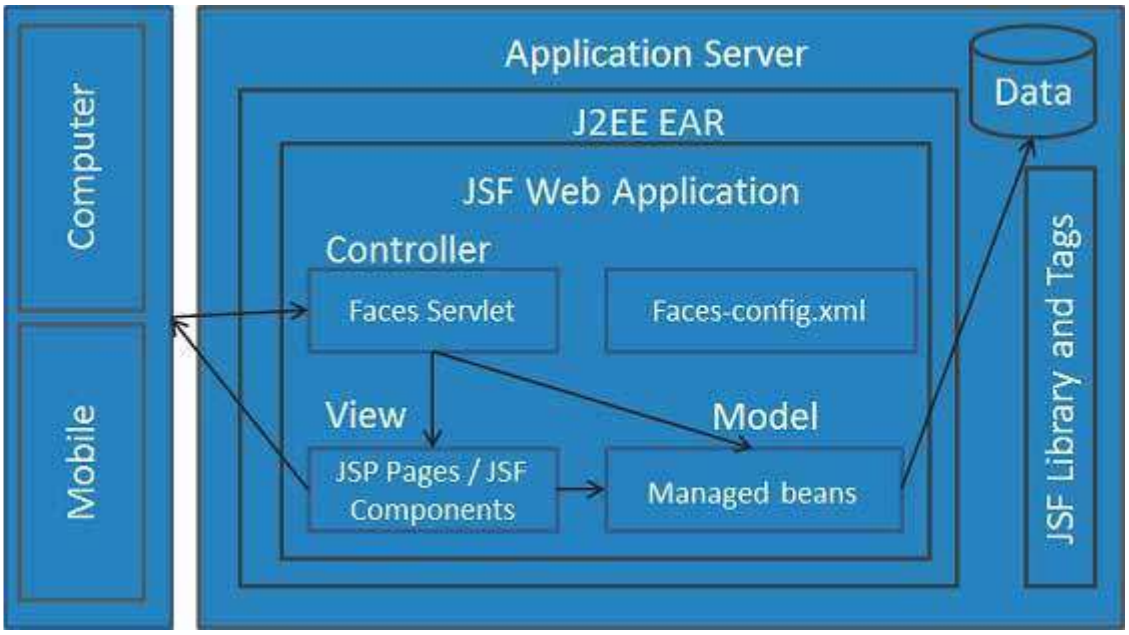
Las fases del ciclo de vida JSF son las siguientes:

1. **Restaurar la vista** (*restore view*). En este paso se obtiene el árbol de componentes correspondiente a la vista JSF de la petición. Si se ha generado antes se recupera, y si es la primera vez que el usuario visita la página, se genera a partir de la descripción JSF.
2. **Aplicar los valores de la petición** (*apply request values*). Una vez obtenido el árbol de componentes, se procesan todos los valores asociados a los mismos. Se convierten todos los datos de la petición a tipos de datos Java y, para aquellos que tienen la propiedad `inmediate` a cierta, se validan, adelantándose a la siguiente fase.
3. **Procesar las validaciones** (*process validations*). Se validan todos los datos. Si existe algún error, se encola un mensaje de error y se termina el ciclo de vida, saltando al último paso (renderizar respuesta).
4. **Actualizar los valores del modelo** (*update model values*). Cuando se llega a esta fase, todos los valores se han procesado y se han validado. Se actualizan entonces las propiedades de los beans gestionados asociados a los componentes.
5. **Invocar a la aplicación** (*invoke application)* . Cuando se llega a esta fase, todas las propiedades de los beans asociados a componentes de entrada (*input*) se han actualizado. Se llama en este momento a la acción seleccionada por el usuario.
6. ***Renderizar* la respuesta** (*render response*). En HTML

JSF Architecture

A JSF application is similar to any other Java technology-based web application; it runs in a Java servlet container, and contains

- JavaBeans components as models containing application-specific functionality and data
- A custom tag library for representing event handlers and validators
- A custom tag library for rendering UI components
- UI components represented as stateful objects on the server
- Server-side helper classes
- Validators, event handlers, and navigation handlers
- Application configuration resource file for configuring application resources



Extensiones

- **RichFaces**
- **ICEfaces**
- **jQuery4jsf**
- **PrimeFaces**
- **OpenFaces**

A JavaBean is just a standard

- All properties private (use getters/setters)
- A public no-argument constructor
- Implements Serializable.
    - a process that converts an instance into a stream of bytes. Those bytes can be stored in files, sent over a network connection, etc, and have enough info to reconstruct the object later -- possibly in a different instance of the application, or even on a whole other machine!

Managed beans

- Managed Bean is a regular Java Bean class registered with JSF. In other words, Managed Beans is a java bean managed by JSF framework.
- The managed bean contains the getter and setter methods, business logic or even a backing bean (a bean contains all the HTML form value).
- Managed beans works as Model for UI component.
- Managed Bean can be accessed from JSF page.
- Eager true el managed bean se crea antes de ser requested, lazy por default solo se crea cuando de lo requested.

Scope	Description
@RequestScoped	Bean lives as long as the HTTP request-response lives. It get created upon a HTTP request and get destroyed when the HTTP response associated with the HTTP request is finished.
@ViewScoped	Bean lives as long as user is interacting with the same JSF view in the browser window/tab. It get created upon a HTTP request and get destroyed once user postback to a different view.
@SessionScoped	Bean lives as long as the HTTP session lives. It get created upon the first HTTP request involving this bean in the session and get destroyed when the HTTP session is invalidated.
@ApplicationScoped	Bean lives as long as the web application lives. It get created upon the first HTTP request involving this bean in the application (or when the web application starts up and the eager=true attribute is set in @ManagedBean) and get destroyed when the web application shuts down.

- **@ManagedProperty** annotation a managed bean's property can be injected in another managed bean.
-