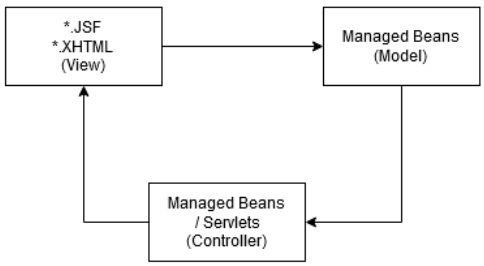
- Are there standard ways of doing things? (Like MVC support or ORM out of the box)

Standard ways for MVC do exist, it is handled with the java faces: (JSF)

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 T
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-trar

<html xmlns = "http://www.w3.org/1999/xhtml">
   <head>
      <title>JSF Tutorial!</title>
   </head>

   <body>
      #{helloWorld.message}
   </body>
</html>
```




```
import javax.faces.bean.ManagedBean;
import javax.faces.bean.RequestScoped;

@ManagedBean(name = "message", eager = true)
@RequestScoped
public class Message {
   private String message = "Hello World!";

   public String getMessage() {
      return message;
   }
   public void setMessage(String message) {
      this.message = message;
   }
}
```

```
import javax.faces.bean.ManagedBean;
import javax.faces.bean.ManagedProperty;
import javax.faces.bean.RequestScoped;

@ManagedBean(name = "helloWorld", eager = true)
@RequestScoped
public class HelloWorld {
   @ManagedProperty(value = "#{message}")
   private Message messageBean;
   private String message;

   public HelloWorld() {
      System.out.println("HelloWorld started!");
   }

   public String getMessage() {

      if(messageBean != null) {
         message = messageBean.getMessage();
      }
      return message;
   }
```

Tutorials:
https://www.tutorialspoint.com/jsf/jsf_quick_guide.htm

For ORM Mapper there exists the default JPA Interface for ORM Mapping. This is coming by default with the JEE context but must also be configured properly to work.

- Can you foresee that it will often be necessary to deviate from the standard way?

Since there are powerful frontend languages like angular react which make life easier to create frontend components we would say yes there are reasons to deviate from the standard. But that hast o do with the current approaches with SPA and improvement of frontend frameworks.

- Are there secure defaults?

There are some defaults but the most time security has to be configured by the developer themselves. By default all the backend logic ist accessible anonymously and through http. Security roles (Authorization and therefore Authentication), https must be configured by the developer.

There Are default security points implemented in the MVC Concept for secure generation of session-keys or hadling CSRF.

- How easy is it to deviate from the standard way securely?

It is very easy tho deviate from the standard since everything is configurable and at very many points a developer can intercept the process a developer is able the deviat at many points from standard or the best practices.

- Does a standard web app have many moving parts that need to be configured/programmed separately? How does that impact security?

From the config point of view there are much configuration properties. The benefit though ist that the configuration is always the same, since everything is defined in the JEE interface. The issue ist hat several implemtnations given by the webapp server may require further specific configuration and is a component more to be handled correctly. Since the framework is so flexible in means of adding modules like persistence layer, authorization types, request handling, etc. one most really be careful that all these things create a secure application as one.

- Where are changes made if you want to change your app from 10 users to 10 000 users? Just config files? Just code? Both?

JEE brings up the EJB concept, which basically are there to handle the server side compute load. They can be made serializeable so they can be shifted around a clustered server setup to support horizontal scalability. However the fact how good the app scales with clustering stronly depends on the application implementation. By default JEE brings up scaleability by default in terms of delegating it tot he server runtime but the application must be able to handle the constraints that come with it.