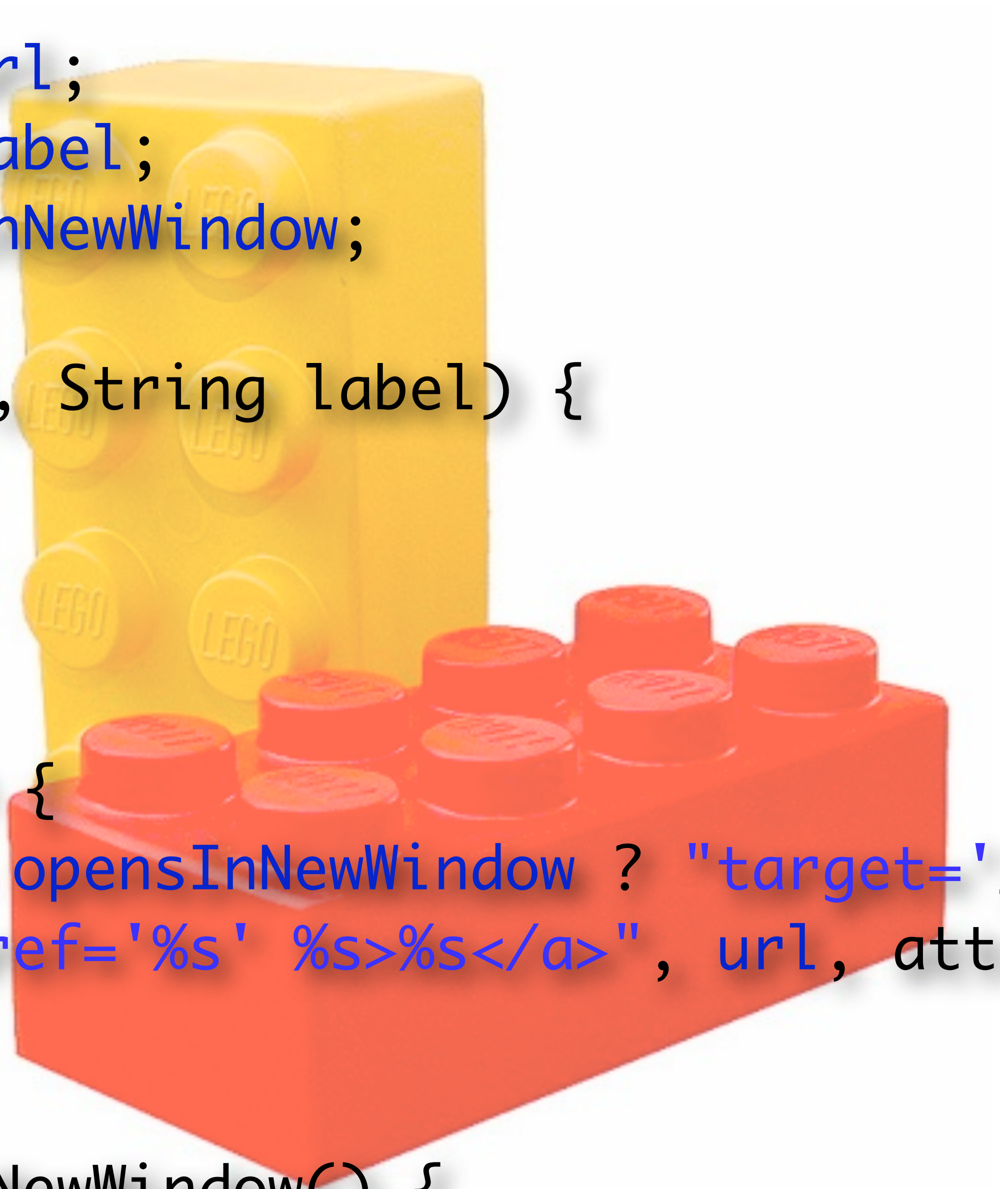
```
class ObjectOrientedCourseView implements CourseView {
   private String title;
   private String description;

   public void setCourseTitle(String title) {
      this.title = title;
   }

   public void setCourseDescription(String description) {
      this.description = description;
   }

   public void renderOn(Writer writer) throws IOException, TemplateException {
      HtmlDivision div = new HtmlDivision().with("id", "course");
      div.add(new Display(title));
      div.add(new TextBlock(description));
      div.renderOn(writer);
   }
}
```

Blocchi da costruzione



```
public class Display implements HtmlElement {

  private String text;

  public Display(String text) {
    this.text = text;
  }

  public String toHtml() {
    return format("<p class='display'>%s</p>", text);
  }
}
```

Altri blocchi



```
public class Link implements HtmlElement {

   private final String url;
   private final String label;
   private boolean opensInNewWindow;

   public Link(String url, String label) {
      this.url = url;
      this.label = label;
   }

   public String toHtml() {
      String attributes = opensInNewWindow ? "target='_blank'" : "";
      return format("<a href='%s' %s>%s</a>", url, attributes, label);
   }

   public void setOpensInNewWindow() {
      opensInNewWindow = true;
   }
}
```

Anche form e layout

```java
public Page getTemperatureConversionPage(Map<String, String> parameters) {
   Display display = new Display(converter.convert(parameters.get("temp")));

   Form form = new Form("/", "get"));
   form.add(new TextField("Temperatura:", "temperature", parameters.get("temp")));
   form.add(new SubmitButton("Converti"));

   Page page = new Page();
   page.addStylesheet("ourstyle");
   page.addDisplay(display);
   page.addForm(form);
   return page;
}
```

In conclusione?

- Usa la forza degli oggetti
- Sfrutta i tuoi strumenti
- Sii consapevole delle conseguenze dei framework

Grazie dell'attenzione!

XPeppers

Extreme Programming: sviluppo e mentoring